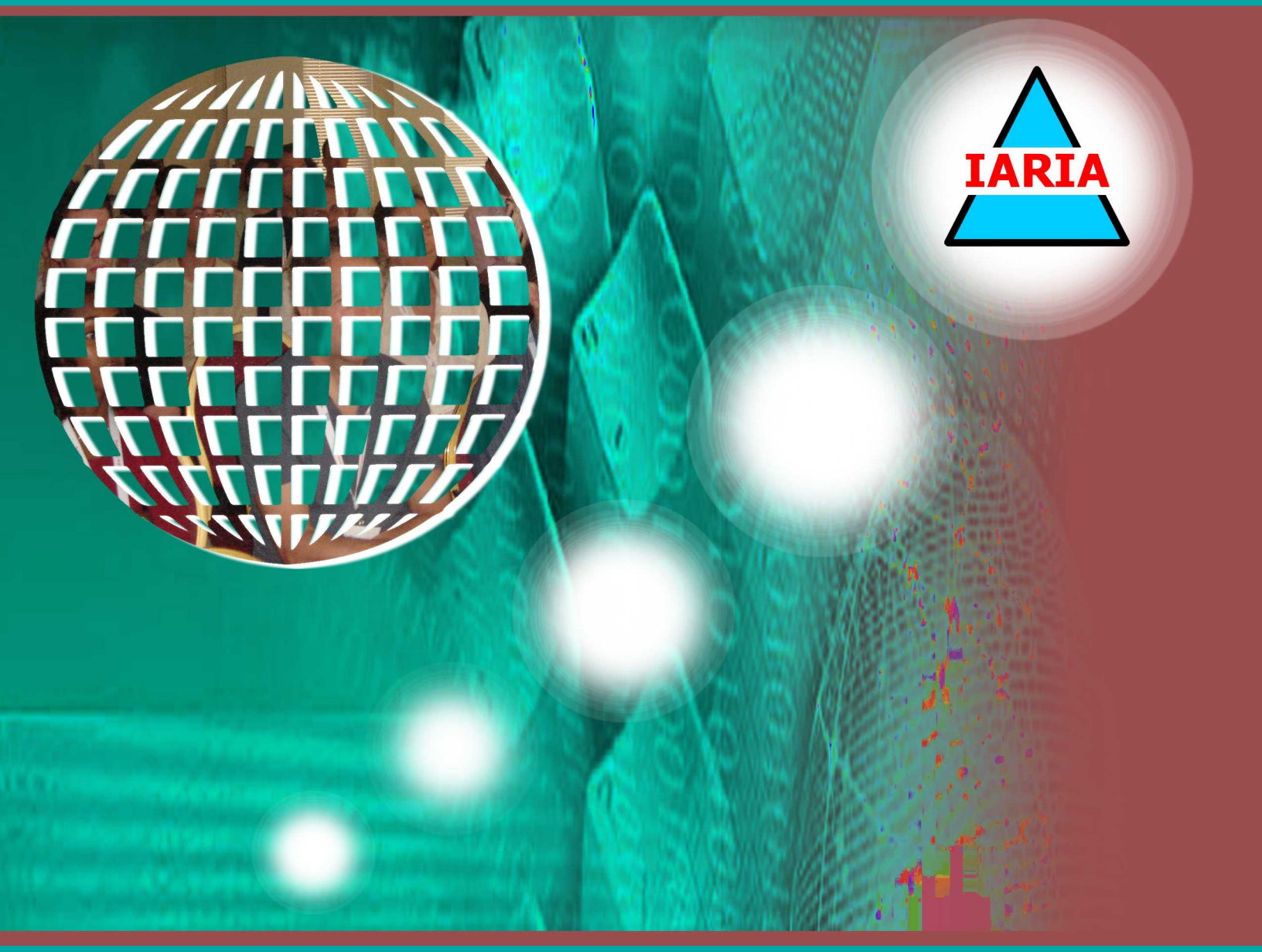
International Journal on Advances in Software



2026 vol. 19 nr. 1&2

The *International Journal on Advances in Software* is published by IARIA.
ISSN: 1942-2628
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Responsibility for the contents rests upon the authors and not upon IARIA, nor on IARIA volunteers, staff, or contractors.

IARIA is the owner of the publication and of editorial aspects. IARIA reserves the right to update the content for quality improvements.

Abstracting is permitted with credit to the source. Libraries are permitted to photocopy or print, providing the reference is mentioned and that the resulting material is made available at no cost.

Reference should mention:

International Journal on Advances in Software, issn 1942-2628
vol. 19, no. 1 & 2, year 2026, http://www.iariajournals.org/software/

The copyright for each included paper belongs to the authors. Republishing of same material, by authors or persons or organizations, is not allowed. Reprint rights can be granted by IARIA or by the authors, and must include proper reference.

Reference to an article in the journal is as follows:

<Author list>, "<Article title>"
International Journal on Advances in Software, issn 1942-2628
vol. 19, no. 1 & 2, year 2026,<start page>:<end page> , http://www.iariajournals.org/software/

IARIA journals are made available for free, proving the appropriate references are made when their content is used.

Sponsored by IARIA
www.iaria.org

Copyright © 2026 IARIA

International Journal on Advances in Software
Volume 19, Number 1 & 2, 2026

Editor-in-Chief

Petre Dini, IARIA, USA

Editorial Board

Chrissanthi Angeli, University of West Attica, Greece
Vincenzo Arceri, University of Parma, Italy
Thierry Badard, Université Laval, Canada
Fabian Barbato, UDELAR, Uruguay
Fernando Boronat Segui, Universitat Politecnica de Valencia, Spain
Mina Boström Nakićenović, Flightradar24, Stockholm, Sweden
Antonio Brogi, University of Pisa, Italy
Carlos Casanova, UTN-FRCU, Argentina
Yoonsik Cheon, The University of Texas at El Paso, USA
Stefano Cirillo, University of Salerno, Italy
Emanuele Covino, Università di Bari, Italy
Mirela Danubianu, "Stefan cel Mare " University of Suceava, Romania
Cláudio de Souza Baptista, Federal University of Campina Grande, Brazil
Maria del Pilar Angeles, Universidad Nacional Autonónoma de México, México
Rafael del Vado Vírseda, Universidad Complutense de Madrid, Spain
Leandro Dias da Silva, Universidade Federal de Alagoas, Brazil
Roland Dodd, CQUniversity, Australia
Ann Dunkin, US Department of Energy, USA
Gledson Elias, Federal University of Paraíba (UFPB), Brazil
Thomas Fehlmann, Euro Project Office, Switzerland
Stoyan Garbatov, OutSystems, Portugal
Apostolos Gkamas, University of Ioannina, Greece
Gregor Grambow, Aalen University, Germany
Christoph Grimm, Kaiserslautern University of Technology, Germany
Andreas Hausotter, University of Applied Sciences and Arts, Hanover, Germany
Tzung-Pei Hong, National University of Kaohsiung, Taiwan
Dayang Norhayati Abang Jawawi, Universiti Teknologi Malaysia (UTM), Malaysia
Ahmed Kamel, Concordia College, Moorhead, USA
Ayad Ali Keshlaf, Sabratha University, Libya
Maqbool Khan, Pak-Austria Fachhochschule - Institute of Applied Sciences and Technology, Pakistan
Radek Koci, Brno University of Technology, Czech Republic
Dmitry Korzun, Petrozavodsk State University, Russia
Jana Kostičová, Comenius University Bratislava, Slovakia
Luigi Lavazza, Università dell'Insubria, Italy
Zheng Li, Queen's University Belfast, UK
Panos Linos, Butler University, USA
Qifeng Lu, Pivotal Commware, USA
Herwig Mannaert, University of Antwerp, Belgium
Adriana Martin, Universidad Nacional de la Patagonia Austral, Argentina
Goran Martinovic, J. J. Strossmayer University of Osijek, Croatia
Paulo Martins, University of Trás-os-Montes e Alto Douro (UTAD), Portugal

José Carlos M. M. Metrolho, Polytechnic Institute of Castelo Branco, Portugal
Jose Manuel Molina Lopez, Universidad Carlos III de Madrid, Spain
Fernando Moreira, REMIT, Universidade Portucalense, Portugal
Roy Oberhauser, Aalen University, Germany
Constantin Paleologu, National University of Science and Technology Politehnica Bucharest, Romania
Elzbieta Pustulka, University of Applied Sciences and Arts Northwestern Switzerland (FHNW), Switzerland
Kornelije Rabuzin, University of Zagreb, Croatia
Piotr Ratuszniak, Koszalin University of Technology, Poland
Hajarisena Razafimahatratra, Ecole Nationale d'Informatique - Université de Fianarantsoa, Madagascar
José Rouillard, University of Lille, France
Claus-Peter Rückemann, Universität Münster / DIMF / Leibniz Universität Hannover, Germany
Sébastien Salva, IUT Clermont Auvergne | University of Clermont Ferrand, France
Patrizia Scandurra, Università degli Studi di Bergamo, Italy
Mu-Chun Su, National Central University, Taiwan
Maryam Tayefeh Mahmoudi, ICT Research Institute, Iran
Mónica Isabel Teixeira da Costa, Technology School | Polytechnic Institute of Castelo Branco, Portugal
Pierre Tiako, Langston University, USA
Božo Tomas, University of Mostar, Bosnia and Herzegovina
Mariusz Trzaska, Polish-Japanese Academy of Information Technology, Poland
Chrisa Tsinaraki, Technical University of Crete, Greece
Miroslav Velev, Aries Design Automation, USA
Mudasser F. Wyne, National University, USA
Martin Zinner, Technische Universität Dresden, Germany

International Journal on Advances in Software
Volume 19, Numbers 1 & 2, 2026

CONTENTS

pages: 1 - 12
Scalable Image Search Using CLIP: Architecture Evolution, GPU Acceleration, and Practical Evaluation of a Command-Line Tool
Yurij Mikhalevich, QA Wolf, Australia

pages: 13 - 27
Automatic Semantic Image Tagging at Scale Using CLIP: Method, Evaluation, and Comparison with Modern Approaches
Yurij Mikhalevich, QA Wolf, Australia

pages: 28 - 39
Extracting Entity Relationship Diagrams from Unstructured Text - A Hybrid Approach
Vaihunthan Vyramuthu, Aalen University, Germany
Gregor Grambow, Aalen University, Germany

pages: 40 - 54
Perceptions of AI-Edited Video News: Evaluation, Classification, and Attitudes
Caspian Jade Moosburner, RheinMain University of Applied Sciences, Germany
Matthias Kowald, RheinMain University of Applied Sciences, Germany
Till Dannewald, RheinMain University of Applied Sciences, Germany
Dennis Quandt, RheinMain University of Applied Sciences, Germany
Wolfgang Ruppel, RheinMain University of Applied Sciences, Germany
Matthias Narroschke, RheinMain University of Applied Sciences, Germany

pages: 55 - 66
KAN Control Framework for Continual Learning
Evgenii Ostanin, Toronto Metropolitan University, Canada
Nebojsa Djosic, Toronto Metropolitan University, Canada
Salah Sharieh, Toronto Metropolitan University, Canada
Alexander Ferworn, Toronto Metropolitan University, Canada

pages: 67 - 81
VR-ML: Visualization of Machine Learning Models and their Evolution in Virtual Reality
Roy Oberhauser, Aalen University, Germany

pages: 82 - 95
Brains, Brawn, and Silicon: Evaluating LLM Performance for Code Generation across Intel and Apple M3 Max Architectures
Miren Illarramendi Rezabal, Mondragon Goi Eskola Politeknikoa S.Coop, Spain
Joseba Andoni Agirre, Mondragon Goi Eskola Politeknikoa S.Coop, Spain
Aitor Picatoste, Mondragon Goi Eskola Politeknikoa S.Coop, Spain
Juan Ignacio Igartua, Mondragon Goi Eskola Politeknikoa S.Coop, Spain

Scalable Image Search Using CLIP: Architecture Evolution, GPU Acceleration, and Practical Evaluation of a Command-Line Tool

Yurij Mikhalevich
QA Wolf
Melbourne, Australia
email: yurij@mikhalevi.ch

***Abstract* - This paper presents an extended study of `rclip`, a practical, scalable, command-line image search engine that leverages Contrastive Language-Image Pre-Training to enable natural language queries over unlabeled image collections. Extending the original conference paper, this work describes the substantial architectural evolution of `rclip`, including its migration from OpenAI's `clip` library to the open-source `open_clip` library, the introduction of graphics processing unit acceleration via Apple's Metal Performance Shaders backend, configurable batched indexing for improved throughput, and a text-only model loading optimization that significantly reduces query latency when indexing is not required. The paper also reports corrected benchmark results for the ImageNet-1k dataset: the original conference paper reported a top-1 accuracy of 31.17% due to a bug in the benchmark code that silently discarded 488 of the 1000 ImageNet classes; the corrected measurement yields 58.78% top-1 accuracy with `ViT-B-32-quickgelu`, the `open_clip` variant that uses the original OpenAI Contrastive Language-Image Pre-Training model weights. Metal Performance Shaders acceleration further reduces indexing time for 1.28 million images on an Apple M1 Max from 4 hours 52 minutes to 2 hours 16 minutes, a 2.15× speedup. Additional improvements include expanded image format support, an improved file system traversal mechanism, and an evolved database schema. The results demonstrate that semantic image search powered by vision-language models is now practical for everyday use on consumer hardware, through an accessible command-line interface, without cloud infrastructure or specialized servers.**

Keywords* - *image search; image indexing; CLIP; natural language processing; computer vision.

I. Introduction

This paper is an extended version of the original conference paper [1], which introduced `rclip`, a command-line tool for searching unlabeled image collections using natural language, and demonstrated its practical viability. It substantially extends that work with new material on `rclip`'s evolution, updated experimental results, and an expanded literature review.

The release of the Contrastive Language-Image Pre-Training (CLIP) model by OpenAI in 2021 has generated significant attention in the field of Natural Language Processing (NLP) and Computer Vision (CV). This model has demonstrated the ability to learn state-of-the-art image representations from a massive dataset of 400 million (image, text) pairs that were collected from the Internet. CLIP can predict the most relevant text snippet, given an image, using natural language instructions without explicitly optimizing for the task. This zero-shot ability is similar to that of GPT-2 and 3 [2]. The authors of CLIP have demonstrated that CLIP performs as well as the original ResNet50 on ImageNet in a zero-shot manner without using any of the original 1.28M labeled examples. This accomplishment overcomes significant challenges in computer vision.

In addition to these capabilities, CLIP allows researchers to perform image searches using natural language queries. This is precisely the application that `rclip` explores, demonstrating the practical viability of searching unlabeled image collections using natural language.

Since the publication of the original conference paper, the CLIP ecosystem has evolved considerably. The open-source `open_clip` library [3] has emerged as a reproducible alternative to OpenAI's original implementation, trained on publicly available datasets such as LAION-5B [4]. GPU acceleration for machine learning workloads on consumer hardware has also become more accessible, with Apple's Metal Performance Shaders (MPS) backend for PyTorch [5] enabling GPU-accelerated inference on Mac devices without requiring dedicated NVIDIA hardware. These developments have motivated a substantial evolution of `rclip`'s architecture and capabilities.

The exponential growth of data, particularly in the form of images, makes this research especially relevant. With the proliferation of digital devices and the Internet, more and more images are being produced and shared online every day. As a result, it is becoming increasingly challenging to find specific images manually. A robust and efficient image search solution can help users and businesses quickly find the images they need amidst this growing sea of data. Moreover, as the volume of photos being shared online continues to increase, it becomes increasingly critical to protect intellectual property and online security. An efficient image search solution can help identify and remove images that violate copyright laws or contain sensitive or inappropriate content. Therefore, with the growing amount of data and images being created, an efficient image search solution is becoming increasingly necessary and relevant.

This extended journal paper makes the following contributions beyond the original conference publication:

- A detailed description of `rclip`'s migration from OpenAI's `clip` library to the open-source `open_clip` library, including the rationale and practical implications.
- An analysis of GPU acceleration through Apple's MPS backend, with quantitative performance comparisons against CPU-only execution.
- Corrected benchmark results for the ImageNet-1k dataset, along with a thorough explanation of the bug that affected

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

the original measurements.

- An expanded implementation description covering batched indexing, text-only model loading, expanded image format support, and database schema evolution.
- An expanded literature review covering recent developments in the CLIP ecosystem, CLIP-based retrieval tools, and GPU acceleration for consumer hardware.

The remainder of this paper is organized as follows. Section II explores related work in image search, CLIP-based systems, and GPU acceleration. Section III describes the proposed method for image search using CLIP, including extensions for query arithmetic. Section IV presents the implementation details, including the architectural changes introduced since the original publication. Section V discusses indexing and search performance, including MPS acceleration results. Section VI presents corrected search quality measurements and discusses the benchmark bug found in the original paper. Section VII situates `rclip` among other CLIP-based retrieval tools. Section VIII discusses practical implications, limitations, and lessons learned. Finally, Section IX concludes the paper and discusses future work.

II. Related Work

There exist multiple methods and solutions for image search. This section reviews both traditional approaches and the more recent developments in the CLIP ecosystem that are most relevant to this work.

A. Traditional Image Search Methods

Traditional image search methods rely on bag-of-features, which are sets of features that describe the contents of an image. These features can be manually specified by annotating the image with descriptive labels. For instance, Fiedler et al. [6] proposed an image tagging software that helps users enter the labels efficiently. Another way to obtain labels is to use labels injected into the photos by camera software, as explored by Tesic [7], who examined camera metadata for consumer photos.

Mobile phone software can also inject useful metadata within photos. Kim et al. [8] explored how this metadata can be leveraged to effectively manage and search photos using mobile smartphones. This metadata can include information about the location, time, or type of camera used, among others. By using such metadata, it's possible to automatically generate labels for images.

One way to automatically generate labels is by using image recognition algorithms, such as Convolutional Neural Networks (CNNs). CNNs can be specifically trained to recognize and classify images based on their contents. For example, Krizhevsky et al. [9] trained a CNN to recognize images based on their visual features. This approach can be combined with other methods, such as using camera metadata or manual annotations, to generate more accurate and diverse labels for images. Lee et al. [10] proposed a scalable method for image annotation that combines manual and automatic approaches to improve the accuracy and scalability of the labeling process.

In image retrieval, search algorithms are applied to the features extracted from images to find relevant images based on a user's query. The choice of search algorithm depends on the type of labels associated with the images and the problem at hand. For instance, if the labels are GPS coordinates, a distance-based search algorithm can be used to find images related to the query location, as demonstrated by Zhang et al. [11].

On the other hand, if the labels are in text form, a text-based search algorithm can be employed to retrieve images based on textual queries. There are various techniques available for text-based search, ranging from substring matching to lemmatization-based search, as shown by Balakrishnan et al. [12], or even using word embeddings, as explored by Günther [13] and Kenter et al. [14].

In practice, combining multiple features and approaches, such as GPS coordinates, text labels, image capture date, camera model, focal distance, etc. can yield more accurate and relevant results. Ismail [15] investigated image annotation and retrieval based on multi-modal feature clustering and found that this approach can significantly improve the retrieval performance of the system.

Therefore, in summary, image retrieval is a complex process that involves the extraction of features from images, the selection of suitable search algorithms based on the type of labels and the integration of multiple features and approaches to improve retrieval accuracy.

The methods mentioned earlier do not facilitate searching for images using a text query without first identifying the features. Moreover, they do not permit searching for a concept that is not included in the image labels, even if the concept exists within the image. These are the problems that OpenAI's CLIP [2] enables us to solve.

B. CLIP and Vision-Language Models

CLIP [2] introduced a paradigm shift in image understanding by jointly training an image encoder and a text encoder on 400 million image-text pairs using a contrastive learning objective. The resulting model maps images and text into a shared embedding space, enabling zero-shot transfer to a wide variety of downstream tasks. The image encoder in CLIP can be either a ResNet or a Vision Transformer (ViT) [16]. The ViT architecture, which treats an image as a sequence of fixed-size patches and processes them with a standard Transformer encoder, has proven particularly effective for learning rich visual representations. CLIP's `ViT-B/32` variant, which uses 32×32 pixel patches, offers a practical balance between computational cost and representation quality, and is the variant used in `rclip`.

Following CLIP, several related vision-language models have been proposed. ALIGN [17] demonstrated that similar zero-shot transfer capabilities can be achieved with noisier but larger-scale training data (1.8 billion image-alt-text pairs), using a simpler dual-encoder architecture without the careful data curation employed by CLIP. BLIP [18] extended the paradigm by bootstrapping captions from a noisy web dataset

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

and introducing a multimodal mixture of encoder-decoder architectures that unifies vision-language understanding and generation. SigLIP [19] replaced CLIP's softmax-based contrastive loss with a sigmoid loss that operates on independent image-text pairs rather than requiring a full pairwise similarity matrix, enabling more efficient training at larger batch sizes.

C. OpenCLIP and Reproducible Models

A significant limitation of the original CLIP release was that OpenAI did not publish its training data or training code, making it impossible for the research community to reproduce or extend the training process. The `open_clip` library [3] addressed this gap by providing a fully open-source implementation of CLIP's training and inference pipeline. Cherti et al. conducted an extensive study of scaling laws for contrastive language-image learning, training models on publicly available datasets including LAION-400M and LAION-2B [4] (a 2-billion-sample subset of LAION-5B). Their work demonstrated that the scaling behavior of CLIP-like models is predictable and reproducible when both code and data are open. The `open_clip` library provides pre-trained model checkpoints with detailed documentation of training configurations, including the specific dataset, image resolution, and number of training samples seen. This transparency was a key motivation for `rclip`'s migration from OpenAI's `clip` library to `open_clip`, as discussed in Section IV.

D. Similarity Search at Scale

For large-scale image retrieval systems, the efficiency of the similarity search step becomes critical. Libraries such as FAISS [20] provide highly optimized implementations of approximate nearest neighbor (ANN) algorithms, including Inverted File Indices (IVF) and Hierarchical Navigable Small World (HNSW) graphs, that can search through billions of vectors in milliseconds. These approaches trade a small amount of retrieval accuracy for dramatic improvements in query latency.

However, ANN indices introduce additional complexity: they require a separate index-building step, consume additional memory, and their accuracy depends on tuning parameters such as the number of clusters or graph connectivity. For `rclip`'s target use case—personal photo collections and moderately sized image catalogs typically containing thousands to low millions of images—the brute-force dot product computation over cached vectors remains fast enough. As shown in Section V, querying 1.28 million images takes approximately 11 seconds on an Apple M1 Max, which is acceptable for an interactive command-line tool. Should future scaling requirements demand sub-second queries over tens of millions of images, integrating an ANN index would be a natural extension.

E. GPU Acceleration on Consumer Hardware

The introduction of the MPS backend for PyTorch [5] in 2022 marked an important step toward accessible GPU acceleration on consumer hardware. MPS enables PyTorch operations to execute on Apple Silicon GPUs (M1, M2, M3, M4, and M5 series) on Mac systems. The unified memory architecture of Apple Silicon is particularly well-suited for machine learning inference workloads, as it eliminates the need to copy data between CPU and GPU memory. This is relevant for `rclip` because many of its users run the tool on Mac laptops and desktops. The MPS backend enables `rclip` to leverage hardware acceleration, significantly improving the indexing throughput as demonstrated in Section V.

III. Method

With CLIP's text transformer, it is possible to convert a text query to a n-dimensional vector (where n differs depending on the CLIP model used). With CLIP's image transformer, it is possible to convert an image to a n-dimensional vector. Then, we can calculate the dot product (Equation (1)) of the normalized query vector and each of the normalized image vectors. After this, we sort the images by the decreasing dot product and take first k images; this gets us the k images that match the query the most.

$$\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \tag{1}$$

where $\boldsymbol{a}$ is the L2-normalized query vector (text or image) and $\boldsymbol{b}$ is the L2-normalized image feature vector.

It is worth noting that because both the query and image vectors are L_2-normalized (i.e., $\|\boldsymbol{a}\| = \|\boldsymbol{b}\| = 1$), the dot product is equivalent to cosine similarity (Equation (2)). This normalization ensures that similarity scores are bounded between -1 and 1, enabling meaningful comparison across different queries and image collections.

$$\text{cosine_similarity}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\|\boldsymbol{a}\| \|\boldsymbol{b}\|} = \boldsymbol{a} \cdot \boldsymbol{b} \tag{2}$$

where the last equality holds when $\|\boldsymbol{a}\| = \|\boldsymbol{b}\| = 1$.

While this approach works reasonably fast on a few images, it may not be scalable when dealing with a large number of images. This is particularly challenging if there is no access to GPUs to run the CLIP model. In order to address this scalability issue, the solution proposed in this paper suggests caching the image vectors. By caching the image vectors, repeated queries can be executed quickly, without having to wait for the image to be processed each time that a query is made.

Caching involves storing the computed image vectors on disk after the initial processing of images so that they can be accessed later. When a new query is received, the system retrieves the cached image vectors and computes only the query vector. This approach reduces the computational overhead associated with image querying and improves the response time for users. This is particularly useful when dealing with large image databases, where repeated queries are common. By leveraging caching, the system can handle repeated queries quickly and efficiently without having to recompute the image vectors each time.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

The proposed solution also allows adding new images to the cache to avoid recomputing the whole cache when the image catalog is updated.

A. Query Arithmetic

Beyond simple text queries, the current version of `rclip` supports query arithmetic, enabling users to refine their search results through additive and subtractive query composition. Users can specify positive queries (which increase the similarity score for matching images) and negative queries (which decrease it). The combined feature vector is computed as:

$$\boldsymbol{f} = \sum_i \boldsymbol{q}_i^+ - \sum_j \boldsymbol{q}_j^- \tag{3}$$

where $\boldsymbol{q}_i^+$ denotes positive query feature vectors and $\boldsymbol{q}_j^-$ denotes negative query feature vectors. This is analogous to the vector arithmetic property observed in word embeddings, where semantic relationships can be expressed as vector operations.

Additionally, `rclip` supports query multipliers, allowing users to assign different weights to individual query components. A multiplied query takes the form `<multiplier>:<query>`, where the multiplier scales the corresponding feature vector before summation. This enables fine-grained control over the relative importance of different query aspects.

B. Multimodal Queries

A distinctive feature of `rclip`'s method is its support for multimodal queries. Because CLIP maps both images and text into the same embedding space, `rclip` can accept not only text queries but also image files and URLs as query inputs. When an image is provided as a query, `rclip` computes its CLIP embedding and uses it as the query vector, effectively performing image-to-image similarity search. This enables use cases such as finding visually similar photos in a collection, identifying near-duplicates, or locating different shots from the same scene.

Text and image queries can also be combined in a single search. For example, a user could search for images similar to a reference photo but with specific characteristics described in text (e.g., providing a landscape photo as input and adding the text query "sunset" to find similar landscapes specifically at sunset). The feature vectors from all query components—text, local image files, and URL-referenced images—are summed element-wise, with optional multipliers for weighting, to produce the final composite query vector.

The main advantage of the proposed method over the methods described in Section II-A is that the method does not require any metadata, labels, or annotations, which enables using it with any image catalog. Moreover, if used on labeled images, the proposed method allows searching for concepts not included in the image labels. The ability to combine text and image queries, apply query arithmetic with positive and negative components, and weight individual query parts with multipliers provides a flexible and expressive query language that enables users to utilize CLIP effectively in real-world scenarios involving image search.

IV. Implementation

The method described above is implemented in the Python utility called `rclip` [21]. `rclip` provides an easy-to-use command-line interface (CLI) that allows users to search images within any directory on a computer where `rclip` is installed. Search within nested directories is also supported. To use it, the user should open the terminal, navigate to the directory containing the files that they want to search through, type `rclip <search query>`, and hit "Enter."

Since the original conference publication, `rclip` has undergone a substantial architectural evolution. The following subsections detail the key changes and their rationale. Figure 1 shows the overall system pipeline.

A. Migration to OpenCLIP

The original version of `rclip` used OpenAI's `clip` library [2] directly to load the `ViT-B/32` model and compute feature vectors. While functional, the library received minimal updates after its initial release, and the training dataset (WebImageText) was never publicly released, making it impossible for the community to reproduce the training process or train new models.

The current version of `rclip` uses the `open_clip` library [3]. While both OpenAI's `clip` and `open_clip` provide open-source inference code, `open_clip` additionally includes the full training pipeline and provides checkpoints trained on public datasets such as LAION-5B [4], enabling fully reproducible model training. The `open_clip` library is also more actively maintained and offers a wider selection of model variants. `rclip` uses the `ViT-B-32-quickgelu` variant with the `openai` checkpoint. The `quickgelu` suffix indicates that this variant preserves the QuickGELU activation function from the original OpenAI CLIP implementation, ensuring that model behavior is identical to the original. This migration also positions `rclip` to adopt fully reproducible, publicly trained checkpoints in the future.

Figure 2 shows the model initialization code, which is where the key differences from the original implementation are concentrated. The original version used `clip.load()` to instantiate the model on CPU; the current version uses `open_clip.create_model_and_transforms()` with an explicit device parameter that enables MPS acceleration (described in Section IV-B). The feature computation methods themselves (`compute_image_features` and `compute_text_features`) follow the same pattern as the original implementation: preprocess, encode, normalize. The notable difference is in text feature computation (Figure 3), which uses a separate text-only model (`self._model_text`) instead of the full vision-language model, as described in Section IV-D.

B. GPU Acceleration with MPS

A significant performance improvement in the new version of `rclip` is the support for GPU-accelerated inference

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

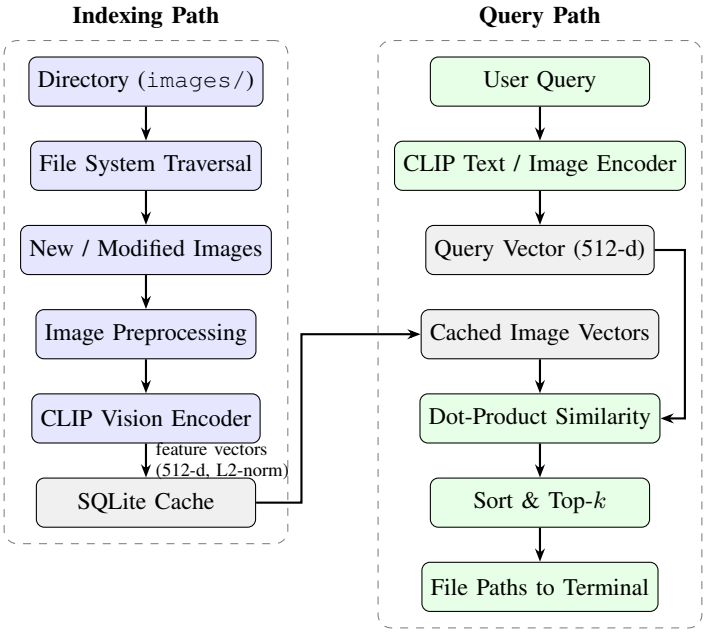


Figure 1. rclip system pipeline. The *indexing path* (left) traverses the directory, filters new or modified images, runs them through the CLIP vision encoder, and stores the resulting 512-dimensional L2-normalized feature vectors in a SQLite cache. The *query path* (right) encodes the user's text or image query with the CLIP encoder, retrieves cached image vectors from SQLite, computes dot-product (cosine) similarity, and returns the top-k matching file paths.

```
def _load_model(self):
  self._model, _, self._preprocess = (
    open_clip.create_model_and_transforms
        (
      self._model_name,
      pretrained=self._checkpoint_name,
      device=self._device,
    ))
```

Figure 2. Model initialization using `open_clip` with device selection.

```
def compute_text_features(self, text):
  with torch.no_grad():
    tokenized_text = self._tokenizer(text
        ).to(self._device)
    text_features = self._model_text.
        encode_text(tokenized_text)
    text_features /= text_features.norm(
        dim=-1, keepdim=True)
  return text_features.cpu().numpy()
```

Figure 3. Text vector computing method using a separate text-only model (`self._model_text`).

through Apple's MPS backend [5]. On macOS systems with Apple Silicon, `rclip` automatically detects MPS availability and uses it as the default compute device. Users can override this behavior by passing the `--device cpu` flag to force CPU-only execution.

The device selection logic is straightforward: `rclip` checks whether the system is running macOS and whether the MPS backend is available in PyTorch. When MPS is available, all model parameters and input tensors are moved to the MPS device, and the GPU handles the computationally intensive CLIP encoding operations. The unified memory architecture of Apple Silicon means that there is no explicit data transfer overhead between CPU and GPU memory, which is particularly beneficial for the image preprocessing pipeline where images are loaded and preprocessed on the CPU before being encoded on the GPU.

The performance impact of MPS acceleration is quantified in Section V.

C. Batched Indexing

The original version of `rclip` processed images in fixed batches of 8 during indexing. The current version introduces a configurable batch size, controllable through the `--indexing-batch-size` command-line argument so that users can tune it based on their available memory and hardware capabilities.

Larger batch sizes improve throughput by amortizing the fixed overhead of model invocation across more images. However, they also increase peak memory consumption, as all images in a batch must be held in memory simultaneously. The configurable batch size allows users to find the optimal tradeoff for their specific hardware configuration.

D. Text-Only Model Loading

A notable optimization in the new version of `rclip` is the ability to load only the text transformer portion of the CLIP model when performing queries that do not require image indexing. The full CLIP model includes both a vision

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

transformer and a text transformer, but when a user passes the `--no-indexing` flag (indicating that no new images need to be indexed), only the text transformer is needed to compute query vectors.

The text-only model is extracted from the full model by creating a deep copy and removing the visual component. This stripped-down model is then serialized to disk using PyTorch's `torch.save` mechanism, along with a version file that tracks the `open_clip` library version used to create it. On subsequent runs, if the `open_clip` version has not changed, `rclip` loads the cached text-only model directly, bypassing the need to download and instantiate the full vision-language model. This optimization significantly reduces startup time for search-only queries, as the text transformer is substantially smaller than the full model.

Table I quantifies the impact of this optimization. The measurements were performed on the test image set included in the `rclip` repository. The text-only model reduces query time by approximately 50% and memory consumption by 3.78×. The RAM reduction comes primarily from not loading the vision transformer and is independent of the number of indexed images. The query time improvement may be less pronounced for larger datasets, where loading cached image vectors from disk becomes the dominant cost.

E. Expanded Image Format Support

The original version of `rclip` supported only JPEG and PNG image formats. The current version adds support for WebP, High Efficiency Image Container (HEIC), and RAW formats (ARW and CR2). HEIC support is enabled through the `pillow-heif` library [22], which registers a HEIF opener with Pillow [23]. RAW image support is provided as an experimental feature, enabled via the `--experimental-raw-support` flag. When RAW support is active, `rclip` intelligently skips RAW files that have a corresponding processed image (e.g., a JPEG alongside an ARW file), avoiding duplicate entries in the index.

F. Improved File System Traversal

The original implementation used Python's `os.walk` function to traverse directories, which requires a separate `os.path.getmtime` and `os.path.getsize` call for each file to obtain its metadata. The current version uses `os.scandir`, which returns `DirEntry` objects that cache file metadata from the initial directory listing, avoiding redundant system calls.

Additionally, the new implementation runs a separate thread to count the total number of image files in the directory tree, enabling a progress bar with accurate completion percentage during indexing. The directory traversal itself uses an iterative stack-based approach rather than the recursive `os.walk`, giving `rclip` more control over the traversal order and enabling the exclusion of directories (such as `.git` and `node_modules`) at the directory entry level rather than after recursion.

TABLE I. Impact of text-only model loading on search-only queries.

Metric	Full Model	Text-Only	Improvement
Query time	3.8s	1.9s	2× faster
RAM usage	1.7 GB	0.45 GB	3.78× less

```
self._con.execute("""
  CREATE TABLE IF NOT EXISTS images (
    id INTEGER PRIMARY KEY,
    deleted BOOLEAN,
    indexing BOOLEAN,
    filepath TEXT NOT NULL UNIQUE,
    modified_at DATETIME NOT NULL,
    size INTEGER NOT NULL,
    vector BLOB NOT NULL
  )
""")
```

Figure 4. Updated structure of the "images" table (schema version 2).

G. Database Schema Evolution

`rclip` features the image vector cache implemented using the `SQLite 3` relational database management system (RDBMS) [24]. The image vector is computed and added to the index only if the cache does not already contain an entry for a given image. The vectors are cached by image paths. This allows the user to execute repeated queries over the same image catalog without waiting for the image vectors to be recomputed.

The database schema has evolved from version 1 to version 2 since the original publication. Figure 4 shows the current schema. The most notable addition is the `indexing` column, which is used to track the indexing state of images during the re-indexing process. Previously, the `deleted` flag was set on all images before re-indexing and then cleared as each image was confirmed to still exist. This approach was problematic because it briefly marked all images as deleted, which could cause issues if the process was interrupted. The new approach uses a separate `indexing` flag: images are marked as "indexing" at the start, the flag is cleared as each image is confirmed, and any images still flagged as "indexing" at the end are marked as deleted. This ensures that the index remains consistent even if the process is interrupted.

The database module also includes a version migration system that automatically applies schema changes when a user upgrades `rclip`. When the stored schema version is lower than the expected version, the migration logic adds the new `indexing` column via an `ALTER TABLE` statement and updates the version number.

When the user executes the `rclip` command, the tool first recursively processes the current directory and all its subdirectories, finds all of the images in them, and, if the image is not already present in the cache, computes their feature vectors and saves them to the SQLite cache database. This step is quick on repeated queries because the images are

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

already cached, but if the user wants to skip the process of checking that the cache is up-to-date, they can pass the `-n` argument to `rclip` to skip the indexing step completely and speed up the `rclip` command execution even more.

Then, `rclip` computes the query vector, fetches from the cache database image vectors for all of the images located in the current directory, computes the similarity score between the query vector and each of the image vectors, sorts the scores by the decreasing order, and gets the first k images with the highest similarity scores.

Finally, `rclip` prints the paths to the images that match the query to the terminal. Users can then open the images in their favorite image viewer or editor.

The `rclip` source code is published on GitHub under the MIT license [21].

V. Performance

This section presents performance benchmarks for the current version of `rclip`, including comparisons with the original version and the impact of MPS GPU acceleration.

A. Model Comparison on Low-End Hardware

As reported in the original conference paper, `rclip` was benchmarked using two different CLIP models, `ViT-B/32` (smaller CLIP model) and `ViT-L/14@336px` (larger CLIP model), on a NAS running `Intel(R) Celeron(R) CPU J3455 @ 1.50GHz`. Table II shows how `rclip` performs when indexing and searching through 269 photos when running on this CPU.

As Table II shows, the `ViT-L/14@336px` performance will not scale well, which makes `rclip` unusable in practical scenarios when running CLIP on low-level and mid-level consumer CPUs. This is why `rclip` uses `ViT-B/32`.

Running `rclip` indexing with `ViT-B/32` on 72,769 photos on the same NAS powered by `Intel(R) Celeron(R) CPU J3455 @ 1.50GHz` took 23 hours. Performing a query over 72,769 photos takes 56 seconds.

B. MPS GPU Acceleration

Table III presents the updated performance measurements on an `Apple M1 Max`, comparing MPS-accelerated indexing against CPU-only indexing on the ImageNet-1k 1.28 million image train set.

The MPS backend provides a 2.15× speedup for indexing 1.28 million images. This improvement comes from offloading the CLIP vision transformer forward pass to the Apple Silicon GPU. The CPU-only indexing rate of 73.1 images per second still represents a significant improvement over the original version, which achieved approximately 42 images per second on the same CPU (indexing 1.28 million images in 8 hours 31 minutes, as reported in the original paper). This additional improvement is primarily attributable to the improved file system traversal mechanism (as described in Section IV-F) that uses `os.scandir` and optimizes the directory search loop.

C. Performance Improvements Summary

Table IV provides a comprehensive comparison of the performance characteristics between the original version of `rclip` (as described in the conference paper) and the current version, all measured on the ImageNet-1k 1.28 million image train set using the `Apple M1 Max`.

The combined effect of file system traversal optimizations and MPS GPU acceleration yields a 3.76× improvement in indexing throughput compared to the original version. Even without GPU acceleration, the CPU-only current version achieves a 1.75× improvement, demonstrating that the software-level optimizations (specifically the `os.scandir` traversal mechanism) contribute meaningfully independent of hardware acceleration.

D. Scalability

To give a better understanding of how `rclip` performance scales, Table V shows how `rclip` performs when indexing and searching through 50k images and 1.28m images on the `Apple M1 Max` CPU. As the Table V shows, the indexing time scales linearly with the number of images when the search time increases only slightly even when going from searching through 50 thousand images to searching through 1.28 million images.

It should be noted that real-life `rclip` application possibilities go far beyond results demonstrated in these benchmarks because `rclip` can be used to execute any queries and not just the ones present in the dataset labels.

VI. Search Quality

This section presents corrected and expanded search quality measurements. An important erratum regarding the original benchmark results is discussed first, followed by the corrected measurements.

A. Erratum: Original ImageNet-1k Benchmark

The original conference paper [1] reported a top-1 accuracy of 31.17% and a top-5 accuracy of 44.80% on the ImageNet-1k [25] 1.28 million images train set. These numbers were incorrect due to a bug in the benchmark code.

The bug was located in the original benchmark script [26]. In the original code, the number of classes was inadvertently

TABLE II. Indexing and search performance on `Intel(R) Celeron(R) CPU J3455 @ 1.50GHz` using different CLIP models.

Model	Indexing Time	Search Time
`ViT-B/32`	3m56.626s	0m18.064s
`ViT-L/14@336px`	125m0.507s	3m19.742s
Difference	x31.70	x11.06

TABLE III. Indexing performance on `Apple M1 Max` using `ViT-B-32-quickgelu`: MPS vs CPU.

Device	Indexing Time	Throughput	Speedup
MPS (GPU)	2h16m01s	157.0 img/s	2.15×
CPU only	4h52m12s	73.1 img/s	1.00×

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE IV. Summary of performance improvements: original vs. current `rclip` on ImageNet-1k 1.28m images (Apple M1 Max).

Metric	Original Version	Current (CPU)	Current (MPS)
CLIP library	OpenAI `clip`	`open_clip`	`open_clip`
Indexing time (1.28m images)	8h 31m	4h 52m	2h 16m
Indexing throughput	∼41.7 img/s	73.1 img/s	157.0 img/s
Speedup vs. original	1.00×	1.75×	3.76×
Text-only model loading	No	Yes	Yes
Batch size	Fixed (8)	Configurable	Configurable
File system traversal	`os.walk`	`os.scandir`	`os.scandir`

derived from `image_features.shape[0]`, which returns the feature vector dimension (512 for `ViT-B/32`), not the number of ImageNet classes (1000). The label-to-class mapping was then constructed using Python's `zip` function, which stops at the shorter iterable. Since the feature dimension (512) is smaller than the number of classes (1000), `zip` silently dropped 488 of the 1000 ImageNet classes from consideration. This meant that every top-1 and top-5 accuracy measurement was computed over only 512 classes instead of the full 1000, systematically underreporting the accuracy.

The CIFAR-100 benchmark results (55.15% top-1, 81.34% top-5) reported in the original paper were not affected by this bug, as the CIFAR-100 benchmark used a different code path that correctly handled all 100 classes.

B. Corrected Results

Table VI presents the corrected search quality measurements using the fixed benchmark code. The results are shown for `ViT-B-32-quickgelu`, the model used by the current version of `rclip`.

The corrected top-1 accuracy on ImageNet-1k is 58.78%, nearly twice the originally reported 31.17%.

C. Context for the Results

It is important to note that `rclip`'s search quality evaluation differs from the standard zero-shot classification approach used in the original CLIP paper. In the CLIP paper, zero-shot classification is performed by computing class description vectors using prompt templates such as "a photo of a {class}"/"a blurry photo of a {class}"/"a bad photo of a {class}" and selecting the class with the highest cosine similarity to the image vector. The original CLIP paper reported 63.2% zero-shot top-1 accuracy on ImageNet with `ViT-B/32`.

`rclip`'s benchmark follows a similar approach but uses a simpler prompt template ("photo of {class}") without the ensemble of prompt templates used in the original CLIP paper, which employed 80 different prompt templates and averaged their embeddings. This methodological difference partially explains the gap between `rclip`'s 58.78% and CLIP's reported 63.2%. Additionally, `rclip` benchmarks against the 1.28 million image train set rather than the 50,000 image validation set, and the train set contains images with varying quality and ambiguity.

TABLE V. Indexing and search performance of the original version of `rclip` on `Apple M1 Max` CPU using `ViT-B/32`.

Dataset	# of images	Indexing Time	Search Time
`ImageNet-1k` validation set	50k	19m24.750s	0m4.04s
`ImageNet-1k` train set	1.28m	8h31m26.680s	0m11.49s
Difference	x25.62	x26.35	x2.84

TABLE VI. Corrected `rclip` search quality.

Dataset	Top-1	Top-5
`ImageNet-1k` 1.28m	58.78%	84.21%
`CIFAR-100` 10k	55.15%	81.34%

D. Qualitative Examples

To get a better understanding of `rclip`'s performance, see Figure 6, Figure 7, and Figure 8 showing search results for a search performed on an unlabeled demo set of 142 images. Figure 5 gives a glimpse into the set and shows that there is a variety of different images present there.

These qualitative results show that `rclip` returns visually and semantically coherent matches for both concrete object queries (Figure 6, "cat") and abstract compositional queries (Figure 7, "leading lines"), and that it also handles longer descriptive prompts effectively (Figure 8).

VII. Comparison with Other CLIP-Based Tools

Several other tools have been developed to leverage CLIP for image search. This section compares `rclip` with the most notable ones and discusses the design philosophy differences that distinguish these approaches.

`clip-retrieval` [27] is a comprehensive toolkit that provides components for computing CLIP embeddings (`clip inference`), building searchable indices (`clip index`), hosting them as a web service (`clip back`), and querying them through a web frontend (`clip front`). It is designed primarily for large-scale, server-based deployments and integrates with FAISS for approximate nearest neighbor search. In contrast, `rclip` is designed as a lightweight, single-user command-line tool that runs entirely locally and requires no server infrastructure. The `clip-retrieval` approach is better suited for organizations that need to serve multiple concurrent users over a shared image collection, while `rclip` targets individual users who want to quickly search

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

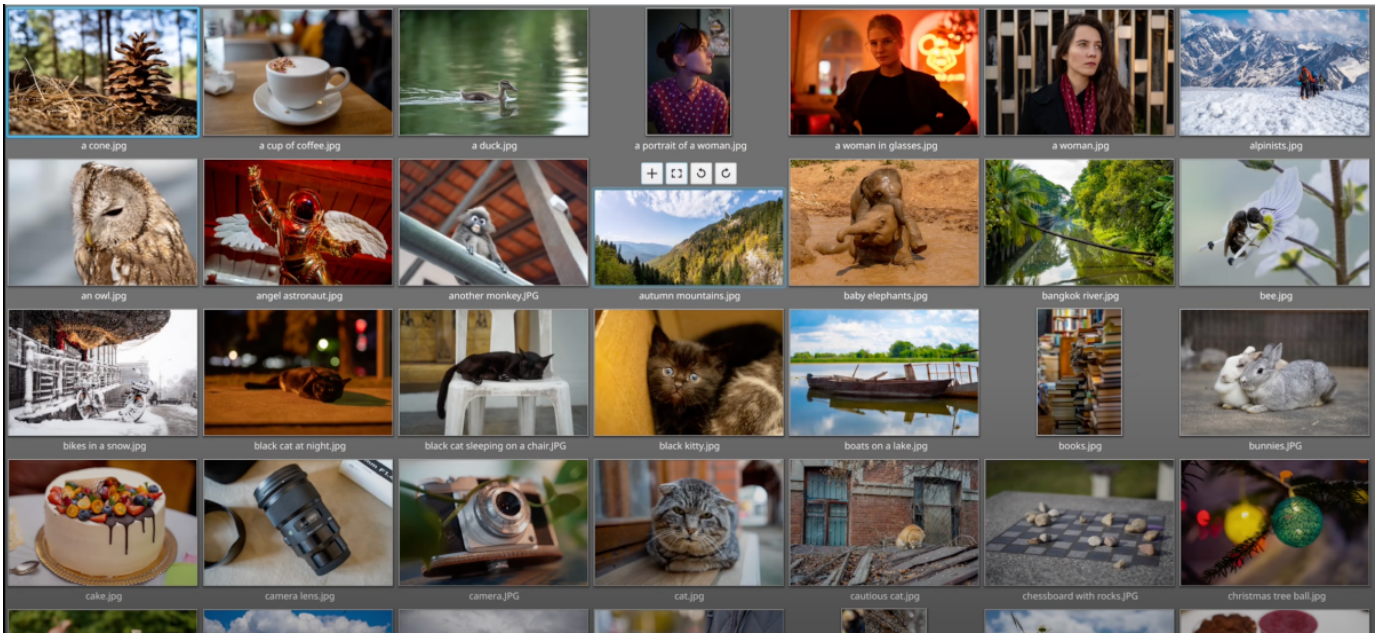


Figure 5. Demo set sample.

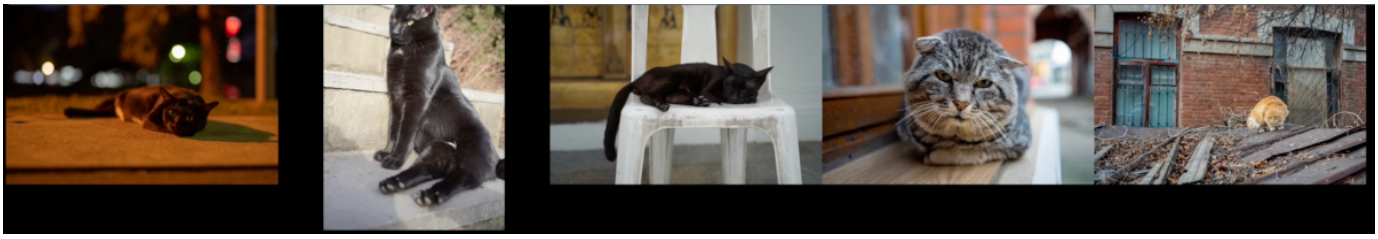


Figure 6. Search result for query “cat”.

Figure 7. Search result for query “leading lines”.

Figure 8. Top result for query “a kitten peeking from behind a corner”.

their personal photo libraries without setting up additional infrastructure.

CLIPPyX [28] is an AI-assisted desktop search tool that supports content-based, text, and visual similarity searches across local and external drives, storing embeddings in a vector database. While it provides a graphical user interface, `rclip` focuses on command-line simplicity and integration with existing terminal workflows. The GUI-based approach of CLIPPyX may be more accessible to non-technical users, but `rclip`’s command-line interface enables scripting, piping results to other tools, and integration into automated workflows—capabilities that are important for power users and developers.

CLIPSE (CLIP-based Image Search Engine) [29] is a minimalistic, research-oriented search engine built with Python. It

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

is designed for simplicity and easy extension, primarily aimed at researchers rather than end users. Unlike `rclip`, CLIPSE separates the indexing process (via a `build_index.py` script) from the search interface, supporting both CLI queries and a web interface. `rclip`, on the other hand, integrates these workflows into a single, zero-configuration executable aimed at everyday use without requiring users to manage separate indexing scripts or virtual environments.

Table VII summarizes the key differences between these tools.

`rclip`'s distinguishing characteristics are its focus on local, command-line usage; its support for incremental indexing that efficiently handles evolving image collections; its MPS GPU acceleration for Apple Silicon; its text-only model loading for fast repeated queries; and its support for query arithmetic and weighted queries. The design reflects a deliberate choice to prioritize simplicity and low friction: a user can install `rclip` via `pip`, navigate to any directory, and immediately search their images without any configuration, database setup, or server management.

VIII. Discussion

This section discusses the practical implications of the results presented in this paper, the limitations of the current approach, and lessons learned during the development and evolution of `rclip`.

A. Practical Considerations for Deployment

`rclip`'s architecture makes it particularly well-suited for several real-world deployment scenarios. For individual users managing personal photo collections, the tool provides an immediate and intuitive way to find specific images without requiring manual tagging or organization. The incremental indexing mechanism means that once a collection is initially indexed, subsequent searches incur minimal overhead even as new photos are added.

For creative professionals such as photographers, graphic designers, and video editors, the ability to search by abstract concepts ("leading lines," "warm tones," "minimalist composition") rather than concrete objects offers a workflow advantage that traditional keyword-based search cannot provide. The query arithmetic feature further enables retrieval of images that match complex aesthetic criteria, such as searching for images that are "similar to this reference photo but without people."

In organizational settings, `rclip` can serve as a lightweight alternative to commercial digital asset management (DAM) systems for teams that need semantic search capabilities without the overhead of deploying and maintaining a full-featured DAM platform. The command-line interface also facilitates automation: `rclip` can be integrated into shell scripts for batch processing, automated curation, or content moderation workflows.

Another practical application is in security and compliance contexts, where organizations need to identify images containing specific content across large file systems. Because `rclip` works with natural language queries and does not require pre-trained classifiers for each content category, it can be quickly deployed to search for new categories of content without retraining.

B. Limitations

Despite the improvements described in this paper, `rclip` has several limitations that users should be aware of.

Model limitations. The search quality is fundamentally bounded by the capabilities of the underlying CLIP model. `ViT-B/32` is one of the smaller CLIP models, and while it offers a good balance of speed and quality, larger models such as `ViT-L/14@336px` would provide better search results at the cost of significantly longer indexing times and higher memory consumption. As shown in Table II, the larger model is approximately 31× slower for indexing on CPU, making it impractical for large collections on consumer hardware. CLIP models also have known biases inherited from their training data, which can affect search results for certain query categories.

Scalability ceiling. While `rclip`'s brute-force similarity search is acceptable for collections up to a few million images, it becomes increasingly slow for much larger collections. The search time scales linearly with the number of indexed images because every image vector must be loaded and compared on each query. For collections exceeding 5–10 million images, integrating an approximate nearest neighbor index (as discussed in Section II-D) would be necessary.

Cache invalidation. `rclip` identifies images by their file path, modification time, and file size. This means that renaming or moving an image file causes it to be treated as a new image and re-indexed, even though its content has not changed. A content-based hashing mechanism could address this limitation but would add overhead to the indexing process.

Single-machine design. `rclip` is designed as a single-machine tool with a local SQLite database. It does not support distributed indexing, shared caches across machines, or concurrent access from multiple users. For multi-user or multi-machine deployments, a tool like `clip-retrieval` [27] with a client-server architecture would be more appropriate.

Image preprocessing. CLIP's image preprocessing pipeline resizes and crops input images to a fixed resolution (224×224 pixels for `ViT-B/32`). This means that fine-grained details in high-resolution images may be lost during preprocessing, potentially affecting search quality for queries that depend on small details within larger images.

IX. Conclusion and Future Work

This extended journal paper has presented the substantial evolution of `rclip`, a command-line image search tool based on CLIP, since its original conference publication. The key contributions of this extended work include:

- The migration from OpenAI's `clip` library to the open-source `open_clip` library, positioning `rclip` to adopt fully reproducible, publicly trained checkpoints in the future.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE VII. Comparison of CLIP-based image search tools.

Feature	rclip	clip-retrieval	CLIPPyX	CLIPSE
Interface	CLI	Web + CLI	GUI	CLI + Web
Deployment	Local	Server	Local	Local
ANN Index	No	Yes (FAISS)	Yes	No
MPS / Apple Silicon Support	Yes	No	Yes (MLX)	No
Incremental Indexing	Yes	No	Yes	No
Query Arithmetic	Yes	No	No	No
Multimodal Queries	Yes	Yes	Yes	No
RAW Image Support	Yes	No	No	No
Text-Only Model Loading	Yes	No	No	No
Persistent Cache	Yes (SQLite)	Yes (various)	Yes	Yes (JSON)

- The introduction of GPU acceleration via Apple's MPS backend, achieving a 2.15× speedup for indexing 1.28 million images on Apple M1 Max.
- A correction to the ImageNet-1k benchmark results: the original paper reported 31.17% top-1 accuracy due to a benchmark bug that silently dropped 488 of 1000 classes; the corrected accuracy is 58.78% with `ViT-B-32-quickgelu`, which uses the original OpenAI CLIP weights.
- Detailed descriptions of implementation improvements including batched indexing, text-only model loading, expanded image format support, improved file system traversal, and database schema evolution.
- A comparison with other CLIP-based image search tools, situating `rclip`'s design choices within the broader ecosystem.

Future work includes:

- Enriching `rclip`'s search capabilities by combining CLIP embeddings with existing image metadata such as GPS coordinates, capture date, and camera model.
- Exploring the integration of approximate nearest neighbor search (e.g., FAISS) for sub-second query performance on collections exceeding tens of millions of images.
- Exploring the use of larger CLIP models (e.g., `ViT-L/14@336px`) on systems with dedicated GPUs, trading increased computational cost for higher search quality.
- Investigating the impact of image corruption, compression artifacts, and unusual aspect ratios on search quality.
- Adding CUDA GPU support for systems with NVIDIA GPUs, which would further reduce indexing times on Linux and Windows workstations.
- Evaluating the text-only model loading optimization on larger datasets to quantify how the relative speed improvement scales as cached vector loading becomes the dominant cost.

In summary, this paper presents a practical and scalable method of searching images using natural language queries based on the CLIP model, details its architectural evolution over multiple release cycles, provides corrected benchmark results, and demonstrates that GPU acceleration on consumer hardware can significantly improve the tool's usability for large image collections. Taken together, these results confirm that large personal image collections can be searched by natural language on consumer hardware through a command-line interface, with no cloud infrastructure or specialized servers required.

References

[1] Y. Mikhalevich, "Using text queries to look up unlabeled images: A command-line search tool based on CLIP," in *Proceedings of CONTENT 2023, The Fifteenth International Conference on Creative Content Technologies*, IARIA, 2023, pp. 7–11, ISBN: 978-1-68558-048-3.

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, et al., "Learning transferable visual models from natural language supervision," arXiv, 2021. Accessed: Mar. 5, 2026. [Online]. Available: https://arxiv.org/abs/2103.00020

[3] M. Cherti, R. Beaumont, R. Wightman, M. Wortsman, G. Ilharco, C. Gordon, et al., "Reproducible scaling laws for contrastive language-image learning," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2023, pp. 2818–2829. DOI: 10.1109/cvpr52729.2023.00276

[4] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, et al., "Laion-5b: An open large-scale dataset for training next generation image-text models," 2022. arXiv: 2210.08402 [cs.CV]. Accessed: Mar. 5, 2026. [Online]. Available: https://arxiv.org/abs/2210.08402

[5] Apple Inc., *Accelerated PyTorch training on Mac*, 2026. Accessed: Mar. 5, 2026. [Online]. Available: https://developer.apple.com/metal/pytorch/

[6] N. Fiedler, M. Bestmann, and N. Hendrich, "Imagetagger: An open source online platform for collaborative image labeling," in *RoboCup 2018: Robot World Cup XXII 22*, Springer, 2019, pp. 162–169.

[7] J. Tesic, "Metadata practices for consumer photos," *IEEE MultiMedia*, vol. 12, no. 3, pp. 86–92, 2005.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

[8] J. Kim, S. Lee, J.-S. Won, and Y.-S. Moon, "Photo cube: An automatic management and search for photos using mobile smartphones," in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, IEEE, 2011, pp. 1228–1234.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[10] B. N. Lee, W.-Y. Chen, and E. Y. Chang, "A scalable service for photo annotation, sharing, and search," in *Proceedings of the 14th ACM international conference on Multimedia*, 2006, pp. 699–702.

[11] J. Zhang, A. Hallquist, E. Liang, and A. Zakhor, "Location-based image retrieval for urban environments," in *2011 18th IEEE International Conference on Image Processing*, IEEE, 2011, pp. 3677–3680.

[12] V. Balakrishnan and E. Lloyd-Yemoh, "Stemming and lemmatization: A comparison of retrieval performances," *Lecture Notes on Software Engineering*, vol. 2, no. 3, pp. 262–267, 2014.

[13] M. Günther, "Freddy: Fast word embeddings in database systems," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 1817–1819.

[14] T. Kenter and M. De Rijke, "Short text similarity with word embeddings," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, 2015, pp. 1411–1420.

[15] M. M. B. Ismail, *Image annotation and retrieval based on multi-modal feature clustering and similarity propagation*. University of Louisville, 2011.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. Accessed: Mar. 5, 2026. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[17] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, et al., "Scaling up visual and vision-language representation learning with noisy text supervision," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 18–24 Jul 2021, pp. 4904–4916. Accessed: Mar. 5, 2026. [Online]. Available: https://proceedings.mlr.press/v139/jia21b.html

[18] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *Proceedings of the 39th International Conference on Machine Learning*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., ser. Proceedings of Machine Learning Research, vol. 162, PMLR, 17–23 Jul 2022, pp. 12 888–12 900. Accessed: Mar. 5, 2026. [Online]. Available: https://proceedings.mlr.press/v162/li22n.html

[19] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," 2023. arXiv: 2303.15343 [cs.CV]. Accessed: Mar. 5, 2026. [Online]. Available: https://arxiv.org/abs/2303.15343

[20] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, pp. 535–547, 2017. Accessed: Mar. 5, 2026. [Online]. Available: https://api.semanticscholar.org/CorpusID:926364

[21] Y. Mikhalevich, *rclip*, version 2.0, 2025. Accessed: Mar. 5, 2026. [Online]. Available: https://github.com/yurijmikhalevich/rclip

[22] A. Piskun and contributors, *Pillow-heif: Python library for reading and writing HEIF images*, https://github.com/bigcat88/pillow_heif, 2026. Accessed: Mar. 5, 2026. [Online]. Available: https://github.com/bigcat88/pillow_heif

[23] A. Murray, H. van Kemenade, wiredfool, J. A. Clark, A. Karpinsky, O. Baranovič, et al., *Python-pillow/pillow: 12.1.1*, version 12.1.1, Feb. 2026. DOI: 10.5281/zenodo.18604291 Accessed: Mar. 5, 2026. [Online]. Available: https://doi.org/10.5281/zenodo.18604291

[24] R. D. Hipp, *SQLite*, version 3.31.1, 2020. Accessed: Mar. 5, 2026. [Online]. Available: https://www.sqlite.org/index.html

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 248–255.

[26] Y. Mikhalevich, *rclip benchmark script (ImageNet) – original (buggy) revision*, 2023. Accessed: Mar. 5, 2026. [Online]. Available: https://github.com/yurijmikhalevich/rclip/blob/f862d9a/benchmarks/benchmark-imagenet.py

[27] R. Beaumont, *Clip retrieval: Easily compute clip embeddings and build a clip retrieval system with them*, https://github.com/rom1504/clip-retrieval, 2022. Accessed: Mar. 5, 2026. [Online]. Available: https://github.com/rom1504/clip-retrieval

[28] U. Ahmed and contributors, *CLIPPyX: AI powered search tool offers content-based, text, and visual similarity system-wide search*, https://github.com/0ssamaak0/CLIPPyX, 2026. Accessed: Mar. 5, 2026. [Online]. Available: https://github.com/0ssamaak0/CLIPPyX

[29] S. Göring, *Clipse – a minimalistic clip-based image search engine for research*, 2025. arXiv: 2504.17643 [cs.CV]. Accessed: Mar. 5, 2026. [Online]. Available: https://arxiv.org/abs/2504.17643

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Automatic Semantic Image Tagging at Scale Using CLIP: Method, Evaluation, and Comparison with Modern Approaches

Yurij Mikhalevich
QA Wolf
Melbourne, Australia
email: yurij@mikhalevi.ch

***Abstract* - This paper presents an extended study of `rtag`, a practical, scalable, command-line tool for automatic semantic image tagging that leverages the Contrastive Language-Image Pre-Training model. Extending the original conference paper, this work contributes a survey of modern vision-language models and image tagging approaches, an in-depth analysis of prompt engineering strategies for zero-shot tagging, and a systematic comparison with existing tagging models and photo management applications. Evaluating three prompt templates on the ObjectNet dataset, the best template improves top-1 accuracy by 4.36 percentage points (from 25.37% to 29.73%) and top-5 accuracy by 5.94 percentage points (from 48.09% to 54.03%), substantially exceeding the improvement previously reported on ImageNet. We further discuss similarity threshold selection and its effect on tagging precision and recall, as well as standard image metadata formats and their role in integrating automated tagging into photo management workflows. The tool caches image features in an embedded database, enabling efficient re-tagging of previously indexed collections; the scalability analysis confirms approximately linear scaling of both indexing and tagging with the image count, with the cache providing a 25.3% speedup for previously indexed images. These results show that flexible, zero-shot image tagging with arbitrary user-defined vocabularies is practical at scale on consumer hardware, benefiting sectors that handle large volumes of visual content such as media and entertainment, digital archiving, healthcare, and personal photo management.**

Keywords* - *image tagging; zero-shot classification; CLIP; vision-language models; image metadata.

I. Introduction

This paper is an extended version of the conference paper [1], which introduced `rtag`, a command-line tool for automatic semantic image tagging based on CLIP. It substantially extends that work with a comprehensive survey of vision-language models, an analysis of prompt engineering and threshold selection, and a systematic comparison with existing tagging tools.

The unveiling of the Contrastive Language-Image Pre-Training (CLIP) model by OpenAI in 2021 has captured widespread interest across the domains of Natural Language Processing (NLP) and Computer Vision (CV). This innovative model is capable of understanding advanced image representations by analyzing a vast collection of 400 million image and text pairs sourced from the internet. CLIP uniquely identifies the most applicable text snippet for an image through natural language guidance without being directly trained for such tasks. Its zero-shot learning capabilities mirror those found in GPT-2 and GPT-3 [2]. The authors of CLIP have showcased its ability to match the performance of the original ResNet50 on the ImageNet dataset in a zero-shot setting without using any of the 1.28 million labeled examples. This breakthrough addresses some of the most daunting challenges in the field of computer vision.

Since CLIP's release, the vision-language model landscape has evolved rapidly. Models such as ALIGN [3], BLIP [4], SigLIP [5], and OpenCLIP [6] have pushed the boundaries of contrastive learning, while specialized image tagging models like Tag2Text [7] and RAM [8] have advanced image tagging from fixed-vocabulary approaches toward open-set recognition. This rapidly evolving landscape motivates a comprehensive review and comparison of approaches for automatic image tagging.

Given its capabilities, CLIP emerges as an ideal foundation for developing a semantic image tagging tool capable of operating in a zero-shot manner with any user-provided tags or images. This article explores this particular application of CLIP.

The rapid expansion of data, particularly image data, underscores the timeliness and significance of this research. Industry estimates suggest that billions of images are shared online daily across social media, messaging platforms, and cloud storage services. The surge in digital device usage and internet accessibility has led to an unprecedented increase in image production and sharing online, complicating the task of manually locating specific images. In a previous study, we introduced a method that employs natural language queries for image searches using the CLIP model, resulting in the development of a tool named `rclip` [9]. While `rclip` has broad applications, there are a lot of other use cases where it is important to be able to tag images to enable their manual cataloging and search using third-party software. This makes an efficient image tagging solution increasingly necessary and relevant.

The key contributions of this extended work beyond the conference paper are as follows:

- A comprehensive survey of modern vision-language models and image tagging approaches, including CLIP, ALIGN, BLIP, SigLIP, RAM, and Tag2Text.
- A detailed analysis of prompt engineering strategies for CLIP-based tagging and their effect on zero-shot classification accuracy.
- An in-depth discussion of similarity threshold selection and its impact on tagging quality, including the trade-off between precision and recall.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

- A systematic comparison with existing image tagging tools and services, highlighting `rtag`'s unique positioning in the ecosystem.
- An extended discussion of image metadata standards (IPTC, XMP, EXIF) and integration with photo management workflows.
- An architecture diagram and enriched qualitative analysis of tagging results.

The remainder of this paper is organized as follows. Section II reviews related works in deep learning, vision-language models, image tagging, and prompt engineering. Section III describes the proposed method for image tagging. Section IV presents the implementation details, including a system architecture diagram and metadata standards. Section V discusses the performance benchmarks. Section VI analyzes the tagging quality results. Section VII provides a systematic comparison with existing image tagging solutions. Section VIII discusses practical applications and limitations. Finally, Section IX concludes the paper and outlines future work.

II. Related Work

The advancement of NLP and CV has been significantly influenced by the development of models capable of understanding and generating insights from both text and images. The CLIP model by OpenAI marks a pivotal moment in this journey, leveraging a novel approach to learn visual concepts from natural language descriptions. This section reviews the literature that contextualizes CLIP within the broader field of image tagging and multimodal learning.

A. Foundations of Deep Learning for Image Understanding

One of the earliest attempts to bridge the gap between vision and language is the work by Socher et al. [10], who introduced a model for generating descriptions of image contents, paving the way for subsequent research in multimodal learning. Their model was among the first to use a neural network to combine visual and textual data, setting a foundation for future exploration in the field.

The concept of using neural networks for image classification was revolutionized by Krizhevsky et al. [11], with the introduction of AlexNet, which demonstrated the potential of deep learning in computer vision tasks. AlexNet achieved a top-5 error rate of 15.3% on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), reducing the previous best error by over 10 percentage points and establishing deep convolutional neural networks (CNNs) as the dominant paradigm for image classification.

Following AlexNet, two architectures significantly advanced the field. VGGNet [12] demonstrated that deeper networks with small (3×3) convolutional filters could achieve strong performance, reaching 7.3% top-5 error on ILSVRC 2014. GoogLeNet (Inception) [13] introduced the inception module, which applies multiple filter sizes in parallel, achieving 6.7% top-5 error while maintaining computational efficiency.

The development of the ResNet model by He et al. [14] represented another major step forward, introducing the concept of residual learning to enable the training of networks with over 100 layers. ResNet won ILSVRC 2015 with a 3.57% top-5 error rate, surpassing human-level performance. The advancements in image classification models like ResNet laid the groundwork for more sophisticated image understanding and tagging capabilities.

In parallel to improvements in image classification, research in NLP was making strides with the development of models like BERT by Devlin et al. [15], which introduced a new method for pre-training language representations using bidirectional transformers. BERT's success in understanding context and nuance in text provided inspiration for exploring similar pre-training approaches in multimodal models.

The intersection of NLP and CV began to take a more defined shape with the introduction of models like ViLBERT by Lu et al. [16], which employed separate pathways for processing visual and textual information before integrating them through co-attentional transformer layers, demonstrating improved performance on tasks requiring both visual and textual understanding.

The Transformer architecture [17], originally developed for machine translation, has proven to be a versatile backbone for multimodal learning. Its self-attention mechanism enables models to focus on relevant parts of the input when generating tags or descriptions, facilitating more nuanced interpretation of visual and textual data.

B. Vision-Language Models and Contrastive Learning

The development of CLIP by Radford et al. [2] stands as a significant milestone in vision-language modeling. CLIP employs a dual-encoder architecture: a Vision Transformer (ViT) [18] encodes images, while a text transformer encodes natural language descriptions. The model is trained using contrastive learning on 400 million image-text pairs collected from the internet (the WebImageText dataset). During training, CLIP learns to maximize the cosine similarity between matching image-text pairs while minimizing it for non-matching pairs. This approach enables remarkable zero-shot transfer: at inference time, the model can classify images into arbitrary categories by computing similarity between image features and text features derived from category descriptions.

CLIP's ViT backbone comes in several variants with different capacity and performance trade-offs. ViT-B/32 achieves 63.2% top-1 accuracy on ImageNet in a zero-shot setting, while ViT-L/14 reaches 75.3%. The choice of backbone directly affects both accuracy and computational cost.

ALIGN [3] scaled contrastive learning to 1.8 billion noisy image-text pairs collected from the web without careful curation. Despite the noise in training data, ALIGN achieved competitive zero-shot performance, demonstrating that scale alone can compensate for imperfect data quality, even with only minimal frequency-based filtering and without careful curation.

BLIP [4] introduced a bootstrapping approach to language-image pre-training, combining a multimodal encoder-decoder architecture with a captioning model that generates synthetic

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

captions and a filter that removes noisy ones. This self-refinement of training data yields improved performance on both understanding and generation tasks.

SigLIP [5] proposed replacing CLIP's softmax-based contrastive loss with a sigmoid loss function that operates on individual image-text pairs rather than requiring the full batch for normalization. This modification simplifies the training procedure, enables more efficient distributed training, and achieves competitive or superior performance compared to the original CLIP formulation. SigLIP 2 [19] extends this approach with multilingual support, improved zero-shot classification, and dense feature extraction capabilities.

OpenCLIP [6] provides an open-source reproduction of CLIP's training procedure, trained on publicly available datasets such as LAION-5B [20]. OpenCLIP established reproducible scaling laws for contrastive language-image learning, showing that both model size and dataset size contribute to improved zero-shot performance. The availability of open-source training code and public checkpoints has made OpenCLIP a foundation for many downstream applications. OpenCLIP's ViT-H/14 variant trained on LAION-2B achieves approximately 78.0% zero-shot top-1 accuracy on ImageNet, surpassing the original OpenAI CLIP models.

C. Image Tagging Models and Approaches

In the domain of image tagging, prior works have explored various approaches for automating the process. Traditional multi-label classification approaches use supervised learning on fixed label sets. Wang et al. [21] introduced a CNN-RNN framework that enhances the accuracy of tagging by leveraging the correlations between labels. Their method predicts multiple relevant tags for a single image by modeling dependencies among tags, producing more accurate and contextually relevant annotations. However, such supervised approaches are limited to the categories seen during training.

The Recognize Anything Model (RAM) [8] represents a significant advancement in image tagging. RAM uses a large-scale annotation pipeline to automatically generate training data from image-text pairs, supporting over 6,400 tag categories. RAM employs a vision transformer backbone with a tag recognition decoder and achieves strong performance, outperforming CLIP and BLIP on standard tagging benchmarks. While RAM is primarily evaluated on its predefined tag set, its use of a CLIP-based text encoder for label queries gives it open-set capability, enabling recognition of tags not seen during training. However, RAM requires GPU resources for inference, with model sizes of approximately 5.6 GB.

RAM++ [22] builds on RAM's open-set foundation by incorporating multi-grained text supervision to significantly improve recognition of diverse and rare open-set categories. This is achieved through a retrieval-augmented approach that combines RAM's semantic tag embeddings with enhanced text representations. While more flexible than the original RAM, RAM++ still relies on a specialized architecture and substantial computational resources.

Tag2Text [7] combines image tagging with image captioning in a unified framework. The model supports 3,429 tag categories and can simultaneously generate descriptive captions and structured tags. Tag2Text demonstrates that the tagging and captioning tasks are complementary: tags provide grounding for more accurate captions, while the captioning objective improves tag recognition.

Weyand et al. [23] introduced PlaNet, a method for geolocating images using a deep neural network. PlaNet demonstrates the potential of using image content to infer metadata beyond simple object recognition, assigning geographic tags based on visual features.

In contrast to these approaches, `rtag` occupies a distinct niche. While RAM offers open-set tagging capability and Tag2Text operates with a fixed vocabulary, both use specialized architectures. In contrast, `rtag` leverages CLIP's general-purpose zero-shot capability to support completely arbitrary, user-defined tags without any specialized training. This flexibility comes at the cost of lower accuracy on fixed-vocabulary benchmarks, but enables use cases that specialized models cannot address, such as tagging with domain-specific or personal categories.

D. Prompt Engineering for Vision-Language Models

The effectiveness of CLIP-based zero-shot classification is sensitive to the textual prompts used to represent categories. Radford et al. [2] demonstrated that using a simple template like "a photo of a {label}" provides a 1.3 percentage point improvement on ImageNet over using bare class names. Furthermore, using an ensemble of 80 prompt templates (e.g., "a photo of a {label}", "a good photo of a {label}", "a rendering of a {label}") yields an additional 3.5 percentage point improvement over the single template, for a total gain of approximately 4.8 percentage points over bare class names. Even simple template modifications have measurable effects: template variations such as "a photo of a {class}" vs. "a photo of {class}" can produce approximately 5 percentage points difference on fine-grained datasets like Caltech-101.

This sensitivity has motivated research on automatic prompt optimization. CoOp [24] replaces hand-crafted prompt templates with learnable continuous vectors that are optimized on a few labeled examples, achieving significant improvements over hand-crafted prompts on several benchmarks. CoCoOp [25] extends this approach with conditional context optimization, generating input-dependent prompts that generalize better to unseen classes.

An alternative direction uses large language models (LLMs) to generate descriptive prompts automatically. Menon and Vondrick [26] proposed using GPT-3 to generate category descriptions (e.g., for "cat": "has whiskers", "has pointed ears", "is a small domesticated carnivorous mammal"), achieving substantial improvements by leveraging the LLM's world knowledge to create more discriminative text representations. This approach can yield up to 5 percentage points improvement on challenging benchmarks without requiring any labeled images.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Prompt engineering is directly relevant to `rtag`'s design. As we show in Section VI, evaluating three prompt templates on ObjectNet reveals a 4.36 percentage point improvement in top-1 accuracy with the "photo of a [tag]" template, substantially exceeding the 1.3 percentage point gain reported by Radford et al. on ImageNet. This suggests that prompt optimization has an outsized effect on real-world photograph datasets and that further optimization could yield additional gains.

III. Method

This section formalizes the approach behind `rtag`, covering the image-tag similarity computation, the threshold-based tag-selection rule, the role of prompt templates, and the optimizations that make the method scalable.

A. CLIP-Based Similarity Computation

CLIP's dual-encoder architecture produces aligned embeddings for images and text in a shared n-dimensional vector space (where n depends on the CLIP model variant; $n = 512$ for ViT-B/32). Given an image i and a text label (tag) t, the CLIP image encoder f_{img} and text encoder f_{text} produce L2-normalized feature vectors $\mathbf{v}_i = f_{\text{img}}(i)$ and $\mathbf{v}_t = f_{\text{text}}(t)$, respectively.

The similarity between an image and a tag is computed via the dot product (Equation (1)) of their normalized feature vectors:

$$\mathbf{v}_i \cdot \mathbf{v}_t = \sum_{k=1}^{n} v_{i,k} \cdot v_{t,k} \tag{1}$$

Since CLIP produces L2-normalized vectors ($\|\mathbf{v}_i\| = \|\mathbf{v}_t\| = 1$), the dot product is equivalent to cosine similarity (Equation (2)):

$$\cos(\theta) = \frac{\mathbf{v}_i \cdot \mathbf{v}_t}{\|\mathbf{v}_i\| \cdot \|\mathbf{v}_t\|} = \mathbf{v}_i \cdot \mathbf{v}_t \tag{2}$$

For a collection of m images and p tags, the full similarity matrix can be computed efficiently as a single matrix multiplication:

$$\mathbf{S} = \mathbf{I} \cdot \mathbf{T}^\top \tag{3}$$

where $\mathbf{I} \in \mathbb{R}^{m \times n}$ is the matrix of image feature vectors and $\mathbf{T} \in \mathbb{R}^{p \times n}$ is the matrix of tag feature vectors. The resulting matrix $\mathbf{S} \in \mathbb{R}^{m \times p}$ contains the cosine similarity score between every image-tag pair.

B. Tag Selection and Threshold Semantics

Given the similarity matrix $\mathbf{S}$, a tag t_j is assigned to image i if and only if their similarity exceeds a configurable threshold τ:

$$\text{assign}(i, t_j) = \begin{cases} 1 & \text{if } S_{ij} > \tau \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The threshold τ controls the trade-off between precision and recall. A higher threshold produces fewer tags per image with higher confidence (higher precision, lower recall), while a lower threshold produces more tags, including less confident ones (lower precision, higher recall).

The default threshold of $\tau = 0.25$ was chosen empirically. In practice, raw CLIP cosine similarity scores for relevant image-tag pairs typically fall in the range of 0.15–0.40, with highly relevant matches reaching 0.30–0.35. A threshold of 0.25 provides a reasonable balance: it captures tags with meaningful semantic relevance while filtering out noise. Unlike classification tasks where the model must select a single best class, image tagging is inherently a multi-label problem where multiple tags can apply simultaneously, making the threshold a critical parameter.

C. Prompt Template Selection

In general, a tag text t can be wrapped in a prompt template T before encoding:

$$\mathbf{v}_t = f_{\text{text}}(T(t)) \tag{5}$$

Template choice matters because CLIP was trained on natural language captions, not isolated labels. A bare label like "cat" lacks the syntactic context of a natural sentence, potentially leading to suboptimal embeddings. Templates such as "photo of a [tag]" bridge this gap by framing the label as a caption-like phrase. However, `rtag` passes tags directly to CLIP's text encoder without wrapping them in a template. This design choice preserves maximum flexibility: tags are not restricted to object labels but can be arbitrary descriptive text such as "warm lighting," "taken outdoors," or "birthday party." The benchmark in Section VI evaluates the effect of different prompt templates on classification accuracy, showing that templates like "photo of a [tag]" substantially improve results when tags are simple object labels.

Template ensembling offers a way to further improve embeddings. Given N templates $T_1, \ldots, T_N$, the ensembled tag embedding is:

$$\mathbf{v}_t^{\text{ens}} = \text{normalize}\left(\frac{1}{N}\sum_{k=1}^{N} f_{\text{text}}(T_k(t))\right) \tag{6}$$

Radford et al. [2] report a 3.5 percentage point improvement on ImageNet using an 80-template ensemble over a single prompt template. User-configurable template strings and ensembling are promising directions for future work (Section IX).

D. Scalability Optimizations

To allow tagging any kind of image with any set of tags, the approach suggests allowing the input of a custom list of tags. While this approach works reasonably fast on a few images and tags, it may not be scalable when dealing with a large number of images or tags, particularly without access to GPUs. To address this scalability issue, we propose two optimizations:

- **Pre-compute tag feature vectors**: Tag embeddings are computed once and reused across all images. Since the

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

number of tags p is typically much smaller than the number of images m, the text encoder inference cost $O(p \cdot c_{\text{text}})$ is negligible compared to image encoding.

- **Utilize `rclip`'s cache layer**: The `rclip` [27] cache stores computed image feature vectors in an SQLite [28] database after initial processing. By reusing these cached vectors, `rtag` avoids the expensive image encoding step entirely for previously indexed images.

Without caching, the total computational cost is $O(m \cdot c_{\text{img}} + p \cdot c_{\text{text}} + m \cdot p \cdot d)$, where c_{img} is the cost of encoding one image, c_{text} is the cost of encoding one text label, and d is the embedding dimension. With caching, the cost reduces to $O(p \cdot c_{\text{text}} + m \cdot p \cdot d)$, since $c_{\text{img}} \gg p \cdot d$ (CLIP image inference dominates). The similarity computation $O(m \cdot p \cdot d)$ is a simple matrix multiplication and is fast even for large datasets.

The `rclip` cache also handles incremental updates: when new images are added to a directory, only the new images require CLIP inference. This is particularly useful when the user needs to retag images with a different set of tags or when the images are already indexed by `rclip`.

Once the tags are selected, they are written to the image metadata, allowing other software to utilize the tags. For example, the user can search for images by tag using the operating system's file manager or a third-party image management application.

IV. Implementation

This section presents how the method is realized in practice, describing the system architecture, the core processing pipeline, the available operating modes, and the image metadata standards used to store the resulting tags.

A. System Architecture

The method described in Section III is implemented in the Python utility called `rtag` [29]. `rtag` provides an easy-to-use Command-Line Interface (CLI) that allows users to tag images within any directory on a computer where `rtag` is installed. Tagging images within nested directories of a complex directory subtree is also supported. To use it, the user opens the terminal, navigates to the target directory, types `rtag`, and hits "Enter."

Figure 1 shows the overall system pipeline. The architecture consists of two input paths—tag processing and image processing—that converge at the similarity computation stage. Tags are encoded once by CLIP's text encoder, while image feature vectors are retrieved from the `rclip` SQLite cache (or computed on the fly for unindexed images). The dot-product similarity between all image-tag pairs is computed as a matrix multiplication. Tags exceeding the threshold are passed to the IPTC metadata writer, which embeds them into the image files.

The solution uses the `rclip` library [27] to compute the feature vectors. `rclip` uses the `ViT-B/32` version of OpenAI's CLIP model [2]. The tool writes the computed tags to the image IPTC metadata [30] using the IPTCInfo3 Python library [31].

B. Core Processing Pipeline

A simplified version of the image processing loop is shown in Figure 2. The pipeline operates as follows: (1) tag feature vectors are computed once by passing all tags through CLIP's text encoder; (2) image feature vectors are retrieved from the `rclip` SQLite database; (3) for each image, the dot product between its feature vector and all tag vectors is computed using a single matrix multiplication (`image_features @ tag_features.T`); (4) tags with similarity above the threshold are selected; and (5) selected tags are written to the image's IPTC metadata.

Images are processed sequentially in the current implementation. Each image's feature vector is a compact 512-dimensional float32 array (2 KB), making the per-image memory footprint negligible. The dominant memory cost is the tag feature matrix, which for the default ImageNet-1k label set (1,000 tags) is approximately 2 MB.

C. Operating Modes and Configuration

Figure 3 shows how tags are loaded. By default, `rtag` uses the list of the ImageNet-1k [32] labels as tags. The ImageNet-1k label set contains 1,000 categories spanning a broad range of everyday objects, animals, vehicles, and natural scenes, making it a reasonable default for general-purpose tagging. The user can provide their own list of tags by specifying the `--tags-filepath` argument, e.g., `rtag --tags-filepath ./tags.txt`, where the file contains a newline-separated list of tags.

`rtag` supports two modes of operation:

- **Append mode** (`--mode append`): Computed tags are merged with existing tags in the image metadata. Duplicate tags are automatically removed. This mode is useful for incremental tagging, e.g., first tagging with general categories and then adding domain-specific tags.
- **Overwrite mode** (`--mode overwrite`): Existing tags are replaced entirely with the computed tags. This mode is appropriate when re-tagging with a completely new tag set.

The `--threshold` argument allows the user to control the number of tags written per image. A higher threshold produces fewer, more confident tags; a lower threshold produces more tags with greater recall. The default value of 0.25 produces optimal results in practice.

The `--dry-run` argument shows what tags would be written without actually modifying any files. This is useful for experimenting with different tag sets and threshold values before committing to changes.

The `rtag` source code is published on GitHub under the MIT license [29].

D. Image Metadata Standards

`rtag` writes tags using the IPTC IIM (Information Interchange Model) standard [30] via the IPTCInfo3 library [31]. The IPTC Photo Metadata Standard defines a "keywords" field specifically designed for descriptive tags, making it the natural choice for image tagging applications.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

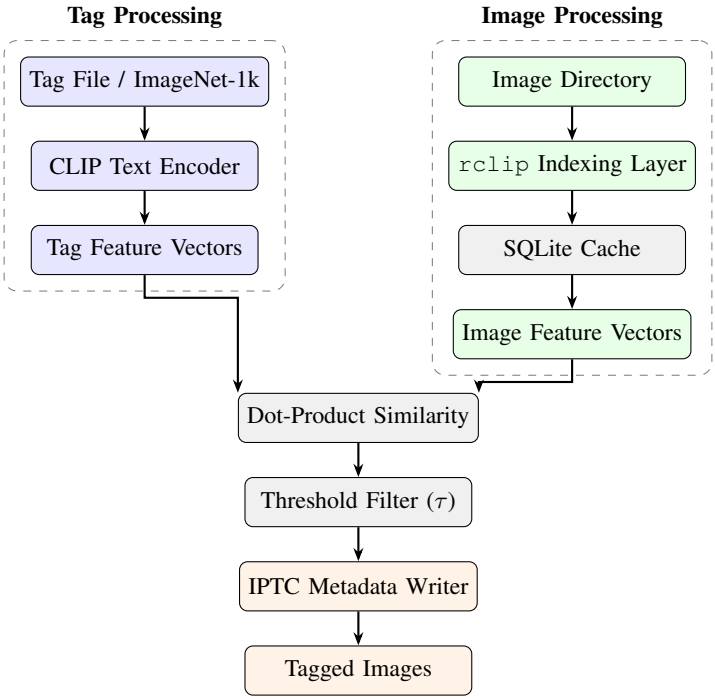


Figure 1. The `rtag` system pipeline. The *tag processing* path (left) encodes user-provided tags (or the default ImageNet-1k label set) using CLIP's text encoder. The *image processing* path (right) retrieves image feature vectors from the `rclip` SQLite cache, computing them on the fly for unindexed images. The similarity computation stage computes the dot product between all tag and image vectors, applies the threshold filter, and writes matching tags to IPTC metadata.

Several metadata standards exist for embedding information in image files:

- **EXIF** (Exchangeable Image File Format): Primarily stores camera settings and capture parameters (exposure, focal length, GPS coordinates). EXIF has limited support for keywords and is not well-suited for tagging workflows.
- **IPTC IIM**: A well-established standard for editorial metadata, including the "keywords" field used by `rtag`. IPTC is broadly supported by photo management software, including Adobe Lightroom, digiKam, macOS Finder, and Windows Explorer.
- **XMP** (Extensible Metadata Platform) [33]: Adobe's more general metadata framework, based on RDF/XML. XMP can represent all IPTC fields and more. XMP metadata can be embedded in the image file or stored in a separate sidecar file (typically with a `.xmp` extension).

IPTC was chosen for `rtag` because of its broad compatibility with existing photo management software and operating system file managers. A key limitation of the current approach is that IPTC metadata is embedded directly in the image file, which modifies the original. For workflows where non-destructive editing is important—particularly with RAW image formats—XMP sidecar files would be preferable, as they store metadata alongside the image without altering it. Support for XMP sidecar output is planned for a future release (Section IX).

V. Performance Analysis

This section describes the benchmarking methodology and reports the resulting processing times, analyzing how `rtag` scales with the image count and how much the feature cache contributes.

A. Benchmarking Methodology

Performance was benchmarked on the ObjectNet [34] dataset, which contains 50,273 images of 313 object categories. ObjectNet was chosen because its images feature controlled variations in background, rotation, and viewpoint, making it more challenging and representative of real-world conditions than standard benchmarks where objects are typically centered and well-lit.

All benchmarks were run on an Apple M1 Max CPU (10-core, 8 performance + 2 efficiency cores, 64 GB unified memory). Before running the benchmarks, the dataset was resized to 224 pixels by smaller dimension and converted to JPG using ImageMagick [35]. The 224-pixel dimension matches CLIP's expected input resolution. Processing times are wall-clock measurements from a single run.

B. Results and Scalability Analysis

Table I shows `rtag`'s performance when tagging previously indexed and unindexed images.

The theoretical scaling factor between the full dataset and the subset is $50{,}273/965 = 52.1\times$. The observed scaling factors are $381.25\text{s}/12.04\text{s} = 31.66\times$ (unindexed) and $284.79\text{s}/9.14\text{s} = 31.15\times$ (indexed). The sub-linear observed

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

```
tag_features = (
  model.compute_text_features(tags)
)

for image in tqdm(
  rclipDB.get_image_vectors_by_dir_path(
      current_directory),
  unit='images',
):
  image_path = image['filepath']
  image_features = np.frombuffer(
    image['vector'],
    np.float32,
  )

  similarities = image_features @
      tag_features.T

  new_tags = []
  for tag, similarity in zip(tags,
      similarities):
    if similarity > args.threshold:
      new_tags.append(tag)

  if not new_tags:
    continue

  image_metadata = IPTCInfo(
    image_path,
    force=True,
  )

  if args.mode == 'append':
    existing_tags = image_metadata['
        keywords']
    image_metadata['keywords'] = [*set(
        existing_tags + new_tags)]
  elif args.mode == 'overwrite':
    image_metadata['keywords'] = new_tags
  else:
    raise ValueError(f'Invalid mode: {
        args.mode}')

  image_metadata.save()
```

Figure 2. Image processing loop.

```
def load_tags_from_file(path: str):
  with open(path, 'r') as f:
    return f.read().splitlines()

tags_filepath = args.tags_filepath or
    get_imagenet_tags_filepath()
tags = load_tags_from_file(tags_filepath)
```

Figure 3. Tags loading.

scaling (31× vs. the theoretical 52×) is expected: fixed overhead costs such as model loading, SQLite database initialization, and Python interpreter startup are amortized over more images in the larger dataset, improving the effective per-image throughput.

The `rclip` cache provides a substantial speedup. For the full dataset, indexed processing (4m44.790s) is 25.3% faster than unindexed processing (6m21.250s). For the subset, the speedup is 24.1% (9.14s vs. 12.04s). This confirms that the caching mechanism provides a consistent benefit regardless of dataset size.

The effective throughput for the full dataset is approximately 177 images per second for indexed images (50,273/284.79) and 132 images per second for unindexed images (50,273/381.25). At these rates, a personal photo library of 50,000 images can be fully tagged in under 5 minutes when previously indexed by `rclip`.

The real-life `rtag` application possibilities go far beyond results demonstrated in these benchmarks because `rtag` can be used with any tags and images and not just the ones present in the dataset.

VI. Tagging Quality Analysis

This section explains the evaluation methodology, presents the quantitative results, examines the effect of prompt engineering, and offers a qualitative analysis of representative tagged images.

A. Evaluation Methodology

Tagging quality was evaluated using top-k accuracy on the ObjectNet [34] dataset. Top-k accuracy measures the fraction of images for which the correct label appears among the k highest-scoring tags. This metric is appropriate for evaluating zero-shot tagging systems because the model must rank the

TABLE I. Tagging performance on Apple M1 Max CPU.

Dataset	# of images	Indexed	Processing time
`ObjectNet` dataset	50,273	No	6m21.250s
`ObjectNet` dataset	50,273	Yes	4m44.790s
`ObjectNet` subset	965	No	0m12.040s
`ObjectNet` subset	965	Yes	0m09.140s

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

correct tag among the entire candidate vocabulary (313 ObjectNet category labels in our evaluation), rather than selecting from a small set of choices.

ObjectNet's ground truth provides a single correct label per image, drawn from 313 categories, of which 113 overlap with the ImageNet-1k label set. The evaluation assigns each image the tags with the highest similarity scores and checks whether the ground truth label appears in the top k. This simulates a realistic use case where the user provides a broad set of candidate tags and relies on the system to identify the most relevant ones.

ObjectNet was specifically designed with controlled biases: objects appear against random backgrounds, in unusual rotations, and from atypical viewpoints. This makes it substantially more challenging than standard benchmarks. Published results show that models typically experience a 40–45% performance drop on ObjectNet compared to their ImageNet accuracy, making it a stringent test of real-world generalization.

Alternative evaluation metrics include mean average precision (mAP) and per-category F1 score, which are more suited to multi-label evaluation. However, ObjectNet provides single-label ground truth, making top-k accuracy the natural choice for this dataset.

B. Results Analysis

Table II presents the tagging quality results.

The best template, `photo of a [tag]`, achieves 29.73% top-1 accuracy and should be interpreted in context. CLIP ViT-B/32 achieves approximately 44.3% top-1 accuracy on ObjectNet in standard zero-shot classification [6], where the model selects from ObjectNet's 113 overlapping categories [34] and is evaluated only on images from those categories. `rtag`'s evaluation uses all 313 ObjectNet category labels as the candidate tag set and evaluates on all 50,273 images from all 313 categories, making the task harder in two ways: the model must rank the correct tag among approximately 2.8 times more candidates, and it must classify images from the 200 additional categories that have no overlap with ImageNet.

The top-5 accuracy of 54.03% means that the correct tag appears among the five highest-scoring tags for over half of all images. This is practically useful: even when the model does not rank the correct tag first, it often ranks it highly, enabling effective tag-based browsing and search.

The gap between top-1 and top-5 accuracy (approximately 24 percentage points) indicates that CLIP frequently assigns high similarity scores to semantically related tags. This is a characteristic of contrastive text encoders: training on image–caption pairs causes embeddings for synonyms and taxonomically related terms (e.g., "speaker" and "loudspeaker") to be mathematically adjacent in the shared embedding space. For example, a "speaker" image may receive high scores for both "speaker" and "loudspeaker," and an image of a camera may score highly for both "camera" and "projector." In a tagging context, this is often desirable: semantically related tags improve discoverability.

TABLE II. `rtag` tagging quality on ObjectNet.

Template	Top-1 accuracy	Top-5 accuracy
`[tag]`	25.37%	48.09%
`photo of [tag]`	26.94%	50.20%
`photo of a [tag]`	29.73%	54.03%

C. Prompt Engineering Effect

Prompt template choice has a substantial effect on tagging accuracy. The three templates evaluated show a progressive improvement: the bare `[tag]` template achieves 25.37% top-1 accuracy, `photo of [tag]` improves this by 1.57 percentage points to 26.94%, and `photo of a [tag]` adds another 2.79 percentage points, reaching 29.73%. The total improvement from bare tags to the best template is 4.36 percentage points in top-1 accuracy (25.37% to 29.73%) and 5.94 percentage points in top-5 accuracy (48.09% to 54.03%). Notably, the addition of the article "a" alone accounts for more than half of the total gain.

Radford et al. [2] reported a 1.3 percentage point improvement on ImageNet with the "a photo of a {label}" template. `rtag`'s observed improvement on ObjectNet is more than three times larger (4.36pp vs. 1.3pp). This amplified effect may reflect the nature of the dataset: ObjectNet consists entirely of real-world photographs, making the "photo of" framing especially appropriate—CLIP was trained on internet image-caption pairs where captions frequently describe photos.

The progressive improvement across all three templates suggests that prompt framing has an outsized effect on datasets of real-world photographs. While template sensitivity is well documented on standard classification benchmarks (e.g., more than 5% in accuracy on Caltech-101 [24]), the improvement on ObjectNet is notable for its magnitude on a highly diverse, real-world dataset with 313 categories.

CLIP's 80-template ensemble yields a 3.5 percentage point improvement on ImageNet over a single prompt template. Notably, `rtag` already exceeds this ensemble gain with a single well-chosen template on ObjectNet (4.36pp), suggesting that template selection may be more impactful than ensembling for datasets well-matched to a specific prompt. Implementing template ensembling in `rtag` could provide further gains at the cost of increased text encoding time (proportional to the number of templates).

Recent work on LLM-generated prompts [26] has shown that using large language models to create category descriptions can yield substantial improvements on challenging benchmarks. Learnable prompt optimization approaches such as CoOp [24] and CoCoOp [25] offer another avenue, though they require labeled examples for the target domain. These directions represent significant potential for improving `rtag`'s tagging quality in future iterations.

D. Qualitative Analysis

Figures 4, 5, and 6 (at the end of the paper) show example images from the ObjectNet dataset tagged by `rtag` using the default ImageNet-1k tag set and a threshold of 0.25.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

The speaker image (Figure 4) is assigned a broad set of tags. True positives include the combined label "loudspeaker, speaker, speaker unit," a direct match for the object. Related but less precise tags such as "radio" and "cassette player" reflect CLIP's semantic associations: speakers are commonly found alongside audio playback equipment. These semantically related tags, while not exact matches for the object, can be useful for browsing—a user searching for audio equipment would find this image through multiple related queries.

The soap image (Figure 5) receives a wide variety of tags. Category confusion between "soap dispenser," "bathtub," and "dishwasher" illustrates CLIP's tendency to associate objects with the contexts in which they commonly appear. Soap is frequently depicted near bathtubs and cleaning contexts, leading CLIP to assign high similarity to these related concepts. This contextual association is a feature of CLIP's training on captioned internet images, where co-occurring objects are frequently mentioned together.

The camera image (Figure 6) receives a cluster of tags centered on optical devices. The presence of "binoculars" and "opera glasses" alongside "reflex camera" and "Polaroid camera" shows CLIP's semantic clustering of optical devices. These objects share visual features (lenses, eyepieces, compact form factors) that lead to similar feature vectors in CLIP's embedding space.

A general observation across these examples is that CLIP tends to assign semantically related tags even when they are not exact matches. In a tagging context, this behavior is often beneficial: users browsing tagged images by related concepts will find relevant results even when the exact tag is not present. The threshold parameter allows users to control the specificity of tags—raising the threshold would retain only the highest-confidence tags, reducing both true and false positives. Despite occasional false positives, the added tags allow users to surface the images they are looking for, the primary goal in a browsing and retrieval context.

VII. Comparison with Existing Image Tagging Solutions

To contextualize `rtag`'s capabilities, we compare it with specialized image tagging models and photo management software that offer automatic tagging features.

A. Specialized Image Tagging Models

RAM (Recognize Anything Model) [8] supports over 6,400 tag categories and uses an automatic annotation pipeline trained on large-scale image-text pairs. RAM employs a Swin Transformer backbone with a specialized tag recognition head, achieving state-of-the-art performance on standard tagging benchmarks. While RAM's open-set architecture allows it to recognize categories beyond its predefined tags, it requires GPU resources for efficient inference and has a model size of approximately 5.6 GB.

RAM++ [22] improves upon RAM's open-set capability by incorporating multi-grained text supervision alongside RAM's semantic tag embeddings. This significantly enhances recognition of diverse and rare categories, though the approach still relies on RAM's specialized architecture.

Tag2Text [7] integrates tagging (3,429 categories) with image captioning. While more versatile than pure tagging models, Tag2Text requires similar computational resources and operates with a predefined category set for its tagging component.

`rtag` takes a fundamentally different approach. Rather than training a specialized tagging model, it leverages CLIP's general-purpose zero-shot capability. This means any text string can serve as a tag—the user is not limited to a predefined vocabulary. While this flexibility comes at the cost of lower accuracy compared to specialized models on their benchmark categories, it enables use cases that fixed-vocabulary models cannot support, such as tagging with highly domain-specific or personal categories.

B. Photo Management Software with Automatic Tagging

Several photo management applications offer built-in automatic tagging:

Google Photos uses proprietary cloud-based ML models trained on billions of photos, offering excellent accuracy for common objects, scenes, and faces. However, it requires uploading all photos to Google's cloud (raising privacy concerns), does not support custom tags, and the ML-generated searchable tags cannot be exported as standard IPTC or XMP metadata (though some metadata such as descriptions and locations can be exported via Google Takeout as JSON sidecar files).

Apple Photos performs on-device ML-based classification for objects, scenes, and faces. While privacy-preserving, it is locked to the Apple ecosystem and provides no user-facing CLI. The macOS version supports custom user-defined keywords via its Keyword Manager, which can be embedded as standard IPTC metadata upon export; the iOS version is more limited, offering only captions. Apple also provides PhotoKit, a developer API for interacting with the Photos library on iOS and macOS, though there is no command-line interface for end users.

digiKam [36] is an open-source photo management application with deep learning-based face recognition and image classification. It reads and writes IPTC and XMP metadata, making it compatible with `rtag`'s output. However, digiKam is GUI-only and has limited zero-shot tagging capability.

Immich [37] is an open-source, self-hosted alternative to Google Photos with ML-based tagging (using CLIP models) and face recognition. It requires server deployment and does not operate as a standalone command-line tagging tool.

PhotoPrism [38] is an open-source photo management application using TensorFlow-based classification. It offers basic automatic tagging but with limited accuracy compared to CLIP-based approaches and requires server deployment.

C. Feature Comparison

Table III summarizes the key differences between `rtag` and existing solutions across several dimensions.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE III. Comparison of image tagging solutions.

Feature	rtag	Google Photos	Apple Photos	digiKam	Immich	PhotoPrism
Interface	CLI	Web/Mobile	Desktop/Mobile/Web	GUI	Web/Mobile	Web
Custom user tags	Yes	No	Yes (manual)	Yes (manual)	Yes (manual)	Yes (manual)
Arbitrary tag vocabulary	Yes	Proprietary	Manual	Manual	Manual	Fixed
Metadata standard	IPTC	Proprietary/JSON	IPTC (export)	IPTC/XMP	XMP/IPTC	YAML/XMP
Offline operation	Yes	No	Yes	Yes	Yes	Yes
Open source	Yes	No	No	Yes	Yes	Yes
GPU required	No	N/A	No	Optional	Optional	Optional

D. Discussion

`rtag` fills a unique niche in the image tagging ecosystem. It is the only tool that combines zero-shot tagging with arbitrary user-defined tags, CLI-first design, standard metadata output (IPTC), and offline CPU-only operation. This combination is particularly valuable for users who need scriptable, automatable tagging workflows that integrate with existing photo management software.

The fundamental trade-off is accuracy vs. flexibility. Specialized tagging models achieve higher accuracy on common categories because they are trained specifically for the tagging task. `rtag` sacrifices some accuracy by using CLIP's general-purpose embeddings, but gains the ability to tag with a virtually unlimited vocabulary—including highly specific niche subjects and arbitrary label combinations that would never appear in a standard closed-vocabulary dataset.

`rtag` is also complementary to photo management software. Users can run `rtag` to pre-tag images with IPTC metadata before importing them into applications like digiKam or Lightroom, which can read and display the tags. This workflow combines `rtag`'s zero-shot flexibility with the rich browsing and editing features of dedicated photo management tools.

VIII. Discussion

This section discusses the practical applications enabled by `rtag` and the limitations that constrain its current tagging quality.

A. Practical Applications

`rtag`'s combination of zero-shot tagging, arbitrary tag vocabularies, and scriptable CLI interface makes it applicable across several domains:

Media and entertainment: Bulk tagging of stock photo libraries with content descriptors, scene types, and mood keywords. Studios and agencies can define custom tag vocabularies tailored to their cataloging needs.

Digital archiving: Libraries, museums, and historical archives can tag digitized photo collections with subject matter, time period, and content descriptors. The ability to define custom tags is critical for specialized archival vocabularies.

Security and surveillance: Tagging extracted stills from CCTV or body camera footage with object and scene categories to enable keyword-based retrieval, useful for reviewing specific incidents.

Healthcare: Categorizing clinical photographs (dermatological images, wound documentation, surgical site photos) with domain-specific tags using a custom tag file. This can assist in organizing and retrieving images for clinical review.

Personal photo management: Tagging personal photo libraries with content categories to enable search via the operating system's file manager. Since `rtag` writes standard IPTC metadata, tags are visible in macOS Finder, Windows Explorer, and most photo management applications.

E-commerce: Product image tagging for catalog management, enabling attribute-based search (color, material, style) using custom tag vocabularies defined by the catalog taxonomy.

B. Limitations

Several limitations should be considered when evaluating `rtag`:

Model capacity: `rtag` currently uses CLIP ViT-B/32, the smallest backbone variant. Larger models (ViT-L/14, ViT-H/14) would improve tagging quality but at the cost of increased inference time and memory usage.

File modification: IPTC metadata is embedded directly in the image file, which modifies the original. No non-destructive option (XMP sidecar files) is currently available.

Single-label evaluation: ObjectNet provides single-label ground truth, limiting evaluation to top-k accuracy. Multi-label evaluation with metrics such as mAP or F1 score would provide a more complete picture of tagging quality but requires a multi-label benchmark dataset.

No GPU utilization: The current implementation runs entirely on CPU. While this ensures broad compatibility, GPU acceleration would significantly improve throughput for large-scale tagging tasks.

English-only: CLIP ViT-B/32 is trained primarily on English text, limiting tag support to English labels. Multilingual CLIP variants such as SigLIP 2 [19] could enable multi-language tagging.

Input resolution: CLIP's 224×224 pixel input resolution may lose fine details that are important for distinguishing visually similar objects.

Threshold sensitivity: The single global threshold τ applies uniformly to all tags. Per-category threshold optimization could improve results but is not currently implemented.

IX. Conclusion and Future Work

This paper presented an extended study of `rtag`, a command-line tool for automatic semantic image tagging based on OpenAI's CLIP model. Building on the original

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

conference publication, we provided a comprehensive survey of modern vision-language models and image tagging approaches, situating `rtag` within the broader landscape of contrastive learning, specialized tagging models, and prompt engineering research.

The core method computes dot-product similarity between CLIP-encoded image features and tag text features, assigning tags that exceed a configurable threshold. By leveraging the `rclip` caching infrastructure, the tool achieves efficient re-tagging with a 25.3% speedup for previously indexed images. The scalability analysis confirmed approximately linear scaling at larger dataset sizes, as fixed overhead costs (model loading, database initialization) are amortized over more images, with throughput of approximately 177 images per second on an Apple M1 Max CPU for indexed images.

Our analysis of prompt engineering effects across three templates showed a progressive improvement, with the "photo of a [tag]" template improving top-1 accuracy by 4.36 percentage points and top-5 accuracy by 5.94 percentage points on the ObjectNet dataset—more than three times the 1.3 percentage point gain reported by Radford et al. on ImageNet. This substantial improvement demonstrates that prompt template selection is a high-impact optimization for real-world photograph datasets, with further gains possible through template ensembling and learnable prompt tuning.

The systematic comparison with existing solutions revealed that `rtag` occupies a unique niche: it is the only tool combining zero-shot tagging with arbitrary user-defined tags, CLI-first design, standard IPTC metadata output, and offline CPU-only operation. While specialized models like RAM achieve higher accuracy on common tag categories, `rtag`'s flexibility and simplicity enable practical use cases without requiring specialized architectures or GPU resources.

The discussion of metadata standards highlighted the importance of using established formats like IPTC for interoperability with photo management software.

Building on the conference paper's [1] original future work items, we identify the following directions for strengthening `rtag` as a versatile tool for automated image organization.

From the conference paper:

- XMP sidecar file support for non-destructive tagging.
- Parallel image processing for improved throughput.
- Preventing re-indexing on file rename.
- Evaluation on additional datasets.

New directions identified in this extended study:

- Support for larger CLIP backbones (ViT-L/14, ViT-H/14) via migration to the `open_clip` library [6].
- GPU acceleration via MPS (Apple Silicon) and CUDA backends.
- User-configurable prompt template strings (e.g., `--template "photo of a [tag]"`) to let users opt into template wrapping when their tags are simple object labels, while preserving the default bare-tag behavior for maximum flexibility.
- Prompt template ensembling as default behavior, averaging embeddings across multiple templates for improved accuracy.
- Tuning of image preprocessing and cropping together with per-class similarity thresholds to reduce incorrect or missed labels.
- Learnable prompt tuning using CoOp [24] or CoCoOp [25] for domain-specific tagging applications.
- Integration with specialized tagging models (RAM, Tag2Text) for hybrid pipelines that combine zero-shot flexibility with specialized accuracy.
- Hierarchical tagging using WordNet or other ontologies to organize tags into taxonomic structures.
- Utilizing existing image metadata (GPS coordinates for location-based tagging, EXIF date for temporal tags) to enrich the tagging process.
- Multi-language tag support via multilingual CLIP models such as SigLIP 2 [19].
- Evaluation on multi-label benchmarks with metrics such as mAP and F1 score for a more comprehensive quality assessment.

`rtag` is open-source and available on GitHub [29].

References

[1] Y. Mikhalevich, "Automatic semantic image tagging at scale: AI-powered command-line tool based on CLIP," in *Proceedings of CONTENT 2024, The Sixteenth International Conference on Creative Content Technologies*, IARIA, 2024, pp. 7–13, ISBN: 978-1-68558-159-6.

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, et al., "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8748–8763.

[3] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, et al., "Scaling up visual and vision-language representation learning with noisy text supervision," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 2021, pp. 4904–4916. [retrieved: Mar. 2026]. [Online]. Available: https://proceedings.mlr.press/v139/jia21b.html

[4] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 162, PMLR, 2022, pp. 12 888–12 900. [retrieved: Mar. 2026]. [Online]. Available: https://proceedings.mlr.press/v162/li22n.html

[5] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, *Sigmoid loss for language image pre-training*, 2023. arXiv: 2303.15343 [cs.CV]. [retrieved: Mar. 2026]. [Online]. Available: https://arxiv.org/abs/2303.15343

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

[6] M. Cherti, R. Beaumont, R. Wightman, M. Wortsman, G. Ilharco, C. Gordon, et al., "Reproducible scaling laws for contrastive language-image learning," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2023, pp. 2818–2829. DOI: 10.1109/cvpr52729.2023.00276 [retrieved: Mar. 2026]. [Online]. Available: http://dx.doi.org/10.1109/CVPR52729.2023.00276

[7] X. Huang, Y. Zhang, J. Ma, W. Tian, R. Feng, Y. Zhang, et al., *Tag2text: Guiding vision-language model via image tagging*, 2023.

[8] Y. Zhang, X. Huang, J. Ma, Z. Li, Z. Luo, Y. Xie, et al., *Recognize anything: A strong image tagging model*, 2023.

[9] Y. Mikhalevich, "Using text queries to look up unlabeled images: A command-line search tool based on clip," in *Proceedings of CONTENT 2023, The Fifteenth International Conference on Creative Content Technologies*, IARIA, 2023, pp. 7–11.

[10] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng, "Grounded compositional semantics for finding and describing images with sentences," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 207–218, 2014.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, et al., "Going deeper with convolutions," 2014. arXiv: 1409.4842 [cs.CV]. [retrieved: Mar. 2026]. [Online]. Available: https://arxiv.org/abs/1409.4842

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[16] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2019.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[19] M. Tschannen, A. Gritsenko, X. Wang, M. F. Naeem, I. Alabdulmohsin, N. Parthasarathy, et al., *Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features*, 2025. arXiv: 2502.14786 [cs.CV]. [retrieved: Mar. 2026]. [Online]. Available: https://arxiv.org/abs/2502.14786

[20] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, et al., "Laion-5b: An open large-scale dataset for training next generation image-text models," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22, New Orleans, LA, USA: Curran Associates Inc., 2022, ISBN: 9781713871088.

[21] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "Cnn-rnn: A unified framework for multi-label image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2285–2294.

[22] X. Huang, Y.-J. Huang, Y. Zhang, W. Tian, R. Feng, Y. Zhang, et al., *Open-set image tagging with multi-grained text supervision*, 2023. arXiv: 2310.15200 [cs.CV]. [retrieved: Mar. 2026]. [Online]. Available: https://arxiv.org/abs/2310.15200

[23] T. Weyand, I. Kostrikov, and J. Philbin, "Planet - photo geolocation with convolutional neural networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, Springer, 2016, pp. 37–55.

[24] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *International Journal of Computer Vision*, vol. 130, no. 9, pp. 2337–2348, 2022.

[25] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Conditional prompt learning for vision-language models," 2022. arXiv: 2203.05557 [cs.CV]. [retrieved: Mar. 2026]. [Online]. Available: https://arxiv.org/abs/2203.05557

[26] S. Menon and C. Vondrick, "Visual classification via description from large language models," 2022. arXiv: 2210.07183 [cs.CV]. [retrieved: Mar. 2026]. [Online]. Available: https://arxiv.org/abs/2210.07183

[27] Y. Mikhalevich, *rclip*, version 1.7.26, Feb. 2024. [retrieved: Mar. 2026]. [Online]. Available: https://github.com/yurijmikhalevich/rclip

[28] R. D. Hipp, *SQLite*, version 3.31.1, 2020. [retrieved: Mar. 2026]. [Online]. Available: https://www.sqlite.org/index.html

[29] Y. Mikhalevich, *rtag*, version 0.1.0, Feb. 2024. [retrieved: Mar. 2026]. [Online]. Available: https://github.com/yurijmikhalevich/rtag

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

[30] *Iptc photo metadata standard*, Jan. 2023. [retrieved: Mar. 2026]. [Online]. Available: https://www.iptc.org/standards/photo-metadata/iptc-standard/

[31] T. Gulacsi and J. Campbell, *IPTCInfo3*, version 2.1.4, 2019. [retrieved: Mar. 2026]. [Online]. Available: https://github.com/james-see/iptcinfo3

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, et al., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[33] Adobe Systems Incorporated, "Xmp specification part 1: Data model, serialization, and core properties," Tech. Rep., 2012, Version 2012. [retrieved: Mar. 2026]. [Online]. Available: https://www.adobe.com/devnet/xmp.html

[34] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, et al., "Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models," *Advances in Neural Information Processing Systems*, vol. 32, pp. 9448–9458, 2019.

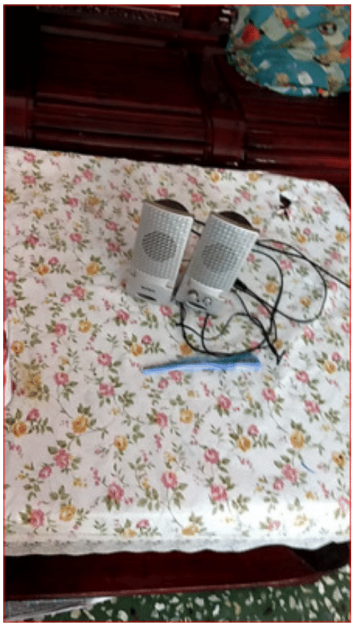
[35] ImageMagick Studio LLC, *Imagemagick*, version 7.0.10, Jan. 4, 2023. [retrieved: Mar. 2026]. [Online]. Available: https://imagemagick.org

[36] digiKam Team, *Digikam: Professional photo management*, 2026. [retrieved: Mar. 2026]. [Online]. Available: https://www.digikam.org/

[37] A. Tran and contributors, *Immich: High-performance self-hosted photo and video management*, 2026. [retrieved: Mar. 2026]. [Online]. Available: https://immich.app/

[38] M. Neubert and contributors, *Photoprism: AI-powered photos app for the decentralized web*, 2026. [retrieved: Mar. 2026]. [Online]. Available: https://www.photoprism.app/

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org



```
Profile-iptc: 334 bytes
  unknown[2,0]:
  Keyword[2,25]: cassette
  Keyword[2,25]: cassette player
  Keyword[2,25]: dial telephone, dial phone
  Keyword[2,25]: electric fan, blower
  Keyword[2,25]: handkerchief, hankie, hanky, hankey
  Keyword[2,25]: iPod
  Keyword[2,25]: loudspeaker, speaker, speaker unit, loudspeaker system, speaker system
  Keyword[2,25]: modem
  Keyword[2,25]: mosquito net
  Keyword[2,25]: notebook, notebook computer
  Keyword[2,25]: radio, wireless
  Keyword[2,25]: sewing machine
  Keyword[2,25]: tape player
```

Figure 4. An ObjectNet [34] image of speakers tagged by `rtag` using the default ImageNet-1k label set (threshold $\tau = 0.25$). The tool correctly assigns the combined label "loudspeaker, speaker, speaker unit" as a top tag. Semantically related tags such as "radio" and "cassette player" reflect CLIP's learned associations between audio equipment concepts.

```
Profile-iptc: 280 bytes
  unknown[2,0]:
  Keyword[2,25]: bathtub, bathing tub, bath, tub
  Keyword[2,25]: dishwasher, dish washer, dishwashing machine
  Keyword[2,25]: maraca
  Keyword[2,25]: plunger, plumber's helper",
  Keyword[2,25]: screwdriver
  Keyword[2,25]: soap dispenser
  Keyword[2,25]: toilet seat
  Keyword[2,25]: tub, vat
  Keyword[2,25]: washbasin, handbasin, washbowl, lavabo, wash-hand basin
  Keyword[2,25]: butternut squash
```

Figure 5. An ObjectNet [34] image of soap tagged by `rtag` using the default ImageNet-1k label set (threshold $\tau = 0.25$). The presence of tags like "bathtub" and "dishwasher" alongside "soap dispenser" illustrates CLIP's contextual associations—soap frequently co-occurs with bathroom and cleaning contexts in the training data.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

```
Profile-iptc: 270 bytes
  unknown[2,0]:
  Keyword[2,25]: binoculars, field glasses, opera glasses
  Keyword[2,25]: bolo tie, bolo, bola tie, bola
  Keyword[2,25]: espresso maker
  Keyword[2,25]: holster
  Keyword[2,25]: lens cap, lens cover
  Keyword[2,25]: loupe, jeweler's loupe",
  Keyword[2,25]: Polaroid camera, Polaroid Land camera
  Keyword[2,25]: reflex camera
  Keyword[2,25]: slot, one-armed bandit
  Keyword[2,25]: tripod
```

Figure 6. An ObjectNet [34] image of a camera tagged by `rtag` using the default ImageNet-1k label set (threshold $\tau = 0.25$). Tags such as “binoculars” and “opera glasses” appear alongside “reflex camera” and “Polaroid camera,” showing CLIP’s semantic clustering of optical devices that share visual features such as lenses and compact form factors.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Extracting Entity Relationship Diagrams from Unstructured Text - A Hybrid Approach

Vaihunthan Vyramuthu
Department of Electronics and Computer Science
Aalen University
Aalen, Germany
email: vaihunthan@gmail.com

Gregor Grambow
Department of Electronics and Computer Science
Aalen University
Aalen, Germany
email: Gregor.Grambow@hs-aalen.de

***Abstract*—Data models are a crucial part of computer science applications that automate real-world processes. An important model for abstractly capturing data sets and their relations is the Entity-Relationship (ER) model. To create these models, real-world entities and their relations have to be abstracted. However, the transformation from real world descriptions in natural language to standardized ER diagrams can be tedious and error-prone. Recently, Natural Language Processing (NLP) has gained much attention, but this specific area is still mostly handled manually by humans. With this contribution, we aim at automating the time-consuming interpretation of textual database scenarios by describing a hybrid system to capture components of the ER model from German texts using NLP. This hybrid system combines rule-based and model-based approaches. This may lead to a more robust and reliable extraction, as errors in one of the approaches can be compensated by the other. We implemented and tested both approaches, whereas the main extraction is performed by the rule-based variant. The results of the model-based approach are used as a comparison to the rule-based results and can be applied to check the correctness and improve the results. We conducted an evaluation using a set of real-world scenarios describing various entities and their relationships and applied a set of metrics suited to the imbalanced nature of the dataset. Our evaluation shows promising results indicating that a hybrid approach can deliver better results than classical approaches.**

***Index Terms*—Entity-Relationship Model; Natural Language Processing; Named Entity Recognition; POS-Tagging; SpaCy; LSTM; Transformer; Imbalanced Data**

I. Introduction

This publication is an extended version of our previous work from 2025 [1]. In today's data-driven world, large amounts of text are generated that may contain valuable information. Examples of this include police reports from online press portals, social media posts or Amazon reviews. Various analyses are possible that can provide insights about future trends, user ratings and sentiment or other questions [2]. Extracting structured data from unstructured text, however, is a complex task that mostly requires manual processing. Automating this process can save time and resources and improve the efficiency of data analysis [3] [4].

At the same time, the use of database systems for storing and managing data is prevalent. A common approach to database modeling is the use of *Entity Relationship Models* (ERM), followed by conversion to a relational model and finally to *Structured Query Language* (SQL) statements to represent real world data in a database [5].

Fig. 1. Overview from plain text to ERM.

Figure 1 presents a transformation of unstructured text into an ERM. The process includes the extraction of relevant entities and relationships from textual requirements, which are then formalized into an ERM. This conceptual model is subsequently translated into a relational schema, preserving the semantic integrity of the original requirements. Finally, the relational schema is implemented using SQL statements, resulting in a database schema. This workflow exemplifies the systematic progression from human-readable specifications to machine-executable data definitions. The primary focus of this publication is the transition from text to ERM generation, as it

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

represents the most complex and interpretation-sensitive step in the workflow.

Transforming textual requirements into an ERM serves several important purposes in the database design process. First, it enables a clearer and more intuitive understanding of the conceptual structure of the data. Second, working at the ER level allows the early identification and elimination of redundant entities or relationships. The ultimate goal is to minimize the number of relational tables, which in turn reduces the need for complex and time-consuming JOIN operations during query execution. Thus, the ERM plays a critical role in optimizing both the design quality and performance of the resulting relational database.

Figure 2 and Figure 3 present extracts of an ER schema for an university which governs how professors, students, lectures, and other entities should be connected. The comparison of Figure 2 and Figure 3 highlights the impact of cardinalities in ER diagrams on the compactness of the resulting relational models. In Figure 2, a many-to-many (n:m) relationship exists between students and lectures, which necessitates the introduction of a separate associative table. This results in a minimum of three relational tables, as the relation "hören" requires a composite primary key consisting of both *student-ID* and *lecture-ID*. In contrast, Figure 3 illustrates a one-to-many relationship between professors and lectures. The "lesen" relationship does not require a separate table, as it can be merged into the *lecture* table, which already uses *lecture-ID* as its primary key. This allows for a more compact representation using only two tables. Clearly defined cardinalities thus support schema simplification by enabling merging of relationships into entities.

Fig. 2. Relationship with minimum 3 tables.

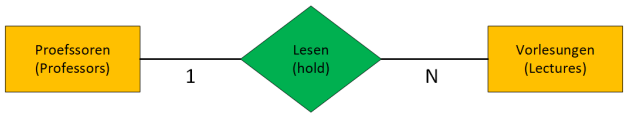


Fig. 3. Relationship with minimum 2 tables.

Manual data extraction involves reading the text and interpreting it correctly. The aim of this paper is to automate data extraction and thereby simplify the process of database modeling and integration. The results of this process are considered successful if the discrepancy between the human and software interpretation of a text with regard to correct ERM generation is as small as possible. This paper proposes a hybrid approach for the identification of entities and attributes, as well as the recording of entity relationships (including cardinalities). A hybrid approach may lead to increased error resistance. Rule-based systems alone are often prone to inaccurate results if the texts do not exactly match the expected patterns. The model-based approach compensates for this with its ability to learn from contexts and recognize variations. The analysis of semantic contexts and complex text structures that go beyond the recording of entities, attributes, and relationships is not part of this work. The implementation of the approach focuses on the processing of texts in German language.

The paper is organized as follows: Section II presents related work, followed by Section III, which contains details on the concept and architecture of the proposed approach. The implementation is presented in Section IV. Section V discusses limitations of the approach, while Section VI presents the evaluation. Finally, the conclusion can be found in Section VII.

II. Related Work

In literature, there exist several approaches that deal with the creation of ER models from natural language. Ghosh et al. [6] propose a method that uses grammatical knowledge patterns as well as lexical and syntactic analyses of request texts to create ERMs. This system assumes that the input text consists only of simple subject-predicate-object sentences for correct information extraction. This sentence structure makes it possible to detect entities (subject), relationships (verb) and attributes or other related entities (object). The approach applies NLP techniques, such as sentence segmentation, word separation (tokenization) or *POS-Tagging*. After that a domain-specific database is used to identify ERM components.

The segmentation in this work is carried out in a way that cases in English, such as "Mr. Mustermann", in which (.) appears after Mr, are not recognized as the end of the sentence. In *POS-Tagging*, the individual words are assigned to the corresponding word types, e.g., noun, verb, pronoun, adjective, preposition etc. The recognition of entities, attributes and relationships is performed using a database and the *Support Vector Machine* (SVM) classifier.

Six tables (word & synonym for each ERM component) are implemented, which contain domain-specific words and synonyms. To capture related entities, synonyms of a word from the table are given the same ID. This prevents redundant SQL tables from being created. Pronouns such as "he", "she" or "it" are already recognized in the *POS-Tagging* phase. In this phase, the pronoun is identified based on the closest previous entity that is present either in the same sentence or in the previous sentences. The technical term for this is *coreference resolution* [7].

In the publication by Kashmira et al. [8], the ERM components are recorded using neural networks. Three main modules are presented, namely the pre-processing module, the machine learning module, and the ER modeling module. In the first step, the pre-processing module is implemented using *NLTK* to pre-process the text. This includes steps such as converting the text into lowercase letters, tokenization into sentences, etc. The machine learning module is implemented using supervised learning. This module is trained with an English dataset where words are categorized into different categories including entity,

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

sub-entity, attribute, and irrelevant category. Four classifiers are taken into account when training the model: *Random Forest*, *Naive Bayes*, *Decision Table*, and *Sequential minimal optimization* (SMO).

To address the problem of attribute selection for entities in an ER diagram, the proposed model uses a combination of *ontology* and *web mining*. By using an ontology, an attempt is made to filter relevant attributes from the extracted entities. In addition, web mining is used to obtain further information that can be helpful in determining the attributes.

The publication by Habib [9] also follows similar preprocessing steps at the beginning. After the parsing process, the grammatical sentence structure is obtained so that the components of the ERM can be determined based on rules. The words are converted into a parse tree structure to understand how the individual parts of the sentence are related to each other. Using appropriate rules for sentence structures, entities, attributes, cardinalities and relationships can be determined.

There are several other publications that go in a similar direction and examine the topic of automatic ERM generation in the context of NLP in more detail. An example by Omar et al. [10] describes heuristic-based analysis options for ERM generation. In contrast, Omar and Abdulla [11] pursue the approach of training a machine learning model that can extract the entities from the text. Depending on the complexity of the input text and the scope of training, the model achieves precision values of up to 85%. The approach of Btoush and Hammad [12] can also be placed in a similar context. Here, a method is presented which, like [9], defines and applies certain rules for extracting information from texts.

The paper by Li et al. [13] proposed a hybrid approach that combines classical classifiers with large language models for relation extraction tasks. In their approach, a classifier is first used to identify potentially relevant entity pairs before the final relation prediction is performed by an LLM. Such combinations can improve the overall extraction quality and reduce unnecessary processing of unrelated entities. However, despite the hybrid architecture, these approaches still mainly rely on neural components and therefore remain largely non-transparent due to the black-box characteristics of LLMs and deep learning models.

Banjac et al. [14] presented *OmniaDB*, a web-based prototype for the automatic derivation of conceptual database models (CDMs) from heterogeneous source artifacts. Their approach combines the outputs of multiple existing tools that derive CDMs from different types of sources and integrates the resulting models into a unified representation. A major challenge addressed by their work is the integration of automatically generated and potentially unreliable CDMs. Initial results indicate that the integrated model can achieve higher precision and completeness compared to models derived from a single source type alone. Although their work does not directly address the extraction of ER components from unstructured natural language text, it nevertheless demonstrates the broader relevance of automated conceptual database design and highlights the importance of combining complementary extraction approaches.

It can be stated that two basic methodological approaches are used for ERM generation. The ERM components are determined either rule-based or using neural networks or artificial intelligence. Both approaches have advantages and disadvantages. The biggest advantage of rule-based extraction is the more time-efficient implementation as, unlike neural networks, no data pre-processing and training is required. Furthermore, the unambiguous definition of the rules ensures high extraction probability. In neural networks, a residual inaccuracy always remains. In contrast, model-based extraction is not limited to a few rules, but can learn complex and nested sentence structures to determine the ERM components. These sentence structures can contain linguistic variations and ambiguities, which can be recognized more easily by neural networks than by fixed heuristics. The heuristic approach is more suitable for small application areas and cannot maintain its effectiveness in large application areas.

Purely rule-based methods are prone to lack of generalization, while purely model-based approaches often depend on large training datasets and have difficulties in capturing rare or complex linguistic structures. This hybrid approach has the potential to overcome these limitations by combining the strengths of both methods: The rule-based method enables accurate extraction, while the model-based method helps to validate and improve the results, resulting in a more robust and adaptable solution to different text scenarios.

Existing rule-based approaches often focus only on selected elements, such as entities and relationships, while other important aspects are frequently omitted. In contrast, our approach systematically considers all core components of an ER model, including entities, relationships, attributes, and cardinalities. In particular, explicit handling of cardinalities distinguishes our approach from many existing rule-based methods. We conducted an additional literature search, but were unable to identify closely related work that applies such a hybrid strategy in the specific context addressed by our paper.

III. Overview of the Architecture

Figure 4 shows a general overview of the individual processes of the proposed approach. The latter is divided into a rule-based and a model-based part. The model-based algorithm can be used either to train new or existing models or to extract the ER components from a text. However, these results are not directly used for the resulting ERM, but can be applied to evaluate the rule-based results. This makes it possible to check whether the final result from the rule-based process may still need to be modified manually.

The input for both parts of the approach is a text that is saved in a *.txt* file. The *output.json* file contains all ER components and relationships found in a structure that can be read by an external ER modeling tool. The artifacts (in the figure: ordinary rectangles or arrow labels) represent the created files or results. These files are required for the subsequent processes.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

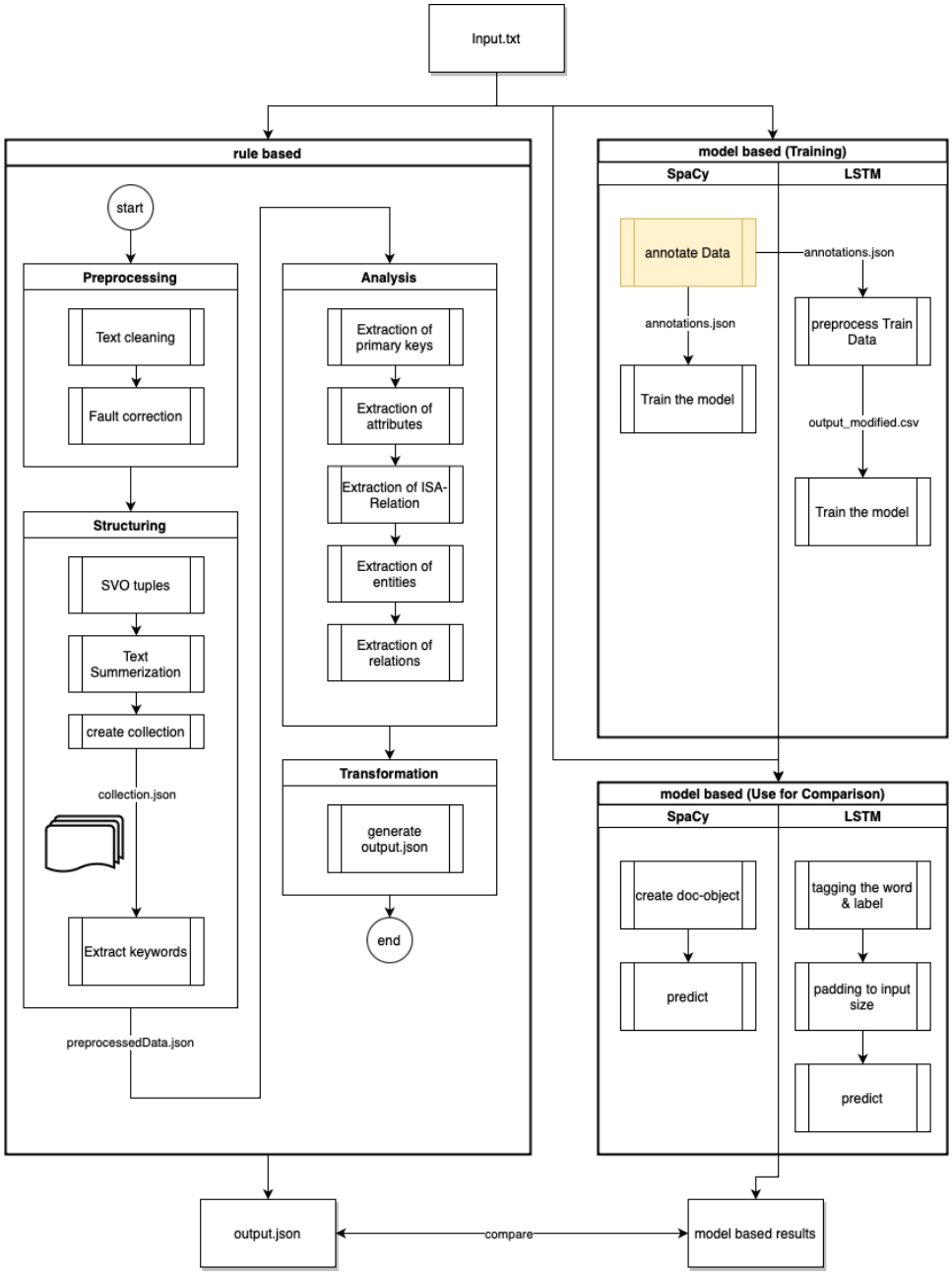


Fig. 4. Overall flow chart of the implementation.

The rule-based approach is made up of the sub-steps pre-processing, structuring, analysis and transformation. During pre-processing, the text data is cleansed and corrected for spelling errors. The structuring phase contains processes that break down the text into more meaningful parts and store them temporarily. In the main block of the analysis, the ER components are extracted one after the other.

In the model-based part, a different process path is carried out depending on the selected model type (*SpaCy*-Transformer [15] or *LSTM*). Due to the individual input requirements of the models, the text data must be converted into the permitted form for both the training case and the use case. The process block highlighted in yellow indicates an annotation process that is performed using an external tool. For the *LSTM* model, the output of the annotation tool must be pre-processed again into a *.csv* file so that it can be used for training. After the training processes, the model can be applied to new text data. For the *SpaCy* model, it is sufficient to convert the text into a doc object. The text for the *LSTM* model, on the other hand, requires additional processing (analogous to the training process), which decodes the words and labels into numbers and scales the input word vector to a specified size (so-called *padding*). This hybrid combination provides an important advantage for continuous optimization. The results

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

of the model-based approach can be used to dynamically adapt and further develop the rules of the rule-based system. This means that the hybrid system can become increasingly precise and effective over time through feedback and new data sets.

The selection of *SpaCy* as a model and NLP-Tool is based on its robust capabilities and user-friendly implementation. SpaCy is recognized for its comprehensive documentation and strong support from an active developer community. *LSTM* were chosen for their effectiveness in handling sequential data and their ability to capture contextual dependencies over extended text passages. This makes them particularly suitable for tasks that require understanding the relationships within long text sequences of information.

LSTM networks are a specialized form of Recurrent Neural Networks (*RNN*) designed to address the vanishing gradient problem. Traditional *RNN* struggle to capture long-range dependencies in sequences because gradients tend to vanish during backpropagation over time, making learning ineffective for long texts [16].

LSTM introduce a gating mechanism—comprising an input gate, forget gate, and output gate that allows the model to selectively retain or discard information over time [17]. This structure enables *LSTM* to capture dependencies across long sequences, which is particularly beneficial for longer sentences or documents with complex syntax. Unlike *RNN* or *LSTM*, Transformers like *SpaCy* eliminate recurrence entirely and instead rely on a self-attention mechanism, which allows the model to weigh the relevance of all other words in the sequence simultaneously [18].

IV. Implementation

Starting with the rule-based part, it can be noted that it is important not to define too many special rules. Otherwise, the implementation will be too application-specific and errors for other text styles will sometimes occur. The first process in data pre-processing is text cleansing. This process is divided into three steps. Each result of the individual pre-processing and structuring steps is saved in the *preprocessedData.json*. As starting point, the unstructured text is filtered from the *.txt* file.

In this first step, existing white-spaces or empty lines are also taken into account. In the second step of text cleansing, individual sentences from the text are found and saved using the *SpaCy* model (*de_core_news_sm*). In the last step of the text cleansing process, unnecessary sentences that do not provide necessary information for the ER diagram design are removed. In this process, sentences are removed using *RegEx* matches of certain words, such as “database” or “modeling”, are sorted out.

The next process in pre-processing is the error correction of words. The *Levenshtein-similarity* is used here. The Python library *pyspellchecker* checks whether there is an error for each word in a sentence. If this is the case, the word is replaced by the closest one.

In the structuring task block, the main focus is on capturing the subject-verb-object (SVO) sentence structure and some important term frequency analyses, with the help of which the text can be broken down into smaller information-rich parts. While only the *normalized word frequency* (TF) is used for the text summary, the *inverted document frequency* (TF-IDF) helps to capture the most significant keywords from the text.

The TF is used to calculate the frequency of a specific term in a text document. TF measures how often a term appears in a document. The calculation of Term Frequency can be performed in various ways, with the most common method being the counting of the number of occurrences of the term in the text. The formula for the calculation is given by [19]:

$$TF_n(t,d) = \frac{TF_a(t,d)}{max_{t_i \in d}(TF_a(t_i,d))}$$

n	relative frequency
a	absolute frequency
t	term
d	document

The numerator represents the number of occurrences of the term t in document d, and the denominator describes the maximum frequency of any term in d. This normalization prevents distortion in large documents by adjusting the absolute frequency. TF yields a value in the range of 0 to 1, where a value close to 1 indicates a high frequency of the term in the document.

The Document Frequency (DF), on the other hand, calculates the frequency of a term within a corpus of text documents. DF measures how many documents contain a particular term and allows assessing the relevance of the term across the entire collection of documents. For example, this is used to identify stop words, which occur in many documents and therefore are of little significance for identifying key terms. The value of DF also ranges between 0 and 1.

The combination of TF and DF leads to the TF-IDF method (Term Frequency-Inverse Document Frequency), which is calculated by the following formulas [19]:

$$IDF(t) = \log \frac{N}{|d : t \in d| + 1}$$

$$TF_IDF(t,d) = TF_n(t,d) \times IDF(t)$$

N	total number of documents
t	term
d	document
n	relative frequency

The denominator of $IDF(t)$ represents the number of documents that contain the term t. Thus, the TF-IDF method is used to evaluate the relevance of a term in a document within a corpus.

Figure 5 illustrates Zipf’s law. This law states that the frequency of a word is inversely proportional to its rank. In other words, the more frequently a word appears, the lower its rank, and vice versa. Luhn [20] proposed a similar idea. In his approach, the distribution of important words in

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

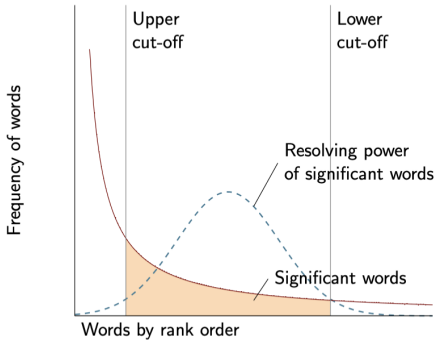


Fig. 5. Relationship between word frequency and rank [19].

documents is bounded by two thresholds that define the range of significance. Words outside these thresholds are either stop words (upper cut-off) or rare proper.

In contrast to SVO generation, these two structuring steps are only used optionally. These results are not used actively in the NLP workflow because it is possible that relevant information may be lost. Another use case for these results is to look for the most essential keywords in the text in order to compare them with the entities and attributes found. It should also be noted that to calculate the TF-IDF for keyword extraction, a document corpus (*collection.json*) must be created in which all existing ER diagram sample texts are stored. The *TfidfVectorizer()* function provided by the *Scikit-learn* library calculates the TF-IDF. No numbers are included in the keywords.

The text summary is implemented chronologically according to the following key points.

- count the number per word in the text (stop words excluded),
- calculate the normalized weight per word used by dividing the respective word count by the maximum word frequency occurring,
- calculate the sentence weight by adding the weight per word,
- search for sentences with the highest weighting.

Individual sentences are scored by adding the normalized TF for each word in the sentence. Depending on the original length of the text, a certain number of sentences is selected in descending order of the evaluation number. For this purpose, a corresponding factor is determined at the beginning with which the number of sentences is calculated. If there are fewer than three sentences in the text, the text is not summarized.

To generate SVO tuples, the sentence must be analyzed using *dependency parsing* and *POS-Tagging* provided in *SpaCy*. Certain commands can be used to analyse the grammatical structure of sentences so that the visualization shown in Figure 6 is displayed. Similarly to the tree structure, the successors or predecessors of a word are addressed with “children” or “parent”.

Based on this structure, a generally valid logic can be developed for the extraction of subject, verb, and object, which

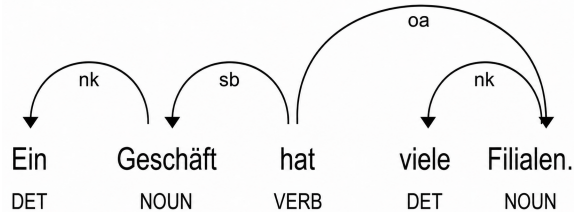


Fig. 6. Visualization of the grammatical sentence structure.

is shown in the structure diagram in Figure 7. The sentences in the text are entered individually into the function. There may be several verbs in each sentence, but each verb must belong to exactly one subject and object. *Lemmatization* can be used for the output of recorded relationships in texts. This involves changing the verb from its inflected form back to its basic form. For example, the German word “angeboten” is “anbieten” after lemmatization.

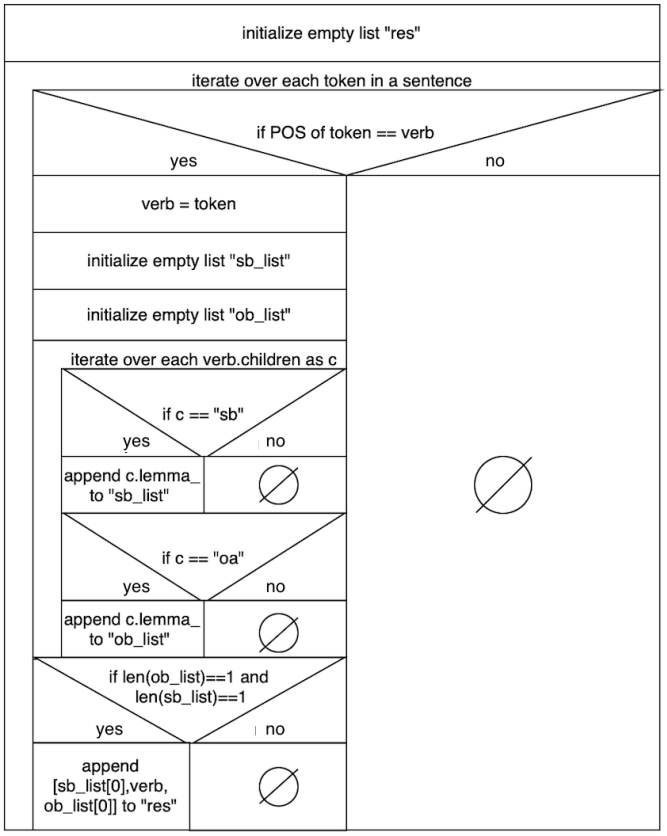


Fig. 7. Structogram for SVO extraction from a record.

A. Primary Key

Although this paper focuses on applications for German texts, English terminology is used throughout this work for reasons of readability and consistency.

The primary keys can be found using a *RegEx* comparison. The words “-id” or “-number” indicate a key candidate (see Figure 8).

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

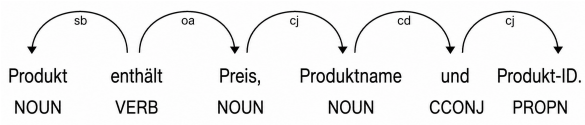


Fig. 8. Example sentence for extracting primary keys

The words are first converted to lower case so that there is a certain amount of leeway in the comparison. However, the limitation is the hyphen, which must be contained in a primary keyword. The following rules are implemented as Python code:

- Determine all nouns and filter primary key via *RegEx*,
- If the record only contains the word "unique", then the closest noun should be the primary key with "noun ID",
- If only the words "ID" or "id" appear, then the nearest noun should be the primary key with "noun ID".

B. Attribute

Attributes are identified as soon as a sentence contains a list of more than two nouns. The first noun that occurs in the sentence is the entity to which the remaining nouns or attributes belong. This simple rule serves as a first step in the identification process, but can be refined and enhanced through model-based results in the future.

C. ISA-Inheritance

To detect the specialization or generalization of entities, the following rules are followed:

- If a sentence contains the following verbs: ["include", "consist", "comprise", "share", "include"], then the sentence describes a generalization,
- If the sentence contains "type" as a word (noun), the first noun or entity is the generalization of the following nouns (entities)

It should be noted that the recognition of ISA relationships using rule-based approaches is limited. For example, the order of membership may be different for the entities, or the ISA description may extend over several sentences.

D. Entity

In the *preprocessed.json* under the item in which the SVO tuples are stored, the subject and object in each record can be either an entity or an attribute. These tuple words are therefore compared again with the attributes and primary keys found so far. If the same tuple word is also contained in the list of attributes or primary keys, it is discarded. The final result is a list that contains only the final entities.

E. Relationship

The SVO tuples can again be used for the relationships. If both the subject and the object are contained in the list of final entities, the verb is a relationship between two entities. There are sentences that are not formulated in an ordinary SVO style, but which define further relations between entities. Figure 9 shows an example sentence.

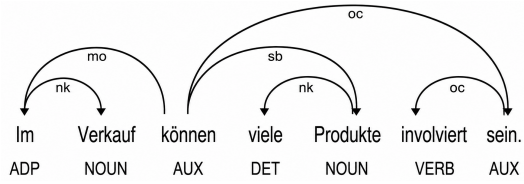


Fig. 9. Example sentence for the extraction of relationships

In order to take such sentences into account as well, a check is made to see whether two nouns occur in a sentence in addition to a verb, which are not contained in the attribute, ISA-Inheritance and primary key list. If this is the case, a relationship tuple can be extracted from this sentence again if it has not already been extracted from the SVO tuple.

F. Cardinality

Two min/max cardinalities must be determined for each relationship between two entities. The sentence in Figure 6 shows that the cardinalities can be taken from the determiners of the nouns. Once the SVO tuples have been reassigned to complete sentences with the help of indexing, the corresponding determiners of each entity can be determined. These are then translated into the corresponding min/max value using a comparison. For the correct min/max notation, the cardinalities of the entities in an SVO tuple must be swapped.

When interpreting the adverbs "at most" and "at least", the following word must also be taken into account, as this defines either the upper (max) or lower limit (min). An example sentence is illustrated in Figure 10.

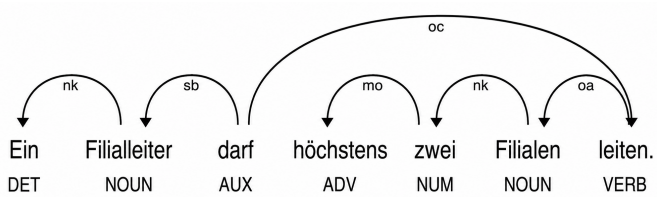


Fig. 10. Example sentence with the adverb "höchstens"

Due to the unlimited possibilities for translating this adverb, no rule-based application is suitable for this. It would make the rules too specialized for one use case.

G. LSTM Architecture

The model-based approach primarily serves as a comparison tool for the ER components found in the rule-based algorithm. Therefore, the hybrid approach is also advantageous for performance reasons. The rule-based methods often deliver faster results as they do not rely on extensive calculations, while the model-based part intervenes where more in-depth analyses are required. This efficient applicability ensures that the goal can be achieved faster without losing accuracy. In the following, only the architecture of the *LSTM* model is examined in detail, as the Transformer model is provided as a pre-trained component by *SpaCy*.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

The model architecture, designed using the *Tensorflow Keras* library [21] and the *Sequential API*, is shown in Figure 11. In Python, this is implemented as depicted in Listing 1:

Listing 1. Model architecture code

```
model = keras.Sequential()
model.add(InputLayer((max_len)))
model.add(Embedding(
    input_dim=num_words,
    output_dim=max_len,
    input_length=max_len
))
model.add(SpatialDropout1D(0.1))
model.add(Bidirectional(
    LSTM(
        units=100,
        return_sequences=True,
        recurrent_dropout=0.1
    )
))
```

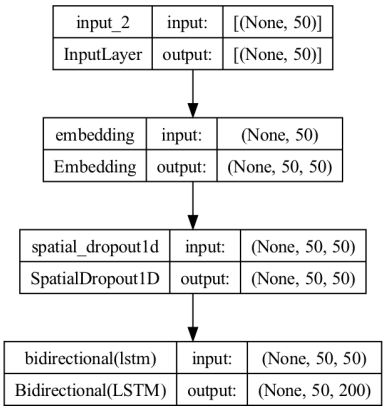


Fig. 11. Architecture of the designed LSTM model

The architecture consists of five layers. The output layer is not depicted in Figure 11. In addition to the input layer, the model includes an embedding layer, a dropout layer, and a bidirectional *LSTM*. The dropout layer prevents overfitting during training by randomly deactivating certain weights in the layer. When the model is applied to test data, the dropout layer is ignored.

The use of a bidirectional *LSTM* allows the model to better capture contextual information from both directions of the input text during training. Furthermore, this increases the number of trainable parameters, which helps reduce the bias error (also referred to as training loss). However, it is important to note that the model should not be overly complex, as this could again lead to overfitting.

The input layer has a size of 88 neurons, corresponding to the number of words. Multiplying this value by the input size of the embedding layer (50 neurons) results in a total of 4400 trainable parameters for this layer.

In NLP, words and phrases are often represented as one-dimensional vectors, where each dimension corresponds to a word in the vocabulary. This method is referred to as *one-hot encoding*. These vectors are high-dimensional, making them inefficient and difficult to work with. Instead of using these sparse high-dimensional vectors, the embedding layer maps each word to a lower-dimensional vector, where each dimension captures specific features of the word [22].

The dropout layer contains no trainable parameters. In contrast, the bidirectional *LSTM* has 120800 trainable parameters. This value is derived as follows:

$$p = 2 \times 4 \times (h \times (h + i) + h) = 120800$$

p	Number of trainable parameters
2	Factor due to bidirectionality
4	Four neural networks (gates) in LSTM
h	100 (number of LSTM cells)
i	50 (embedding dimension) + 100 (recurrent input) = 150 (input dimension)

A fixed input dimension is defined for the input layer. Based on the distribution of average sentence lengths (see Figure 12), a value of 50 is selected as the maximum sentence length or input dimension. The boxplot displayed above the histogram provides an additional statistical summary of the distribution. The central line inside the box represents the median sentence length, while the box itself corresponds to the interquartile range (IQR), containing the middle 50% of all sentence lengths. The whiskers indicate the typical spread of the data outside the quartiles, and the isolated point on the left side represents an outlier. Overall, the boxplot confirms that the sentence lengths are relatively concentrated around the median.

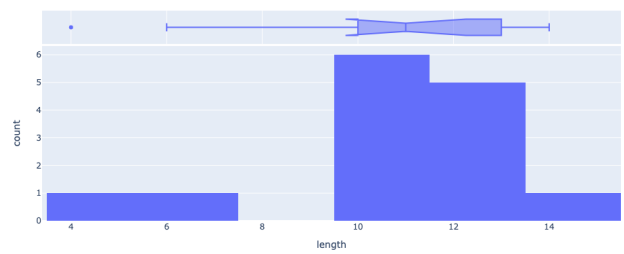


Fig. 12. Distribution of sentence lengths in the small dataset used for testing. The median is at length 11.

This means that the word vector of each sentence must be adjusted to this fixed size. After the tagging process, the words are decoded as numerical values. Due to varying sentence lengths, the sequences need to be expanded to the uniform input size of 50 by padding. These processes are illustrated in Figure 4, both in the model based (training) block and in the model based (use for comparison) block.

For updating the weight values, the Gradient Descent algorithm is used in combination with the *Sparse Categorical Crossentropy* loss function, as this function internally handles one-hot encoding. Moreover, this loss function is particularly suitable for classification problems because, compared to

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Mean Squared Error, it enables a significantly faster adjustment of the weight parameters.

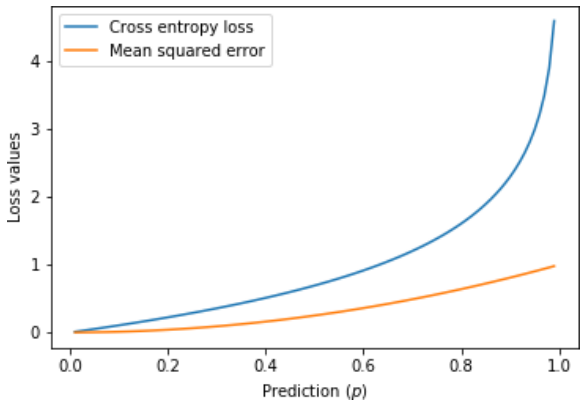


Fig. 13. Comparison of the loss functions Mean Squared Error and Cross Entropy. The Ground Truth is at 0. [23]

The reason lies in the characteristics of the *Categorical Cross Entropy* function: due to the logarithmic transformation of values between 0 and 1, errors result in substantially higher loss values than with the same input using *Mean Squared Error* (see Figure 13). As a result, the *Categorical Cross Entropy* function enables larger step sizes in the gradient descent process, thereby accelerating the training.

V. Limitations

The following aspects were not taken into account in the rule-based algorithm:

- Attributes for relationships,
- special cardinalities ("two", "three", etc.),
- weak entities, relationships, attributes, etc.,
- described ISA-Inheritance across several sentences,
- multi-valued or complex attributes.

In the future, the results of the model-based approach can be used to include more ER components in the results. An extension to the rule-based approach for this would be rather difficult, as a generally valid formulation of the rules is difficult and often tied to a specific use case. Moreover, limitations such as coreference resolution and those listed above can be better addressed by model-based approaches. Current coreference resolution tools are primarily optimized for English, while there is still a lack of robust solutions for German texts. Accurately interpreting the meaning of sentences presents a significant challenge, as words and phrases can change meaning depending on context. Ambiguity and polysemy further complicate the precise analysis of text. For example, the German sentence

"Er sieht den Verkäufer mit dem Fernglas"

("He sees the salesman with the binoculars") is structurally ambiguous, as it is unclear whether the binoculars are associated with the observer's action or with the salesman as an attribute. Furthermore, words often need to be correctly recognized and interpreted despite spelling errors. Techniques such as the Levenshtein distance can be applied in this context.

TABLE I
Word-level challenges for NLP

Word Type	Meaning	Examples
Homonyms	same spelling; different meaning	Schale, Bank
Homophones	same pronunciation; different meaning	mahlen/malen, leeren/lehren
Homographs	same spelling; different pronunciation	Montage (Pl. Montag / Zusammenbau)
Synonyms	different words for the same concept	essen, speisen, konsumieren
Compounds	composition of multiple words	Kapitänsmütze
Abbreviations	shortened forms of words	u.s.w.
Proper Nouns	individual names	Angela Merkel, Apple corp.
Inflections	grammatical adaptation of words	ich singe, ich sang
Anglicisms	words borrowed from English	surfen, boostern

Another major challenge in natural language processing is the handling of lexical ambiguities. Polysemy refers to words that have multiple meanings. In contrast, homonymy describes different words that share a similar spelling or pronunciation but have different meanings. Correctly understanding and assigning such ambiguities in a given context can be very challenging. Table I illustrates various types of words that can cause difficulties in NLP-based interpretation for German language.

This highlights the need for further research and development to improve the processing of German-language input in this context.

VI. Evaluation

The evaluation is based on several German texts that describe specific DB scenarios. This means that they contain specific formulations that describe the ER components. The rule-based algorithm only extracts ER components that correspond to the defined grammatical regularities. In contrast to the model-based approach, emphasis is placed on a qualitatively correct ER extraction instead of a quantitative result set.

Fig. 14. Rule-based extraction of ER components from a German database scenario text.

Figure 14 shows the output of the rule-based approach applied to a German-language text describing a specific database scenario. The extracted ER components—entities, relationships, attributes, cardinalities, and keys—are highlighted based on their semantic roles. The text is rich in domain-specific formulations, making it well-suited to evaluate ER extraction. The visualization also highlights that many relevant ER

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

components are already effectively captured by the rule-based method, indicating that the underlying rules are well-defined and cover a broad range of relevant grammatical structures.

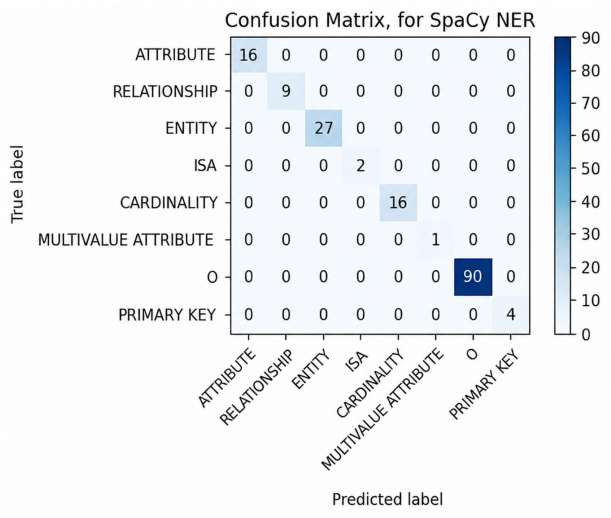


Fig. 15. Ideal reclassification result in the case of training and testing with the same data.

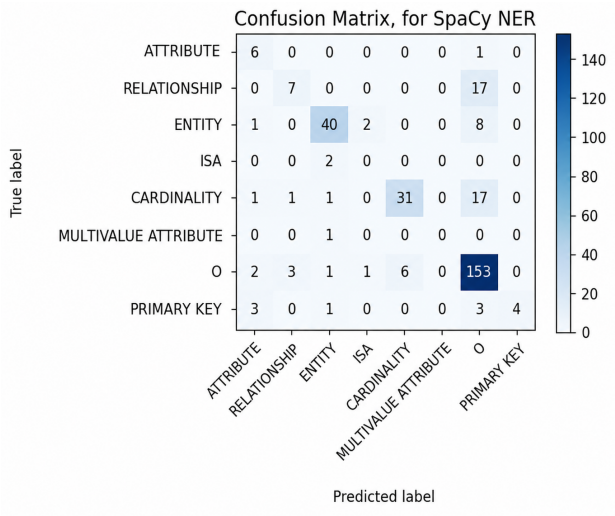


Fig. 16. Results of Crossvalidation with larger and different training and test data sets.

The Confusion Matrix in Figure 15 shows the reclassification case in which all previously labeled ER components are correctly classified during testing. However, the Confusion Matrix in Figure 16 provides more information about the generalization capability of the *SpaCy* model, because the cross-validation also examines text data that the model does not yet know and was therefore not part of the training process.

As shown in Figure 17, one of the main reasons why the metrics stagnate in this experiment is the small amount of training data. With only 77 training sentences, the model does not see enough examples to learn robust patterns, which limits its ability to improve beyond a certain point. Furthermore,

```
============================ Training pipeline ============================
ℹ Pipeline: ['tok2vec', 'ner']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ------  ------------  --------  ------  ------  ------  ------
  0       0          0.00     69.20    8.95    7.49   11.11    0.09
 21     200        182.58   2663.95   56.36   65.96   49.21    0.56
 47     400         71.33     58.70   57.78   65.66   51.59    0.58
 79     600         95.29     29.06   58.15   65.35   52.38    0.58
```

Fig. 17. Model performance with limited training data.

there is a class imbalance problem, where some entity types occur far less frequently than others. This makes it harder for the model to generalize and leads to uneven performance across different labels. As a result, both precision and recall remain stuck at moderate levels instead of improving over time.

```
============================ Training pipeline ============================
ℹ Pipeline: ['tok2vec', 'ner']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ------  ------------  --------  ------  ------  ------  ------
  0       0          0.00     66.20    0.00    0.00    0.00    0.00
  8     200        166.09   3079.88   98.41   98.41   98.41    0.98
 18     400        290.66    339.56   99.21   99.21   99.21    0.99
 31     600        147.25    100.29  100.00  100.00  100.00    1.00
```

Fig. 18. Model performance with extended training data.

In Figure 18, it can be observed that the *SpaCy* model reached very high precision and recall values after only 8 epochs, both around 98%. This shows that the model was able to generalize effectively and recognize entities with high accuracy at an early stage of training. The model was trained with an extended dataset of 206 training sentences and evaluated on 77 test sentences. Such a strong performance after a relatively small number of epochs underscores the quality of the dataset, while the effect of a potential imbalance in the data set played only a minor role.

Similarly to the *SpaCy* model, the *LSTM* model is also evaluated using the learning curves shown in Figure 19. After 10 training epochs, a solid validation accuracy of 92.1% and a validation error of 0.53 are achieved.

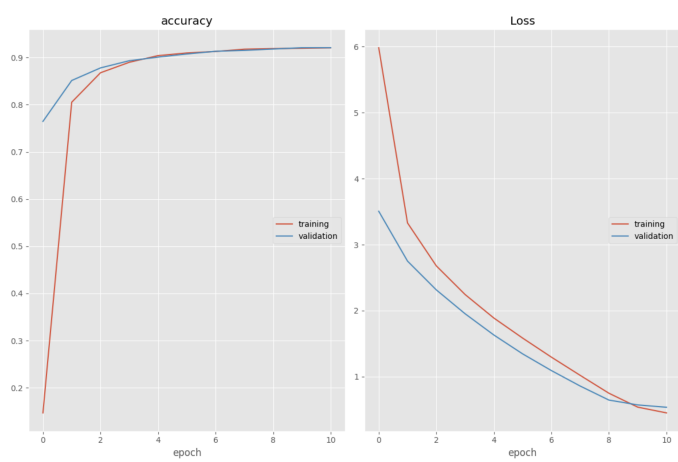


Fig. 19. Learning curves from the *LSTM* model training.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

This evaluation already shows promising results, but can only serve as a first step. As part of our future work, we plan to compare our results to the results created by domain experts for a larger set of different models.

The results of the two approaches, both rule-based and model-based, complement each other very well to form a complete tool due to their complementarity. It can be stated that the challenge of the rule-based method lies in the assignment of the nouns found to the individual components. Furthermore, the connection of attributes and relationships to certain entities was not entirely trivial. On the other hand, the model-based comparison offers possibilities to correctly capture different cardinalities or ISA inheritances due to the increased flexibility in sentence structure and grammar structure.

VII. Conclusion

Extracting information from unstructured text is a complex task. Manual processing of large amounts of text is time-consuming, error-prone, and not scalable. Recently, numerous approaches for automating this task have been proposed. However, there exist many cases where very specific information has to be extracted from unstructured text. These imply challenges based on the specifics and rules applicable for the desired result. One of these cases is the extraction of ER models. There exist several approaches, but they still lack different features for a complete and usable automation.

We proposed a hybrid approach to extract meaningful ER data from unstructured text. Two subsystems were developed, both of which can extract ER components from unstructured texts. Special NLP methods were used for the rule-based extraction of entities, attributes, and relationships. Due to the higher reliability of the rule-based results, these were used for the final ER model. *SpaCy* and *LSTM* models can be used to validate the rule-based results. In the future, these results can be supplemented by an automated comparison with the results of the model-based approach. Additionally, this extended article provides a more detailed view of the LSTM model architecture, includes more illustrative examples, and presents a meaningful evaluation of the model-based results.

In the literature, there are still research gaps in the area of automatic extraction of ERM from texts using NLP techniques. This includes the integration of contextual information, the consideration of ambiguity, and the additional complexity posed by the German language, such as compound nouns and flexible sentence structure. Furthermore, current approaches mainly focus on the extraction of ERM, while object-oriented representations such as class diagrams are rarely considered. Modern software systems and web frameworks commonly rely on object-oriented paradigms and frequently use Object-Relational Mappers (ORMs) to derive ERM directly from source code. Investigating the automatic generation of class diagrams and other object-oriented representations from textual descriptions is planned as part of future work. Another fundamental improvement is the training of the models with even more text data, so that even rarely occurring ER components, such as multi-value attributes, can be better learned. The pretrained models can be trained for other domains according to the principle of transfer learning so that the models can be used for other purposes, e.g., to create knowledge graphs.

Our future work will include the mentioned gaps: We will expand our approach to include more specific ER concepts. In addition, we will focus on enabling a broad applicability for different domains. We will also investigate options for automated integration of the results from the two approaches. To enhance the robustness of our evaluation, we will also include more use cases with real-world texts and compare the results of our approach with ER models created by human experts. Another area of future work is the extension of the approach to also incorporate other models. As a first step, we plan to integrate UML class diagram creation.

Recent advances in large language models (LLMs) have shown that such models are capable of extracting entities and relationships from textual descriptions with promising results, often even without task-specific fine-tuning. Nevertheless, the focus of this work was intentionally placed on a hybrid approach combining rule-based and classical model-based techniques. An important reason is the improved explainability and traceability of the extraction process, as rule-based methods allow a clearer understanding of why specific entities or relationships were identified. In contrast, LLM-based approaches often operate as black-box systems, making validation and error analysis more difficult. Furthermore, our approach enables a more controlled integration of linguistic rules tailored to German language characteristics. Therefore, integration and evaluation of LLM-based extraction methods will be considered as part of future work.

References

[1] V. Vyramuthu and G. Grambow, "Towards extracting entity relationship diagrams from unstructured text using natural language processing," in *Proc 17th International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA), Lisbon, Portugal, March 09-13*, 2025, pp. 42–47.

[2] M. Kanakaraj and R. M. R. Guddeti, "Performance analysis of ensemble methods on twitter sentiment analysis using NLP techniques," in *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*. IEEE, 2015, pp. 169–170.

[3] A. Rajput, "Natural language processing, sentiment analysis, and clinical analytics," in *Innov in Health Informatics*. Elsevier, 2020, pp. 79–97.

[4] R. Suganya, R. Krupasree, S. Gokulraj, and B. Abinesh, "Product review analysis by web scraping using NLP," in *Smart Data Intelligence: Proceedings of ICSMDI 2022*. Springer, 2022, pp. 427–436.

[5] V. T. N. Chau and S. Chittayasothorn, "A bitemporal SQL database design method from the enhanced entity-relationship model," in *2021 7th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*. IEEE, 2021, pp. 85–90.

[6] S. Ghosh, P. Mukherjee, B. Chakraborty, and R. Bashar, "Automated generation of ER diagram from a given text in natural language," in *2018 Int'l Conf on ML and Data Eng (iCMLDE)*. IEEE, 2018, pp. 91–96.

[7] V. Bryl, C. Giuliano, L. Serafini, and K. Tymoshenko, "Supporting natural language processing with background knowledge: Coreference resolution case," in *The Semantic Web–ISWC 2010: 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I 9*. Springer, 2010, pp. 80–95.

[8] P. Kashmira and S. Sumathipala, "Generating entity relationship diagram from requirement specification based on NLP," in *2018 3rd Int'l Conf on Information Technology Research (ICITR)*. IEEE, 2018, pp. 1–4.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

[9] M. K. Habib, "On the automated entity-relationship and schema design by natural language processing," *Int'l J Eng Sci*, vol. 8, no. 11, pp. 42–48, 2019.

[10] N. Omar, J. Hanna, and P. McKevitt, "Heuristic-based entity-relationship modelling through natural language processing," in *Proc. of the 15th Artificial Intelligence and Cognitive Science Conference (AICS-04)*. Artificial Intelligence Association of Ireland, 2004, pp. 302–313.

[11] M. Omar and A. Abdulla, "The entities extraction for entity relationship models from natural language text via machine learning algorithms," in *Proc 4th Int'l Conf of Basic Sci and Their Appl, Elbeida, Libya*, 2020.

[12] E. S. Btoush and M. M. Hammad, "Generating ER diagrams from requirement specifications based on natural language processing," *Int'l J of Database Theory and Application*, vol. 8, no. 2, pp. 61–70, 2015.

[13] X. Li, K. Chen, Y. Long, and M. Zhang, "Llm with relation classifier for document-level relation extraction," *IEEE Transactions on Big Data*, 2026.

[14] G. Banjac, D. Brdjanin, and D. Banjac, "Towards automatic conceptual database design based on heterogeneous source artifacts," in *European Conference on Advances in Databases and Information Systems*. Springer, 2023, pp. 487–498.

[15] M. Honnibal, I. Montani, S. Vanlandeghem, and A. Boyd, "spacy: Industrial-strength natural language processing in python," doi: 10.5281/zenodo.1212303, 2020.

[16] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[19] C. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[20] H. P. Luhn, "The automatic creation of literature abstracts," in *IBM Journal of Research and Development*. Ibm, 1958, vol. 2, no. 2, pp. 159–165.

[21] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[22] E. Zvornicanin, "What are embedding layers in neural networks?" May 2023. [Online]. Available: https://www.baeldung.com/cs/neural-nets-embedding-layers

[23] S. Weidman, "Deep learning from scratch." [Online]. Available: https://www.oreilly.com/library/view/deep-learning-from/9781492041405/ch04.html

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Perceptions of AI-Edited Video News: Evaluation, Classification, and Attitudes

Caspian J. Moosburner
RheinMain University of Applied Sciences
Wiesbaden, Germany
email: crescentcaspianarts@gmail.com

Dennis Quandt
RheinMain University of Applied Sciences
Wiesbaden, Germany
email: dennis.quandt@hs-rm.de

Matthias Kowald
RheinMain University of Applied Sciences
Wiesbaden, Germany
email: matthias.kowald@hs-rm.de

Wolfgang Ruppel
RheinMain University of Applied Sciences
Wiesbaden, Germany
email: wolfgang.ruppel@hs-rm.de

Till Dannewald
RheinMain University of Applied Sciences
Wiesbaden, Germany
email: till.dannewald@hs-rm.de

Matthias Narroschke
RheinMain University of Applied Sciences
Wiesbaden, Germany
email: matthias.narroschke@hs-rm.de

***Abstract*—With video news gaining more and more popularity, Artificial Intelligence (AI) video editing tools could be implemented to accelerate video news production. A challenging issue, however, is the quality of AI-edited video news and the acceptance of such news by media consumers. A cross-sectional survey with 2,219 participants across three independent waves was conducted in Germany to assess audience perceptions of video news clips edited by AI-based systems and by professional human editors. All survey participants were recruited by a commercial survey company. Participants were generally unable to reliably identify whether video clips were edited by humans or AI systems. Across several quality dimensions, ratings of AI-edited and human-edited clips were descriptively similar, with overlapping confidence intervals on multiple indicators. Future research can benefit from investigating fatigue related to AI tools and testing algorithmic performance improvements.**

Keywords-Artificial Intelligence; AI; video editing; video news; algorithms; AI evaluation.

I. INTRODUCTION

Expanding upon the presented survey results of a preliminary survey testing Artificial Intelligence (AI) video news editing algorithms in [1], this paper highlights the changes in survey design based on previous results, presents more detailed results from the first survey wave and includes additional survey waves conducted since the publication of the first conference paper at AIMEDIA 2025.

Video news has been established as a strong contender for attention in the digital landscape, aided by the rise of TikTok and their short form video content that is making a jump to other social media platforms [2]. This creates a need for quick video news reporting, a need that could be aided by implementing AI editing tools.

With 40% of young people preferring social media search engines over traditional means [2], the market share of video news reporting on social platforms will continue to grow. This is aided by media publishers who put a bigger focus on digital video production [2] making it clear that the video news reporting demand will increase, and the market is preparing to supply by offering more and more AI tools. One example for this is the launch of the AI writing tool “ChatGPT” in 2022, establishing AI tools as more and more popular and expected to reach a global market share of “more than 2.5 trillion USD by 2032” [3]. Creative industries have implemented generative AI and AI algorithms, for example Adobe Photoshop offering options like generative fill [4], Canva that has its own AI image generation [5] or on countless social media websites via algorithmic AI [6].

The conducted survey consists of three survey waves focussing on different aspects of AI news editing algorithms and their evaluation. Each survey wave consisted of roughly 750 participants split into two groups to investigate whether the tedious process of editing video news clips could be delegated to AI models and how AI editing technology is perceived by humans. In detail, the study measures subjective receptions of video news and survey participants’ evaluation of the quality of editing algorithms as well as attitudes towards AI tools in the media landscape.

This paper details the construction and field deployment of the three survey waves testing two different algorithms [7] against a human and a random edit and presents results on consumer news habits, the performance of the algorithms and attitudes towards the employment of AI in media landscapes, particularly in video news. The focus is put on the survey protocol allowing for statistical analysis of the gathered data. In Section II, the employed algorithms are introduced while Section III introduces the survey protocol by detailing the video experiment and general questions. Additionally, it

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

discusses the details of all three survey waves and a video-based choice experiment questioning the creator of the presented video clips. Section IV presents the results from all survey waves discussing them as well. The presented results, however, are mostly based on bivariate statistics and want to provide an overview on the sample and indicate promising paths for multivariate analysis and statistical models. Section V concludes the compiled research, and Section VI gives a brief outlook in potential future research.

II. EMPLOYED ALGORITHMS

Editing video news clips to employ within the study, comprises of two main steps. In a first step, typically 10 to 15 most suitable scenes for a news story are selected from up to 200 scenes in raw footage. In the second step, the selected scenes are compiled into a video sequence, which is then accompanied by a voice over of a news text to create the final video news clip. The core of the conducted survey waves investigates how automated AI-based editing of video news clips scores against human news editing as a high anchor and a random editing as a low anchor. Hereby, two AI models are considered, which both are described in detail in [7]. The first AI model is the CLIP (Contrastive Language–Image Pre-training) model [8], pairing images of video shots with snippets of news text. The second AI model is denoted as KIGVI (Automatic AI-supported composition of video news clips) model and has been developed by the RheinMain University of Applied Sciences [7]. The KIGVI model was trained using a dataset of 12354 video clips from news segments that ranged from 30 seconds to 5 minutes covering multiple categories of news from the years 2012 to 2025. The second version of the algorithm was further improved by expanding the data set it was being trained on to then compose new news clips that were employed in survey wave 3 [8]. The KIGVI model uses shot detection to split footage into scenes and follows learned professional video editing rules. These rules include varying shot sizes, an initial establishing shot illustrating the main topic and the inclusion of a human-like editing rhythm when composing the video news clip. This marks a major improvement over previous algorithms e.g., the CLIP algorithm that failed to align information sourced in the visual component as well as the textual one. To keep participants engaged, no clip is longer than 35 seconds ensuring that concentration can be maintained throughout the whole survey.

A previous experiment with a small sample size (n = 38 participants) scored the KIGVI algorithm against human edits based on professionality of the edit, choice of scenes and how well the video sequences illustrated the voiceover [7]. The findings of this experiment warranted further investigation with a bigger sample size and a larger survey that asks respondents about their opinion on AI, video news consumption habits and general media reception. Thus, a more comprehensive survey was developed. Human edited news clips were produced by professional editors. In random editing, scenes are randomly selected and compiled into a video news clip. All edited video news clips are voiced over by a professional German recording studio.

III. SURVEY PROTOCOL

A. Basic Principle

The survey is designed as an online survey. The participants of the survey are asked to assess video news clips according to a list of criteria, and to identify the editor of the respective video clip. Field work was done in three waves to allow adaptation of the survey instrument between the waves. The survey waves are designed as three consecutive waves with a similar survey instrument in each wave but different participants. It thus follows a cross-section design and is not a panel study. In general, wave one lead users through questions about news habits, then AI attitudes and AI in news opinions before introducing and conducting the video experiment and closing with sociodemographic factors. Within the video experiment section, users are tasked with attempting to identify the creator of the video clip they were shown. In survey wave 2, the creator identification question was no longer suitable and is therefore eliminated. Survey wave 3 group B excludes the identification question as well.

B. Samples

As a survey area, the greater RheinMain area in Germany is used. The RheinMain area is a multi-state area in southern Germany including major cities like Frankfurt and Wiesbaden, mid-sized cities such as Russelsheim, as well as a number of smaller municipalities. A market research institute was tasked to recruit survey participants living in or around the RheinMain area aged from 16 to 70+ for all survey waves. The frame population thus included participants aged 17 years and older from the online access panel, currently living in the RheinMain area.

C. Survey Structure

In an experiment portion of the questionnaire video news clips are shown, which cover three overall categories of news. These are:

- *Traffic news*
- *Local news*
- *News concerning politics*.

Each of these three categories of news consists of four video news clips, and each video news clip is edited in four different variants, which are:

- *Human editing*
- AI-based editing using the *KIGVI model*
- AI-based editing using the model based on *CLIP*
- *Random editing*.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

This results in a pool of 48 video news clips (3 categories × 4 video news clips × 4 variants) in waves 1 and 3 or 36 video news clips (3 categories x 4 video news clips x 3 variants) in wave 2 as shown in Figure 1.

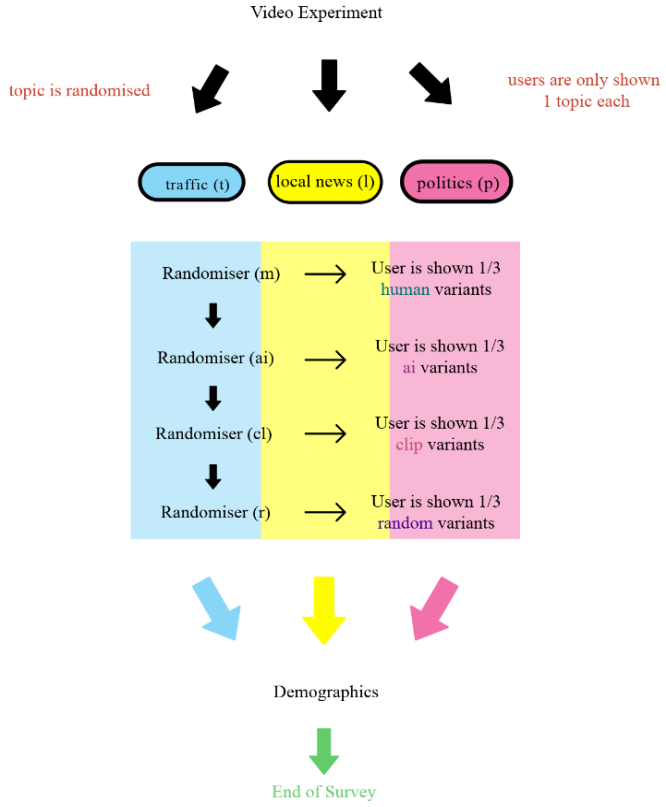


Fig. 1. Survey Protocol outlining randomisation strategy (AI refers to the KIGVI algorithm).

Prior to the experiment portion, the participants received a short introduction before facing a screen-out question that determines whether they agreed to data collection and reside in the greater RheinMain area. Thereafter the participants were asked about their news consumption habits and preferences, how familiar they are with AI in general as well as attitudes towards the technology in general, accounting for both algorithmic and generative AI. The experiment portion was introduced to ensure participants understood the task of evaluating the algorithmic AI technology utilised in the editing process. Matrix questions followed each editing variant in order to assess the quality of a video news clip. A matrix question is the judgement of a specific statement on the Likert-Scale. Eight statements are used in the survey, which are shown in Table I. Response to each question was measured on a six-point Likert-Scale ranging from 1, indicating a strong disagreement to 6 for a strong agreement. An even number of levels was chosen to avoid a neutral position and foster a pro or con decision from the participants.

TABLE I. STATEMENTS BEING JUDGED BY A PARTICIPANT ON A LIKERT-SCALE TO ASSESS THE QUALITY OF A VIDEO NEWS CLIP.

Number	Statement
(1)	The voiceover matches the visual material
(2)	The scenes in the video illustrate all the important information in the program
(3)	The video looks professionally edited
(4)	The video is similar to video news I've seen before
(5)	Aspects of the report seem inconsistent / incorrect
(6)	I will probably remember the video report
(7)	The video makes the report more interesting to me
(8)	The report triggers emotion in me

The Likert-scaled items were inherited from a previous quality survey conducted in [8]. Additional questions such as whether participants noticed technical errors like flickering, black screens, timing errors, etc. accompany the inherited questions. Furthermore, the survey instrument included questions evaluating whether the video offered additional value to the news report or evoked emotions. Sociodemographic data, including age, highest level of education and job status were assessed as factors, measured in categories rather than having participants answer freely. For age specifically, category means were used to calculate means and standard deviations, thereby skewing the data. Gender identity was prioritised over gender assigned at birth, however identities beyond male or female were too low in number to be considered.

D. Experimental Design

Responses of the participants to the experiment portion were accompanied by additional Likert-scaled items asking for general attitudes towards news, news reception, artificial

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

intelligence and AI in news. Again, these items were measured on an even scale from *Strongly Disagree* (1) to *Strongly Agree* (6). The experiment portion was randomised. A first randomisation assigned a category of news per participant while a second randomisation within each category selected one video news clip per editing variant meaning every participant was shown one random editing variant, one human editing variant, one AI editing variant using the KIGVI model and one AI editing variant using the model based on CLIP. The video news clips were embedded in the experiment in a way that the participants were able to view them multiple times to secure the responses. Participants were tasked to evaluate the presented video news clip via the matrix questions. Following this evaluation, a choice was presented whether participants believed a human or an AI to be the editor of the before seen video news clip and how confident they were in their choice via Likert scale. Tracking the time, the participants spent on a specific page aimed to make sure that they at least viewed each video news clip once. The first survey wave gathered a general overview of AI evaluation, news habits and algorithmic performance. The second survey wave tested a potential bias towards AI content by flagging the same videos as Human for group A or AI for group B and evaluated perceived differences in algorithmic performance across both groups. The last survey wave received videos edited by an improved algorithm and tested both algorithmic improvements as perceived by audiences as well as influences of removing the AI label from the evaluation process.

E. Survey Waves

The field work was divided into three survey waves. The employment of consecutive waves was chosen arbitrarily to allow adaptations of the survey instrument between the waves. This was necessary to allow the development of the KIGVI-algorithm in reaction to measuring results from a survey wave. Additionally, the chance to adapt the survey instrument within later waves helped focus on specific topics with more or less intensity. In wave 1 sub survey B, the video experiment portion was pulled up in the survey flow to precede the questions about news habit and co. in an effort to potentially rule out effects of fatigue on the data. With survey wave 2 focussing on a potential bias towards AI tools, participants were shown three variants (human, random and KIGVI algorithm) instead of the previous four that still included the CLIP algorithm, and each subgroup received a different introduction. Wave 2 group A received an introduction saying every single video they would be shown was edited via an algorithm despite being shown the three variants. Wave 2 group B received the opposite introduction; they were told all videos were edited by humans. In wave 3, the improved algorithm [9] was employed for both video experiments, wave 3 group A received the same questionnaire as wave 1 group B with the only distinction being the video clips edited via the improved algorithm while wave 3 group B eliminated every mention of AI or algorithmic tools, therefore once again no longer asking participants on whether the watched videos were considered human-edited or AI-edited. All adjustments are detailed in Table II.

TABLE II. SURVEY ADJUSTMENTS PER WAVE.

Wave	Adjustments
1A	Full survey, general questions ahead of video experiment, featuring variants Human, KIGVI (AI), CLIP and Random
1B	Full survey, video experiment ahead of general questions featuring variants Human, KIGVI (AI), CLIP and Random
2A	Shortened survey, shortened Statement matrix only featuring statements 1, 2, 3 and 5 (see Table I), video experiment ahead of general questions with "all videos are edited by AI" introduction featuring variants Human, KIGVI (AI) and Random
2B	Shortened survey, shortened Statement matrix only featuring statements 1, 2, 3 and 5 (see Table I), video experiment ahead of general questions with "all videos are edited by a human" introduction featuring variants Human, KIGVI (AI) and Random
3A	Identical to wave 1B with evolved KIGVI algorithm replacing previous KIGVI clips, featuring variants Human, KIGVI (AI), CLIP and Random
3B	Shortened survey eliminating every mention of AI in both general questions and video experiment, video experiment ahead of general questions, featuring variants Human, KIGVI (AI), CLIP and Random

IV. FIELD WORK AND PARTICIPANTS

Completing the online survey took participants 12 minutes on average. The questionnaires were distributed via online links hosted on the platform Qualtrics. The selection of participants was outsourced to market research company Horizoom. Horizoom distributed the survey links among their registered community, offering points for completing the survey that can culminate in cash earnings as a reward. The Qualtrics survey includes a captcha to ensure human participants only and flags survey responses that it considered to be completed with or by a bot while respondent IDs were generated to ensure privacy and lock previous survey respondents out of completing future survey waves. This guaranteed new recipient pools for every survey wave. Survey wave 1 ran from December 2024 to April 2025, Survey wave 2 ran throughout July 2025 and the final survey wave 3 ran from November 2025 until mid-December 2025. It is important to recognize that the consecutive survey waves employed similar survey instruments but different survey participants. The survey design is thus cross-sectional and not a panel. The sample sizes of validly completed questionnaires were 794 (wave 1), 722 (wave 2) and 703 (wave 3), resulting in a slightly higher sample size than the aimed for of 750 for the first wave and slightly lower sample sizes for waves 2 and 3 when it came to 100% completion. Within each wave, both sub-surveys had a mostly equal valid sample size.

A. Wave 1

In the sample of 794 individuals, 57.63% of survey participants reported identifying as female. The mean age of participants in Wave 1 was 41.69 (*SD* = 21.75). 45.80% of participants reported having finished their A-levels, 23.79% had one or multiple university degrees and 5.60% completed

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

any vocational training. For job placements, 61.53% of participants reported currently being employed, 11.21% reported being unemployed, 10.57% chose pensioneer as an option while 8.79% considered themselves self-employed.

B. Wave 2

In the sample of 722 individuals, 58.24% of survey participants reported identifying as female. The mean age of participants in Wave 2 was 44.48 (*SD* = 20.42). 41.35% of participants reported having finished their A-levels, 16.76% had one or multiple university degrees and 7.14% completed any vocational training. For job placements, 62.36% of participants reported currently being employed, 5.49% reported being unemployed, 4.26% chose pensioneer as an option while 7.14% considered themselves self-employed.

C. Wave 3

In the sample of 703 individuals, 56.53% of survey participants reported identifying as female. The mean age of participants in Wave 1 was 45.88 (*SD* = 21.68). 51.14% of participants reported having finished their A-levels, 22.22% had one or multiple university degrees and 6.41% completed any vocational training. For job placements, 64.18% of participants reported currently being employed, 8.40% reported being unemployed, 12.96% chose pensioneer as an option while 7.55% considered themselves self-employed.

D. Comparison Between Respondents per Wave

Table III provides an overview on selected socio-demographic characteristics of the respondents´ per survey wave. Although some differences can be identified, a chi-square tests and Kolmogorov–Smirnov tests indicate no statistically significant differences in age and gender distributions across the three survey waves. However, significant differences were observed in education levels between waves ($\chi^2 = 137$; $p < 0.001$). This variation is therefore considered a potential confounding factor in the analytical strategy.

TABLE III. GENDER, AGE & EDUCATION PER WAVE

	Wave 1 *N* = 794	**Wave 2** *N* = 722	**Wave 3** *N* = 703
Gender			
Female	58.23%	58.24%	56.61%
Male	40.23%	41.62%	42.96%
Diverse	1.54%	0.14%	0.43%
Age (mean)	41.69	44.48	45.88
Education Level			
Entry	23.16%	16.62%	18.66%
Medium	69.59%	58.17%	73.36%
High	7.25%	25.21%	7.98%

The particular relevance of the influence of education on media use and media literacy has been empirically proven many times internationally (see, for example [9]). Numerous studies show that formal education is systematically linked to usage patterns, media literacy, and the ability to critically evaluate media content. Against this background, a comparable distribution of educational levels between samples is central to the interpretation of differences in media use. Since the second wave of the survey differs significantly from the other two waves in terms of the distribution of the educational characteristic, direct comparisons can only be interpreted to a limited extent. To avoid distortions due to structural sample differences and to ensure the comparability of the results, the second wave is not considered in the following analysis. The results of the study therefore refer exclusively to the first and third waves of the survey, which are much more consistent in terms of educational structure. In the following we present three sections with different analyses that build upon each other to illustrate influences on the video experiment with more detail.

V. ANALYSIS OF MEDIA USAGE BEHAVIOUR

A. Patterns of Media Consumption

The study revealed differences in the respondents' perception of public versus private media offerings. An overview of the distribution is presented in Table IV. A chi-square test indicates a significant connection between the survey wave and the usage behaviour reported by the respondents ($\chi^2 = 7.08$; $p < 0.05$). Since this effect could be attributed to both a possible time effect and differences in the composition of the respective samples, the decision to use exclusively public or exclusively private media offerings was examined separately in a future analysis step, which is beyond the scope of the present paper, employing binary logistic regression. Specifically, the analysis showed that there is no clear effect of the survey wave or the survey date on the stated choice between public and private media offerings once the level of education is controlled for. In line with the existing literature, however, both models revealed a significant influence of the education characteristic on this decision, confirming the fundamental relevance of education-related differences in media use (Public News: $\chi^2 = 11.21$; $p < 0.01$; Private News: $\chi^2 = 11.47$; $p < 0.01$). In addition, a clear differentiation between user groups emerged: people with a lower level of education significantly more often prefer private media offerings than people with a medium level of education, while the opposite pattern can be observed for public media offerings. For people with a high level of education, however, no clear correlation can be established on the basis of the available analyses. Interestingly, there was no significant temporal effect on the stated choice of public media offerings, while a significant relationship with the time of the survey or the survey wave was observed for private media offerings ($\chi^2 = 5.88$; $p < 0.05$).

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE IV. PREFERANCE TOWARDS PUBLIC OR PRIVATE NEWS AMONGST PARTICIPANTS IN ALL THREE SURVEY WAVES.

	Public News	Private News	Public and Private News
Wave 1	42.69%	12.96%	44.35%
Wave 3	46.76%	8.68%	44.56%

In addition, the influence of gender and age on voting decisions was examined. Age was found to have a fundamental effect on media choice (Public News: $\chi^2 = 9.66$; $p < 0.01$; Private News: $\chi^2 = 11.45$; $p < 0.01$), but this effect varied depending on the type of media offered: while public media offerings were more frequently chosen by younger people, the results indicate that older people tend to use private media offerings. However, a weak significant gender-specific difference was only found in the choice of private media offerings, with a larger proportion of women than men opting for private media offerings ($\chi^2 = 4.09$; $p < 0.1$). Participants were asked to indicate which platforms they use for news consumption, ranging from digital-only sources (news websites, social media, and apps) to more established platforms (television and radio). Most respondents reported using multiple platforms, while substantial subgroups relied exclusively on social media or exclusively on television for news consumption. These patterns were consistent across both survey waves. Regarding video news consumption, an average of 45% of participants reported watching video news daily, while approximately 30% indicated consumption at least once per week. Across waves, women reported daily video news use more frequently than men but also showed higher proportions of very infrequent use (less than once per month). This pattern was observed again in wave 3, although the proportion of men reporting very infrequent use increased. Age-related differences were pronounced: a large share of participants aged 61 years and older reported never watching video news, whereas younger age groups predominantly reported daily or monthly use. Differences by education were less pronounced, though participants with lower educational attainment more frequently reported weekly or monthly use, while those with medium and higher education levels more often reported daily consumption. When asked about satisfaction with the available news content, women more frequently reported being satisfied or mostly satisfied compared to men. By age group, participants aged 41 years and older expressed the highest levels of dissatisfaction, while those aged 31–40 reported the highest satisfaction. In terms of education, respondents with lower and medium educational attainment were less satisfied with the current news offering, whereas those with higher education levels reported greater satisfaction. Regarding familiarity with generative AI, 47% of participants reported having used or currently using AI tools, and approximately 16% indicated that they consume AI-generated content. Around 30% stated that they had no direct contact with AI tools. In light of the general difficulty participants showed in identifying AI-generated or AI-assisted content, this self-reported lack of exposure should be interpreted cautiously, as it may reflect limited awareness rather than actual absence of contact with AI systems.

B. Attitudes towards AI

Attitudes toward the implementation of AI in the media landscape were largely consistent across waves, showing an overall tendency toward critical evaluations (see Table V). However, this result should be interpreted cautiously, as the item was only included in wave 3A and therefore based on a smaller subsample than in previous waves. A chi-square test indicated a significant but weak association between survey wave and response pattern ($\chi^2 = 7.36$, $p < .10$), which may be attributable to differences in the educational composition of the samples. Across waves, respondents more frequently selected the moderate response categories ("rather positive" and "rather negative") than the more extreme positions. While wave 1 included both the highest proportion of clearly negative evaluations and the highest proportion of rather positive attitudes, respondents in wave 3 expressed a generally more critical stance toward the use of AI in media.

TABLE V. EVALUATION OF AI USAGE IN MEDIA CONTEXTS BY PARTICIPANTS IN SURVEY WAVES 1 AND 3.

	Positive	Rather Positive	Rather Negative	Negative
Wave 1	14.27%	34.37%	33.85%	17.51%
Wave 3	10.09%	33.83%	40.95%	15.13%

To gain a better understanding of the underlying mechanisms, the same question was subsequently analysed separately for male and female participants, revealing a stronger differentiation across all three waves (see Table VI). A weak association between gender identity and attitudes toward AI was observed in the analysis of variance (ANOVA; Type I sum of squares = 2.34, $p < .10$; Type III sum of squares = 2.43, $p < .10$). Given the small effect size and the marginal level of statistical significance, these results should be interpreted with caution. Descriptively, male participants in wave 1A expressed slightly more positive attitudes toward AI, whereas female participants tended to evaluate AI more critically. In wave 1B, both genders evaluated AI somewhat more negatively, although the differences were minimal. In wave 3, negative evaluations increased across both gender groups, suggesting a general shift in attitudes rather than a gender-specific effect.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE VI. EVALUATION OF AI USAGE IN MEDIA CONTEXTS BY PARTICIPANTS DIVIDED BY GENDER AND SUB-WAVE.

Attitude towards AI	Gender	
	Female	Male
Wave 1A		
Positive	6.79%	6.53%
Rather Positive	19.58%	13.05%
Rather Negative	21.15%	12.27%
Negative	8.88%	6.53%
Wave 1B		
Positive	7.33%	6.54%
Rather Positive	20.16%	12.83%
Rather Negative	20.42%	10.47%
Negative	8.64%	9.16%
Wave 3A		
Positive	4.02%	5.75%
Rather Positive	17.82%	14.37%
Rather Negative	23.85%	15.52%
Negative	8.33%	5.75%

Attitudes toward AI were additionally examined by education level (see Table VII). For analytical clarity, education was grouped into three categories: low (below A-levels), medium (A-levels, vocational training, or university degree), and high (completed PhD). As with the gender analysis, these results are interpreted descriptively, as the uneven distribution of education levels across waves limits the strength of causal inferences. A significant association between education level and attitudes toward AI was identified using analysis of variance (ANOVA; Type I sum of squares = 4.69, $p < .10$; Type III sum of squares = 5.81, $p < .05$). Given the modest effect sizes and the variation in sample composition across waves, these results should be interpreted cautiously. Descriptively, participants with lower educational attainment evaluated the use of AI in media more critically in wave 1A, whereas participants with medium education levels were more evenly divided and those with higher education tended toward more critical evaluations. In wave 1B, lower and higher education groups expressed slightly more positive attitudes, while the medium education group showed a more negative tendency. In wave 3, attitudes toward AI became more critical overall, particularly among participants with low and medium education levels, while attitudes among the highly educated changed only marginally.

TABLE VII. EVALUATION OF AI USAGE IN MEDIA CONTEXTS BY PARTICIPANTS DIVIDED BY EDUCATION LEVEL AND SUB-WAVE.

Attitude towards AI	Education Level		
	Entry	*Medium*	*High*
Wave 1A			
Positive	3.09%	9.26%	0.95%
Rather Positive	6.89%	22.80%	2.85%
Rather Negative	9.26%	22.80%	1.19%
Negative	4.51%	9.26%	2.14%
Wave 1B			
Positive	5.67%	7.22%	1.03%
Rather Positive	7.99%	22.68%	2.32%
Rather Negative	6.70%	22.68%	1.80%
Negative	5.15%	11.34%	1.03%
Wave 3A			
Positive	0.85%	7.37%	1.42%
Rather Positive	5.67%	24.08%	2.55%
Rather Negative	7.93%	28.33%	2.83%
Negative	3.40%	9.63%	1.42%

In addition to education, attitudes toward AI in media were also examined across age groups (see Table VIII). A significant association between age group and attitudes toward AI was identified using analysis of variance (ANOVA; Type I sum of squares = 25.55, $p < .001$; Type III sum of squares = 26.14, $p < .001$). Across waves, participants aged 61 years and older consistently expressed more negative attitudes toward AI use, whereas participants aged 16–30 tended to evaluate AI more positively, with the exception of wave 3A, in which this pattern was reversed. Participants aged 31–60 generally followed the overall wave-specific trend, expressing more positive attitudes in waves 1A and 1B and slightly more negative attitudes in wave 3A. Although these patterns are robust in terms of statistical significance, they should be interpreted cautiously given the cross-sectional design of the study.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE VIII. EVALUATION OF AI USAGE IN MEDIA CONTEXTS BY PARTICIPANTS DIVIDED BY AGE GROUP AND SUB-WAVE.

Attitude towards AI	Age Group		
	16-30	*31-60*	*61+*
Wave 1A			
Positive	4.77%	8.54%	0.25%
Rather Positive	10.30%	18.34%	3.52%
Rather Negative	7.79%	16.58%	8.79%
Negative	5.53%	6.78%	3.77%
Wave 1B			
Positive	6.46%	6.72%	0.52%
Rather Positive	13.95%	17.05%	2.07%
Rather Negative	9.04%	16.28%	5.94%
Negative	3.36%	9.82%	4.39%
Wave 3A			
Positive	1.14%	7.14%	1.43%
Rather Positive	7.71%	19.43%	5.14%
Rather Negative	8.57%	18.29%	12.29%
Negative	3.43%	9.14%	1.71%

In addition, participants were asked to assess how they perceived attitudes toward AI within their immediate social environment. The prevailing perception was that others held rather positive views of AI technologies, although a notable proportion of respondents indicated uncertainty. Only a small minority assumed that their social surroundings held predominantly negative opinions of AI. Self-reported AI usage patterns largely mirrored attitudinal differences. Across all waves, women reported slightly higher usage of AI tools than men, while consumption of AI-generated content was more evenly distributed across genders. Age-related differences in usage aligned with attitudes: participants in the youngest (16–30) and middle (31–60) age groups reported the highest levels of AI use, whereas those aged 61+ most frequently reported neither using AI tools nor consuming AI-generated content. Differences were also evident across education levels, with participants holding higher educational qualifications more frequently reporting AI usage or consumption, while those with lower educational attainment more often reported no direct contact with AI. Medium education levels likewise showed comparatively high levels of AI usage.

C. Experiences in Video Editing and AI-use in Media

Participants who reported having some or extensive knowledge of video editing were asked to identify potential advantages and disadvantages of using AI tools in media production. The most frequently mentioned advantages included time savings, opportunities for innovation and idea generation, cost reduction, and easier access to media production for novices. Commonly cited disadvantages were the potential replacement of human workers by AI technologies, risks related to copyright infringement, and the reproduction or amplification of biases in AI systems trained on biased data. In addition, participants with a background in video editing were asked to evaluate a set of statements regarding the integration of AI in media production, as presented in Table IX.

TABLE IX. STATEMENTS BEING JUDGED BY A PARTICIPANT ON A LIKERT-SCALE TO ASSESS WISHES TOWARDS THE INTEGRATION OF AI IN MEDIA LANDSCAPES.

Number	Statement
(1)	I want more AI in media landscapes
(2)	I want content that involves AI to be clearly flagged in media landscapes
(3)	I want less/no AI in media landscapes
(4)	I perceive AI in media landscapes as problematic
(5)	I worry that news that were produced with the help of AI tools are not verified upon their factual correctness by a team of human experts

Agreement or Disagreement including standard deviation are pictured in Figure 2. Statement 1 received comparatively low agreement on the Likert scale, indicating a perceived saturation of media landscapes and a general preference for maintaining or reducing the current level of AI use. In contrast, strong agreement was observed for Statement 2, reflecting a clear demand for transparent labelling of AI-generated or AI-assisted content. Statements 3 and 4 were rated on the agreeing side of the scale, suggesting that participants perceive the increasing use of AI in media as potentially problematic or disruptive. Notable agreement was also found for concerns about the absence of human expert oversight in verifying the factual accuracy of news content produced or supported by AI systems.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

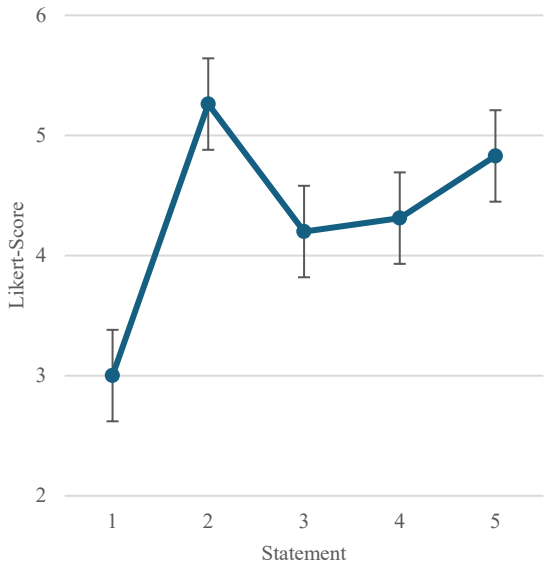


Fig 2. Average scores on the Likert-Scale versus judged statement of Table IX (n = 922 respondents).

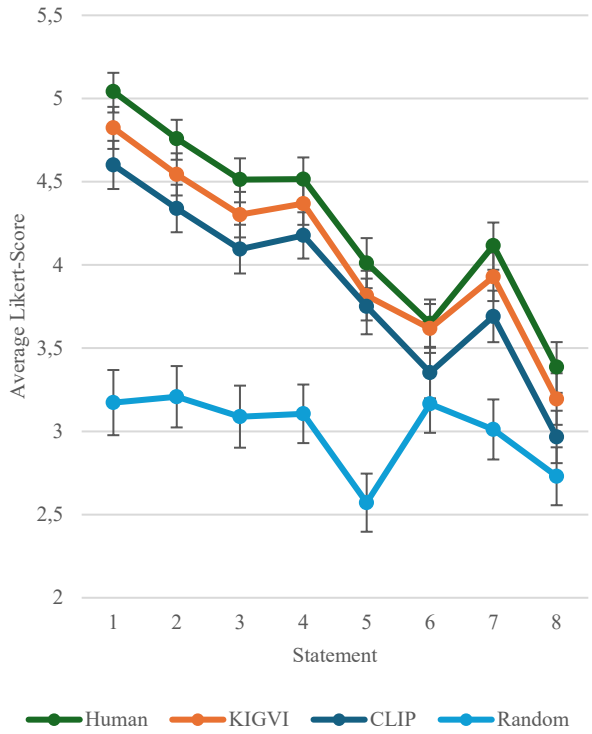


Fig 3. Average scores on the Likert-Scale in wave 1 versus judged statement of Table 1.

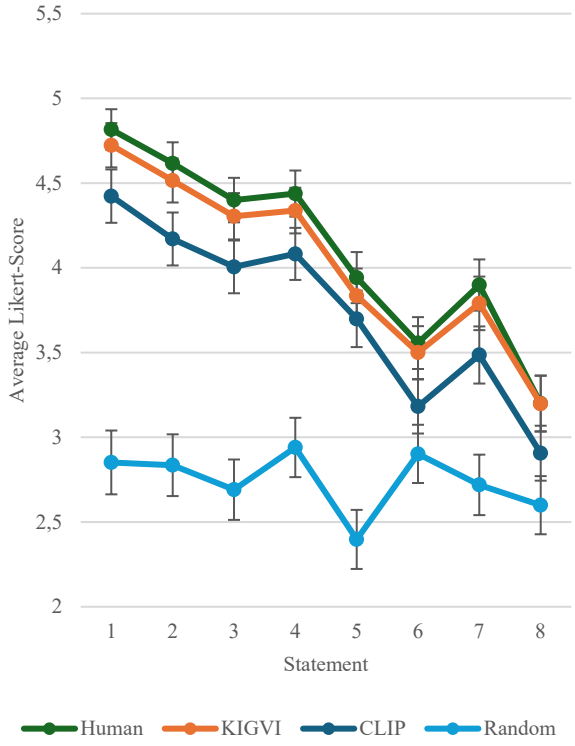


Fig. 4. Average scores on the Likert-Scale in wave 3 versus judged statement of Table 1.

D. Video Experiment

The video experiment assessed participants' agreement with the statements shown in Table I, separately for each editing variant. Evaluations were conducted across three independent survey waves and are displayed with 95% confidence intervals in Figure 3 showing results for wave 1 and Figure 4 showing results for wave 3. The confidence intervals are presented for descriptive purposes only and are not interpreted as evidence of statistical equivalence or difference between variants. Statement 5 was recoded to ensure consistent graphical interpretation; accordingly, lower values on this item indicate a higher number of perceived inconsistencies or errors. Across all waves, the random edit (low anchor) consistently received the lowest ratings across all eight statements, indicating that participants were able to discriminate between clearly divergent quality levels. The CLIP-based edits were evaluated less favourably than both the human edits and the KIGVI-based edits on all statements. This suggests that CLIP-based edits may approximate familiar stylistic conventions without achieving higher perceived quality on other dimensions. Across all variants, a downward trend was observed for memorability (statement 6) and emotional engagement (statement 8). Based on this pattern and to reduce respondent burden, statements 4, 6, 7, and 8 were omitted in wave 2. This decision was made after wave 1 had produced stable results for these items and after the experimental block was moved to the beginning of the questionnaire to minimise fatigue effects. For the same reason, the CLIP condition was also excluded in wave 2. All omitted statements were reintroduced in wave 3 to allow renewed assessment under comparable conditions, as shown in Figure 4. The results for wave 3, presented in Figure 4,

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

largely replicate the patterns observed in Figure 3. The random edit again received the lowest evaluations, while the human and KIGVI edits were rated more favourably across most dimensions. Although higher mean ratings for KIGVI-based edits compared to CLIP-based edits are observed in wave 3, these differences are interpreted descriptively only and do not constitute evidence of algorithmic improvement at this point. Future work will address longitudinal analysis and causal performance comparisons. Across waves, the human edit and the KIGVI edit showed similar evaluation patterns on several indicators, including perceived professionality (statement 3), similarity to existing video news (statement 4), memorability (statement 6), interest generation (statement 7), and emotionality (statement 8), with overlapping confidence intervals. Both variants were also associated with the lowest levels of perceived technical errors or inconsistencies. In addition, participants reported high agreement that the voiceover matched the visual content and that scene selection adequately represented the narrated information. These results indicate perceptual convergence between human-edited and AI-edited variants on multiple quality dimensions.

Another focus of the analysis was the comparison of audience evaluations of CLIP- and KIGVI-edited clips. In wave 1, KIGVI-edited clips received higher average ratings than CLIP-edited clips across all measured dimensions as pictured in Figure 5 and Figure 6.

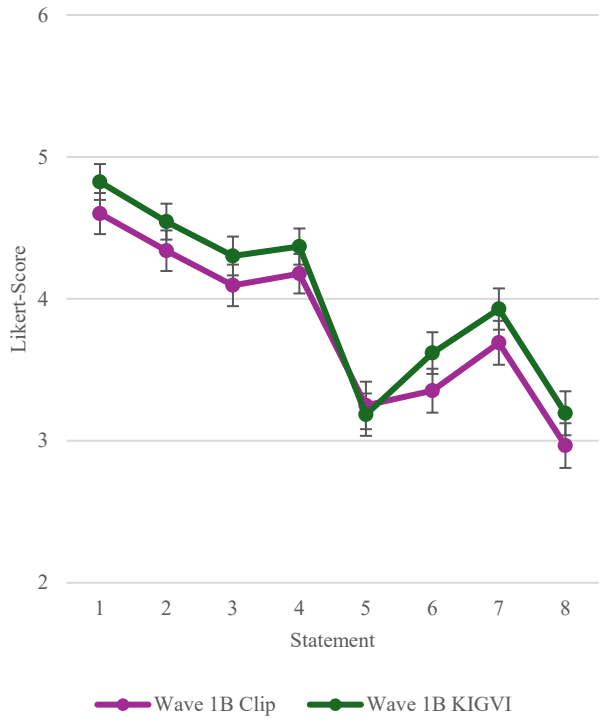


Fig. 5. Average scores on the Likert-Scale of the KIGVI and the CLIP algorithm in wave 1 versus judged statement of Table I.

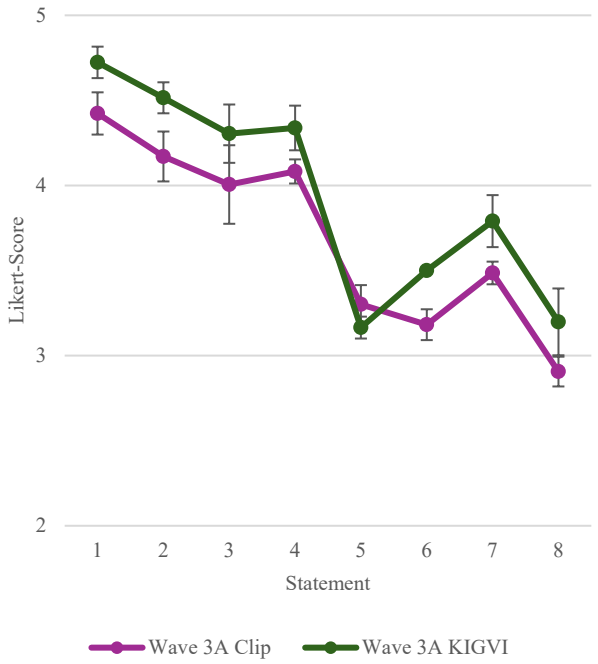


Fig. 6. Average scores on the Likert-Scale of the KIGVI and the CLIP algorithm in wave 3 versus judged statement of Table 1.

These Figures are a closer look at just the two employed algorithms as featured in Figures 3 and 4 above already to allow for a closer identification of performance evaluation across the matrix. While the KIGVI algorithm in version 1 already performs better than the CLIP algorithm across all measures, these margins widen for iteration 2 in wave 3 once more. The results in wave 3 widen especially for statement 4 detailing a high comparability level with existing video news. Memorability (6) scores lower in wave 3 than 1 but still shows a larger margin between the two algorithms. Interest in the topic due to the video edit (7) has the KIGVI algorithm score much higher too and emotionality (8) shows by far the widest gap in evaluation in wave 3 where wave 1 had the algorithm evaluation much closer.

While comparing the evaluation of the performance of the KIGVI algorithm in wave 1 versus wave 3 does not actually produce a significant improvement, looking at the CLIP algorithm evaluation and the random evaluation paints a different picture. Since only the KIGVI algorithm was changed for wave 3 but every other component stayed constant, the other variants can be read as an anchor value too with the significantly lower evaluation of the CLIP algorithm and the random edit whereas the KIGVI algorithm received the same scores in wave 1. However, these differences are reported descriptively and cannot be interpreted as clear statistical evidence of algorithmic improvement due to the cross-sectional design. The identification of the creator of the clips was another consideration within waves 1A, 1B and 3A. In wave 1A, 45.02% of users considered the KIGVI algorithm to be human edited and only 35.32% identified it as AI edited.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

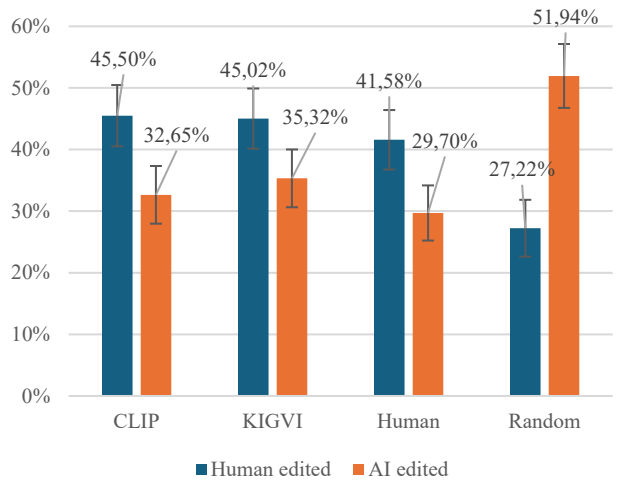


Fig. 7. Percentage of survey participants voting "human creator" (blue), "AI creator" (orange) in wave 1A.

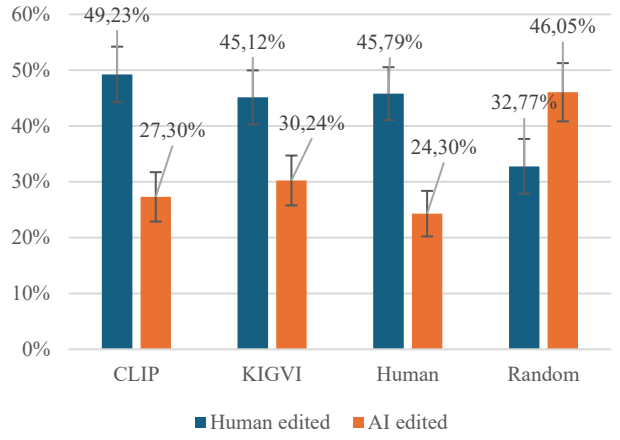


Fig. 8. Percentage of survey participants voting "human creator" (blue), "AI creator" (orange) in wave 2B.

A similar number of participants considered the CLIP algorithm as human edited as well while 29.70% considered the human edit to be done by AI. The random edit was labelled as AI by 51.94% of users. These percentages are shown in Figure 7. In wave 1B, these numbers change slightly as shown in Figure 8. Here, the CLIP algorithm is considered human by 49.23% and the KIGVI algorithm by 45.12%. The random edit is still mostly considered as AI edited despite numbers sinking. The human edit is identified as human more confidently as well.

In wave 3, the random edit was more frequently classified as AI-edited than in wave 1, with nearly 60% of participants selecting AI as its creator. At the same time, confidence in identifying the human-edited clips decreased, while KIGVI-edited clips were more often classified as both human-edited and AI-edited than in earlier waves. Overall, the proportion of participants classifying KIGVI-edited clips as human-edited increased in wave 3. CLIP-edited clips continued to receive

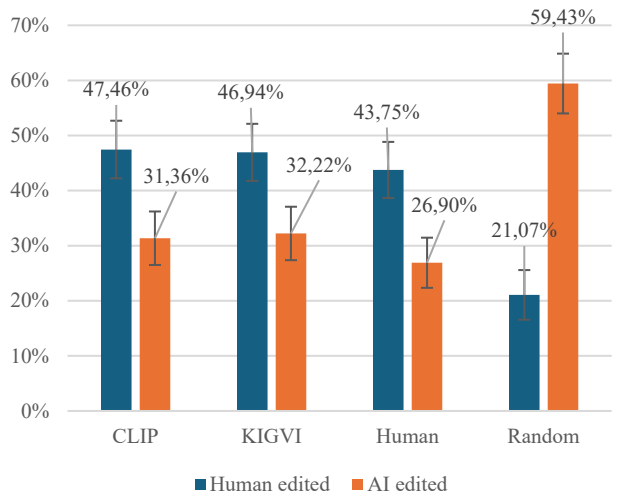


Fig. 9. Percentage of survey participants voting "human creator" (blue), "AI creator" (orange) in wave 3A.

the highest proportion of human-edited classifications, although this proportion was lower than in wave 1B. Additionally, voting behaviour was tested against gender identity excluding options other than male or female due to the small number of responses. This focus was chosen over a testing of educational levels or age since differences were significant within the gender testing. The results for wave 1A are shown in Figure 10. Since Figure 10 does not account for the actual creator of the video, it merely offers insight into voting behaviour by gender displaying a disparity between men and women voting AI as the creator of the clip they were shown with men voting for AI less often and women voting for human slightly more often. The same testing was applied to wave 1B and wave 3A as shown in Figures 11 and Figure 12. In wave 1B, women were even more likely to vote human than men were while in wave 3B, the percentage of women voting for AI climbs higher than in both previous sub-waves. Male participants show a trend towards voting AI as the creator less across all three sub-waves.

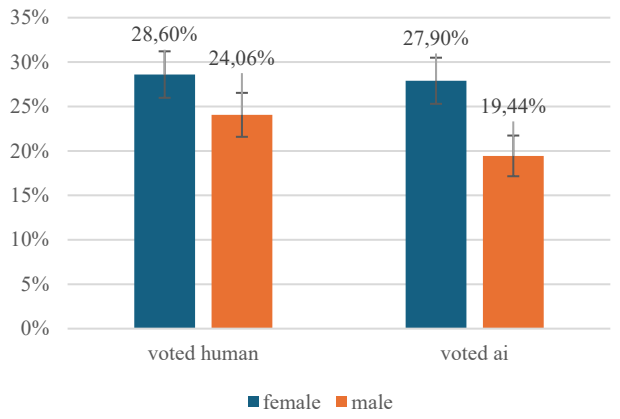


Fig. 10. Percentage of survey participants voting "human creator" (blue), "AI creator" (orange) by gender in wave 1A.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

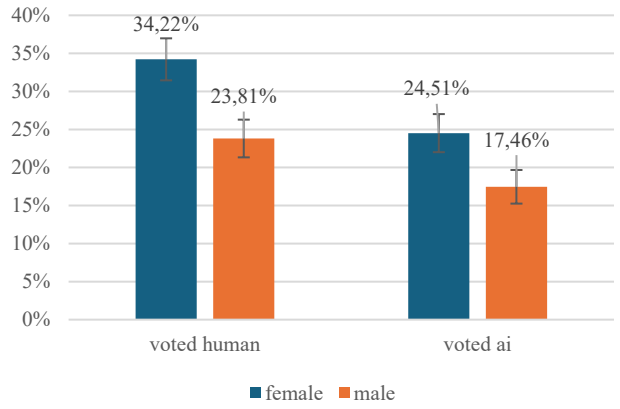


Fig. 11. Percentage of survey participants voting "human creator" (blue), "AI creator" (orange) by gender in wave 1B.

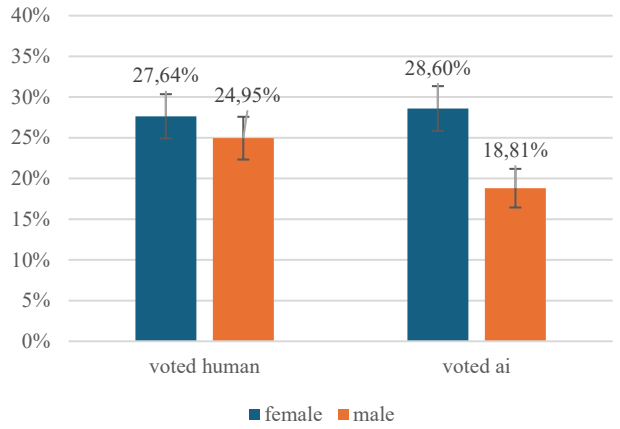


Fig. 12. Percentage of survey participants voting "human creator" (blue), "AI creator" (orange) by gender in wave 3A.

However, the AI selection distinction across gender is significant in all three sub-waves while the human selection is only significant in wave 1B. When users were asked to select whether they paid more attention to the video and its edit or the content of the video, the majority of users (51.07% in wave 1, 49.58% in wave 3) answered in favour of the video content. The question was cut in wave 2.

V. DISCUSSION

A. Patterns of Media Consumption

The observed differences in media choice by age and gender can be plausibly classified in light of existing research on digital inequalities and media preferences. The opposite age effect – with younger people making greater use of public media offerings and older people preferring private offerings – is consistent with findings that media use is not only generation-specific but also influenced by different media socialization and routines (cf. [11]; [12]) The significant gender-specific difference in the use of private media offerings, with women choosing private offerings more frequently, also suggests that media preferences are influenced not only by structural factors such as education and age, but also by gender-specific usage patterns, as described in research on differential media appropriation [13]. Overall, these results underscore that media use must be understood as socially differentiated behaviour and that simple linear explanations are insufficient. The observed patterns of news platform use, and video news consumption (see Table IV) reflect well-established forms of differentiated media use along demographic and educational lines. The broad reliance on multiple platforms, alongside distinct subgroups relying exclusively on social media or television, is consistent with research showing that contemporary news consumption is increasingly hybrid, combining legacy and digital media rather than replacing one with the other [14]. At the same time, the continued existence of single-platform users suggests persistent inequalities in access, routines, and media repertoires [15].

B. Attitudes towards AI

Generally, the attitude towards AI in media industries was evaluated as rather positive and rather negative much more than the definitive stances of positive or negative as shown in Tables V-VIII with later waves recording a slight change in attitudes towards a more negative view generally. Another interesting finding are gender differences within the results for attitudes towards AI and news habits as presented in Table VI or creator identification in Figures 11-12. Even with all three waves featuring a slightly higher number of female participants, a tangible difference can be found with women being more critical of AI but also voting for a human creator more often while men are shown more open towards AI technologies also voting towards a human creator more often. Interestingly in education levels, entry level and higher level often mirror each other's attitudes when it comes to AI in media with a slightly more critical view than medium level education participants as shown in Table VII. Table VIII displays attitudes within age groups showing a rather negative attitude towards AI technologies within people aged 61 and older. This pattern may reflect generational differences in media technology familiarity, as discussed in prior research (ref. [16]).

C. Experiences in Video Editing and AI-use in Media

One of the most prominent findings is that of recipients who have some or a lot of experience in video editing voting in favour of clearly flagged AI content in media instead of having to guess whether AI was utilised in the creation process of anything as shown in Figure 1. Another statement that received high agreement is the worry of news factuality when AI replaces human workforces highlighting the importance of employing human experts to verify AI products when it comes to consumer trust. Taken together, the findings in Figures 2 and Table IX point to a broader ambivalence toward AI in media production. Such ambivalence has been widely documented in research on automation and algorithmic systems, where users simultaneously recognise functional benefits while expressing normative and trust-related concerns ([17]; [18]). The comparatively low agreement with the call for greater AI use is consistent with recent evidence

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

of technology fatigue and growing scepticism toward pervasive automation in everyday media environments [19].

Participants' evaluations of statements comparing AI tools with human expertise further underscore this ambivalence. Responses indicate uncertainty regarding whether AI can achieve results comparable to or better than those of human editors, while also revealing that users care about the origin of media content, suggesting a lack of neutrality toward AI-produced material. This aligns with prior research showing that audiences often judge algorithmic systems not only on performance but also on symbolic and professional expectations associated with journalism ([20]; [21]).

When asked directly about preferences for human versus AI-based video news editing, participants expressed a clear, albeit moderate, preference for human editing without fully rejecting AI-assisted approaches. This pattern is consistent with studies on conditional acceptance of automation, which show that audiences are willing to accept algorithmic support when human oversight remains visible and meaningful [21]. Finally, the slight agreement with the statement that AI tools may reduce emotional bias in decision-making reflects the well-documented tendency to attribute objectivity and rationality to automated systems, even in the presence of broader scepticism [22].

D. Video Experiment

A key finding is that average ratings of human-edited and AI-edited clips were descriptively similar across multiple dimensions. Both were ranked much higher than random editing as shown in Figures 3 and 4. When forced to assign a "human-edited" or "AI-edited" label to the different variants, a majority of the participants assigned the label "human-edited" to both AI edited video news clips and human edited video news clips. Surprisingly, the human edited video news clips were assigned the "AI-edited" label too, as shown in Figures 7-9. All results are displayed with 95% confidence intervals for descriptive purposes. In Figures 3 and 4, the average Likert-Score is illustrated for each of the eight judged statements, (compare Table I), alongside the corresponding 95% confidence intervals. Likert-scale means are reported in rounded form for visual clarity, while all analyses were conducted on the original values.

Overall, AI edited, and human edited video news clips delivered similar average scores on the Likert-scale, recording stronger agreement with the statements except for the inconsistency/incorrectness potential (5) whereas the randomly edited video news clips were scored lower in average for most statements. The more content-related statements (see Table I, statement (1)-(3)) are scored highest in average for AI edited and human edited video news clips. Especially the matching of the voiceover to the visual material (1) received highest scores in average, closely followed by the scene selection illustrating the categories of news well (2). Professionality of the edit (3) and resemblance to other news programs (4) also received high scores with AI-edited and human-edited clips showed similar average ratings on these dimensions. Inconsistencies / incorrectness (5) were scored much lower than random and met more disagreement. Memorability (6) was scored lower in average than other aspects with the smallest difference between all three editing variants. The human editing scores were slightly higher in average when it comes to making the video interesting (7), and emotionality (8) was scored low across all three variants. Figures 5 and 6 present the progression of the video experiment employing both KIGVI iteration 1 in wave 1 and KIGVI iteration 2 in wave 3. With different users participating in wave 3 compared to wave 1, values shifted across the whole matrix bringing the human edit and the KIGVI algorithm edit even closer together. Figures 7-9 especially highlight the lack of congruent creator identification of the presented clips with the KIGVI algorithm passing as human more often than as AI and even the human edit receiving the AI level at around 20% across all three sub-waves. Participants' lack of correct identification of AI and human products was already discussed in reference [23] which aligns well with these findings, widening the research area to moving images from the previous text and still image generation. It is noted that Figures 10-12 display voting behaviour without accounting for whether the selected choice (human-edited or AI-edited) corresponds to the actual editor of the video news clips.

In this paper, we report findings from a cross-sectional survey with 2,219 participants across three independent waves, examining audience perceptions of video news clips edited by AI-based systems and by professional human editors. A total of 48 video news clips is **used** in the study, containing three categories of news, four video news clips per category, and four editing variants of each video news clip with three editing variants in wave 2. Two AI editing variants are applied in waves 1 and 3 with one editing variant receiving a second iteration. In addition, two anchor variants are used. The professionally human edited video news clips serve as a high anchor. Randomly edited video news clips serve as a low anchor. The participants were asked to judge eight statements on a Likert-Scale to assess the quality of a video news clip. Across multiple quality dimensions, ratings of AI-edited and human-edited clips were descriptively similar, with overlapping confidence intervals on several indicators. Participants were generally unable to reliably identify whether clips were edited by humans or AI systems. Furthermore, even when scoring AI edited variants lower, an identification as AI does not always happen.

The results indicate that, based on audience perceptions and the applied evaluation criteria in Table I, AI-edited clips were often rated similarly to human-edited clips, without implying equivalence in technical or professional performance. Lower ratings for memorability and emotional engagement across all variants suggest that these dimensions may be less sensitive to differences in editing style within short-form news formats. Additionally, the low scoring random edit could indicate faithful responses, as the random variant videos were lower in quality and included clips that did not fit the categories of news of the overall videos. Seeing this represented in the matrix questions allows to interpret, that users filled out the video experiment portion genuinely paying attention to the videos. When comparing audience evaluations of CLIP- and KIGVI-edited clips, both variants were more frequently classified as human-edited than as AI-edited. Across the statement matrix, KIGVI-edited clips

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

received higher average ratings than CLIP-edited clips on the measured dimensions, although these differences are reported descriptively and do not constitute evidence of algorithmic performance differences. Park, Oh, and Kim [24] already found social media users to be incapable of distinguishing AI and human made content, in their case generative AI created images and Instagram captions. These findings align with our findings of recipients not being able to distinguish the AI from the human edit. Later samples expressed more critical attitudes toward AI tools in media, which may reflect growing scepticism, although this cannot be interpreted as a temporal change.

VI. LIMITATIONS AND OUTLOOK

The presented results offer a closer insight into the attitudes towards AI in media landscapes and give a first view on the performance of the KIGVI algorithm in the previous and the improved iterations. These findings are unique in showcasing reception of algorithmic editing for video news as opposed to the previous research focus on generative AI usage for texts and images only. The presented findings help define guidelines for public and private news companies on how to use AI technologies in editing, offer a first perspective on recipient acceptance of the technologies and showcase that integrating such strategies can result in acceptance but requires transparent flagging of the displayed content. In the interpretation one must keep in mind that all results are mostly based on bivariate analysis and effects from multiple correlations, multicollinearity, and interactions are not considered. With the significant difference in education levels found for wave 2, these drastically different levels impact all findings especially for survey wave 2 highlighting that survey participants were different across all waves. This in turn presents in the data leading to the choice to mostly exclude the data for wave 2 in the results.

Future research will focus on the improvements of the KIGVI algorithm much more in detail while including further methods of analysis including statistical models and a Difference-in-Differences [25] approach. Another angle to consider would be the potential of oversaturation and the possible fatigue of recipients when it comes to AI tools in general.

ACKNOWLEDGEMENT

The authors thank the German Federal Ministry of Research, Technology and Space, and Qvest GmbH for funding this work via the program FH-Kooperativ 2-2019, contract number 13FH544KX9.

REFERENCES

[1] C. J. Moosburner, D. Quandt, M. Kowald, W. Ruppel, T. Dannewald and M. Narroschke, “Evaluating AI Editing Algorithms for Video News Reporting,” AIMEDIA 2025, pp. 57–61.

[2] N. Newman, “Journalism, media and technology trends and predictions 2023,” Reuters Institute for the Study of Journalism. https://doi.org/10.5287/bodleian:NokooZeEP [retrieved: 2025-06-05]

[3] Y. Hayashi, “Prospects for Revolutionary and Popular AI Technology following the Launch of ChatGPT in 2023,” Electronics, 13(2), 290, pp. 1–2. https://doi.org/10.3390/electronics13020290 [retrieved: 05, 2025]

[4] J. O. Adigun et al., “Enhancing Academic Performance with AI-Powered Tools: A Comparative Study of Adobe Photoshop and Lightroom in Educational Technology Photography,” Journal of Science Research and Reviews, 1(1), pp. 43–48. http://dx.doi.org/10.70882/josrar.2024.v1i1.9 [retrieved: 2025-06-05]

[5] Free online AI image Generator. (n.d.). Canva. https://www.canva.com/ai-image-generator/ [retrieved: 01, 2026]

[6] E. Theophilou et al., “AI and narrative scripts to educate adolescents about social media algorithms: insights about AI overdependence, trust and awareness,” In European Conference on Technology Enhanced Learning, pp. 415–429. Cham: Springer Nature Switzerland. http://dx.doi.org/10.1007/978-3-031-42682-7_28 [retrieved: 2025-06-05]

[7] D. Quandt, P. Altmeyer, W. Ruppel, and M. Narroschke, “Automatic Text-based Clip Composition for Video News,” In Proceedings of the 2024 9th International Conference on Multimedia and Image Processing (ICMIP '24). Association for Computing Machinery, New York, NY, USA, pp. 106–112. https://doi.org/10.1145/3665026.3665042 [retrieved: 2025-06-05]

[8] D. Quandt, W. Ruppel, and M. Narroschke, “News Clip Composition with Mixture-of-Experts”, 2025 IEEE 8th International Conference on Multimedia Information Processing and Retrieval (MIPR), San Jose, CA, USA, 2025, https://www.doi.org/10.1109/MIPR67560.2025.00047 [retrieved: 2026-02-05]

[9] A. Radford et al., “ Learning Transferable Visual Models from Natural Language Supervision,” In Proceedings of the 38th International Conference on Machine Learning, in Proceedings of Machine Learning Research 139, pp. 8748–8763, Available from https://proceedings.mlr.press/v139/radford21a.html [retrieved: 05, 2025]

[10] L. d'Haenens, J. Vissenberg, M. Puusepp, N. Edisherashvili, D. Martinez-Castro, E. J. Helsper, L. Tomczyk, T. Azadi, S. Opozda-Suder, E. Maksniemi, G. Spurava, K. Salmela-Aro, N. Sormanen, S. Tiihonen, T. Wilska, L. Hietajärvi, G. Martínez, N. Larrañaga, M. Garmendia, R. Olveira-Araujo, V. Donoso, M. Pedaste, and D. Sepielak, “Fostering Media Literacy: A Systematic Evidence Review of Intervention Effectiveness for Diverse Target Groups,” Media and Communication, 13(8901). https://doi.org/10.17645/mac.8901 [retrieved: 2026-01-23]

[11] E. Helsper, “The Digital Disconnect: The Social Causes and Consequences of Digital Inequalities,” SAGE Publications Ltd. https://doi.org/10.4135/9781526492982 [retrieved: 2026-01-23]

[12] A. J. A. M. van Deursen and J. A. G. M. van Dijk, „The digital divide shifts to differences in usage. New Media & Society,” 16(3), pp. 507–526. https://doi.org/10.1177/1461444813487959 [retrieved: 2026-01-23]

[13] E. Hargittai and K. Dobransky, “Old dogs, new clicks: Digital inequality in skills and uses among older adults,” Canadian Journal of Communication, 42(2), pp. 195–212. https://doi.org/10.22230/cjc.2017v42n2a3176 [retrieved: 2026-01-23]

[14] N. Newman, R. Fletcher, C. T. Robertson, K. Eddy and R. K. Nielsen, “Digital News Report,” Reuters Institute. Available from: https://reutersinstitute.politics.ox.ac.uk/digital-news-report/2023 [retrieved: 2026-01-23]

[15] U. Hasebrink and H. Domeyer, “Media repertoires as patterns of behaviour and as meaningful practices,” Participations, 9(2),

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

pp. 757–779. Available from: https://www.participations.org/09-02-41-hasebrink.pdf [retrieved: 2026-01-23].

[16] J. Jacobson, L. Z. Chang and R. McEwen, "Aging with technology: Seniors and mobile connections," Canadian Journal of Communication 42(2), pp. 331–357. https://doi.org/10.22230/cjc.2017v42n2a3221 [retrieved: 2026-01-16]

[17] T. Bucher, "If… then: Algorithmic power and politics," Oxford University Press. https://doi.org/10.1093/oso/9780190493028.001.0001 [retrieved: 2026-01-23]

[18] N. Diakopoulos, "Automating the news," Harvard University Press. Available from: https://www.hup.harvard.edu/books/9780674976986 [retrieved: 2026-01-23]

[19] M. Skovsgaard and K. Andersen, „Conceptualizing news avoidance," Journalism Studies, 21(4), pp. 459–476. https://doi.org/10.1080/1461670X.2019.1686410 [retrieved: 2026-01-23]

[20] M. Carlson, "The robotic reporter," Digital Journalism, 3(3), pp. 416–431. https://doi.org/10.1080/21670811.2014.976412 [retrieved: 2026-01-23].

[21] S. C. Lewis, "Journalism in an era of big data," Digital Journalism, 3(3), pp. 321–330. https://doi.org/10.1080/21670811.2014.976399 [retrieved: 2026-01-23]

[22] A. L. Guzman and S. C. Lewis, "Artificial intelligence and communication," New Media & Society, 22(1), pp. 70–86. https://doi.org/10.1177/1461444819858691 [retrieved: 2026-01-23]

[23] J. M. Logg, J. A. Minson and D. A. Moore, "Algorithm appreciation: People prefer algorithmic to human judgment," Organizational Behavior and Human Decision Processes, 151, pp. 90-103, https://doi.org/10.1016/j.obhdp.2018.12.005 [retrieved: 2026-01-23]

[24] J. Park, C. Oh, and H. Y. Kim, "AI vs. human-generated content and accounts on Instagram: User preferences, evaluations, and ethical considerations," Technology in Society, 79, 102705, pp. 10–11. http://dx.doi.org/10.1016/j.techsoc.2024.102705 [retrieved: 2025-06-05]

[25] S. Rothbard, J. C. Etheridge, and E. J. Murray, "A Tutorial on Applying the Difference-in-Differences Method to Health Data," Current Epidemology Reports, 11, pp. 85–95. https://doi.org/10.1007/s40471-023-00327-x [retrieved: 2026-01-16]

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

KAN Control Framework for Continual Learning

Evgenii Ostanin
Toronto Metropolitan University
Toronto, Canada
email: eostanin@torontomu.ca

Nebojsa Djosic
Toronto Metropolitan University
Toronto, Canada
email: nebojsa.djosic@torontomu.ca

Salah Sharieh
Toronto Metropolitan University
Toronto, Canada
email: salah.sharieh@torontomu.ca

Alexander Ferworn
Toronto Metropolitan University
Toronto, Canada
email: aferworn@torontomu.ca

***Abstract*—This study extends the concept of progressively reducing catastrophic forgetting in Kolmogorov-Arnold Networks (KAN) by introducing the KAN Control Framework (KCF). In KCF, the stability-plasticity trade-off is described through a control vector that separates memory-related controls (replay policy, buffer size M, replay intensity ρ) from plasticity-related controls (freezing granularity, freeze ratio k, and importance scoring). Using a two-task Split-MNIST protocol with multi-seed evaluation, replay-only behavior is first mapped over (M, ρ) for standard and balanced replay, where retention improvements are largely driven by rehearsal strength. Progressive spline freezing is then studied as an additional control, and the largest gains are observed in constrained replay scenarios, where memory or replay throughput may be limited, while sufficiently strong replay-only settings can still achieve the lowest forgetting overall. Finally, the way in which the freezing granularity (point-level vs. tensor-level) and scoring rules shape the retention-adaptation trade-off are analyzed and practical guidance is provided for selecting KCF settings under common resource limits. Overall, KCF is intended to make CF configurations in KANs more explicit and reproducible.**

Keywords-Continual Learning; Catastrophic Forgetting; Kolmogorov-Arnold Networks; KAN Control Framework; Spline Freezing; Memory Retention.

I. Introduction

This paper extends a previous publication dedicated to progressive spline freezing for reducing Catastrophic Forgetting (CF) in Kolmogorov-Arnold Networks (KAN) [1]. Continual Learning (CL) is commonly framed as learning from a stream of tasks or shifting data distributions without repeatedly retraining from scratch [2], [3]. In many practical settings, CF can be observed, where after optimization and training on new data, performance in earlier tasks can drop substantially [4]–[6]. This effect is often more problematic when shift is expected (e.g., evolving cyber threats, changing user behavior, or non-stationary sensor streams) and when storage, retraining cadence, or latency constraints make full replay or repeated re-optimization difficult.

A broad family of CL approaches has been proposed to soften interference by limiting updates or reintroducing past information during training. Regularization-based approaches (e.g., Elastic Weight Consolidation and related synaptic-importance methods) penalize changes to parameters estimated to be important for earlier tasks [4], [5]. Gradient-projection and orthogonalization methods restrict the update direction so that interference between tasks is reduced [7]. Architectural strategies add capacity or isolate task-specific subnetworks so that earlier mappings are preserved more directly [8], [9]. Another group of replay-based approaches retain a small buffer of past samples and re-use them with new data so that representations may remain preserved over time [10], [11]. Replay is frequently effective, although its impact tends to depend on the memory budget and on how the buffer is sampled; balanced and uncertainty-guided variants have recently been explored to improve stability when buffers are small [12].

At the same time, KANs have emerged as a structurally distinct alternative to Multilayer Perceptrons (MLPs), replacing fixed node activations with learnable univariate functions placed on edges, commonly parameterized as splines [13]. This shift in the location of the capacity has been associated with favorable trade-offs between accuracy and interpretability in some tasks [13]. At the same time, it also changes where and how plasticity is expressed during sequential training.

Recent studies have started to examine the behavior of KANs under noise and adversarial perturbations, where both robustness opportunities and sensitivity patterns have been reported [14]–[18]. Compared to robustness analysis, however, CL for KANs is still less explored, particularly when reliable retention is needed under limited memory and limited replay throughput. Our prior work introduced progressive freezing strategies for KANs in a CL setting, where subsets of spline parameters are selectively frozen based on importance scores and can be combined with replay. It was observed that freezing can reduce forgetting, but the interaction with replay is not always straightforward: when replay is abundant, freezing may provide only marginal benefit, whereas under constrained replay budgets it can offer more noticeable improvements [1]. These observations motivated a broader framing in which freezing is treated as one of several knobs that may be used to manage stability-plasticity trade-offs.

This extended paper develops that framing into the KAN Control Framework (KCF) for controlled adaptation. The central idea is to represent CL in KANs as a controlled update

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

process governed by a small set of explicit, configurable controls such as replay policy and budget, freezing granularity, freezing fraction, and importance scoring (among others). By making these controls explicit, KCF is intended to support systematic exploration of trade-offs, repeatable comparisons under matched constraints, and decision guidance for selecting configurations when memory and computation are limited. A controlled Split-MNIST benchmark [5] is used as a testbed, where the MNIST [19] (Modified National Institute of Standards and Technology) handwritten-digit dataset is partitioned into two sequential tasks comprising digits 0-4 and 5-9. The multi-seed evaluation is reported so that not only mean behavior but also variability can be inspected. This matters because small differences in configuration may lead to noticeably different retention outcomes, especially in the low-budget regimes that often motivate CL in the first place.

As compared to the original paper [1], this extended paper makes the following contributions:

- KCF: CL in KANs is formalized as a controlled adaptation problem, and a compact control vector is specified so that replay and freezing choices can be expressed in a unified way.
- Reliability under constraints: multi-seed, matched-budget experiments are provided to characterize when progressive freezing is likely to help, and when replay alone may be sufficient, so that freezing can be interpreted as a constraint-aware control rather than a universal improvement.
- Granularity-aware analysis: freezing is compared at different granularities (point-level vs. curve/tensor-level) and across importance scoring choices, highlighting stability-plasticity patterns that appear to be specific to spline-parameterized KAN layers.
- Actionable selection guidance: the empirical findings are translated into practical recommendations for choosing KCF settings when memory and replay throughput are constrained.

This paper is organized in the following way: Section II reviews CL mechanisms and KAN fundamentals outlined in the related work. Section III presents the KCF and the controlled adaptation formulation. Section IV describes datasets, models, and evaluation protocol. Section V reports results and analyses, and Sections VI and VII discuss implications, limitations, and directions for controlled adaptation in non-stationary settings, and provide conclusions and future work directions. With this background in place, related work is reviewed next to position KCF within the broader CL landscape.

II. Related Work

A. CL and CF

CL studies how models may be trained on a sequence of tasks or data distributions while previously learned capabilities are preserved [2], [3], [20]. A recurring difficulty is CF, where optimization on new data introduces interference that degrades performance in earlier tasks [6]. The literature is often summarized into (i) regularization and importance-based constraints, (ii) gradient-constraint or projection methods, (iii) architectural and parameter-isolation strategies, and (iv) replay-based stabilization, with combinations being common [3], [20].

Regularization-based methods reduce forgetting by discouraging changes in parameters that appear to matter the most for the prior tasks. Elastic Weight Consolidation approximates the importance of the parameters via a Fisher-information estimate and penalizes updates along directions that are treated as critical [4]. Related approaches such as Synaptic Intelligence track the contributions of parameters during training and then protect weights that seem to carry prior knowledge [5]. In addition to regularization, gradient-projection techniques aim to prevent destructive interference by constraining the update direction. For example, this can be achieved by enforcing the orthogonality of gradients with respect to stored task-relevant directions [7].

Architectural and parameter-isolation approaches preserve prior knowledge by allocating or masking parameters for new tasks. Progressive networks add new capacity per task, while earlier columns remain frozen [8]. Other methods learn task-conditioned masks over a shared network, which can allow reuse while interference is limited [9], [21]. Across these directions, a common theme is the value of selective update control. Restricting plasticity in the right place may result in a more reliable knowledge retention.

B. Replay and memory-constrained rehearsal

Replay-based methods mitigate forgetting by reusing examples (or features) from previous tasks while training on the current task [10], [11]. Because storage is often limited in practical settings, substantial attention has been paid to how replay effectiveness scales with the memory budget and to sampling strategies that may improve stability under tight constraints [10], [11]. Recent work also explores replay variants that balance representativeness and uncertainty. For example, this is demonstrated through balanced replay combined with uncertainty-guided selection [12]. These results also hint that replay may be more naturally viewed as a configurable control (budget plus policy) than as a single, fixed technique.

C. KANs and stability under distributional stress

KANs introduce a structural alternative to MLPs by placing learnable univariate nonlinearities on edges and aggregating them at nodes, commonly implemented as spline-parameterized functions [13]. Since representational capacity is shifted toward structured function parameters, it is plausible that interference during sequential adaptation will present somewhat differently than in standard MLPs. In parallel with rapid adoption, several studies have examined KAN stability under perturbation-based stressors, including adversarial attacks and noise, to characterize robustness patterns and implementation-dependent effects [14]–[18]. While these studies focus on input perturbations and threat models, they also suggest that KAN behavior can vary meaningfully with

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

architectural and implementation choices, which strengthens the case for explicit adaptation controls in non-stationary settings.

D. Controlled adaptation and selective plasticity for KANs

Despite growing interest in KAN robustness, CL behavior for KANs remains comparatively underexplored. Our prior work investigated CF in KANs on Split-MNIST [5], [19] and introduced progressive spline-freezing strategies, where subsets of spline parameters are frozen based on importance scores and may be combined with replay [1]. The observed pattern is consistent with broader CL experience: replay can be effective, but when replay is constrained, additional update controls may become valuable for retention [1]. This extended paper builds on that foundation by positioning freezing and replay within a unified KCF, where replay policy/budget and freezing granularity/fraction are treated as explicit controls that can be tuned to manage the stability-plasticity trade-off. Because CL systems are increasingly deployed in regulated socio-technical environments, it may also be useful to view controlled adaptation through a broader compliance and governance lens. Saidane and Al-Sharieh provide a compliance-driven framework for privacy and security in highly regulated e-government environments, emphasizing structured controls and accountability [22]. Although the focus here remains on technical controls for model adaptation, that governance-oriented view supports the design of mechanisms that are explicit, configurable, and auditable - qualities that align directly with KCF's emphasis on controllable, inspectable adaptation.

III. KAN Control Framework

In this section, KCF is introduced as a unifying view of CL in KANs as a controlled adaptation process. Building on our earlier progressive freezing results [1] and the spline-parameterized KAN layer structure [13], the stability-plasticity trade-off is made more explicit by exposing a small set of tunable controls that govern (i) what past information is retained and replayed, and (ii) how strongly model plasticity is restricted during adaptation. The aim is not to propose a single best setting, but to provide a compact vocabulary for describing and comparing settings under matched constraints.

A. Controlled adaptation view

A two-phase CL protocol (Task A followed by Task B) is considered, and controlled adaptation is framed as choosing a control vector that determines replay behavior and plasticity restriction during Task B training. Figure 1 summarizes the resulting pipeline. After training on Task A, samples may be stored into a bounded memory buffer. During Task B training, Task B minibatches are interleaved with replayed samples at a prescribed intensity, while plasticity may be restricted by freezing a subset of spline parameters according to an importance score. In the evaluation phase, Task A retention after Task B, Task B acquisition, and aggregate performance are measured.

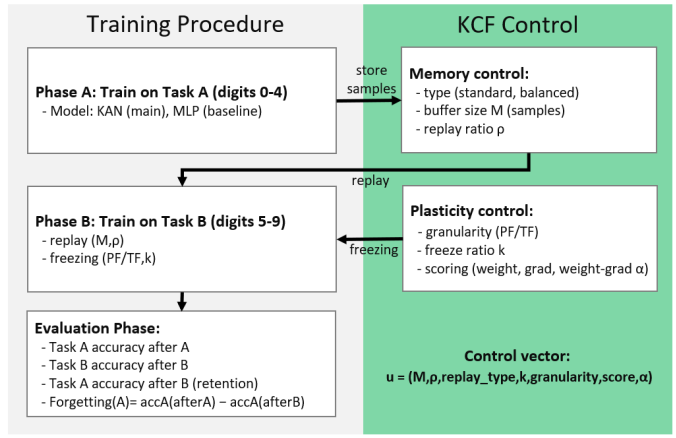


Figure 1. KCF controlled adaptation pipeline for Split-MNIST. KCF separates memory control (replay type, buffer size M, replay ratio ρ) from plasticity control (freezing granularity, freeze ratio k, scoring, and weighting α), yielding an explicit control vector $\mathbf{u}$.

B. Control vector

The KCF control vector is defined as

$$\mathbf{u} = (M, \rho, \texttt{replay_type}, k, \texttt{granularity}, \texttt{score}, \alpha), \quad (1)$$

where M bounds the replay buffer size, ρ specifies replay intensity, `replay_type` selects the sampling policy, and $(k, \texttt{granularity}, \texttt{score}, \alpha)$ governs progressive freezing. In the present setting, freezing targets the spline-parameter components of KAN layers, i.e., the parameters that define the learned edge functions. Two granularities are considered: (i) Point-level Freezing (PF), which freezes individual spline coefficients, and (ii) Tensor-level Freezing (TF), which freezes entire coefficient vectors associated with an input-output feature pair, as introduced in our prior work [1]. The freeze ratio $k \in [0, 1]$ specifies what fraction of coefficients (PF) or curves (TF) are frozen according to an importance score, and α (when used) controls how multiple score components are combined or weighted within the scoring rule.

C. KCF trade-off intuition

KCF is organized around a practical observation: reducing CF typically requires either more rehearsal (larger M and/or higher ρ) or stronger restrictions on plasticity (larger k), and both options come with different cost profiles. Figure 2 illustrates this intuition. High memory and high replay intensity can reduce forgetting but increase storage and training-time overhead; strong plasticity restriction can also prevent forgetting but may limit new-task learning or lead to under-adaptation. In KCF, the goal is often an efficient retention regime (a "sweet spot") where modest replay budgets combined with targeted freezing yield strong retention without relying on large buffers or overly restrictive freezing.

D. Control knobs and operational meaning

Table I summarizes the KCF knobs, their practical meaning, expected effects on forgetting and adaptation, and the main

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

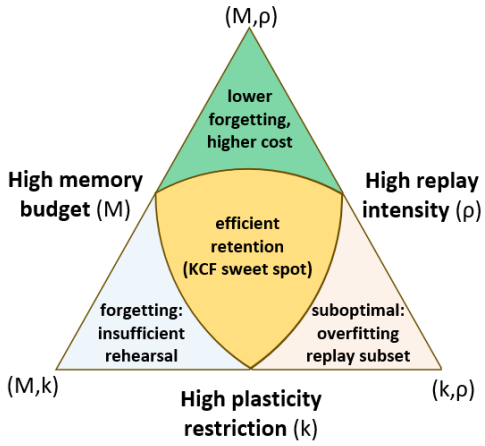


Figure 2. Conceptual KCF trade-off between memory budget (M), replay intensity (ρ), and plasticity restriction (k). The goal is efficient retention: low forgetting at controlled cost.

cost implications. This explicit mapping is central to KCF. CL configuration is treated as a constrained design choice, where $\mathbf{u}$ is selected to balance retention, learning, and operational cost.

E. Outcome measures for control selection

Given $\mathbf{u}$, controlled adaptation is evaluated using retention and acquisition measures that are standard in CL [3], [20]. In the two-task Split-MNIST protocol [5], [19], Task A accuracy after Task A (initial learning), Task B accuracy after Task B (new-task acquisition), Task A accuracy after Task B (retention), and forgetting (the drop in Task A performance after learning Task B) are reported [1]. In later sections, these metrics are used to compare KCF configurations under matched constraints (fixed M and ρ) and to identify regimes where progressive freezing may provide the largest benefit relative to replay alone.

IV. Methodology

A. Task protocol: Split-MNIST sequential learning

CL is evaluated on the Split-MNIST two-task protocol [5], [19]. Task A contains digits $\{0, 1, 2, 3, 4\}$ and Task B contains digits $\{5, 6, 7, 8, 9\}$. Training proceeds in two rounds, and the same model instance is carried forward between rounds (i.e., parameters are updated continuously rather than reinitialized).

In Phase A (the first training round), the model is trained only on Task A until convergence under the chosen training setup, and Task A accuracy after Phase A ($A1$) is recorded. Phase B (the second training round) then continues training on Task B under a selected KCF control vector. After Phase B, Task A accuracy is re-evaluated ($A2$) to quantify retention, and Task B accuracy is also recorded to capture post-adaptation performance on the new task. Forgetting is measured as the drop in Task A performance induced by Phase B training, computed as $A1 - A2$ (so larger values indicate greater forgetting, while values closer to zero indicate stronger retention).

TABLE I
KCF control knobs and meaning.

Param.	Meaning	Impact
M	Replay buffer size (stored samples)	Retention: more coverage → less forgetting (diminishing returns at high M). Cost: storage + I/O overhead.
ρ	Replay intensity during Task B	Retention: more rehearsal → less forgetting. Cost: extra training steps → higher time/compute.
`type`	Replay policy (standard vs. balanced)	Retention: balanced sampling may improve class coverage when M is small. Cost: minor sampling/stratification overhead.
k	Freeze ratio (fraction frozen)	Retention: higher k can reduce interference. Adaptation: too high k may limit Task B learning. Cost: low runtime overhead (mainly masking).
`gran.`	Freezing granularity (PF vs. TF)	Retention/adaptation: PF protects fine-grained coefficients; TF preserves structured curves; trade-offs can be regime-dependent. Cost: similar overhead for both.
`score`	Importance scoring (w, g, w×g)	Behavior: determines which parameters/curves are protected; may affect stability and variance. Cost: small bookkeeping to compute scores.
α	Score mixing weight (if used)	Behavior: tunes sensitivity to magnitude vs. gradient signal (stability vs. plasticity). Cost: negligible.

Figure 3 illustrates the baseline sequential protocol in which Phase B training is performed on Task B without replay and without any freezing. In this setting, the dashed arrow denotes that the same model continues to be optimized on the new task, but Task A samples are no longer available during Phase B. The metric $A1$ is computed immediately after Phase A, and $A2$ is computed after completing Phase B; the difference $A1 - A2$ summarizes the retention loss attributable to task-incremental training.

Figure 4 illustrates replay-based CL. After Phase A, a bounded replay buffer is populated with at most M Task A samples (selected according to the replay policy used in the experiment, e.g., standard or balanced replay). During Phase B, Task B optimization is performed while replayed Task A samples are mixed with Task B minibatches at intensity ρ. In this view, (M, ρ) jointly control the amount of past information reintroduced during adaptation: M limits the available memory of prior data, while ρ determines how strongly that memory is rehearsed during Phase B updates.

Finally, Figure 5 shows the replay+freezing variant used for controlled adaptation in KANs. In addition to replay with buffer size M and intensity ρ, Phase B updates are restricted by a freeze mask that reduces plasticity by preventing updates

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

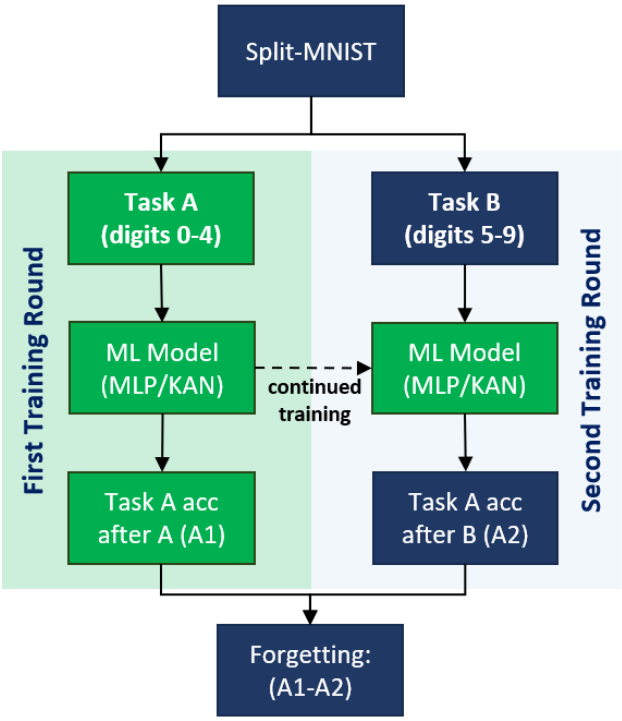


Figure 3. Baseline CL protocol on Split-MNIST (no replay, no freezing). Forgetting is measured as the drop in Task A performance after training on Task B ($A1 - A2$).

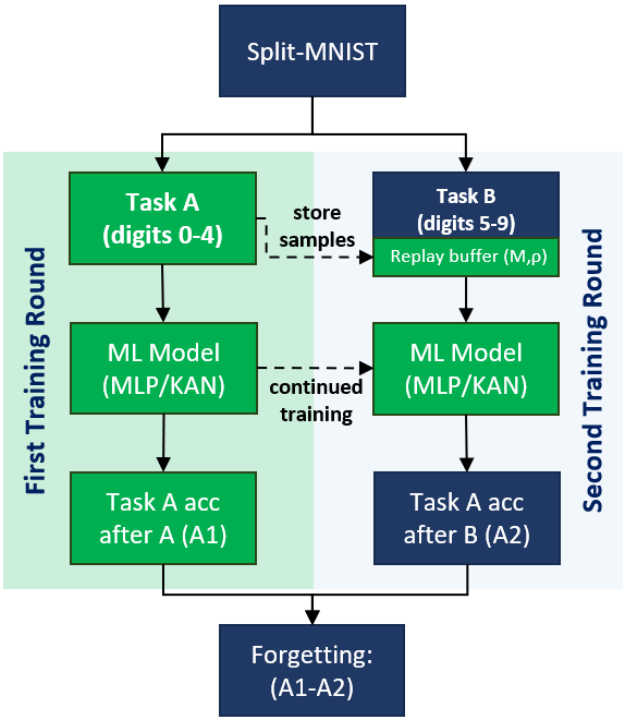


Figure 4. Replay-based CL. After Phase A, samples are stored into a bounded buffer (M). During Phase B, replayed samples are interleaved with Task B training at intensity ρ.

to a selected subset of spline parameters. The freeze ratio k specifies the fraction of parameters to be frozen, and the freezing granularity (PF/TF) determines whether freezing is applied at the point level or at a coarser tensor/curve level as discussed later. In this configuration, replay contributes to retention by reintroducing Task A samples, while freezing constrains which parts of the model are allowed to change during Task B training, thereby shaping the retention-adaptation trade-off.

B. Models: MLP baseline and KAN main model

A standard MLP baseline is compared against a KAN model used as the primary architecture under KCF. Both models operate on 28×28 grayscale images flattened to 784 input features and output 10 logits. The focus is not on optimizing architectures, but on comparing adaptation behavior under fixed, simple models.

a) MLP baseline: The MLP uses a single hidden layer with 128 neurons and fixed activations on nodes (Figure 6). This baseline represents the standard fully-connected feed-forward setting that is often used to study CF in simple benchmarks.

b) KAN model: The KAN replaces fixed node activations with learnable univariate nonlinear functions on edges (Figure 7), parameterized as spline-based functions as in prior KAN formulations [13]. The architecture is configured as [784, 128, 10]: one hidden layer of 128 nodes between 784 flattened inputs and 10 output logits, using cubic B-spline edge functions (spline order $p = 3$, grid size $G = 5$) with a SiLU residual activation, yielding $G + p = 8$ spline coefficients per input–output feature pair and approximately 1.02M parameters in total. In this CL setting, plasticity is primarily governed through the spline parameterization, which allows targeted freezing strategies [1].

C. Memory control (replay control)

As outlined in Table I, replay is governed by $(M, \rho, \texttt{replay_type})$:

- Buffer size M: the maximum number of samples stored after Phase A.
- Replay intensity ρ: the relative rate at which replayed samples are reused during Phase B (i.e., the fraction/ratio of replayed batches or samples injected alongside Task B training).
- Replay type: standard (uniform sampling from the buffer) versus balanced (class-balanced sampling *within the replay buffer*), plus none (no replay).

Task B minibatches are sampled normally; `standard` vs. `balanced` affects only how replay samples are drawn from the buffer. These controls reflect a common operational constraint: retention improvements may need to be achieved under strict limits on stored data and replay throughput. In that scenario, the same change in ρ can sometimes matter more than a modest change in M, since ρ directly increases how often older data is revisited during optimization.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

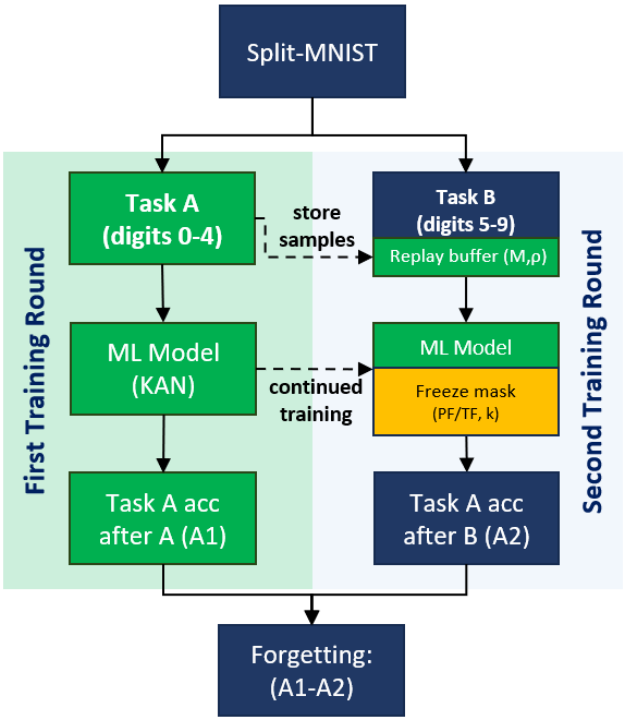


Figure 5. Replay + progressive freezing. During Phase B, replay is combined with a freeze mask that restricts plasticity by freezing a fraction k of spline parameters at a selected granularity (PF/TF).

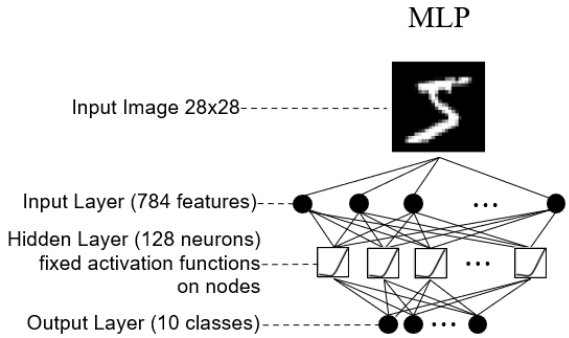


Figure 6. MLP baseline schematic. Fixed activation functions are applied at nodes in a standard fully-connected network.

D. Plasticity control (progressive freezing)

Plasticity control is implemented by freezing a subset of spline parameters during Phase B. Two freezing granularities are considered (Figure 8):

- TF: whole spline "curves" are frozen (blocks associated with an input-output feature pair).
- PF: individual spline coefficients are frozen (fine-grained points) within curves.

A freeze ratio $k \in [0, 1]$ specifies the fraction of curves (TF) or coefficients (PF) to freeze, selected by an importance score. Multiple scoring rules are evaluated: weight, grad, and

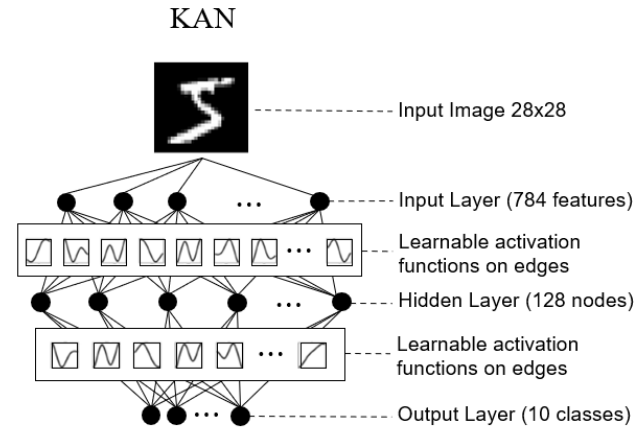


Figure 7. KAN architecture schematic. Learnable activation functions are placed on edges, enabling structured control of functional parameters.

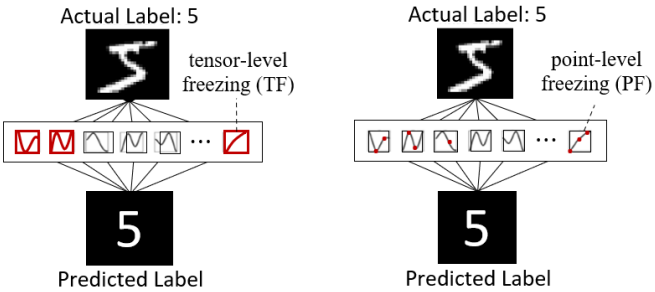


Figure 8. Freezing granularity intuition. TF freezes whole spline curves (structured blocks), while PF freezes individual spline coefficients (fine-grained points) within curves.

weight_grad (weight-gradient product), consistent with the progressive freezing formulation in [1]. TF tends to enforce structured restrictions that preserve entire functional subcomponents, while PF provides finer control that may protect local parameter importance at higher resolution. Neither option is assumed to be universally better; rather, their behavior is treated as part of the control space.

E. Training configuration and evaluation metrics

All configurations are trained for a fixed number of epochs in Phase A and Phase B with the same optimizer and learning rate, and are repeated across multiple random seeds. Task A accuracy after Phase A ($A1$), Task A accuracy after Phase B ($A2$), Task B accuracy after Phase B, and forgetting are reported:

$$\text{Forgetting}(A) = A1 - A2. \tag{2}$$

Aggregate accuracy after Phase B is also reported when overall performance comparisons are needed.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE II
FULL EXPERIMENT GRID (CONTROLS AND VALUES).

Control	Values
M	{50, 100, 200, 500}
ρ	{0.05, 0.1, 0.2, 0.3}
replay_type	{none, standard, balanced}
granularity	{TF, PF}
k	{0.0, 0.05, 0.1, 0.25, 0.5, 0.75}
score	{weight, grad, weight_grad}
seeds	{42, 27, 101}

F. Full experiment grid

Table II lists the complete configuration grid explored in this work, comprising 270 experiments in total (including three random seeds per configuration). The grid spans replay budgets (M), replay intensities (ρ), replay policies, freezing granularities and ratios, and importance scoring rules. To keep a controlled, grid-level study of KCF knobs under a fixed computational budget, a deliberately low-epoch training protocol is used (3 epochs for Phase A and 3 epochs for Phase B) across all configurations. This setting is not intended to maximize absolute accuracy. Instead, it tends to amplify interference effects and provides a repeatable stress-test where differences attributable to replay intensity (M, ρ) and plasticity restriction $(k, \texttt{granularity}, \texttt{score})$ can be seen more clearly under matched training time. Using the same epoch budget for every run also helps reduce the chance that retention gains are simply artifacts of longer optimization, and it enables a fairer comparison between replay-only and replay+freezing regimes when replay throughput is itself a cost driver. The conclusions in this paper are therefore framed as trends within this controlled Split-MNIST protocol [5], [19]: replay remains the dominant lever for retention, while progressive freezing tends to provide its largest benefit in constrained replay regimes by improving retention efficiency under fixed compute.

V. RESULTS

Multi-seed results (mean $\pm$ std) are reported for the Split-MNIST CL protocol described in Section IV. The objective is to evaluate KCF under matched constraints and to characterize when progressive freezing appears to provide reliable retention gains beyond replay alone. Some variability across seeds is expected, so emphasis is placed on consistent trends rather than on single-run outcomes.

A. Baseline catastrophic forgetting without controls

The baseline sequential training setting (no replay, no freezing) is considered first. In Figure 9, both KAN and MLP learn Task A well after Phase A, and Task B is learned after Phase B, yet Task A performance collapses after Phase B, yielding near-complete forgetting (Task A accuracy after B ≈ 0). The two-task split therefore induces strong interference and can serve as a compact stress-test for controlled adaptation.

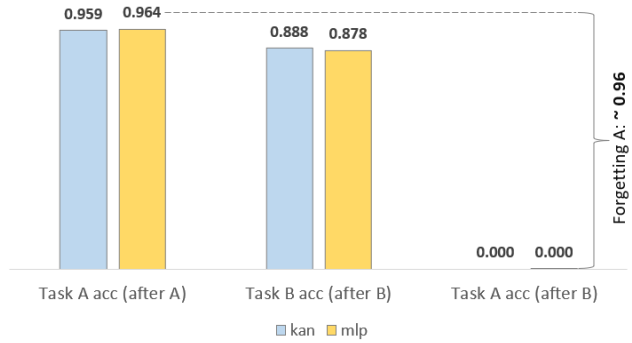


Figure 9. Baseline sequential training (no replay, no freezing). Both models learn Task A and Task B in isolation, but Task A accuracy after training on Task B collapses, indicating near-complete CF.

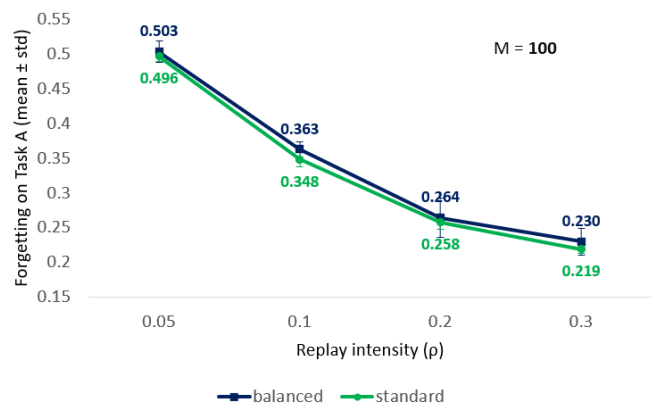


Figure 10. Replay-only results at $M = 100$: forgetting on Task A decreases as replay intensity ρ increases. Standard and balanced replay show similar trends, with small differences that depend on ρ.

B. Replay establishes a controllable retention baseline

Replay-based rehearsal provides the primary memory control in KCF. In Figure 10, increasing replay intensity ρ substantially reduces forgetting for a fixed memory budget ($M = 100$). Across the tested range, forgetting decreases as replay intensity increases, which suggests that rehearsal strength is a major driver of retention in this benchmark scenario. Differences between replay policies (standard vs. balanced) are comparatively modest at this budget, although small regime-dependent differences can be seen at particular ρ values.

C. Replay performance landscape across budgets and intensities

To better understand replay-only behavior, the retention and forgetting landscape over (M, ρ) is visualized for both replay policies. Figures 11–14 summarize replay-only outcomes as heatmaps. Two patterns are observed repeatedly. First, increasing either M or ρ improves retention (Task A accuracy after Phase B) and reduces forgetting, with the best

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

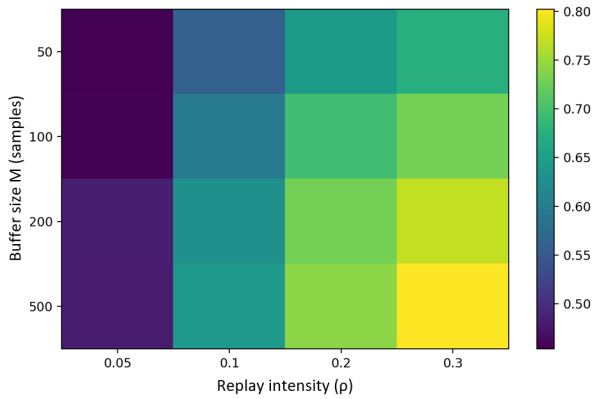


Figure 11. Replay-only retention landscape (balanced replay): Task A accuracy after Phase B (mean) as a function of memory budget M and replay intensity ρ.

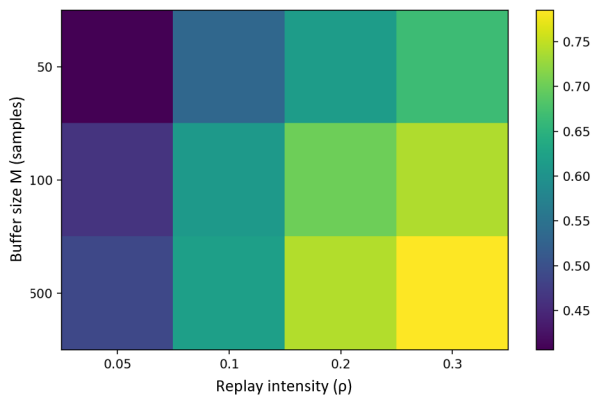


Figure 12. Replay-only retention landscape (standard replay): Task A accuracy after Phase B (mean) as a function of M and ρ.

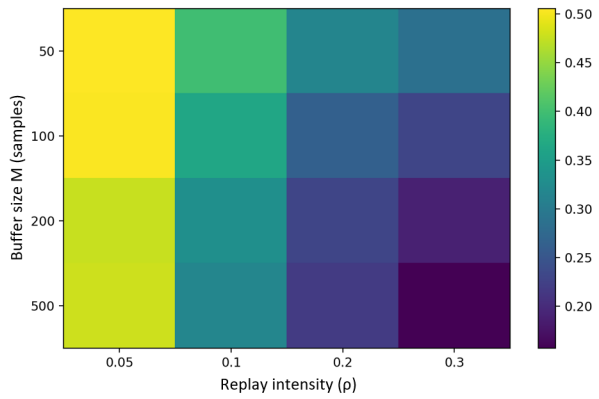


Figure 13. Replay-only forgetting landscape (balanced replay): forgetting on Task A (mean) as a function of M and ρ. Lower values indicate better retention.

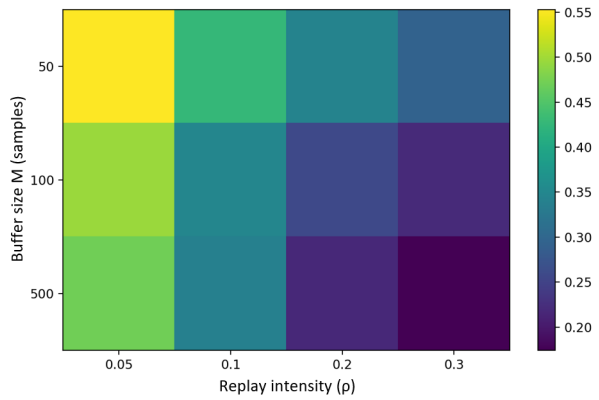


Figure 14. Replay-only forgetting landscape (standard replay): forgetting on Task A (mean) as a function of M and ρ. Lower values indicate better retention.

outcomes occurring when both are large. Second, for a fixed M, increasing ρ often yields a stronger improvement than increasing M by a comparable step, which is consistent with replay throughput being a particularly influential lever in this setting.

Balanced and standard replay lead to similar qualitative landscapes, although small differences can appear in low-budget corners. When both M and ρ are small, both policies remain in a high-forgetting regime. As budgets increase, retention improves and the two policies tend to converge. These corners are also where complementary controls, such as freezing, may matter most.

D. KCF gains under constraints: freezing helps when replay is limited

A central hypothesis of KCF is that progressive freezing may be most useful under constrained replay budgets, where replay alone does not fully prevent forgetting. Figure 15 compares replay-only and replay+KCF (replay plus progressive freezing) in two constrained operating points, alongside an unconstrained reference where replay-only is allowed to choose the best (M, ρ). In the constrained settings, adding KCF reduces forgetting relative to replay-only, yielding a consistent retention gain. In contrast, the unconstrained replay-only best-case provides the lowest forgetting overall, which suggests that freezing is not necessarily meant to beat "unlimited replay". Instead, it may help improve retention more efficiently when memory or replay throughput is capped.

E. Effect of plasticity restriction k and importance scoring

Next, the plasticity restriction knob k is examined together with the importance scoring rule. Figures 16 and 17 report forgetting as a function of k for TF and PF under a fixed replay configuration (standard replay, $M = 100$, $\rho = 0.2$). Across both granularities, increasing k tends to reduce forgetting, although the magnitude of improvement and stability across seeds can depend on the scoring rule. In this regime, gradient-based and combined (weight$\times$grad) scoring generally yield more favorable forgetting profiles than weight-only scoring, which suggests that update-sensitivity signals may be helpful when selecting parameters to protect during adaptation.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

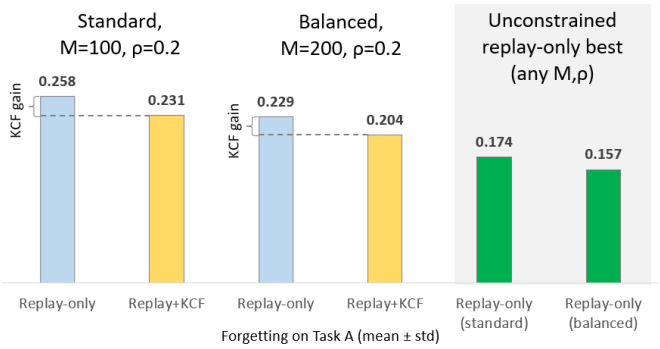


Figure 15. Constrained vs. unconstrained regimes. Under constrained replay budgets, adding KCF (replay + progressive freezing) reduces forgetting relative to replay-only. Under unconstrained replay (best (M, ρ)), replay-only achieves the lowest forgetting, supporting KCF's role as a constraint-aware control.

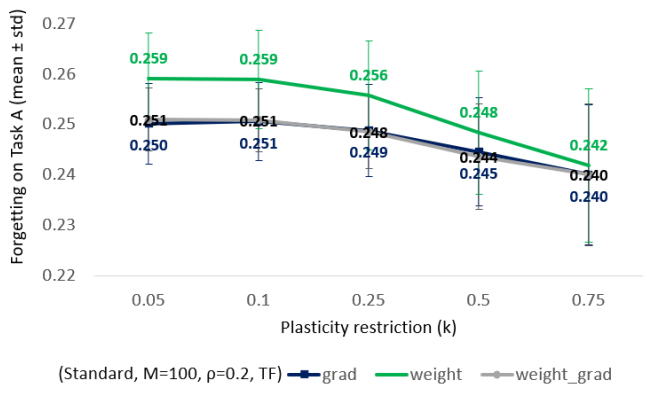


Figure 16. Forgetting vs. plasticity restriction k under TF (standard replay, $M = 100$, $\rho = 0.2$). The scoring rule influences the retention trajectory; gradient-informed scoring yields consistently lower forgetting in this regime.

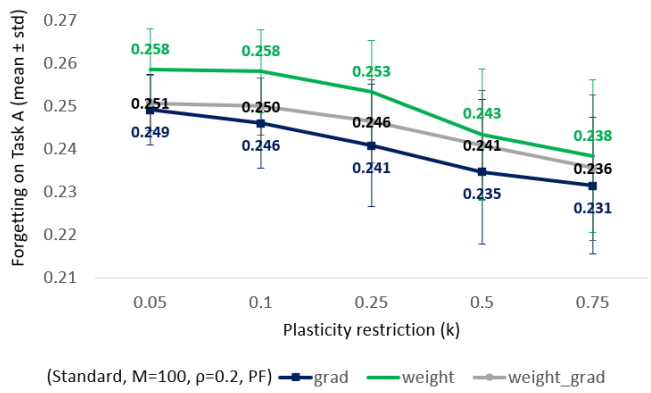


Figure 17. Forgetting vs. plasticity restriction k under PF (standard replay, $M = 100$, $\rho = 0.2$). PF shows a similar monotonic forgetting reduction with larger k, with differences across scoring rules.

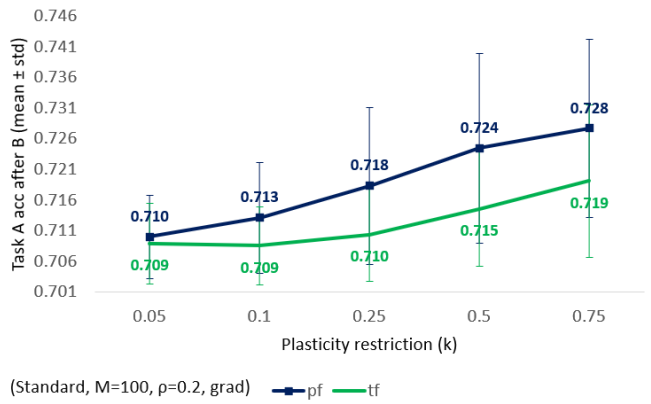


Figure 18. Task A retention (accuracy after Phase B) vs. plasticity restriction k under standard replay ($M = 100$, $\rho = 0.2$, grad). PF and TF both improve retention with larger k, with PF showing a modest advantage at higher k in this regime.

F. Granularity trade-offs: PF vs. TF impacts retention and new-task learning

KCF includes freezing granularity as a structural control: TF freezes whole spline "curves", whereas PF freezes individual spline coefficients. This trade-off is evaluated by comparing retention (Task A accuracy after Phase B) and new-task acquisition (Task B accuracy after Phase B) as k varies.

Figures 18 and 19 report Task A retention after Phase B for standard ($M = 100$) and balanced ($M = 200$) replay settings (both with $\rho = 0.2$ and gradient-based scoring). In both regimes, retention improves with larger k, and PF often yields slightly higher Task A retention than TF at larger k. This is consistent with PF providing finer-grained protection for previously learned function components, although the size of the gap varies across regimes.

Stronger plasticity restriction can also reduce Task B acquisition. Figures 20 and 21 show Task B accuracy after Phase B as k increases under the same settings as above. A mild but consistent downward trend is visible as k increases, which reflects the expected stability–plasticity trade-off: protecting prior knowledge may limit how fully the model adapts to the new task. In some regimes, TF preserves Task B performance slightly better than PF at higher k, which is consistent with TF applying more structured constraints than freezing many individual points.

G. Summary of empirical takeaways

Across the full experiment grid, four recurring patterns can be summarized: (i) without controls, both KAN and MLP exhibit near-complete CF; (ii) replay strength (through M and especially ρ) is the dominant driver of forgetting reduction and defines a smooth performance landscape over (M, ρ); (iii) progressive freezing contributes most under constrained replay budgets, where it can deliver more efficient retention gains relative to replay-only; and (iv) the KCF knobs expose a concrete stability–plasticity trade-off: increasing k tends to improve retention but can mildly reduce new-task acquisition, while PF/TF and scoring rules shift where a favorable balance may occur for a given constraint regime. These trends moti-

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

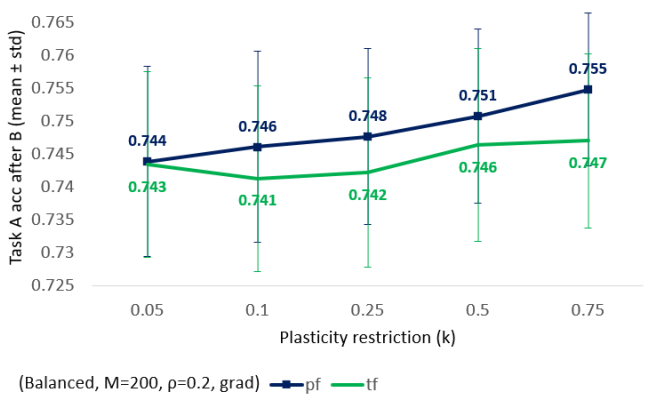


Figure 19. Task A retention vs. plasticity restriction k under balanced replay ($M = 200$, $\rho = 0.2$, grad). Retention increases with k for both PF and TF; the relative gap depends on k and the replay regime.

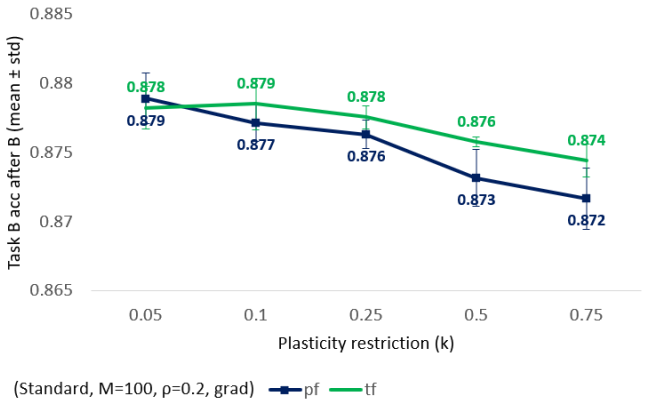


Figure 20. Task B accuracy after Phase B vs. k under standard replay ($M = 100$, $\rho = 0.2$, grad). Increasing k restricts plasticity and can slightly reduce new-task performance; PF and TF exhibit different sensitivity profiles.

vate the selection guidance and cost-of-adaptation discussion developed in the next section.

VI. Discussion

In KCF, the stability-plasticity trade-off is made operational: rather than treating replay or freezing as standalone techniques, a compact control vector is exposed that can be tuned under explicit resource constraints. The Split-MNIST experiments provide a small but fairly transparent setting in which these controls can be inspected, and several implications can be drawn for both CL research and for practitioners who may be considering KANs in dynamically changing environments, where data distributions may shift over time.

In many settings, sufficiently strong replay may be effective on its own. When replay is limited, other controls may become more important, and the practical question shifts to which choices offer the best return under a given budget.

Across the replay-only landscapes, increasing rehearsal strength through (M, ρ) consistently reduces forgetting. This is in line with rehearsal-based CL and supports the view that replay defines a strong baseline in this benchmark regime

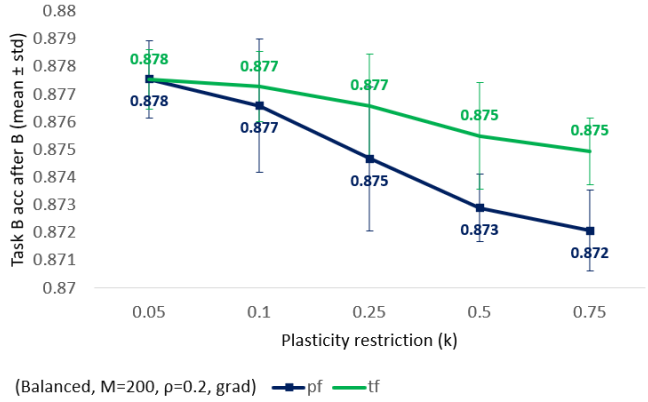


Figure 21. Task B accuracy after Phase B vs. k under balanced replay ($M = 200$, $\rho = 0.2$, grad). As k increases, Task B accuracy decreases modestly, highlighting the stability–plasticity trade-off under stronger freezing.

[10], [11]. Replay, however, is not free. Storage costs scale with M, training-time overhead tends to scale with replay throughput ρ, and operational policies may also place limits on what can be retained (data minimization) or how long it can be stored (retention windows). KCF is therefore best interpreted as a constraint-aware mechanism: when replay cannot be increased arbitrarily, progressive freezing can deliver measurable retention gains without requiring large buffers or high replay intensity. In that sense, replay is not replaced; rather, it may be made more efficient in the low-budget corners where CF is otherwise severe.

The plasticity control k exhibits the expected stability–plasticity behavior: stronger freezing reduces forgetting but can mildly reduce Task B performance. This aligns with a general CL principle that limiting representational drift may help retention at the cost of flexibility [3]–[5]. The contribution of KCF is mainly practical. Instead of optimizing only for minimal forgetting, k (together with replay controls) can be chosen to meet an operational objective. In some deployments, a small drop in new-task accuracy may be acceptable if retention is improved substantially. In others, fast adaptation may be prioritized and a smaller k (or replay-only) might be preferred. Neither choice is automatically correct, and it is often the constraint profile that decides.

A KAN-specific point is that freezing granularity can influence the retention/adaptation balance. TF preserves structured function components, while PF enables fine-grained protection of individual spline coefficients. These controls are not merely implementation details. They correspond to different assumptions about how knowledge is stored in KAN edge functions: PF protects localized parameter importance, while TF preserves entire functional substructures that may be reused across tasks. The observed differences suggest that CL for KANs may benefit from controls that operate at the level of learned functions rather than only at the level of conventional weights [13]. It is also plausible that these effects may depend on the task stream and the type of shift; the present benchmark is intentionally simple, so more complex regimes will be

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

needed for stronger claims.

The scoring rule (`weight`, `grad`, `weight_grad`) shapes which spline parameters are protected. In the reported experiments, gradient-informed scoring generally yields more favorable forgetting profiles than weight-only scoring in the tested regime. One possible interpretation is that gradient statistics capture update sensitivity during adaptation, providing a useful proxy for parameters likely to be affected by interference. More broadly, scoring can be seen as connecting classical importance-based CL (e.g., Elastic Weight Consolidation or synaptic-importance methods) with KAN-specific freezing: both aim to identify and protect knowledge-bearing components, but KCF applies this principle to spline-function parameters with tunable granularity [1], [4], [5].

The results support a simple selection heuristic:

- If replay budgets are generous (large M and/or high ρ), replay-only may be sufficient and will often be the simplest deployment choice.
- If replay is constrained, adding plasticity control may help: a moderate k with a gradient-informed scoring rule can provide retention gains, and PF/TF can be selected depending on whether retention or new-task learning is prioritized.
- KCF can be used as a design tool: configurations can be compared under matched budgets (fixed M and ρ) to identify an "efficient retention" operating point rather than a single global optimum.

This heuristic is consistent with the KCF "sweet spot" intuition: efficient retention is often obtained by combining modest rehearsal with targeted restrictions, rather than by maximizing a single knob.

This study uses Split-MNIST [5], [19] as a controlled two-task benchmark chosen deliberately to isolate CF dynamics under a transparent, reproducible protocol. This is a recognized limitation: Split-MNIST is a relatively simple benchmark, and the extent to which KCF's benefits generalize to high-dimensional data (e.g., Split-CIFAR, ImageNet subsets) or natural language processing tasks remains to be established. Extending KCF evaluation to multi-task class-incremental streams, domain-shift benchmarks, and non-vision settings is therefore the most important empirical next step, and the control-vector formulation is designed to transfer without modification to those settings.

A second limitation concerns the static nature of the freeze ratio k. In the current formulation, k is pre-configured before Phase B begins and does not change in response to observed drift or task difficulty. An adaptive scheduling strategy—one that conditions k on drift-detection signals or on running estimates of forgetting severity—could provide a more favorable retention/adaptation balance when distributional change is gradual or irregular. This is a concrete planned extension of KCF.

Third, all experiments use a two-task sequential protocol. Evaluating KCF over longer task sequences (e.g., 10+ tasks under class-incremental or domain-incremental protocols) is needed to characterize how accumulated freezing constraints interact with ongoing plasticity demands. The present results should therefore be interpreted as characterizing KCF behavior in constrained, short-horizon regimes; stronger claims about long-horizon stability require dedicated multi-task experiments.

Finally, while the present paper emphasizes CL, KCF is motivated as a broader abstraction for controlled adaptation in dynamically changing environments. A longer-term direction is to integrate KCF with drift detection and governance mechanisms so that when and how models adapt can be made explicit and auditable in deployed systems [22].

VII. Conclusion

In this paper, the KCF was presented as a controlled adaptation view of CL in KANs. Extending our prior work on progressive spline freezing [1], this paper makes four novel contributions relative to that work: (i) KCF is formalized as a compact, unified control vector that separates memory and plasticity controls into an explicit, reproducible specification; (ii) a multi-seed reliability characterization is provided that exposes variance patterns absent from single-run comparisons in our previous work; (iii) a granularity-aware analysis is introduced, distinguishing point-level (PF) and tensor-level (TF) freezing behaviors and their regime-dependent trade-offs; and (iv) actionable selection guidance is derived for common resource-constrained operating scenarios.

Through multi-seed experiments on Split-MNIST [5], [19], it was found that replay establishes a strong retention baseline and defines a smooth performance landscape over (M, ρ), while progressive freezing tends to provide its largest benefits under constrained replay budgets, where it can improve retention efficiency without relying on maximal replay. Overall, the results support KCF as a practical lens for configuring KAN continual learning: matched-budget comparisons are enabled, the stability–plasticity trade-off is clarified, and selection guidance can be derived for common resource constraints. Future work may benefit from longer task streams and more realistic shifts, from adaptive control schedules, and from tighter integration of controlled adaptation with governance and compliance objectives in deployed systems.

Acknowledgment

General-purpose online and cloud-based tools, including AI-assisted writing assistants, were used during the preparation of this work solely for language editing and text refinement; they were not used for experiment design, data collection, statistical analysis, or generation of results, figures, or tables. All scientific contributions, experimental findings, and conclusions reflect the authors' own work.

References

[1] E. Ostanin, N. Djosic, F. Hussain, S. Sharieh, A. Ferworn, and M. Sharieh, "Progressively Overcoming Catastrophic Forgetting in Kolmogorov–Arnold Networks", in *IARIA Congress 2025: The 2025 IARIA Annual Congress on Frontiers in Science, Technology, Services, and Applications*, Jul. 2025, pp. 138-143.

[2] B. Liu, "Lifelong machine learning: A paradigm for continuous learning", *Frontiers of Computer Science*, vol. 11, no. 3, pp. 359–361, 2017.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

[3] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review", *Neural Networks*, vol. 113, pp. 54–71, 2019.

[4] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks", *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[5] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence", in *Proc. 34th Int. Conf. on Machine Learning (ICML)*, vol. 70, pp. 3987–3995, 2017.

[6] Z. Li and D. Hoiem, "Learning without forgetting", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2018.

[7] M. Farajtabar, N. Azizan, A. Mott, and A. Li, "Orthogonal gradient descent for continual learning", in *Proc. 23rd Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, vol. 108, pp. 3762–3773, 2020.

[8] A. A. Rusu et al., "Progressive neural networks", *arXiv preprint arXiv:1606.04671*, 2016.

[9] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks", in *Proc. 6th Int. Conf. on Learning Representations (ICLR)*, 2018.

[10] A. Chaudhry et al., "On tiny episodic memories in continual learning", *arXiv preprint arXiv:1902.10486*, 2019.

[11] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning", in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[12] L. Liu, L. Liu, and Y. Cui, "Prior-free balanced replay: Uncertainty-guided reservoir sampling for long-tailed continual learning", *arXiv preprint arXiv:2408.14976*, 2024.

[13] Z. Liu et al., "KAN: Kolmogorov-Arnold networks", in *Proc. 13th Int. Conf. on Learning Representations (ICLR)*, 2025.

[14] E. Ostanin, N. Djosic, F. Hussain, S. Sharieh, and A. Ferworn, "Evaluating the robustness of Kolmogorov-Arnold networks against noise and adversarial attacks", in *Proceedings of the SECURWARE 2024, The Eighteenth International Conference on Emerging Security Information, Systems and Technologies*, Nov. 2024, pp. 11–16.

[15] N. Djosic, E. Ostanin, F. Hussain, S. Sharieh, and A. Ferworn, "KAN vs KAN: Examining Kolmogorov-Arnold Networks (KAN) Performance under Adversarial Attacks", in *Proceedings of the SECURWARE 2024: The Eighteenth International Conference on Emerging Security Information, Systems and Technologies*, Nov. 2024, pp. 17-22.

[16] E. Ostanin, N. Djosic, F. Hussain, S. Sharieh, and A. Ferworn, "From Theory to Practice: Evaluating and Enhancing Kolmogorov-Arnold Networks (KAN) Robustness Under Adversarial Conditions", *International Journal on Advances in Security*, vol. 18, no. 1 & 2, pp. 77-91, Jun. 2025.

[17] A. D. M. Ibrahum, Z. Shang, and J.-E. Hong, "How resilient are Kolmogorov–Arnold networks in classification tasks? A robustness investigation", *Applied Sciences*, vol. 14, no. 22, article 10173, 2024.

[18] T. Alter, R. Lapid, and M. Sipper, "On the robustness of Kolmogorov-Arnold networks: An adversarial perspective", *arXiv preprint arXiv:2408.13809*, 2024.

[19] L. Deng, "The MNIST database of handwritten digit images for machine learning research", *IEEE Signal Processing Magazine*, vol. 29, pp. 141–142, Jun. 2012.

[20] M. Delange et al., "A continual learning survey: Defying forgetting in classification tasks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3366–3385, 2022.

[21] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights", in *Proc. European Conference on Computer Vision (ECCV)*, pp. 67–82, 2018.

[22] A. Saidane and S. Al-Sharieh, "A compliance-driven framework for privacy and security in highly regulated socio-technical environments: an e-government case study", in *Research Anthology on Privatizing and Securing Data*, IGI Global, pp. 933–962, 2021.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

VR-ML: Visualization of Machine Learning Models and their Evolution in Virtual Reality

Roy Oberhauser[0000-0002-7606-8226]
Computer Science Dept.
Aalen University
Aalen, Germany
e-mail: roy.oberhauser@hs-aalen.de

Abstract – **Machine Learning (ML) models can be difficult to grasp, and especially for novices or infrequent users. Between its inputs and outputs, various ML models can seem abstract and mathematical and not readily understandable and explorable, especially across diverse models. To make diverse ML models accessible and support comprehension and analysis for further stakeholders, this paper contributes VR-ML, a solution concept for immersive multi-model ML visualization in Virtual Reality (VR). A prototype is used to exhibit its feasibility, demonstrating support for diverse ML model visualizations including Neural Networks, Random Forest, and Naive Bayes. A case-based evaluation provides insights into its capabilities and potential for visualizing diverse ML models immersively and concurrently, including loading, building or configuring, training, comprehension, analysis, exploration, scalability, and highlights collaboration opportunities.**

Keywords – virtual reality; visualization; machine learning; artificial neural networks; naive bayes; random forest.

I. INTRODUCTION

This paper extends our prior work VR-ANN [1], incorporating VR viewpoint support for multiple Machine Learning (ML) models, illustrated with Artificial Neural Networks (ANN), Naive Bayes (NB), and Random Forest (RF) models, and exhibiting support for additional capabilities such as model evolution, history, and comparison.

ML is a subset of Artificial Intelligence (AI) [2] that focuses on having machines learn from data, improving their performance over time without reprogramming. A learning algorithm can optimize its model's parameters to improve its performance, which can improve pattern detection, predictions, decisions, etc. Various models can be applied in the area of ML. Artificial NNs (ANN) [3], referred to as just Neural Networks (NN) in this paper, are inspired by biological NNs and consist of nodes (referred to as artificial neurons) connected via weighted links or edges (i.e., synapses), with other nodes and are aggregated into layers. Numbers are used to represent signals, while a node's activation function determines a neuron's output based on its inputs and their associated link weights. Between the input and output layer are intermediate (hidden) layers. Dense layers are fully-connected to the preceding layer. Dropout layers address overfitting by randomly setting a fraction of the input units to that layer to 0 during training. Hidden layers are any that are neither input nor output layers. Deep Learning (DL) is a branch of ML utilizing deep NNs having multiple hidden layers. Convolutional Neural Networks (CNNs) are a type of DL network especially relevant for image processing. Feedforward NNs (FNNs) are a type of NN where all neurons are fully connected to the next layer with unidirectional information flow. Recurrent NNs (RNNs) are a type of NN for processing sequential data where its order matters. Naive-Bayes (NB) [4] is a supervised probabilistic ML classification algorithm, based on Bayes' theorem, that assumes features are independent and assigns probabilities. Application areas include medical diagnosis, recommendation engines, spam filtering, software defect detection, and sentiment analysis. Random Forest (RF) classification [5] is a supervised ML ensemble method that creates multiple decision trees during training using different random subsets of the data and features, and then combines predictions to produce a final classification based on the majority. Applications areas of RF include credit scoring, medical diagnosis, and fraud detection, and weather forecasting [6]. Each has its strengths: NNs can capture complex nonlinear patterns in large datasets, NB is simple, fast, and effective for high-dimensional data with approximate independence, and RFs offer robust performance with good accuracy and resistance to overfitting. Yet each also has its liabilities: NNs require more data, tuning, and computation and are difficult to interpret, NB assumes strong independence that affects accuracy when features are correlated, while RF can become computationally expensive and less interpretable than NB, while often not achieving the accuracy of well-tuned NNs for complex tasks.

Yet the complexity of ML models can be difficult for various stakeholders to understand, analyze, and trace their evolution. Even seemingly slight configuration changes can affect performance, and a visual comparison of model types could be beneficial in understanding performance differences. An immersive visualization of ML models can make such models accessible for stakeholders and support comprehension, analysis, and learning. While various tools support 2D, there has been relatively little investigation into the potential offered by Virtual Reality (VR). To address the visualization of multiple heterogeneous ML models and their evolution, this paper proposes and investigates an immersive visualization solution concept for diverse ML model types, which, due to VR's spatial scalability, can readily depict multiple models and their evolution concurrently.

In prior work, we investigated the application of VR to various other areas. A selection of our prior VR-related contributions include: in the Software Engineering (SE)

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

space, VR-SDLC [7] models development lifecycles, VR-Git [8] models Git repositories, VR-DevOps [9] models Continuous Development pipelines, VR-SBOM [10] models Software Bill of Materials (SBOM) and software supply chains, and VR-EA+TCK [11] and VR-EvoEA+BP [12] exemplify enterprise modeling and business processes. Our VR-ANN [1] focused only on ANN models. This paper contributes a solution concept for immersive multi-ML model and model evolution visualization of in VR. A prototype is used to exhibit its feasibility, demonstrating support for diverse ML model visualizations including NN, NB, and RF. A case-based evaluation investigates its capabilities and potential for supporting multiple model and model evolution depiction, comprehension and analysis.

The remainder of this paper is structured as follows: the next section discusses related work. In Section III, our solution is described. Section IV presents our realization and is followed by our evaluation in Section V. And finally, a conclusion is provided.

II. Related Work

In their survey of the application of XR to AI, Hirzle et al. [13] screened 2619 publications (2017-2021) and reviewed 311 in depth. They found only seven papers that applied XR to AI problems (2.3%), five of which visualize AI methods for immersive analytics, or to improve the understanding of neural networks for non-expert users by visualizing them in VR (the rest are discussed below). The authors state XR "methods are promising to facilitate the interaction with neural networks for novices." The 2025 survey by Yim and Su [14] of K-12 learning tools for AI examined 46 papers, but and makes no mention any XR/VR tool. The 2021 survey by Reiners et al. [15] identified 36 papers that combined XR and AI, of which only two were non-domain-specific and related to visualizing DL in XR (discussed below). The survey on AI in VR by Inkarbekov et al. [16] identified a research gap for more user-friendly, intuitive, and adaptable VR tools that can accommodate complex and high-dimensional AI models. From these surveys we conclude that VR-based visualization of AI has not been thoroughly investigated nor readily adopted and remains relatively unexplored.

VR-related NN visualization work includes Bellgardt et al. [17], who depict convolutional ANNs in VR as node-link diagrams by stacking circular layers for a robotic image processing case. InteractML [18] is a node-based tool for creative practitioners to train an ML model for movement interactions in VR using real-time gesture demonstrations. AIive [19] uses a force-directed graph visualization and sonification to enable VR users to manipulate NN parameters via virtual hands and auditory feedback. Towards non-experts, Meissler et al. [20][21] depict a simplified CNN model in 3D in a closed room virtual environment, with information in 2D anchored to different areas of the room. Schreiber and Bock [22] use the Unreal Engine to display a NN in 3D, whereby connections between layers are not visualized. While VR is mentioned in the title, there is no mention of VR or immersion in the paper, which focuses on 3D. Queck et al. [23] create a virtual room in VR with areas providing CNN information. VR4DL [24] can train and test CNNs with a focus on biomedical image classification; it is not intended to be generic for arbitrary CNNs, and is tailored to users with little to no knowledge of ML. DeepVisionVR [25] visualizes CNNs in VR with a focus on image processing. Common 2D NN model visualization tools include (TensorFlow) Deep playground, TensorBoard, Netron, Comet, and neptune.ai.

NB visualization work includes Možina et al. [26], who propose 2D nomograms. Balfer and Bajorath [27] propose an interactive 2D pie chart visualization with feature groups as slices, points representing a feature colored by influence (negative or positive), pole distance corresponding to absolute log odds ratio, and point size for likelihood. For single user assessment in VR simulators, Ferreira et al. [28] propose a double-weighted fuzzy hypergeometric NB network. However, VR visualization aspects are not directly addressed.

RF visualization work includes Welling et al. [29], who propose 3D forest floor visualizations that depict feature contributions and utilize color gradients. Nsch et al. [30] propose 2D botanically-inspired colored binary tree visualizations. iForest [31] is an interactive 2D visualization for analyzing RF models and predictions. ReFINE [32] consists of several 2D RF visualization views: proximity, interactions, and prototypes.

In summary, most prior work has focused on 2D ML visualization and typically focused on a single model. Where immersive VR has been used in conjunction with ML, it is often regarding ML-based assessment of users during simulation and training sessions (such as in medicine). Hence, VR-based immersive ML model visualization with multi-model and evolution support has not been sufficiently investigated and remains relatively unexplored.

III. Solution Concept

Our VR-ML solution concept is shown (highlighted in blue) in the AI area (green) relative to our other prior VR solutions in our conceptual map of Figure 1.

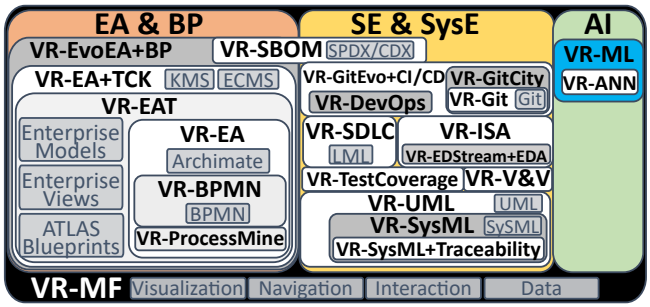


Figure 1. Conceptual map of our various published VR solution concepts with VR-ML highlighted in blue.

Our generalized VR Modeling Framework (VR-MF, black area, detailed in [33]) is the foundation, which provides a domain-independent hypermodeling framework, which addresses the VR aspects of visualization, navigation, interaction, and data integration. Our VR-based solutions specific to the Enterprise Architecture (EA) and Business Process (BP) space (EA & BP, red area) include: VR-EA [33] for mapping EA models to VR, VR-BPMN [34] for BPMN models, VR-EAT for enterprise repository integration, VR-

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

EA+TCK [11] for knowledge and content integration, and VR-EvoEA+BP [12] for EA evolution and business process animation, VR-ProcessMine, and VR-SBOM [10]. Solutions in the SE and Systems Engineering (SE & SysE, orange area) areas include: VR-Git [8], VR-GitCity [35], VR-DevOps [9], VR-GitEvo+CI/CD [36], VR-V&V (Verification and Validation), VR-TestCoverage, VR-SDLC [3], VR-ISA for informed software architectures, and VR-UML and VR-SysML for software and systems modeling.

A. Solution Concept Grounding in VR-Related Research

As previously described in [36], our rationale for integrating VR into our solution concept stems from existing VR research in fields we consider relevant to modeling, analysis, and collaboration, of which several are summarized here. Akpan and Shanker [37] in their systematic meta-analysis demonstrate that VR and 3D provide marked advantages in discrete event modeling, including model development, analysis, and verification and validation, with common findings pointing to the positive impact of 3D/VR model analysis and V&V. Across 23 studies on 3D analysis, 95% found 3D more effective and conducive to enhanced analysis compared to 2D, notably when assessing model behavior or conducting what-if scenarios; there was broad agreement that 3D/VR effectively communicates results to decision-makers with clarity and conviction; 74% of 19 papers concluded that 3D/VR significantly enhances model development tasks, benefiting team support, and sharpening precision and clarity. In exploring VR applicability for analytical tasks within an information architecture, Narasimha et al. [38] conducted a card sorting collaboration study. Their findings indicated VR matched or exceeded in-person card sorting for certain variables, surpassing both traditional and video-based settings. Qualitative insights on awareness suggested that during collaboration, participants maintained awareness of tasks, people, and the environment, mimicking in-person interactions, with positive perceptions of VR. These findings indicate that VR facilitates a sense of presence and collaboration akin to face-to-face settings. Fonnet and Prie's [39]survey of Immersive Analytics (IA) reviewed 177 papers and found that for complex graph and spatial data, IA offers advantages over non-IA methods, though for multi-dimensional data, benefits vary with the task. They highlight that while IA supports the exploration of extensive data environments, the underutilization of context-aware navigation techniques is problematic, despite their importance to users. Müller et al. [40] compared VR vs. 2D for a software analysis task, finding no significant decrease for VR in comprehension and analysis time. While interaction time was less efficient, VR improved the user experience, was more motivating, less demanding, more inventive/innovative, and more clearly structured.

Consequently, we infer that an immersive, context-rich VR experience holds significant promise for comprehensively visualizing 3D models, while also enhancing comprehension, awareness, analysis, and the inclusion of and collaboration with stakeholders.

B. Visualization in VR

Our solution concept leverages extensible *VR Viewpoints* (VR:VP) to support different forms of immersive visualization for different ML model types and stakeholder concerns. Viewpoints are generic and provide conventions, rules, or templates governing view types, whereas ML views are then specific to a certain ML model and address specific stakeholder concerns. The VR viewpoints offer clarity about how to support the visualization of a certain ML type (kind), while not hindering further additional VR viewpoints to support additional ML types. Our solution concept is extensible, in that it can theoretically support an unlimited set of viewpoints and ML model types, from 1-to-1 through M-to-N relations. Each unique ML model type is mapped to at least one VR Viewpoint. A VR Viewpoint can either be unique to that ML model type, reused (aggregating/supporting multiple types), or adapted (templated and specialized from another similar viewtype, perhaps conceptually superclassed or abstracted). Hence, viewpoints can aggregate ML model types, can readily reuse visualization support from other viewtypes. Moreover, additional crosscutting or specialized ML viewpoints are feasible. Given the unlimited visual space of VR, this enables a much larger scale of multi-ML-model visual support for comprehension, analysis, and collaboration.

To assist with visually delineating the various models from one another, an invisible (glass) 3D Bounding Box (BB) showing outlines is used.

For simplification and for the sake of brevity, we will focus on three common ML types and associated viewpoints to convey our concept, but the concept is extensible with additional VR viewpoints for any given ML model.

VR Viewpoint Neural Network (VR:VP:NN): This viewpoint is concerned with portraying NN models, offering support for depicting construction, configuration, layers, connections, execution, prediction, training, data, versioning, evolution, etc. in VR-based views. To better delimit the context of a NN model in a multi-NN VR space, each NN model is visualized within an outlined 3D transparent Boundary Box (BB). This enables further additional portfolio of NNs to be visualized contemporaneously. Contained within the NN BB are outlined transparent 3D BBs, each representing a NN layer, which vertically delineates a set of spherical nodes (neurons). Lines are used as connectors to indicate which nodes are connected between layers.

VR Viewpoint Naive-Bayes (VR:VP:NB): This viewpoint is concerned with portraying a NB model, offering support for depicting classification, data, configuration, etc. in VR-based views. Visualization utilizes a standard 3D top-down hierarchically-layered tree (model, classes, features).

VR Viewpoint Random Forest (VR:VP:RF): This viewpoint is concerned with portraying RF models, offering support for depicting trees, data, configuration, etc. in VR-based views. As any tree in a RF can be structured differently with different depths, for RF model visualization, vertical 2D trees are radially separated and dispersed, and trees utilize vertical space expanding further out radially, analogous to an explosion.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

To support model evolution and version tree visualization, we chose to use the intuitive analogy of growing trees with the root at the bottom and newer versions spreading upwards, with connectors between BBs showing relations. To help visually differentiate changes, green is used to indicate any additions to a model on the new version, while red is used to mark what no longer exists in the next version. Furthermore, panels at the front of a BB are used to retain version-specific information for that BB, enabling detailed parameter, data, or metrics comparisons between versions. These can be collapsed or expanded individually using the affordance or for all via the VR-Tablet. The anchored panels offer clear context association to their model when multiple models are involved, while the VR-Tablet can be used to adjust overall parameters.

C. Navigation in VR

Dual navigation modes are incorporated in our solution: default gliding controls for fly-through VR, while teleporting instantly places the camera at a selected position in space. While teleporting can be potentially disconcerting, it may reduce the likelihood of VR sickness.

D. Interaction in VR

User-element interaction is supported primarily through VR controllers and the incorporation of a VR-Tablet. The VR-Tablet provides detailed context-specific element information. Various tabs in the VR-Tablet enable support for loading, training, executing, or configuring models and viewing related metrics. Since for our examples no text entry and keyboard were required, a virtual keyboard was not included. However, our implementation can be readily enhanced with a virtual keyboard for text entry using laser pointer key selection, as demonstrated in our other VR solutions. A small control sphere is placed as an affordance on the edge of a BB for dragging, collapsing (to reduce visual clutter), or expanding a BB.

IV. Realization

For our prototype realization, the VR visualization aspects were implemented using Unity in C#, referred to as our frontend, as shown in Figure 2.

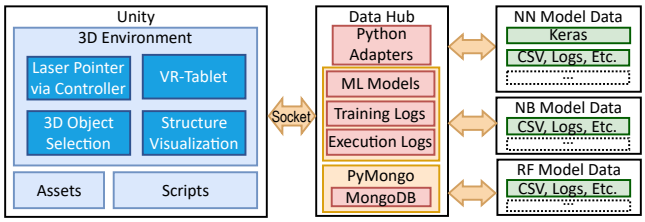


Figure 2. VR-ML logical architecture.

It connects via a Socket to our backend Data Hub, which is based on the hexagonal (or ports-and-adapters) design pattern and is implemented in Python. The Data Hub integrates and stores all data via PyMongo to a MongoDB database in JSON/BSON format utilizing Python adapters to utilize and convert the various ML models, APIs, and data formats to our internal format. This permits the frontend to quickly render its various visualizations. Docker images and containers can be readily used on the backend if available for the various ML models. Various formats and serialization are supported and used, including PKL, CSV, JSON, and Joblib.

For realizing *VR:VP:NN* support, the Keras API was used as a high-level DL API primarily due to its popularity and flexibility, since it supports multiple backends, such as TensorFlow, PyTorch, OpenVINO, and JAX (Python library for high-performance ML). Initially for prototyping, the Sequential model with various layer types placed in sequence is supported, but support for additional models can be readily added. Data can be imported from or exported to the common keras format (.keras extension), which is a zip archive containing JSON-based configuration, H5-based state file containing layers and weights, model weights, and metadata. Additionally, the CSV file format is supported. Further formats can be readily supported via adapters. The models can either be pretrained, loaded, and executed in VR (load and execute), or can be (re)configured and trained in a VR session via our Model Builder support mode.

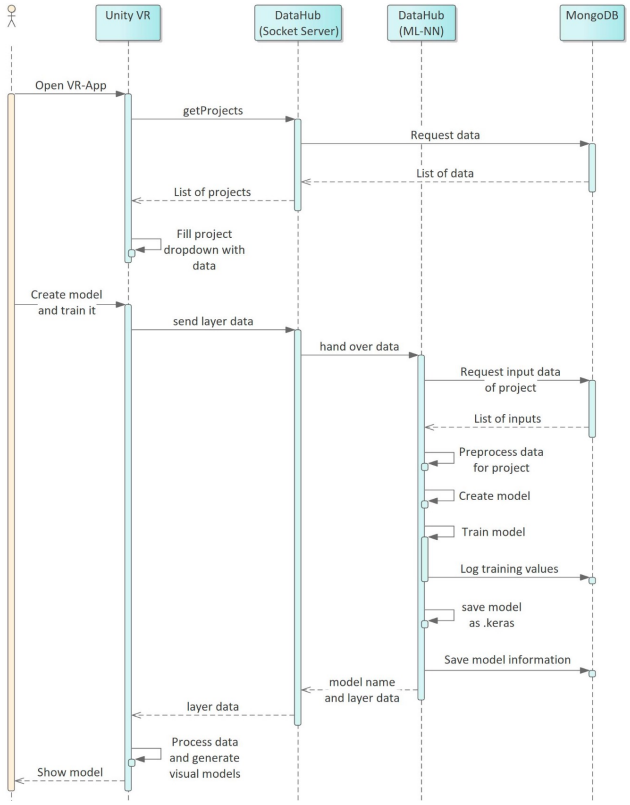


Figure 3. Example VR-ML sequence diagram for a NN training session.

To exemplify the internal dynamics of the architecture, a sequence diagram for a VR-centric NN training session is shown in Figure 3. Initially, the list of available NN projects is retrieved from the Data Hub. If the user creates a new model and starts a training session, then the layer data is sent to the Data Hub, the training inputs are retrieved, and a model is created and trained via the Keras API, training values are logged, and the model is saved or exported. The acquired layer data is returned to Unity via the socket, and is visualized as a model, which can be further analyzed.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Our VR-Tablet supports interaction in VR. This is exemplified for a VR:VP:NN context in the bottom right in Figure 4. The tab groups Load, Training, Execution, Display options, and Graphs are shown, as is the scrollbar. Here, under the selected Display options tab, the epoch slider is shown as well as options to show or color connections. Also shown in this figure, the model layer BBs top side provide layer information (number and type) and metrics (number of neurons) and containing spheres as neurons evenly spaced along a single vertical plane with slots for the next best-fitting square matrix, filled from the bottom left (upper right may have empty slots), while dropout layers are depicted as an opaque slab. Connectors are shown by default in blue; the diameter of the lines indicate the relative weighting to the next layer. In execution, the most active (top five) routes as nodes and connectors are colored (darker to lighter) green. Connectors can optionally be colored (iterated list of 32 colors) to more easily follow a connector between nodes.

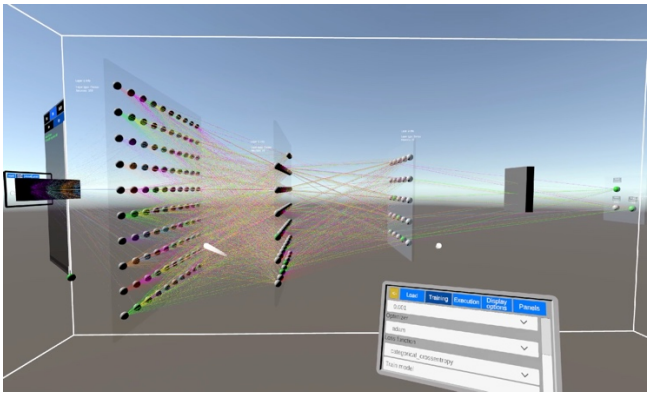


Figure 4. Boundary Box (BB) enclosing a ML model (NN), with the VR-Tablet showing tab groups and input options.

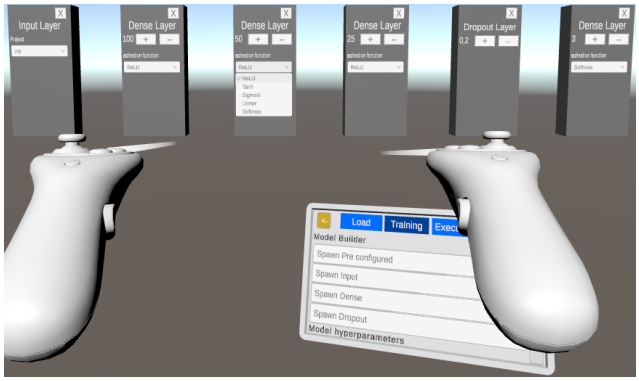


Figure 5. VR:V:NN:MB Model Builder VR View for configuring a NN.

VR:VP:NN also includes support for a Model Builder VR View (VR:V:NN:MB). For this, opaque 3D boxes represent layers ordered from left to right, each of which offers appropriate options based on type, as shown in Figure 5. To avoid requiring text input and simplify interaction, the default values can be adjusted with plus/minus buttons and dropdown lists offer selection options. Layers can be removed by an X button at the top right, and new layer types can be inserted via the VR-Tablet. The ordering of the layers can be adjusted by dragging the boxes. RNN support could be readily included but was not yet implemented. All applicable parameters can be selected and configured in the VR-Tablet, including epochs, learning rate, optimizer, loss function, etc.

VR:VP:RF support in the Python adapters was implemented utilizing the sklearn.ensemble RandomForestClassifier. For RF visualization, the root node of each 2D vertical tree is radially equidistantly dispersed on an invisible horizontal circle near the center of the BB, as shown in Figure 6. Each RF tree can be structured differently and is vertically depicted from the center with nodes and colored connectors. Gray connectors represent RF sequence connection, in the trees white(black) connectors indicate a condition was fulfilled(unfulfilled), blue is used to indicate a selected node for the "Node Inspector", and green is used for paths to follow a prediction. Additional helper nodes are provided near the corners of the BB to gather values from the various trees and support analysis.

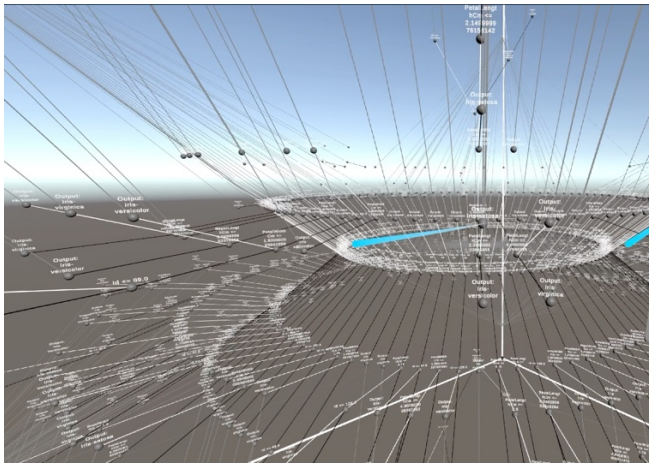


Figure 6. RF model view depicted as vertical trees in radial layout.

VR:VP:RF also includes support for a Model Builder VR View (VR:V:RF:MB), where training aspects for previous trainings is shown as blocks and the associated training parameters are shown and can be adjusted in the VR-Tablet, as in Figure 7. New training parameters can be (individually) set and reset, and training started with a button, as seen in Figure 8.

Figure 7. RF Model Builder VR View (VR:V:RF:MB) showing in the VR-Tablet previous training parameters.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

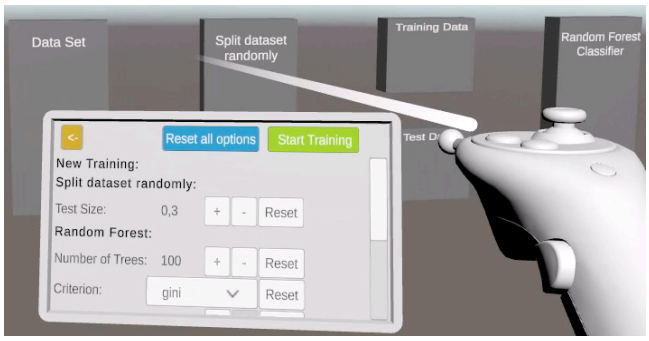


Figure 8. RF Model Builder VR View (VR:V:RF:MB) showing in the VR-Tablet new training parameters.

VR:VP:NB support in the Python adapters was implemented with sklearn.naive_bayes, including support for the NB variants: Gaussian, Bernoulli, Multinomial, Complement, and Categorical. NB visualization was realized as 3D hierarchical layered tree as shown in Figure 9. Blue is used to indicate the currently selected node for node inspection in the VR-Tablet. If it is a class, then the associated features are marked in yellow.

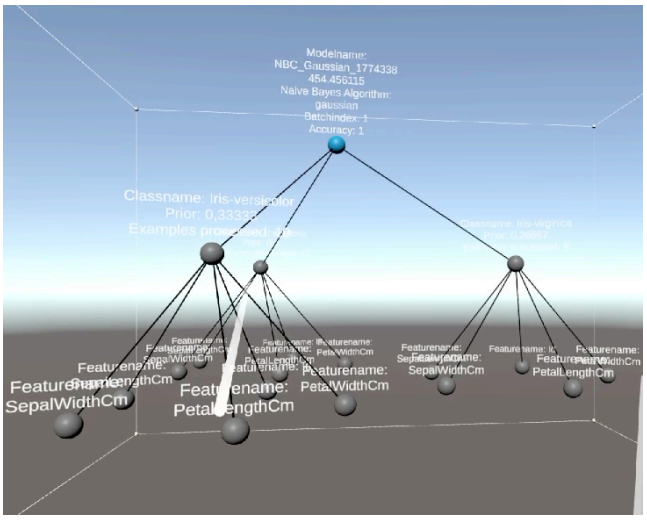


Figure 9. NB model view depicted as 3D hierarchical layered tree.

To support evolution and version differentiation and analysis, chose to anchor information panels and an embedded control panel to the front of a BB. This was done to minimize the risk of user errors when many BBs with only slight differences are involved. The left information panel relates to data flow and operational status (train, test, predict) with subtabs (data, logs) for input and output data and any generated logs. The right panel supports analysis of hyperparameters and performance/metric graphs (accuracy, loss). Furthermore, at the front of each BB an embedded control tablet offers hyperparameter adjustment, and triggering or retraining or predictions directly. That way it is obvious for which BB any hyperparameter change, training, or execution is being enacted.

V. Evaluation

The evaluation of our VR-ML solution concept is based on the design science method and principles [41], in particular a viable artifact, problem relevance, and design evaluation (utility, quality, efficacy). A scenario-based case study is used for each of the implemented VR viewpoints (VR:VP:NB, VR:VP:RF, VR:VP:NN), model evolution, model scaling, and concurrent multi-model support.

For the evaluation dataset, an Iris flower dataset on Kaggle [42] was used. For this, 50 samples each of three Iris species (Setosa, Versicolor, Virginica) are classified in the output based on four properties: SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm. One species is linearly separable from the other two, but they are not linearly separable from each other.

A. VR:VP:NB Case

This case demonstrates and assesses VR Viewpoint support for NB models. A NB project and model are loaded via the Load tab in the VR-Tablet. In the Training tab, the training parameters (if already trained) are shown, and training data and applicable parameters can be configured natively in VR, and a training session started via the “Start training” button, as shown in Figure 10.

Figure 10. NB training configuration via VR-Tablet.

The NB is depicted as a 3D tree structure, shown in Figure 11. A single project node at the top offers a clear starting point for navigation to the possible classes (as well as current project parameters), the classes on the mid-level, and the features determining a class on the bottom level. These can be explored to comprehend and analyze a classification result. As shown in the VR-Tablet, the Inspector tab offers additional node-specific information.

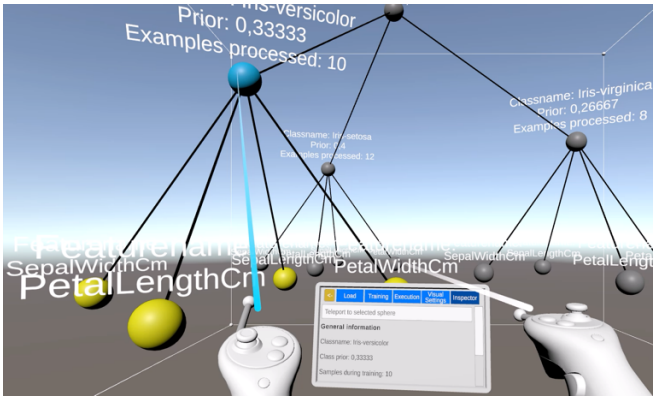


Figure 11. NB node selection colors selected class node blue and corresponding features yellow (Iris-versicolor); node info shown on VR-Tablet.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

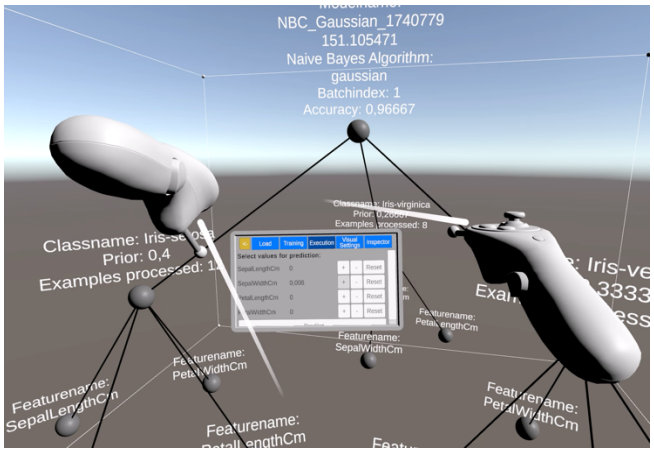


Figure 12. NB input parameters can be natively adjusted before prediction.

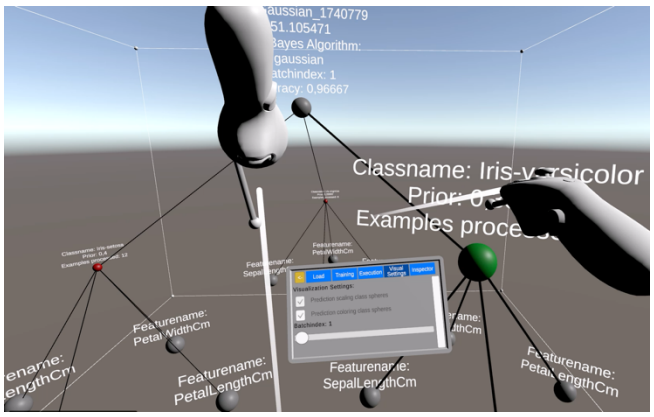


Figure 13. Execution on inputs (left) and result in the output layer (right).

For execution support, a tab in the VR-Tablet supports manually adjusting input parameters, as shown in Figure 12. Prediction results can be visualized and explored, as shown in in Figure 13. Furthermore, visual settings are configurable as shown here: node scaling to indicate probability, and node colorization (green for the most likely class, red for the others).

B. VR:VP:RF

This case demonstrates and assesses VR Viewpoint support for RF models. A RF project is selected and loaded via the VR-Tablet. In the Model Builder View, various training options are selected, as was shown in Figure 8. The resulting model is then depicted with input in the center, as shown in Figure 14. The input is connected (in grey) to the root of every tree radially. An individual tree can be explored, as shown in Figure 15. Nodes are labeled with the tree and condition, gray connectors represent an RF sequence connection, white/black for condition fulfilled/unfulfilled.

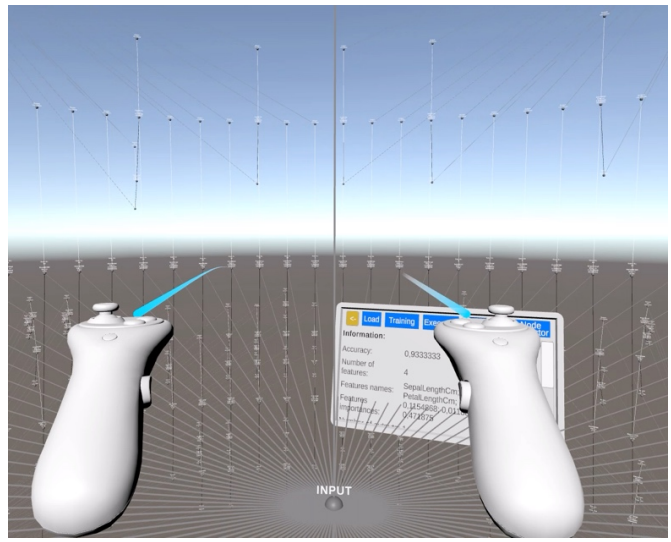


Figure 14. RF model depiction as radially dispersed 2D vertical trees.

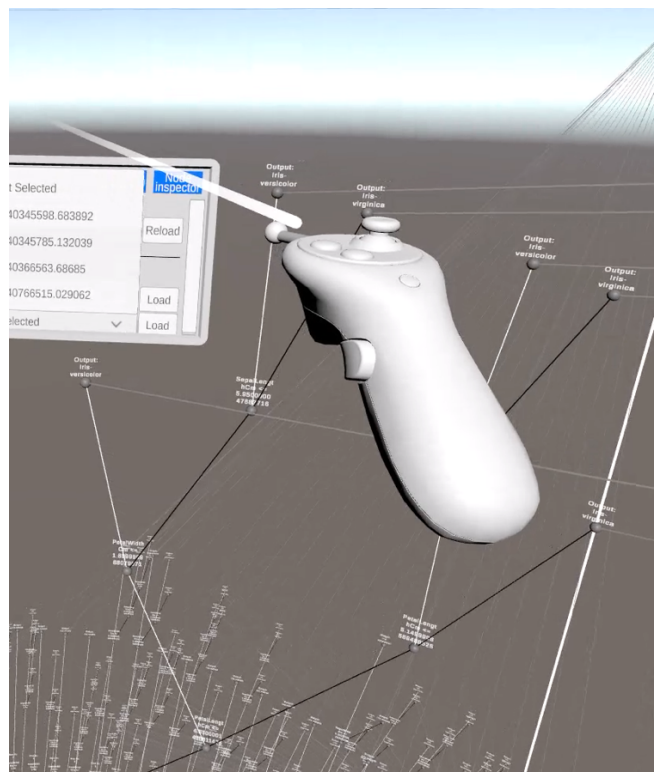


Figure 15. RF tree, each node labeled with tree number and condition with connectors to rest of tree (white if fulfilled, black unfulfilled).

Helper nodes placed near corners of the BB are used to aggregate the tree results via grey connectors to nodes, such as the output collector shown in Figure 16. The majority vote node and resulting final output node are shown in Figure 17.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

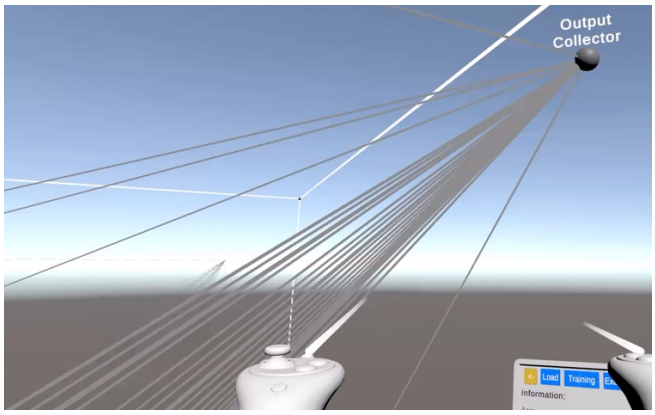


Figure 16. Collector nodes aggregate tree results to support analysis.

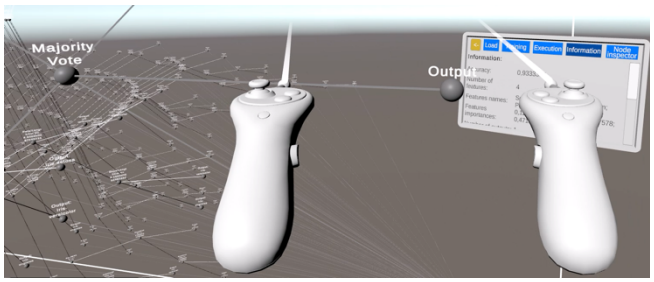


Figure 17. RF depicting majority vote and output nodes.

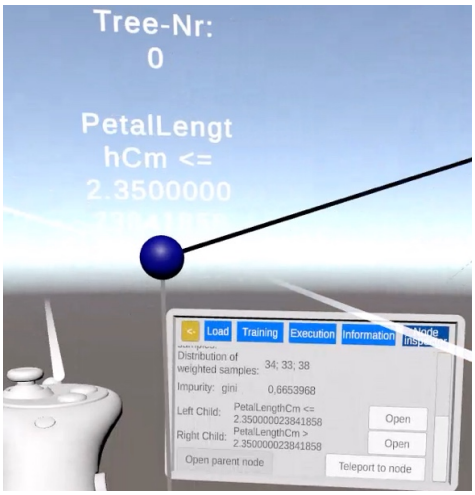


Figure 18. Analyzing RF: selected node information.

To support analysis, an individual node can be selected (blue) and inspected in the VR-Tablet, as shown in Figure 18. It offers information on and navigation to children and parent nodes as well as teleporting.

To support prediction and execution analysis in VR natively, inputs parameters can be adjusted and then the "Run Prediction" button pressed, as shown in Figure 19. Green nodes and branches show which were active. For this case, setosa was predicted by 99 trees, while 1 tree predicted versicolor. To determine which tree, one can iterate thru the tree index to see each tree prediction (80 was setosa), as seen in Figure 21, whereas tree 79 was unique in predicting versicolor, as shown in Figure 22.

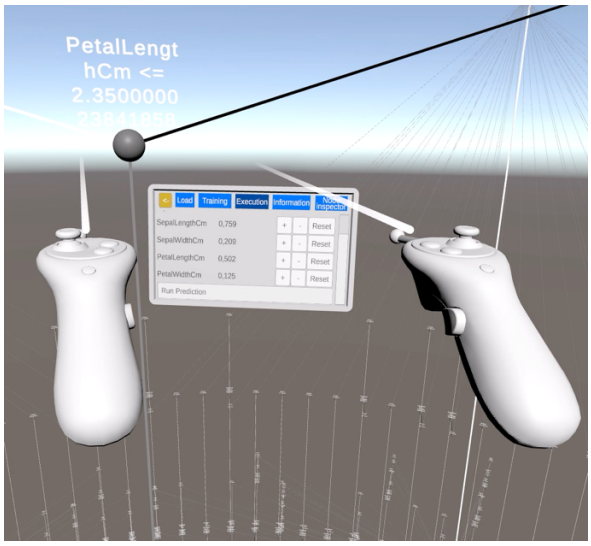


Figure 19. RF: Adjusting inputs in VR to run a prediction.

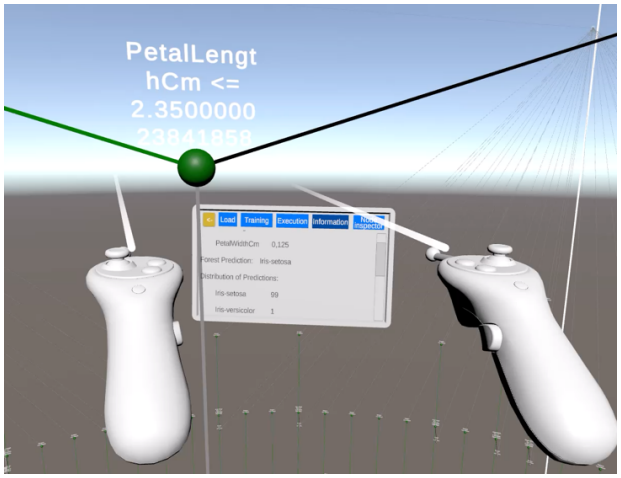


Figure 20. RF execution results in VR-Tablet and green nodes/connectors.

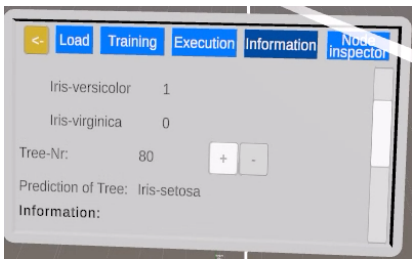


Figure 21. RF analysis: tree 80 predicts Iris-setosa.

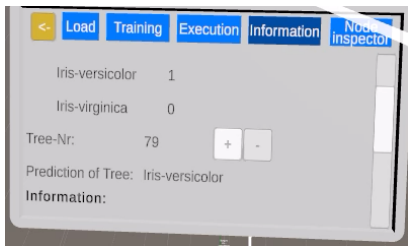


Figure 22. RF analysis: tree 79 predicts Iris-versicolor.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

C. VR:VP:NN Case

This case demonstrates and assesses VR Viewpoint support for NNs.

1) Comparison of 2D vs. VR-ML Visualization

A typical 2D NN visual representation is shown via the TensorFlow Playground in Figure 23. A VR View was shown in Figure 4. In both mediums, a NN can be readily investigated and analyzed. Much of the information seen in the 2D playground is available immersively in VR by interacting with the VR-Tablet, elements (layer, nodes), or panels anchored to the BB. Hence, there are no significant impediments to analysis in VR vs. 2D. Given the many NN connections, we contend it to be more straightforward to immersively follow a connector separated spatially in 3D space.

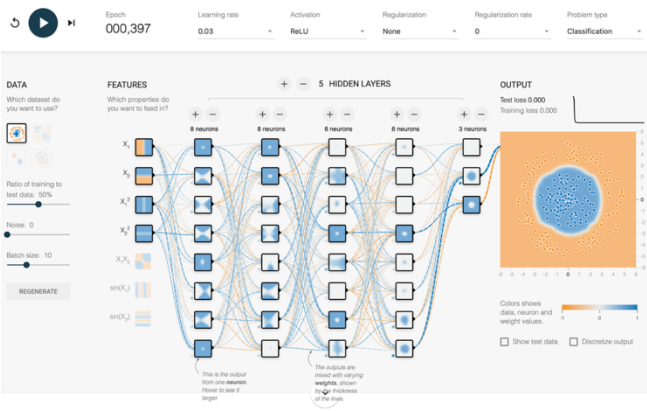


Figure 23. Screenshot of a NN model in TensorFlow playground [43].

2) Build and Train NN Model Support

VR:V:NN:MB support for building and configuring NNs was shown in Figure 5. Via the VR-Tablet, a preconfigured existing model can be loaded and layer types flexibly inserted, removed, or adjusted. After NN training, optionally triggered natively within VR, the training results can be loaded.

3) NN Analysis Support

The model, with all its layer types and neurons, is visualized within a BB as was shown in Figure 4. This helps to readily in comprehend of the total number of layers and their type, size, and ordering. Furthermore, each layer is labeled with its name, type, and size, as shown in Figure 24.

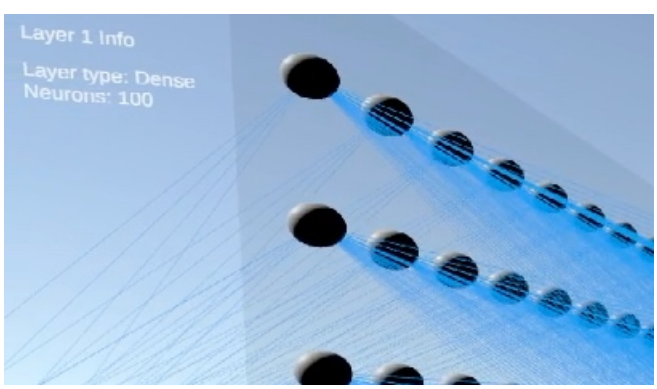


Figure 24. NN layer labels.

To support analysis, optional panels anchored to the front of a BB offer additional model information and insights, as shown in Figure 25. This provides better contextual clarity than a single VR-Tablet alone would offer when multiple models are involved, especially with regard to supporting evolution and version differentiation.

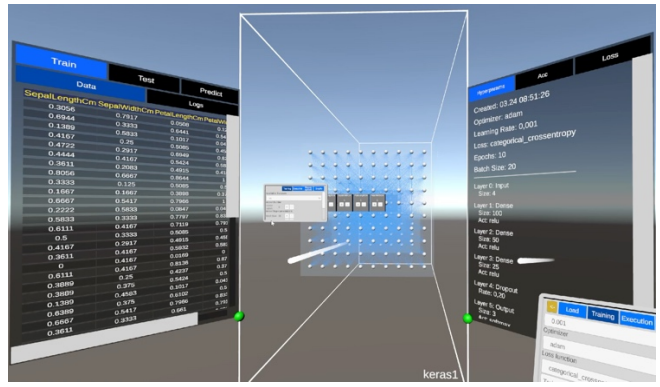


Figure 25. Information panels: data and operational state left (train, test, predict) with subtabs (data, logs), on the right hyperparameters and graphs (accuracy, loss).

The left scrollable information panel relates to data flow and operational status (Train, Test, Predict) with subtabs (Data or Logs). Test Data input (e.g., LengthCm_PetalWidth) and output columns (e.g., true_Species, predicted_Species) are shown in Figure 26. To support prediction analysis, the left panel offers a Predict tab and Data subtab with inputs and outputs as seen in Figure 27. Log output for training, testing, and prediction, can be analyzed, as seen for the Train tab Logs subtab shown in Figure 28.

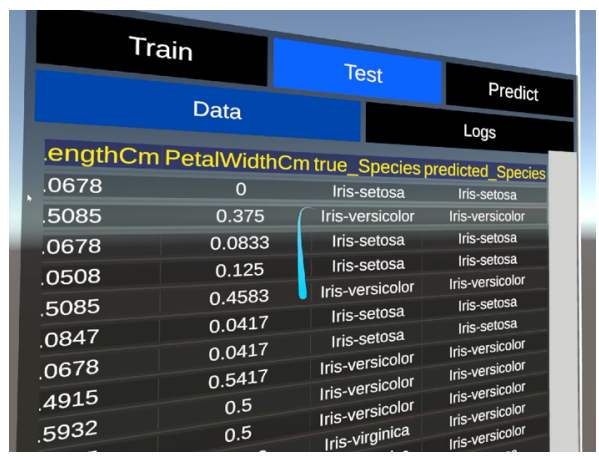


Figure 26. Left panel Test tab with subtab Data.

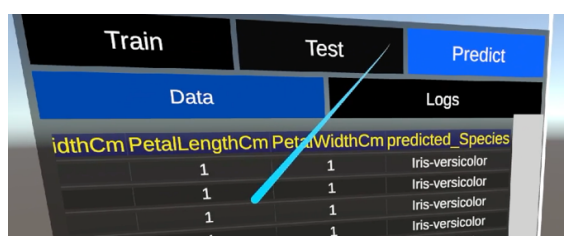


Figure 27. Left panel Predict tab with subtab Data.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

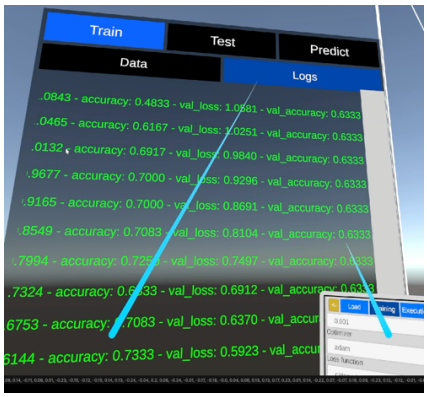


Figure 28. Right panel Train tab with Logs subtab.

After NN execution or prediction, the top five most frequent activation paths (pathways, routes) are indicated via (darker-to-lighter) green nodes and connectors. Within the BB, the model highlights the top five activation nodes and routes in shades of green. Input values for native VR inputs can be adjusted on the front of the BB with +/- buttons, and a prediction run via the front BB control panel, as seen in Figure 29. The output can be viewed the Predict tab Data subtab, or within the BB in the output layer, as shown in Figure 30.

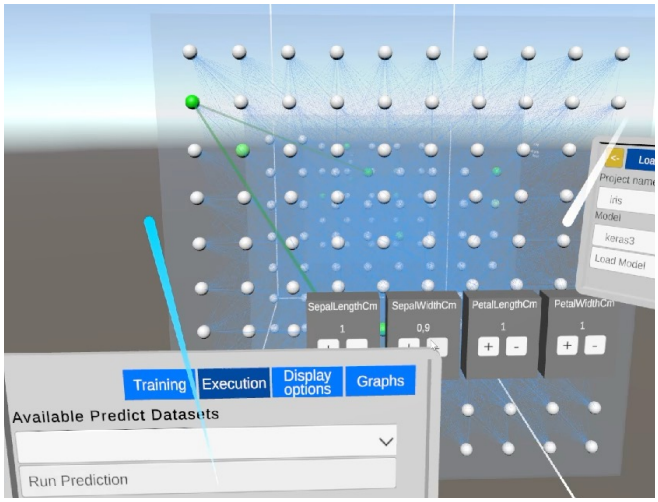


Figure 29. Predict values input values model in BB highlights top five activation nodes/routes in green.

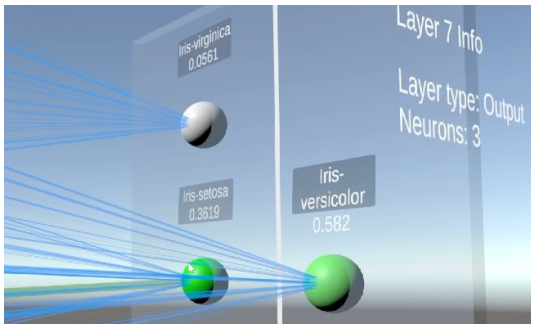


Figure 30. Prediction result output (setosa, 0.3896) depicted on output layer (higher-weighted are larger), top activation paths are highlighted in green.

On the right panel, the Hyperparameters tab supports analysis of hyperparameters, as shown in Figure 31. Furthermore, performance and metric graphs can be viewed in the Accuracy and Loss tabs, as shown in Figure 32. Furthermore, at the front of each BB an embedded control tablet offers hyperparameter adjustment, and triggering or retraining or predictions directly.

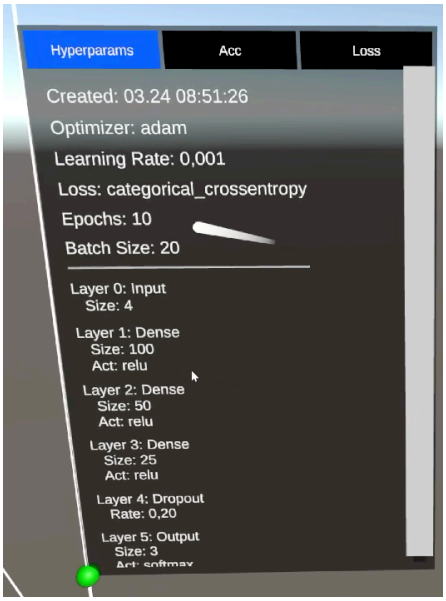


Figure 31. Right panel tabs: Hyperparameters.

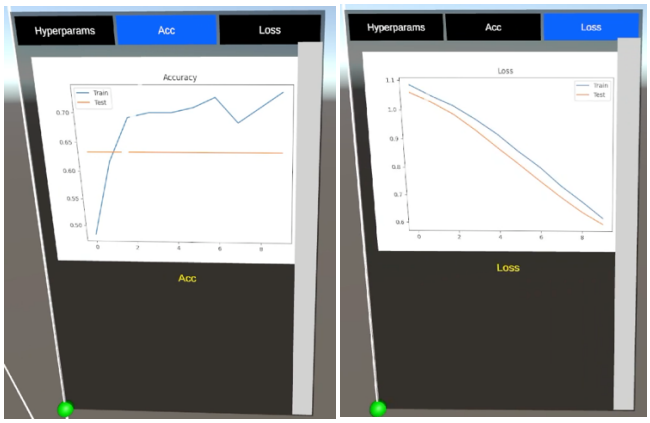


Figure 32. Right panel tabs: Accuracy graph (left) and Loss graph (right).

To support immersive analysis, connections between neurons can be optionally randomly colored for differentiation. To support analyzing a single node, selecting a single node in the model will cause only connections related to that node to be shown and all others to be hidden (to reduce visual distractions), as seen for colorized connections in Figure 33. Furthermore, the weight values of all the connected input nodes are displayed above each input node in the previous layer (higher-weighted connections are visualized larger), and metadata about the selected node is depicted. Also, the VR-Tablet offers an Epoch slider, allowing one to adjust the epoch and see the weights adjust accordingly.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

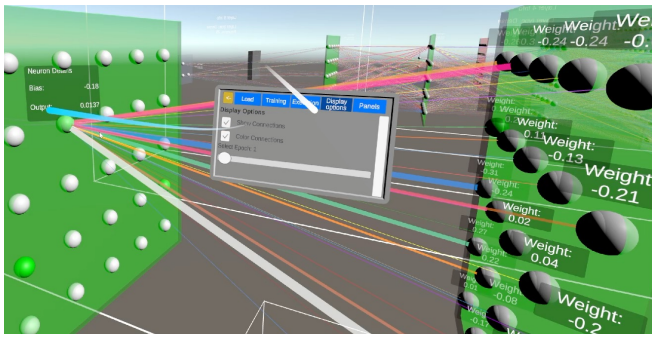


Figure 33. Single node selection displays relevant (colorized) connections with input weight values from previous layer (output connections not seen).

D. Model Evolution Support

Model evolution capability support is showcased via NNs to demonstrate its feasibility and illustrate realization features, but can be equivalently realized for other model types.

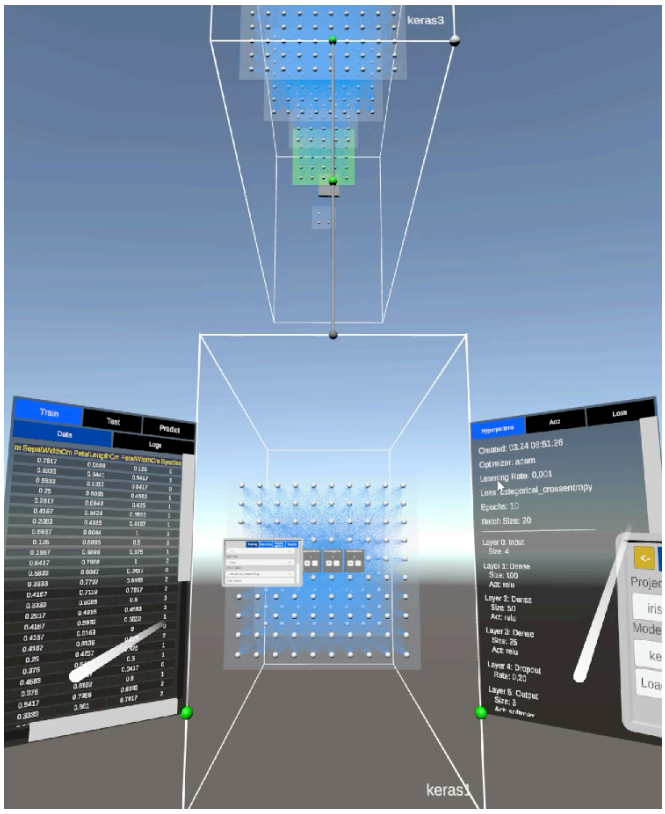


Figure 34. NN model evolution, green/red layers indicate addition/removal.

After changing various parameters in the initial (keras1) model in VR and pressing "Train Model", a new model (keras3) appears above it with a parent-child connector to visual indicate model basis for changes as shown in Figure 34. As seen, a green panel indicates an additional layer vs. the previous (its basis, lower BB) model. Furthermore, the panels also indicate any textual differences to the basis in yellow, as shown in Figure 35. A red panel visually indicates a panel removed in later models (higher BB), as seen in Figure 36.

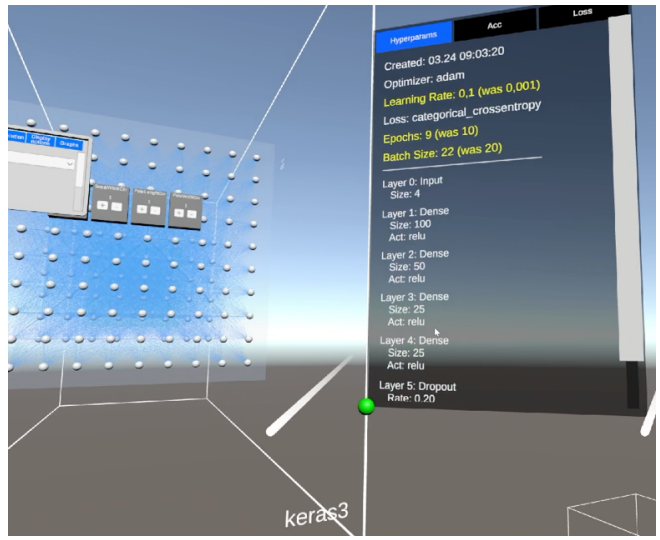


Figure 35. Information panels: text differences to prior version are depicted in yellow.

E. Scaling Support

To evaluate the scaling ability of VR to support large NN models, a mid-sized model that consists of ten (8 hidden) layers with 432 neurons counts (4, 100, 50, 25, 100, 50, 25, 50, 25, dropout, 3) was used. A large model consisting of 982 neurons across ten (8 hidden) layers (4, 100, 200, 25, 100, 200, 100, 50, 200, 3) with the connectors was colored. Our principle is to initially depict a model's reality with its inherent complexity, and then let the user decide which items to hide. Via immersive navigation, selection of an element of interest, and display filtering (such as turning off connectors or colorization), visual overload can be addressed. Based on the stakeholder's interest and intentionality, comprehension or issue analysis for large models can be supported and stakeholder collaboration opportunities with a common immersive model utilized.

F. Concurrent Multi-Model Support

To demonstrate the visual support for analysis capabilities that VR offers, loading another model starts a separate visual evolution tree, as shown in Figure 37. The capability to load and compare diverse ML models, such as comparing a RF to a NB model, is shown in Figure 38. This can support comprehension and analysis of model differences, output difference analysis, and offer decision support for potential optimizations and model choices. Furthermore, model evolution trees can be concurrently loaded with other models, as seen in Figure 39. This supports the optimization and traceability of models, which can support governance and model transparency for larger organizations managing a plethora of models.

VI. Conclusion

This paper described our VR-ML contribution, a solution concept towards immersive visualization of multiple diverse ML models and their evolution concurrently in VR. The VR-ML solution concept utilizes VR Viewpoints to provide ML

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

model type-specific guidance for immersive visualization, configuration or building, training, prediction, analysis, and interaction. Our prototype demonstrates its feasibility by realizing various immersive model viewpoints, specifically NB, RF, and NN. The case-based evaluation provides insights into the solution concept's capabilities and potential for immersively supporting VR Viewpoints using NB, RF, and NN, model evolution support, scaling, and concurrent multi-model support. This makes ML models accessible to diverse stakeholders to better comprehend, explore, analyze, and collaborate on diverse ML models and their evolution.

Future work includes realizing support for additional ML models (e.g., RNN, CNN), dynamic enhancements, support for additional ML frameworks and ML formats, and a comprehensive empirical study.

ACKNOWLEDGMENT

The author would like to thank Daniel Godeck, Marc Jaenke, Nils Jaenke, and Lukas Roufos for their assistance with the design, implementation, and screenshots.

REFERENCES

[1] R. Oberhauser, "VR-ANN: Visualization of Artificial Neural Network Models in Virtual Reality," In: Proceedings of The Second International Conference on AI-based Systems and Services (AISyS 2025), IARIA, 2025, pp. 60-65.

[2] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Global Edition, 4th ed. Pearson Education, 2021.

[3] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," Journal of microbiological methods, 43(1), 2000, pp.3-31. doi: 10.1016/S0167-7012(00)00201-3

[4] I. Wickramasinghe and H. Kalutarage, "Naive Bayes: applications, variations and vulnerabilities: a review of literature with code snippets for implementation," Soft computing, 25(3), 2021, pp.2277-2293.

[5] T. K. Ho, "Random decision forests," In: Proceedings of 3rd international conference on document analysis and recognition, vol. 1, IEEE, 1995, pp. 278-282.

[6] A. Ziegler and I. R. König, "Mining data with random forests: current options for real-world applications," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 4(1), Wiley, 2014, pp. 55-63. doi: 10.1002/widm.1114

[7] R. Oberhauser, "VR-SDLC: A Context-Enhanced Life Cycle Visualization of Software-or-Systems Development in Virtual Reality," In: Business Modeling and Software Design (BMSD 2024), LNBIP, vol 523, Springer, Cham, 2024, pp. 112-129, doi: 10.1007/978-3-031-64073-5_8.

[8] R. Oberhauser, "VR-Git: Git Repository Visualization and Immersion in Virtual Reality," 17th Int'l Conf. on Software Engineering Advances (ICSEA 2022), IARIA, 2022, pp. 9-14.

[9] R. Oberhauser, "VR-DevOps: Visualizing and Interacting with DevOps Pipelines in Virtual Reality," Nineteenth International Conference on Software Engineering Advances (ICSEA 2024), IARIA, 2024, pp. 43-48.

[10] R. Oberhauser, "VR-SBOM: Visualization of Software Bill of Materials and Software Supply Chains in Virtual Reality," In: Business Modeling and Software Design (BMSD 2025), LNBIP, vol 559, Springer, Cham, 2025, pp. 52-70, doi: 10.1007/978-3-031-98033-6_4.

[11] R. Oberhauser, M. Baehre, and P. Sousa, "VR-EA+TCK: Visualizing Enterprise Architecture, Content, and Knowledge in Virtual Reality," In: Business Modeling and Software Design (BMSD 2022), LNBIP, vol 453, Springer, 2022, pp. 122-140. doi: 10.1007/978-3-031-11510-3_8.

[12] R. Oberhauser, M. Baehre, and P. Sousa, "VR-EvoEA+BP: Using Virtual Reality to Visualize Enterprise Context Dynamics Related to Enterprise Evolution and Business Processes," In: Business Modeling and Software Design (BMSD 2023), LNBIP, vol 483, Springer, 2023, pp. 110-128, doi: 10.1007/978-3-031-36757-1_7.

[13] T. Hirzle et al., "When XR and AI Meet - A Scoping Review on Extended Reality and Artificial Intelligence," In: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), ACM, Article 730, 2023, pp. 1–45. doi: 10.1145/3544548.3581072.

[14] I. Yim and J. Su, "Artificial intelligence (AI) learning tools in K-12 education: A scoping review," Journal on Computers in Education, Vol. 12, pp. 93–131, 2025, doi: 10.1007/s40692-023-00304-9.

[15] D. Reiners, M. R. Davahli, W. Karwowski, and C. Cruz-Neira, "The combination of artificial intelligence and extended reality: A systematic review," Frontiers in Virtual Reality, 2, 721933, 2021, doi: 10.3389/frvir.2021.721933

[16] M. Inkarbekov, R. Monahan, and B. A. Pearlmutter, "Visualization of ai systems in virtual reality: A comprehensive review," arXiv preprint, 2023, arXiv:2306.15545.

[17] M. Bellgardt, C. Scheiderer, and T. W. Kuhlen, "An Immersive Node-Link Visualization of Artificial Neural Networks for Machine Learning Experts," 2020 IEEE International Conf. on Artificial Intelligence and Virtual Reality (AIVR), IEEE, 2020, pp. 33-36, doi: 10.1109/AIVR50618.2020.00015.

[18] C. Hilton et al., "InteractML: Making machine learning accessible for creative practitioners working with movement interaction in immersive media," In: Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology (VRST '21), ACM, Article 23, 2021, pp. 1–10. doi: 10.1145/3489849.3489879.

[19] Z. Lyu, J. Li, and B. Wang, "AIive: Interactive Visualization and Sonification of Neural Networks in Virtual Reality," 2021 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), IEEE, 2021, pp. 251-255, doi: 10.1109/AIVR52153.2021.00057.

[20] N. Meissler, A. Wohlan, N. Hochgeschwender, and A. Schreiber, "Using Visualization of Convolutional Neural Networks in Virtual Reality for Machine Learning Newcomers," 2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), IEEE, 2019, pp. 152-1526, doi: 10.1109/AIVR46125.2019.00031.

[21] A. Wohlan, N. Hochgeschwender, and N. Meissler, "Visualizing Convolutional Neural Networks with Virtual Reality," In: Proc. 25th ACM Symposium on Virtual Reality Software and Technology (VRST '19). ACM, Article 100, pp. 1–2, 2019, doi: 10.1145/3359996.3364817

[22] A. Schreiber and M. Bock, "Visualization and exploration of deep learning networks in 3D and virtual reality," In: Human-Computer Interaction International 2019 (HCII 2019) Posters, Proc. 21st International Conference on HCI (HCII 2019), CCIS, Springer International Publishing, 2019, pp. 206-211.

[23] D. Queck, A. Wohlan, and A. Schreiber, „Neural Network Visualization in Virtual Reality: A Use Case Analysis and Implementation," In: Human Interface and the Management of Information: Visual and Information Design (HCII 2022), LNCS, Springer, 2022, vol 13305, pp. 384-397, doi: 10.1007/978-3-031-06424-1_28.

[24] K. VanHorn and M. C. Çobanoğlu, "Democratizing AI in biomedical image classification using virtual reality," Virtual Reality, 26(1), pp. 159-171, 2021., doi: 10.1007/s10055-021-00550-1

[25] C. Linse, A. Hammam, and T. Martinetz, "A walk in the black-box: 3D visualization of large neural networks in virtual

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

reality," Neural Computing and Applications, vol. 34, no. 23, pp. 21237-21252, 2022.

[26] M. Možina, J. Demšar, M. Kattan, and B. Zupan, "Nomograms for visualization of naive Bayesian classifier," In: European conference on principles of data mining and knowledge discovery, Springer Berlin Heidelberg, 2004, pp. 337-348.

[27] J. Balfer and J. Bajorath, "Introduction of a methodology for visualization and graphical interpretation of Bayesian classification models," Journal of chemical information and modeling, 54(9), 2014, pp. 2451-2468.

[28] J. Ferreira, L. S. Machado, and R. M. de Moraes, "A new double weighted fuzzy hypergeometric naive bayes network and its application for user's assessment in virtual reality simulators," International Journal of Computational Intelligence Systems, 18(1), 2025, pp. 90-107.

[29] S. Welling, H. Refsgaard, P. Brockhoff, and L. Clemmensen (2016). Forest floor visualizations of random forests. arXiv preprint arXiv:1605.09196.

[30] R. H. Nsch, P. Wiesner, S. Wendler, and O. Hellwich, "Colorful Trees: Visualizing Random Forests for Analysis and Interpretation," 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2019, pp. 294-302, doi: 10.1109/WACV.2019.00037.

[31] X. Zhao, Y. Wu, D. L. Lee, and W. Cui, "iForest: Interpreting Random Forests via Visual Analytics," in IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 1, pp. 407-416, 2019, doi: 10.1109/TVCG.2018.2864475.

[32] N. Kuznetsova, M. Westenberg, K. Buchin, K. Dinkla, and S. J. Van Den Elzen, "Random forest visualization," Eindhoven University of Technology, 2014.

[33] R. Oberhauser and C. Pogolski, "VR-EA: Virtual Reality Visualization of Enterprise Architecture Models with ArchiMate and BPMN," In: Business Modeling and Software Design (BMSD 2019), LNBIP, vol. 356, Springer, Cham, 2019, pp. 170-187, doi: 10.1007/978-3-030-24854-3_11.

[34] R. Oberhauser, C. Pogolski, and A. Matic, "VR-BPMN: Visualizing BPMN models in Virtual Reality," In: Shishkov, B. (ed.) Business Modeling and Software Design (BMSD 2018), LNBIP, vol. 319, Springer, 2018, pp. 83–97, doi: 10.1007/978-3-319-94214-8_6.

[35] R. Oberhauser, "VR-GitCity: Immersively Visualizing Git Repository Evolution Using a City Metaphor in Virtual Reality," International Journal on Advances in Software, 16 (3 & 4), 2023, pp. 141-150, ISSN: 1942-2628.

[36] R. Oberhauser, "VR-GitEvo+CI/CD: Visualizing the Evolution of Git Repositories and CI/CD Pipelines in Virtual Reality," International Journal on Advances in Software, 18 (1 & 2), 2025, pp. 82-96, ISSN: 1942-2628.

[37] I. J. Akpan and M. Shanker, "The confirmed realities and myths about the benefits and costs of 3D visualization and virtual reality in discrete event modeling and simulation: A descriptive meta-analysis of evidence from research and practice," Computers & Industrial Engineering, vol. 112, pp. 197-211, 2017.

[38] S. Narasimha, E. Dixon, J. W. Bertrand, and K. C. Madathil, "An empirical study to investigate the efficacy of collaborative immersive virtual reality systems for designing information architecture of software systems," Applied Ergonomics, vol. 80, pp. 175-186, 2019.

[39] A. Fonnet and Y. Prie, "Survey of immersive analytics," IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 3, pp. 2101-2122, 2019.

[40] R. Müller, P. Kovacs, J. Schilbach, and D. Zeckzer, "How to master challenges in experimental evaluation of 2D versus 3D software visualizations," In: 2014 IEEE VIS International Workshop on 3Dvis (3Dvis), IEEE, 2014, pp. 33-36.

[41] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," MIS Quarterly, 28(1), 2004, pp. 75-105.

[42] R. Jain. Visualization -> Machine learning -> Deep learning. [Online]. Available from: https://www.kaggle.com/code/ranjeetjain3/visualization-machine-learning-deep-learning 2026.05.09

[43] TensorFlow Playground. [Online]. Available from: https://playground.tensorflow.org 2026.05.0

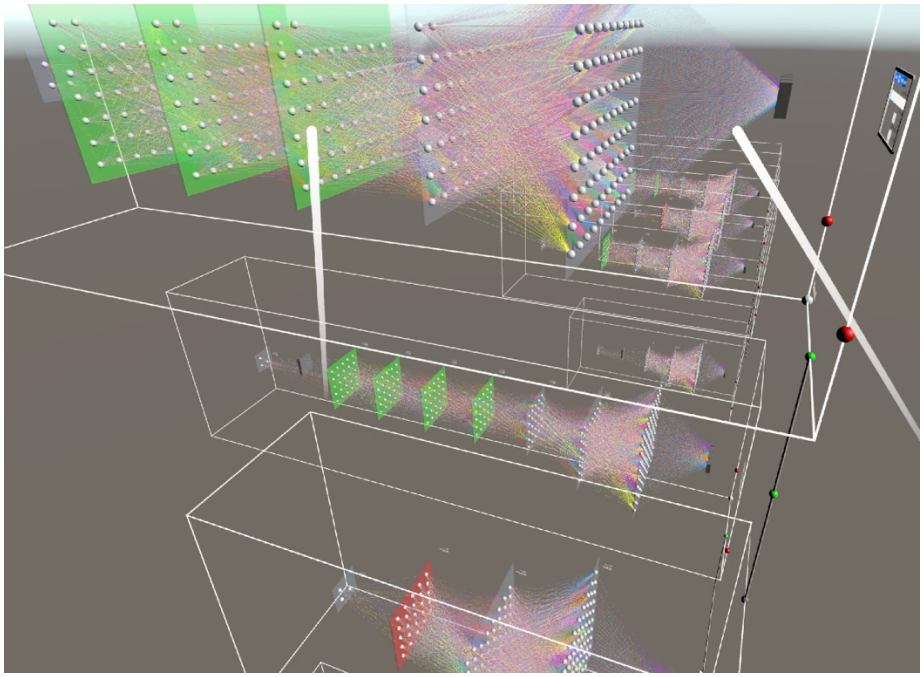


Figure 36. NN evolution view side perspective with colorized connectors. Red/green panels indicate removal/addition.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

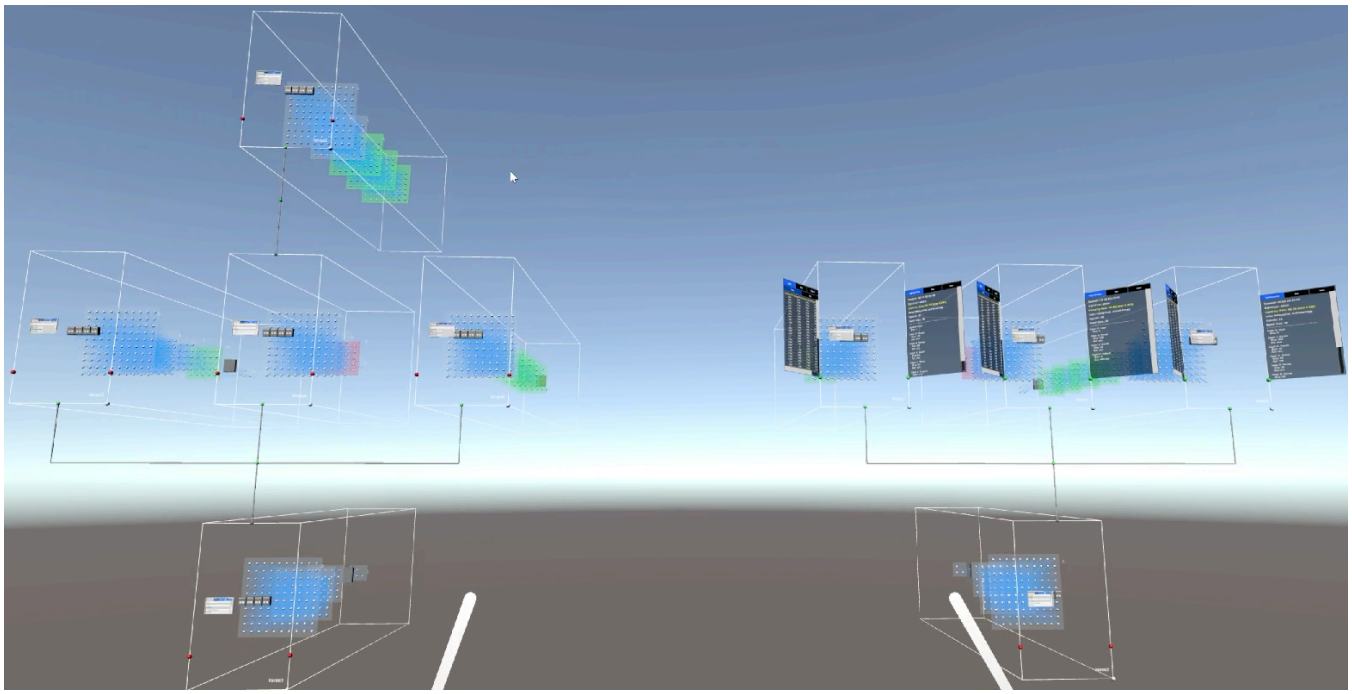


Figure 37. Loading another model (keras2 on right) starts a new evolution tree once any changes are made. Panels can be optionally turned on/off.

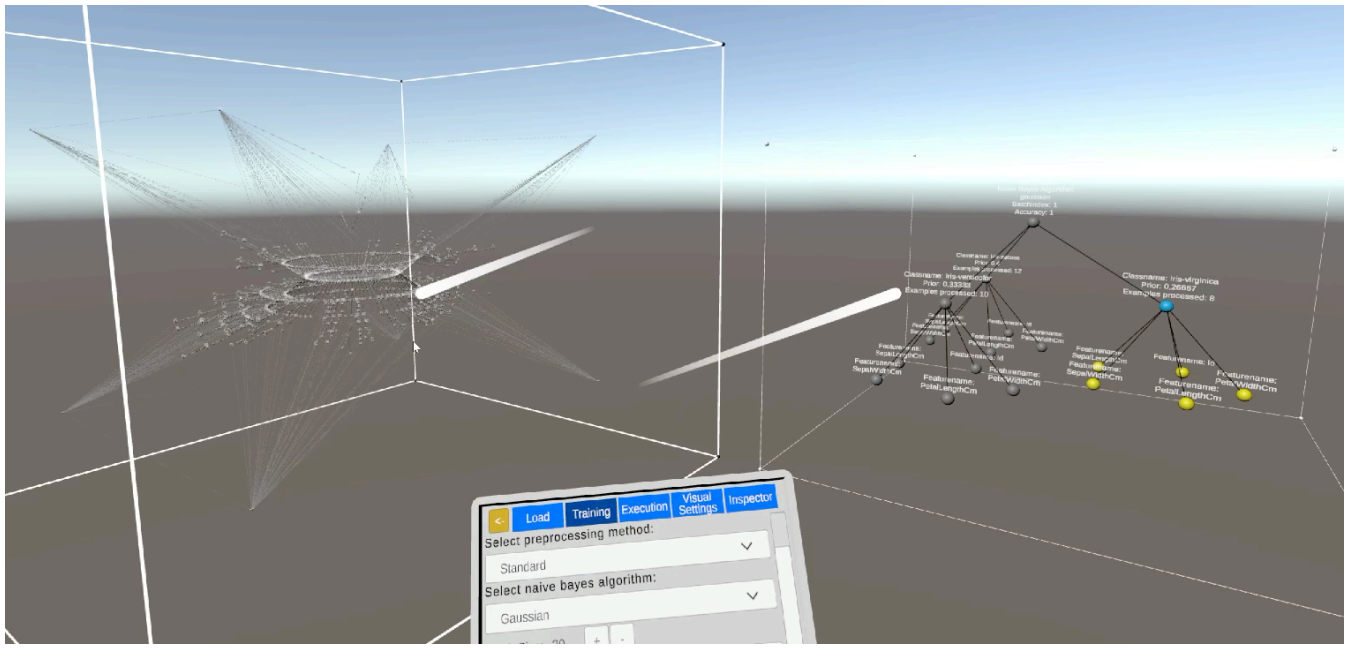


Figure 38. Concurrent dual heterogeneous models (RF left, NB right).

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

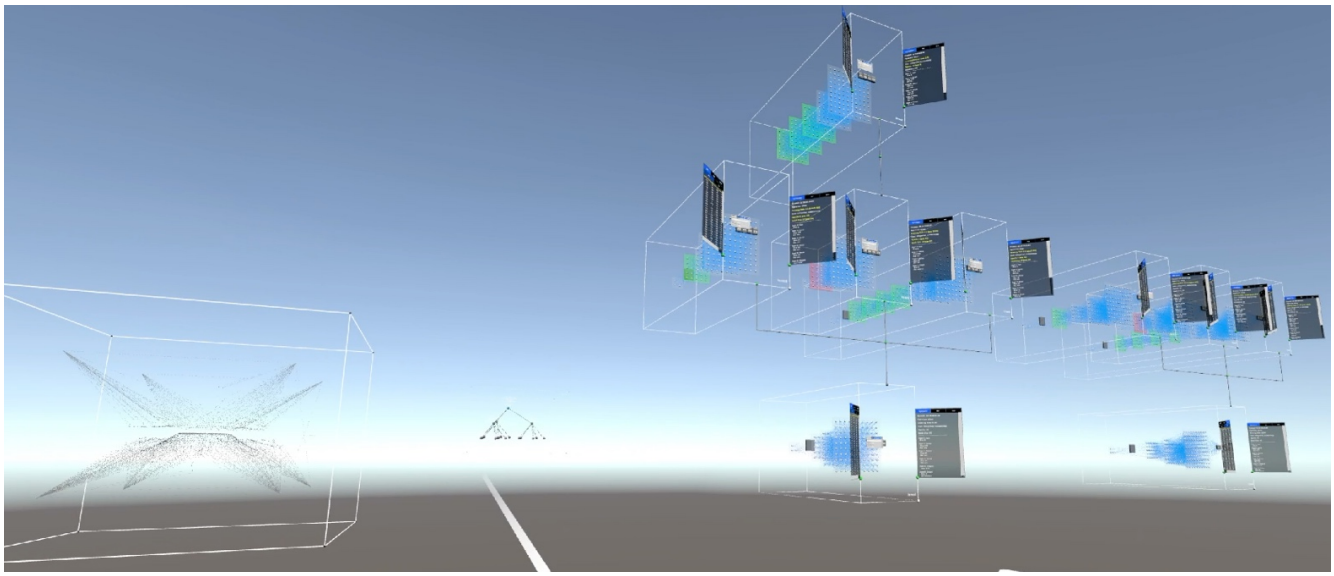


Figure 39. Diverse ML models (RF, NB, NN) visualized concurrently while depicting homogeneous model type history and evolution (two models, right).

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Brains, Brawn, and Silicon: Evaluating LLM Performance for Code Generation across Intel and Apple M3 Max Architectures

Miren Illarramendi[1], Joseba Andoni Agirre[1], Aitor Picatoste[2], Juan Ignacio Igartua[2]
[1]Software and Systems Engineering Research Group, Mondragon University, Spain
Email: {millarramendi, jagirre}@mondragon.edu
[2]Circular Economy and Industrial Sustainability, Mondragon University, Spain
Email: {apicatoste, jigartua}@mondragon.edu

***Abstract*—This research presents a comparative analysis of various Large Language Models (LLMs) for code generation tasks executed without dedicated Graphics Processing Units (GPUs), focusing exclusively on CPU-based environments. While GPUs are typical for large-scale inference, many users rely on CPUs due to resource constraints. However, high-performance chips like the Apple M3 Max introduce a new paradigm for local execution. This study addresses the gap in understanding how LLMs perform across different portable hardware tiers, focusing on the trade-offs between model logic and hardware power. The study evaluates key performance metrics including inference time, code generation accuracy, CPU and memory usage, and energy consumption. To assess the influence of hardware architecture on model efficiency, experiments were conducted across two distinct hardware tiers: a resource-constrained Intel x86 CPU platform and a high-performance ARM-based system powered by the Apple M3 Max. By performing repeated experiments under controlled conditions, we analyze the impact of model size, architectural optimization, and hardware characteristics on computational efficiency. Energy consumption is measured using tools such as CodeCarbon, enabling a detailed examination of the environmental footprint associated with CPU-based LLM execution. The comparison between traditional x86 architectures and modern unified memory ARM systems provides insights into the trade-offs between model precision, computational resource usage, and energy efficiency. The findings contribute practical guidance for developers and researchers seeking to balance performance, sustainability, and hardware constraints in low-resource or portable computing environments, highlighting how architectural differences influence both operational efficiency and environmental impact.**

Keywords-LLMs; GenIA; GreenComputing; Code Generation; Energy Consumption; Sustainability; Apple M3 Max.

I. Introduction

The rapid advancement of LLMs has revolutionized various fields, including Natural Language Processing (NLP), code generation, and even machine translation. Models like GPT-3, DeepSeek, and Llama have shown remarkable capabilities in tasks ranging from text generation to understanding and generating code. However, these models are computationally expensive and require significant resources, particularly during the training and inference stages [2] [3].

While GPUs are typically the hardware of choice for running large-scale machine learning models due to their high parallel processing capabilities, not all environments have access to dedicated GPUs. Many users, particularly those in resource-constrained settings or utilizing cloud computing, must rely on local CPUs for model inference. CPUs, though less powerful than GPUs in terms of parallel processing, are widely available and more energy-efficient in certain use cases, especially for smaller models or lightweight tasks [4].

Despite the growing use of LLMs in production environments, there is a lack of comprehensive analysis comparing the performance and sustainability of these models when executed on CPUs versus GPUs. The existing literature focuses mainly on GPU-based performance, leaving a gap in understanding how LLMs perform in real-world scenarios where only CPU resources are available. Some research has pointed out that the energy consumption of LLMs is often underestimated in most studies, with the environmental impact becoming a significant concern when deploying models at scale.

This study aims to address this gap by conducting a comparative analysis of the performance of various LLMs for code generation tasks when executed in GPU-based environments remotely. Specifically, we will focus on several key metrics, including inference time, code generation accuracy, energy consumption, and computational cost. The initial phase will involve measuring the cost of running inference from our local CPU to understand the energy and computational efficiency of remote execution. The next step will be to extend this analysis by deploying the LLMs directly on our CPU for inference, allowing us to compare performance and resource usage when running these models in resource-constrained environments. Models to be evaluated include "gpt-4o", "gpt-4-turbo", "gpt-3.5-turbo", "gpt-4o-mini", "meta-llama/Meta-Llama-3-8B-Instruct", "alpindale/WizardLM-2-8x22B", and "Qwen/Qwen3-235B-A22B-Instruct-25072". These models have been selected due to their variety in size and diverse platforms (OpenAI, HuggingFace), providing a comprehensive comparison of different model architectures and inference performance across various levels of complexity and computational demands.

The primary contribution of this research is the systematic evaluation of large language models (LLMs) for code generation under real inference conditions, with particular emphasis on accuracy, execution efficiency, energy consumption, and associated CO_2 emissions. Unlike studies that focus solely on benchmark accuracy, this work integrates performance and environmental sustainability metrics, providing a holistic perspective on LLM deployment trade-offs.

Specifically, we measure and monitor code generation tasks executed across different LLM families—both proprietary cloud-based models and open-weight models—on heteroge-

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

neous hardware platforms. By analyzing inference behavior in resource-constrained environments, we aim to quantify how model scale, hardware architecture, and deployment configuration jointly influence the accuracy–efficiency–sustainability balance. This integrated analysis provides practical guidance for researchers, developers, and organizations seeking cost-effective and environmentally responsible AI deployment strategies.

It is important to clarify that this study focuses exclusively on *inference-time evaluation*. Model training and fine-tuning phases are not considered. Furthermore, part of the experimental evaluation relies on remote execution (API-based inference), where energy measurements may be influenced by external factors such as network latency, backend infrastructure optimizations, and limited transparency into proprietary data-center energy usage. These constraints are acknowledged as inherent limitations of current cloud-based evaluation methodologies.

A. The Hardware Democratization Challenge

Although state-of-the-art (SotA) LLMs have demonstrated remarkable capabilities, their widespread adoption is constrained by substantial hardware requirements. Traditionally, efficient execution of frontier models has required high-end data-center infrastructures equipped with specialized accelerators such as NVIDIA H100 or A100 GPUs. While suitable for centralized cloud services, this paradigm raises concerns regarding data privacy, operational costs, vendor dependency, and inference latency.

As a result, there is a growing interest in *Local AI*, where models are executed directly on user devices. However, democratizing access to LLMs requires addressing the fundamental hardware limitations of consumer-grade systems.

B. The CPU Inference Gap

For most users, local hardware consists primarily of laptops and workstations relying on Central Processing Units (CPUs), rather than dedicated AI accelerators. Transformer-based architectures are highly parallel and computationally intensive, typically benefiting from GPUs equipped with tensor cores and high-bandwidth memory (HBM). CPUs, by contrast, are traditionally perceived as suboptimal for large-scale matrix operations.

Despite this limitation, CPUs remain the most ubiquitous computing resource worldwide. Enabling efficient LLM inference on CPUs would significantly expand accessibility, reducing dependence on specialized infrastructure. This study investigates whether modern CPU architectures can narrow the so-called "CPU inference gap," particularly when evaluating energy efficiency alongside performance.

C. The Rise of ARM Architectures and Unified Memory

The emergence of Apple Silicon (M1, M2, M3 series) represents a paradigm shift in personal computing. These ARM-based architectures introduce a unified memory model in which RAM is dynamically shared between CPU and GPU components. This design reduces data transfer overheads associated with PCI-Express bottlenecks typical in traditional x86 systems.

This research evaluates whether such architectural innovations translate into measurable improvements in inference efficiency for LLM workloads. By comparing Intel x86 systems with Apple ARM-based systems, we aim to determine whether modern consumer hardware can deliver near-accelerator efficiency for inference tasks without relying on discrete GPUs.

D. Research Objectives

To address these challenges, this paper conducts a structured comparative evaluation focusing on four main dimensions:

1) **Performance Profiling:** Measuring inference time and throughput across different models and hardware configurations to understand latency scaling behavior.
2) **Architectural Comparison:** Quantifying performance and efficiency differences between traditional Intel x86 platforms and Apple ARM-based architectures.
3) **Sustainability Analysis:** Estimating energy consumption (kWh per inference) and CO_2 emissions to promote Green AI practices and environmentally responsible deployment strategies.
4) **Quality Evaluation:** Assessing functional correctness of generated code using Pass@1 metrics to ensure that efficiency gains do not compromise output quality.

By integrating these dimensions, this study contributes to bridging the gap between high-performance AI research and sustainable, hardware-aware deployment strategies.

E. Next Steps: Local Execution Benchmarking

As a continuation of this work, the next experimental phase will involve running selected open-weight models entirely locally on both CPU and GPU configurations. This will enable direct comparison between remote API-based execution and on-device inference, further clarifying trade-offs in latency, energy consumption, carbon footprint, and operational control.

Such an extension will allow the identification of optimal model–hardware pairings for resource-constrained environments and contribute to the development of reproducible and sustainable AI evaluation methodologies.

The remainder of the paper is organized as follows: Section II and Section III cover the background and related work, respectively, providing the foundational context for this study. In Section IV, we describe the experimental setup and methodologies employed. Section V presents the experimental results, focusing on performance, throughput, code generation accuracy, and energy efficiency. Section VI offers an in-depth discussion of the evaluation, interpreting the results and their implications. Finally, Section VII concludes the paper and proposes directions for future work.

II. Background

The rapid evolution of Large Language Models (LLMs) has been driven by advances in deep learning architectures, large-scale pretraining, and the availability of massive datasets. Foundational models such as GPT-3 [5] and BERT [6] demonstrated that scaling parameter counts and training data significantly improves performance across a broad range of natural language

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

processing (NLP) tasks, including text generation, translation, and question answering. Subsequent models, such as Codex [7], extended these capabilities to software engineering tasks, enabling natural language–to–code generation.

Despite their impressive performance, LLMs require substantial computational resources. Training large-scale models involves distributed GPU clusters and significant energy expenditure. Although inference is less demanding than training, it remains computationally intensive, particularly for real-time or interactive applications. GPUs are typically used to accelerate inference through parallel matrix operations; however, many deployment environments—such as edge devices, educational settings, or small-scale development infrastructures—rely primarily on CPUs.

As model sizes continue to grow, concerns regarding computational cost and environmental impact have intensified. For instance, GPT-3, with 175 billion parameters [5], illustrates the scaling trend that improves performance but also increases energy demand. Prior research has highlighted the environmental footprint associated with large-scale models, emphasizing the importance of energy-aware AI practices [2], [3]. Consequently, optimizing inference efficiency—particularly in CPU-based or resource-constrained settings—has become a critical research direction.

At the same time, the demand for automated code generation has increased significantly. LLM-based coding assistants can accelerate development by generating boilerplate code, suggesting completions, automating refactoring, and assisting debugging. Models trained on large code repositories, such as Codex [7], demonstrate that LLMs can produce syntactically valid and semantically meaningful programs from natural language prompts. This capability has practical implications for productivity and software quality.

However, deploying LLMs for code generation introduces unique constraints. Unlike natural language, source code must be syntactically precise and logically consistent; even minor token-level errors can invalidate the output. Furthermore, interactive development workflows demand low latency. In integrated development environments (IDEs), delays of several seconds can disrupt developer productivity, regardless of the underlying model's accuracy.

A. Evolution of Large Language Models

The progression from Recurrent Neural Networks (RNNs) to the Transformer architecture introduced by Vaswani et al. marked a turning point in language modeling. The Transformer's attention mechanism enabled parallelization and efficient scaling, paving the way for large models such as BERT [6] and GPT-3 [5].

Empirical scaling laws revealed that increasing model size often leads to emergent capabilities, including few-shot reasoning and improved generalization. However, this scaling also introduces substantial hardware requirements. Models with hundreds of billions of parameters require significant GPU memory (VRAM), effectively limiting local execution to specialized infrastructure.

In response, recent research has explored smaller yet efficient alternatives—often referred to as Small Language Models (SLMs)—such as Llama-3-8B or Mistral-7B. These models aim to preserve strong performance while reducing memory footprint and enabling deployment on consumer-grade hardware.

B. LLMs for Code Generation

Code generation represents a specialized application domain with distinct evaluation requirements. Benchmarks such as HumanEval and MBPP assess functional correctness of generated programs, typically through automated test suites. While these benchmarks provide standardized accuracy metrics, they often overlook computational cost and environmental considerations.

Moreover, existing literature largely focuses on benchmark accuracy without systematically analyzing inference efficiency across different hardware configurations. There is limited empirical work comparing execution time, energy consumption, and carbon footprint of LLM-based code generation in CPU-centric environments.

An additional gap concerns remote inference. Many proprietary models operate via cloud-based APIs, where computational resources are abstracted from the end user. In such scenarios, developers query remote systems from local CPU environments, but the energy cost and environmental impact of this interaction remain partially opaque. Understanding these trade-offs—between local execution and remote cloud-based inference—is essential for informed and sustainable deployment decisions.

These gaps motivate the need for comprehensive evaluations that integrate accuracy, latency, and sustainability metrics across both local and remote execution contexts. Such analyses are critical for determining the practical feasibility of LLM-assisted software development in resource-constrained and environmentally conscious settings.

C. Selected Models

The selection of models for these experiments provides a comprehensive comparison across different architectures and computational demands. The evaluation includes proprietary models from OpenAI (`gpt-4-turbo`, `gpt-3.5-turbo`, `gpt-4o-mini`) and open-source models from Hugging Face (`Meta-Llama-3-8B`, `WizardLM-2-8x22B`, `Qwen-2`).

1) Proprietary Models: These models are accessed via API and represent the current state-of-the-art in closed-source LLMs.

- GPT-4 Turbo: The immediate predecessor to GPT-4o, this model features a 128k context window and was trained with a cutoff date up to December 2023. It utilizes a Mixture-of-Experts (MoE) architecture to balance high performance with inference efficiency.
- GPT-3.5 Turbo: A highly optimized, efficient model that served as the industry standard for speed and cost-effectiveness prior to the release of the GPT-4o mini series. It is a dense model known for its reliability in lower-complexity tasks.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

- GPT-4o mini: OpenAI's most cost-efficient small model, designed to replace GPT-3.5 Turbo. It supports multimodal inputs and offers reasoning capabilities superior to GPT-3.5, leveraging the architectural advancements of the GPT-4o family.

2) Open-Source Models: The selected open-weight models cover a range of sizes, from efficient 7B parameter models to large-scale Mixture-of-Experts systems.

- Meta-Llama-3-8B: The 8-billion parameter iteration of the Llama 3 family. It is a standard dense decoder-only transformer trained on a massive dataset of over 15 trillion tokens. It employs a tokenizer with a vocabulary size of 128k, significantly improving performance on multilingual tasks compared to Llama 2.
- WizardLM-2-8x22B: A large-scale Mixture-of-Experts (MoE) model based on the Mixtral 8x22B architecture. It is fine-tuned specifically for complex instruction following and reasoning. Despite having a total of 141B parameters, it only uses 39B active parameters per token, allowing for efficient inference relative to its total size.
- Qwen-2: A dense transformer architecture employing SwiGLU activation, Rotary Positional Embeddings (RoPE), and QKV bias. The model is noted for its strong performance in coding and mathematics benchmarks and its support for a context window of up to 32k tokens in standard configurations.

TABLE I: Summary of Selected Models

Model	Type	Arch.	Access
GPT-4 Turbo	Prop.	MoE	API
GPT-3.5 Turbo	Prop.	Dense (est.)	API
GPT-4o mini	Prop.	–	API
Llama-3-8B	Open	Dense (8B)	Local/HF
WizardLM-2-8x22B	Open	MoE (141B)	Local/HF
Qwen-2	Open	Dense	Local/HF

D. Justification of Model Selection and Theoretical Baseline

The selected models have been chosen to represent three different capacity regimes: large-scale foundation models, medium-sized open models, and compact lightweight models. The objective is to theoretically analyze how model dimension and training knowledge influence expected accuracy and computational efficiency.

From a theoretical standpoint, model size (number of parameters) correlates with representational capacity. Larger models typically capture more complex patterns and exhibit stronger reasoning and generalization capabilities due to broader training corpora. However, this increased capacity results in higher computational cost, increased inference latency, and greater energy consumption.

Conversely, smaller models are expected to be computationally efficient and suitable for edge or energy-constrained environments, though potentially at the expense of reduced reasoning accuracy in complex tasks.

This categorization defines a theoretical baseline that will later be validated through empirical experimentation.

a) Theoretical Hypotheses:

- H1: The large-scale model is expected to achieve the highest accuracy due to its larger parameter space and broader training knowledge.
- H2: The small-scale model is expected to provide the highest energy efficiency and lowest latency.
- H3: The medium-scale model is expected to offer the best trade-off between accuracy and efficiency.

These theoretical expectations define the reference framework for subsequent experimental validation, where accuracy, inference time, and energy consumption will be quantitatively measured.

III. Related work

The growing reliance on LLMs for tasks, such as code generation, text generation, and question answering has significantly advanced the field of artificial intelligence. However, these models, especially large-scale ones like GPT-3 [5], Codex [7], and BERT [6], have raised concerns regarding their environmental impact due to their substantial energy consumption and carbon footprint. As these models become larger, the need for energy-efficient deployment methods becomes critical, particularly when leveraging resources such as CPUs instead of GPUs, which are commonly used in research environments.

A. Energy Consumption and Sustainability in AI

The environmental impact of LLMs has been a topic of growing concern in recent research. Strubell et al. [2] highlighted the significant energy consumption required to train and run models like GPT-3, estimating that the carbon emissions of training such models can rival those of several cars over their lifetimes. This study emphasizes the need for developing models that are not only accurate but also energy-efficient, promoting the idea of Green AI. However, this research focuses primarily on the training phase and the larger-scale infrastructures typically used for training these models, rather than on their inference phase or CPU-based execution.

Schwartz et al. [3] further expanded on the concept of sustainable AI, advocating for a shift toward models that prioritize resource efficiency. They call for reducing the carbon footprint of deep learning models and propose that energy-efficient algorithms should be a focus in model design. However, their work lacks a focus on real-world inference scenarios, particularly in environments where GPUs are unavailable or impractical for deployment.

Xu et al. [8] provided a comprehensive survey on strategies to improve energy efficiency in deep learning models, addressing the growing need to reduce the environmental impact of AI systems. They reviewed a variety of techniques, including model compression, pruning, quantization, and efficient data usage, all aimed at optimizing the energy consumption of machine learning models. These methods can significantly

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE II: Theoretical Comparison of Selected Models

Category	Example Model	Params	Expected Accuracy	Latency	Energy Efficiency
Large-scale	GPT-4 class (>50B)	Very High	Very High	High	Low
Medium-scale	DeepSeek / LLaMA-13B	High	High	Medium	Medium
Small-scale	TinyLlama / 7B models	Moderate	Medium	Low	High

reduce the computational load during inference, particularly for large-scale models. While this work offers valuable insights into improving the energy efficiency of deep learning systems, it does not specifically focus on the trade-offs involved in running LLMs, such as GPT-3 or Codex, on CPUs for code generation tasks—an area central to our research.

B. CPU-Based Inference Optimization and Resource-Aware Deployment

Making LLM inference viable on CPU-centric systems has become an active area of research, particularly as organizations seek to reduce hardware dependency and environmental impact. Several optimization techniques have emerged to improve execution efficiency on commodity hardware.

One of the most widely adopted strategies is **quantization**, which reduces numerical precision from 16-bit floating point (FP16) to lower-bit representations such as 8-bit, 4-bit, or even 2-bit integers. This significantly decreases memory footprint and computational overhead, allowing larger models to run on systems with limited RAM. Frameworks such as `llama.cpp` have demonstrated that even 70B-parameter models can operate on machines with approximately 32GB of RAM through aggressive quantization techniques.

Another important approach is **distillation**, where smaller "student" models are trained to replicate the behavior of larger "teacher" models. Distillation reduces model size and inference cost while attempting to preserve task performance. This technique is particularly relevant in CPU-based environments, where memory and computation resources are constrained.

More recently, **speculative decoding** has been proposed as a latency-reduction strategy. In this method, a smaller model generates draft tokens that are subsequently verified or corrected by a larger model. This hybrid mechanism reduces average inference time while maintaining output quality. Although promising, the energy implications of speculative decoding across different hardware architectures remain underexplored.

While these optimization strategies improve feasibility on CPUs, there is limited literature systematically comparing their energy efficiency across x86 and ARM-based architectures under controlled experimental conditions. Most existing evaluations focus primarily on latency and memory reduction rather than carbon footprint or energy-per-token metrics.

Beyond algorithmic optimizations, inference efficiency is also closely linked to resource allocation strategies. Patterson et al. [9] examined the energy and carbon implications of large-scale deep learning systems, highlighting trade-offs between CPU and GPU utilization. Their work emphasized that hardware selection significantly influences environmental impact. However, the analysis was primarily centered on training workloads and did not explicitly address LLM inference or code generation tasks.

Subsequent work [4] proposed carbon-aware machine learning practices, including scheduling and hardware-aware deployment strategies for energy-efficient inference. While these studies provide important conceptual frameworks for sustainable AI, they do not specifically evaluate large language models in CPU-based settings or compare architectural differences such as x86 versus ARM processors.

In parallel, the integration of energy monitoring tools into DevOps pipelines has gained attention. Tools such as CodeCarbon [10] enable real-time estimation of energy consumption and CO_2 emissions during training and inference. When embedded into continuous integration workflows, these tools allow practitioners to track environmental impact alongside performance metrics. However, such monitoring is typically applied to smaller models or research experiments and is rarely systematically integrated into large-scale LLM inference pipelines, particularly in CPU-centric deployments.

Similarly, MLFlow has been explored for experiment tracking and resource management in production systems [11]. By logging performance metrics and managing model versions, MLFlow facilitates adaptive deployment strategies and cost optimization. Nevertheless, current literature does not sufficiently combine performance tracking with energy-aware metrics in the context of LLM-based code generation tasks.

Overall, while substantial progress has been made in inference optimization techniques and resource management frameworks, a comprehensive evaluation integrating CPU-based optimization, cross-architecture comparison (x86 vs. ARM), and energy monitoring remains largely absent. Addressing this gap is essential for developing sustainable and hardware-aware deployment strategies for LLM-driven code generation systems.

C. Code Generation with LLMs

The application of LLMs for code generation has gained significant attention, particularly with models such as Codex [7], which are specifically designed to generate programming code from natural language prompts. Codex has shown great potential in automating code completion, bug fixing, and refactoring tasks, but there is a lack of research on how these models perform when executed on CPU-based systems as opposed to GPUs.

A recent study by Arora et al. [12] introduced SetupBench, a benchmark designed to evaluate the ability of LLM agents to bootstrap development environments autonomously. The benchmark comprises 93 tasks spanning various programming

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

languages, database engines, and multi-service orchestration scenarios. The evaluation of OpenHands, a state-of-the-art coding agent, revealed low success rates across task categories, particularly in repository setup and local database configuration. The study identified substantial inefficiencies in agent exploration strategies, with a significant percentage of actions being unnecessary compared to optimal human behavior. These findings highlight gaps in current agents' practical environment-bootstrap capabilities.

However, the research does not investigate the inference efficiency of these models when deployed in resource-constrained environments or on CPUs, which is a critical gap in the current body of literature. Furthermore, their focus was mainly on cloud-based models and did not consider the potential environmental impact of running these models in cloud infrastructures, where energy consumption and carbon footprint can vary significantly depending on the hardware used.

D. Gap in Literature

While the literature provides a strong foundation for understanding the energy consumption and performance of deep learning models, particularly in large-scale environments using GPUs, there is limited research specifically addressing the trade-offs and performance of LLMs for code generation when executed in CPU-only environments. Most of the existing studies focus on training and GPU-based inference, overlooking the operational efficiency and sustainability of running LLMs in low-resource environments where only CPUs are available.

Energy-efficient deployment strategies using tools like MLFlow and CodeCarbon remain underexplored for LLMs, particularly in real-time inference tasks like code generation. This paper addresses this gap by comparing the CPU-based performance and energy efficiency of various LLMs, focusing on code generation and incorporating energy monitoring through MLFlow and CodeCarbon to evaluate inference performance and environmental impact in resource-constrained environments.

IV. Experimental Setup

This section describes the experimental setup for evaluating the performance, energy efficiency, and environmental impact of various LLMs in code generation tasks. The selected models include proprietary LLMs accessed via remote APIs as well as open-weight models available through Hugging Face repositories. Performance monitoring is conducted using MLFlow, while energy consumption and estimated CO_2 emissions are tracked using CodeCarbon to ensure reproducible and standardized measurement of sustainability metrics.

Given that the models are accessed remotely through API calls, the evaluation focuses on the interaction between the client-side hardware platform (Intel x86 vs. Apple ARM architectures) and the observed latency, local energy usage, and output correctness. The goal is to understand whether differences in hardware architecture influence inference behavior and sustainability when invoking cloud-hosted LLMs.

A. Research Questions

To structure the experimental analysis, we formulate the following Research Questions (RQs):

- **RQ1 (Performance):** How does inference latency for remote API-based LLM calls vary when executed from different client systems (Intel x86 vs. Apple ARM architectures)?
- **RQ2 (Sustainability):** What is the local energy consumption and estimated CO_2 footprint associated with remote inference requests when initiated from x86 versus ARM-based systems?
- **RQ3 (Accuracy):** Does model accuracy remain consistent across remote API calls when accessed from different hardware platforms, or are there observable differences in output correctness due to system-level factors?

B. Hardware and Operating System

The experimental evaluation has been conducted on two alternative computing platforms in order to analyze potential variations in performance and energy behavior across different hardware architectures and operating systems.

- System 1 (Legacy x86):
  - **CPU:** Intel Core i5-1135G7 (4 cores, 8 threads, up to 4.2 GHz).
  - **RAM:** 16 GB DDR4.
  - **OS:** Windows 11 Pro.
  - **Context:** Represents a resource-constrained environment common in general office settings.
- System 2 (Modern ARM):
  - **CPU:** Apple M3 Pro (12-core CPU, 18-core GPU - GPU not used for main inference logic).
  - **RAM:** 36 GB Unified Memory.
  - **OS:** macOS Sonoma.
  - **Context:** Represents a high-performance portable workstation capable of loading larger models into memory.

The inclusion of both a resource-constrained x86-based architecture (Intel i5, Windows) and a high-performance ARM-based architecture (Apple M3 Max, macOS) enables a comprehensive comparative analysis of inference latency and energy efficiency across heterogeneous computing environments. This distinction is particularly relevant for addressing the "Hardware Democratization Challenge," as it allows for an empirical evaluation of the "CPU inference gap" by comparing ubiquitous office hardware against modern systems featuring unified memory architectures. By utilizing this multi-tier setup, the study transitions from a Phase 1 remote API-based baseline to Phase 2 local execution, facilitating a deeper assessment of how specific architectural designs influence both the performance-per-watt efficiency and the environmental sustainability of large language model workloads.

From a Green AI perspective, evaluating the same models under different processor architectures allows us to examine the trade-off between accuracy and energy consumption in realistic deployment scenarios. This comparison contributes to understanding whether architectural differences (CISC vs.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

ARM-based designs) significantly impact performance-per-watt efficiency during inference.

C. LLMs Selected for the Experiments

The models shown in Table III will be evaluated for code generation in C programming tasks from OpenAI and HuggingFace (via Novita as inference provider):

These models represent a range of architectures, including large-scale models like gpt-4-turbo and optimized models like gpt-4o-mini.

D. Performance Monitoring and Energy Consumption Measurement

To evaluate the energy consumption and carbon emissions, the following tools will be employed:

- MLFlow will be integrated to monitor and track the inference time, accuracy, and computational cost of each model.
- CodeCarbon will be used to track CO_2 emissions and energy consumption for each inference task, helping assess the environmental impact of running LLMs.

E. Inference Tasks for the LLMs

The models will generate C source code for a set of problems, which are as follows:

- Prime Number Check: *Prompt: Write a C program that checks if a number is prime. The program validation returns 1 if it is prime and 0 if not. The number to check is 11.*
- Finding the Greatest of Three Integers: *Prompt: Write a C program that defines three integer variables and prints the greatest of them. The numbers to check are 11, 22, and 33.*
- Even/Odd Check: *Prompt: Write a C program that returns 1 if the input number is even, and 0 if it is odd. The input number for testing will be 122.*
- Absolute Difference Calculation: *Prompt: Write a C program that calculates the absolute difference between two numbers. The input numbers are -122 and 11.*
- Sum of Digits: *Prompt: Write a C program that calculates the sum of the digits of the input number. The input number is 123.*

Each task was repeated 10 times for each model to ensure that the results are statistically significant and to account for potential variations in model performance across multiple runs. The generated code will be validated using gcc to ensure correctness, and the inference time, accuracy, energy consumption, and computational cost will be tracked.

F. Evaluation Framework and Experimental Design

This study evaluates the performance, sustainability, and practical feasibility of large language models (LLMs) for code generation tasks under remote inference conditions. The experimental design is structured around clearly defined evaluation criteria, measurement tools, and phased experimentation to ensure methodological consistency and reproducibility. Figure 1 summarizes the overall evaluation workflow, from prompt execution to metric collection and cross-platform comparison.

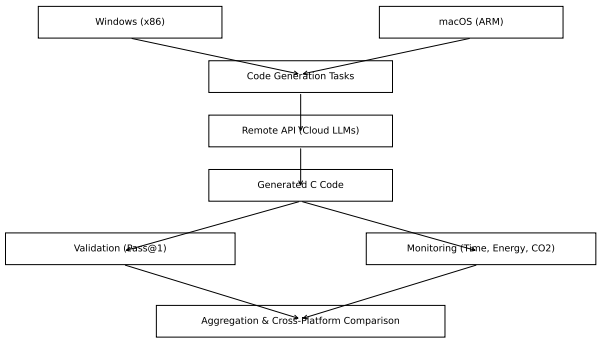


Figure 1: Evaluation pipeline for remote API-based inference from two client systems (Windows x86 and Apple Silicon ARM).

1) Evaluation Criteria: The assessment focuses on four main dimensions:

- **Inference Time:** The total time required by the model to generate the requested C code.
- **Code Generation Accuracy:** The functional correctness of the generated code, verified through compilation and execution against predefined unit tests.
- **Energy Consumption:** The client-side energy usage associated with performing remote inference requests.
- **Computational Cost:** The monetary cost of executing inference through remote APIs (e.g., GPT-4o and related cloud-based models).

2) Experimental Phases: The experimental workflow is structured in two phases:

- **Phase 1 – Remote API-Based Inference:** The primary phase of this study evaluates models accessed through remote APIs. Inference latency, accuracy, energy consumption, and cost are measured when invoking the models from two different hardware platforms (Intel x86 and Apple ARM architectures).
- **Phase 2 – Local CPU Deployment:** As a continuation of this research, future experiments will involve executing open-weight LLMs locally on CPU-based systems. This phase will enable direct comparison between remote and local execution, assessing trade-offs in latency, energy efficiency, and operational cost.

The current study focuses on Phase 1, establishing a baseline for remote inference analysis before extending the evaluation to local CPU deployment scenarios and it is the extension of a previous work presented in [1].

3) Metrics and Measurement Tools: A structured set of quantitative metrics is employed to ensure rigorous evaluation.

a) Performance Metrics:

- **Inference Time (T_{inf}):** The total wall-clock time measured from sending the prompt to receiving the final generated token.
- **Throughput (TPS):** Tokens per second, calculated as:

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE III: Selected LLMs for the Experiments

From OpenAI	From Hugging Face
gpt-4-turbo	meta-llama/Meta-Llama-3-8B-Instruct
gpt-3.5-turbo	alpindale/WizardLM-2-8x22B
gpt-4o-mini	Qwen/Qwen3-235B-A22B-Instruct-2507

$$TPS = \frac{N_{tokens}}{T_{inf}} \quad (1)$$

b) Energy Metrics: Energy consumption is measured using the **CodeCarbon** Python package. On Intel systems, measurements rely on the RAPL (Running Average Power Limit) interface, while on macOS systems the `powermetrics` tool is used.

Total energy consumption is computed as:

$$E_{total} = \int_{t_{start}}^{t_{end}} P(t)\, dt \quad (2)$$

where $P(t)$ represents instantaneous power draw in Watts during inference.

c) Correctness Metric: Functional correctness is evaluated using the **Pass@1** metric, validated against a suite of predefined unit tests for each C programming task:

$$\text{Pass@1} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{1}(s_i \text{ passes all tests}) \quad (3)$$

where k is the number of evaluated samples and $\mathbf{1}(\cdot)$ is the indicator function.

By combining performance, energy, cost, and correctness metrics within a phased experimental design, this framework enables a comprehensive analysis of the trade-offs involved in deploying LLMs for code generation tasks in resource-constrained environments.

V. Results

The experimental results are presented considering two different hardware platforms: a Windows-based x86 system and an Apple Silicon (M3 Pro) system. Since hardware architecture may significantly influence latency and energy behavior, the evaluation is structured in three stages: (i) results on the Windows system, (ii) results on the macOS system, and (iii) a cross-platform comparative analysis.

The evaluation focuses on three key dimensions: Performance and Throughput, Code Generation Accuracy, and Energy Efficiency.

The analyzed models include proprietary and open-weight large language models obtained from OpenAI and Hugging Face, with parameter sizes ranging from 3.8B to 235B parameters. All models were executed under the same experimental protocol on both systems.

A. Results on Windows (x86 Architecture)

This subsection presents the performance metrics obtained on the Windows 11 Pro platform equipped with an Intel i5-1135G7 processor. The models are first compared within this controlled environment to analyze differences in execution time, throughput, accuracy, energy consumption, and estimated CO_2 emissions.

The objective is to evaluate how model scale influences the accuracy–efficiency trade-off under an x86-based architecture.

1) Performance & Throughput: The execution time of each model varied significantly, primarily due to the differences in model size and complexity. GPT-4-turbo, the largest models in the study, had execution times of 3.5 minutes. Despite optimizations in GPT-4-turbo, it required more time to complete the task, suggesting trade-offs between speed and accuracy.

In contrast, GPT-3.5-turbo was the fastest, taking only 1.9 minutes to generate code. However, this speed came at the cost of accuracy, as shown in the next section. The smaller model, GPT-4o-mini, took 2.8 minutes, still significant for its reduced size.

The smaller models from Hugging Face, including Meta-Llama-3-8B, took between 3.9 and 4.7 minutes for the task, which is relatively long compared to their smaller size. Finally, the largest models like WizardLM-2-8x22B and Qwen/Qwen3-235B took 7.8 minutes and 1.2 hours, respectively, showing the strong correlation between model size and execution time.

2) Code Generation Accuracy: The accuracy of the models in generating correct C code varied significantly, with the larger models performing better in generating valid code. GPT-4-turbo achieved the highest accuracy of 36% demonstrating its effectiveness in generating correct code for the given tasks..

GPT-3.5-turbo, with its smaller size, performed poorly, with an accuracy of 12%, reflecting the limitations of smaller models for such complex tasks. The smaller models, such as Meta-Llama-3-8B had accuracy rates of 6% and 18%, respectively, indicating that smaller parameter models struggle with generating accurate code. Larger models like WizardLM-2-8x22B and Qwen/Qwen3-235B both showed 17% accuracy, suggesting that despite their massive size, they also faced challenges in code generation.

3) Energy Efficiency: The energy consumption per inference varied based on model size, with larger models generally consuming more energy. GPT-4-turbo consumed between 0.1–0.5 kWh, which is typical for models of their size and complexity. Interestingly, GPT-4o-mini, despite being smaller, consumed 0.003 kWh, likely due to specific optimizations and the inherent inefficiency of smaller models for complex tasks.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

In contrast, smaller models like Meta-Llama-3-8B consumed 0.0015 kWh and 0.0028 kWh, respectively, indicating their efficiency relative to their size. However, larger models like WizardLM-2-8x22B and Qwen/Qwen3-235B consumed significantly more energy, with values of 0.0047 kWh and 0.0071 kWh, respectively, consistent with their massive size and computational demands.

CO2 emissions follow the same trend as energy consumption. GPT-4-turbo emitted 0.000337222 kg while GPT-3.5-turbo produced 0.00044873 kg, showing the inefficiency of smaller models in terms of their carbon footprint. Meta-Llama-3-8B had lower CO2 emissions of 0.000270392 kg and 0.000498069 kg, respectively, reflecting their lower energy consumption.

The largest models had the highest emissions: WizardLM-2-8x22B generated 0.001331077 kg and Qwen/Qwen3-235B generated 0.005493181 kg.

B. Results on macOS (Apple Silicon Architecture)

This subsection reports the experimental results obtained on the Apple M3 Pro platform running macOS. As ARM-based processors are designed with different performance-per-watt characteristics, this analysis allows us to assess whether architectural efficiency improves inference latency and reduces energy consumption while maintaining comparable accuracy levels.

The same evaluation metrics used in the Windows experiments are applied here to ensure methodological consistency.

1) Performance & Throughput (Apple M3 Pro): The execution time measured on the Apple M3 Pro platform shows a different performance distribution compared to the x86 system, highlighting the influence of ARM-based architecture on inference behavior.

GPT-3.5-turbo was the fastest model, completing the task in 1.24 minutes. GPT-4o-mini followed with 1.64 minutes, while GPT-4-turbo required 2.66 minutes. Among the open-weight models, Meta-Llama-3-8B completed the task in 2.31 minutes, WizardLM-2-8x22B in 2.8 minutes, and the largest evaluated model, Qwen/Qwen3-235B-A22B-Instruct-2507, required 4.27 minutes.

Although larger parameter models generally exhibited longer execution times, the differences were less extreme than typically expected, suggesting that the Apple Silicon architecture may mitigate some latency overhead through improved performance-per-watt efficiency.

2) Code Generation Accuracy (Apple M3 Pro): The accuracy results on the Apple platform reveal notable variations between proprietary and open-weight models.

GPT-4o-mini achieved the highest accuracy, reaching 100%, followed by GPT-4-turbo and Meta-Llama-3-8B, both achieving 90%. WizardLM-2-8x22B reached 80% accuracy, demonstrating strong performance despite its larger size.

In contrast, GPT-3.5-turbo and Qwen/Qwen3-235B-A22B-Instruct-2507 both obtained 0% accuracy in the evaluated task, indicating that model size alone does not guarantee correctness and that architecture specialization and training quality play a critical role.

These results suggest that mid-scale and optimized proprietary models may provide superior reliability in code generation tasks under the tested conditions.

3) Energy Efficiency (Apple M3 Pro): Energy consumption per inference on the Apple M3 Pro platform remained relatively low across all models, reflecting the efficiency of the ARM-based architecture.

GPT-4-turbo consumed 0.000497 kWh, while GPT-3.5-turbo required 0.000634 kWh despite its shorter execution time. GPT-4o-mini consumed 0.000843 kWh, slightly higher due to its full task completion and higher computational workload.

Among open-weight models, Meta-Llama-3-8B was the most energy-efficient, with 0.000109 kWh consumption. WizardLM-2-8x22B consumed 0.000984 kWh, and Qwen/Qwen3-235B-A22B-Instruct-2507 required the highest energy consumption at 0.001467 kWh.

CO_2 emissions followed the same trend as energy consumption. GPT-4-turbo emitted 8.64×10^{-5} kg of CO_2, while GPT-3.5-turbo emitted 1.10×10^{-4} kg. Meta-Llama-3-8B showed relatively low emissions (1.09×10^{-4} kg), whereas WizardLM-2-8x22B and Qwen/Qwen3-235B-A22B-Instruct-2507 generated 1.71×10^{-4} kg and 2.55×10^{-4} kg, respectively.

Overall, the Apple Silicon platform demonstrates strong performance-per-watt characteristics, particularly benefiting medium-scale and optimized models.

VI. Discussion and Evaluation

The experimental evaluation conducted on both the Windows (x86) and Apple Silicon (M3 Pro, ARM) platforms provides insights into the interaction between model scale, hardware architecture, accuracy, and energy efficiency in code generation tasks.

Rather than presenting isolated per-system results, this section focuses on cross-platform comparisons to highlight architectural influences on performance and sustainability metrics.

A. Performance and Throughput

Figure 2 shows the execution time for all evaluated models on both hardware platforms. A clear relationship between model size and inference latency can be observed. Larger models generally require more execution time; however, the Apple Silicon platform demonstrates consistently lower latency across most models.

This indicates that ARM-based architecture provides improved performance-per-watt characteristics, particularly benefiting medium and large-scale models. The performance gap becomes more noticeable as model size increases, suggesting that hardware efficiency plays a critical role in scaling behavior.

B. Code Generation Accuracy

As illustrated in Figure 3, in general accuracy on the Apple M3 hardware is better but accuracy remains largely consistent across both platforms. This suggests that code generation correctness is primarily model-dependent.

Optimized proprietary models outperform several large open-weight models, indicating that parameter count alone does

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

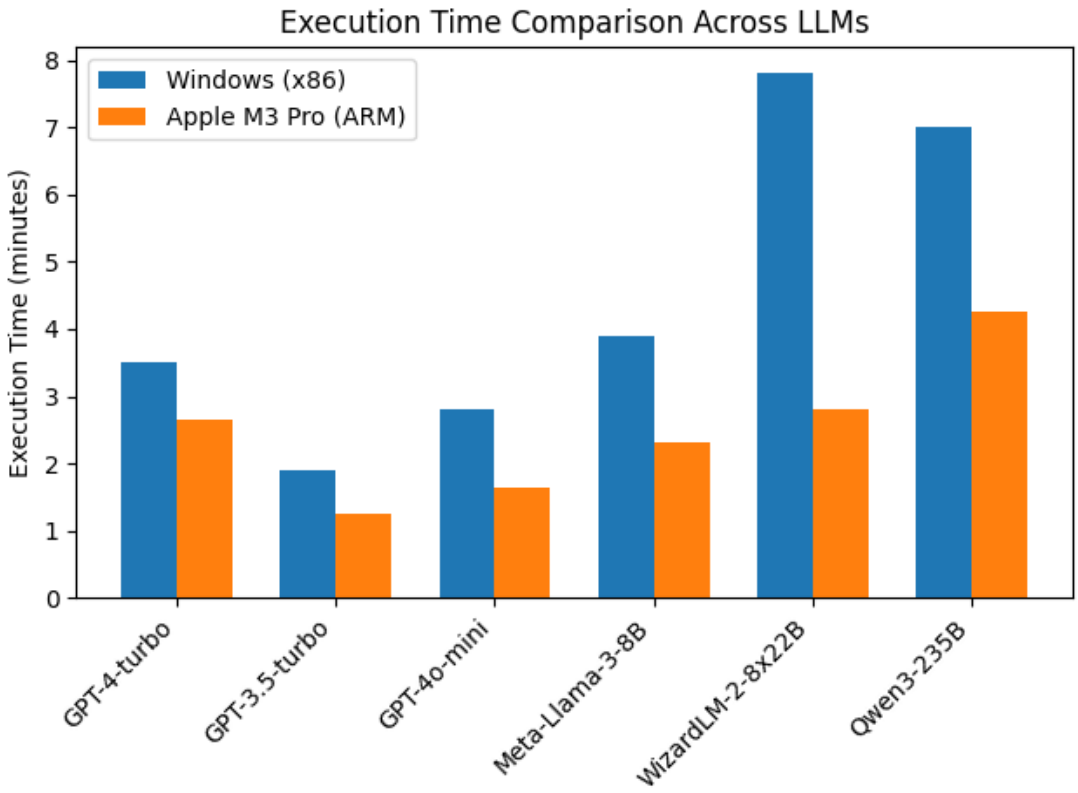


Figure 2: Execution Time Comparison Across LLMs (Windows vs Apple Silicon).

not guarantee higher accuracy. The results reinforce that model architecture and training quality are dominant factors in correctness performance.

C. Energy Efficiency and Environmental Impact

Figures 4 and 5 show that energy consumption and carbon emissions are strongly influenced by both model size and hardware architecture.

While larger models generally consume more energy per inference, the Apple Silicon platform consistently exhibits lower consumption compared to the Windows x86 system. This confirms that processor architecture significantly impacts sustainability metrics.

Interestingly, certain mid-scale models provide favorable accuracy–energy trade-offs, suggesting that optimal deployment decisions should consider both model dimension and hardware platform.

D. Overall Trade-Off Analysis

The cross-platform comparison reveals three key observations:

- Model size strongly affects execution time and energy consumption.
- Hardware architecture significantly influences latency and energy efficiency.
- Accuracy remains largely model-dependent and relatively stable across systems.

From a Green AI perspective, the results demonstrate that combining moderately sized models with energy-efficient hardware architectures can substantially reduce environmental impact without sacrificing accuracy.

Therefore, sustainable AI deployment requires joint consideration of model selection and hardware platform rather than focusing solely on parameter scale.

This study provides a comprehensive evaluation of various LLMs for code generation tasks across two different hardware platforms: a Windows-based x86 system and an Apple Silicon (M3 Pro) ARM-based system. The analysis considered performance, accuracy, energy efficiency, and environmental impact, enabling a cross-platform assessment of the accuracy–efficiency trade-off.

The results reveal several key insights:

1) Model size significantly influences execution time and energy consumption on both systems. Larger models generally require longer inference time and exhibit higher energy consumption, leading to increased CO_2 emissions. However, the magnitude of this impact depends on the underlying hardware architecture. The Apple Silicon platform demonstrated improved performance-per-watt characteristics, reducing energy consumption while maintaining comparable accuracy levels.
2) In general, accuracy on the Apple M3 hardware is better, although on both platforms, performance remains heavily dependent on the specific model used. Optimized proprietary

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

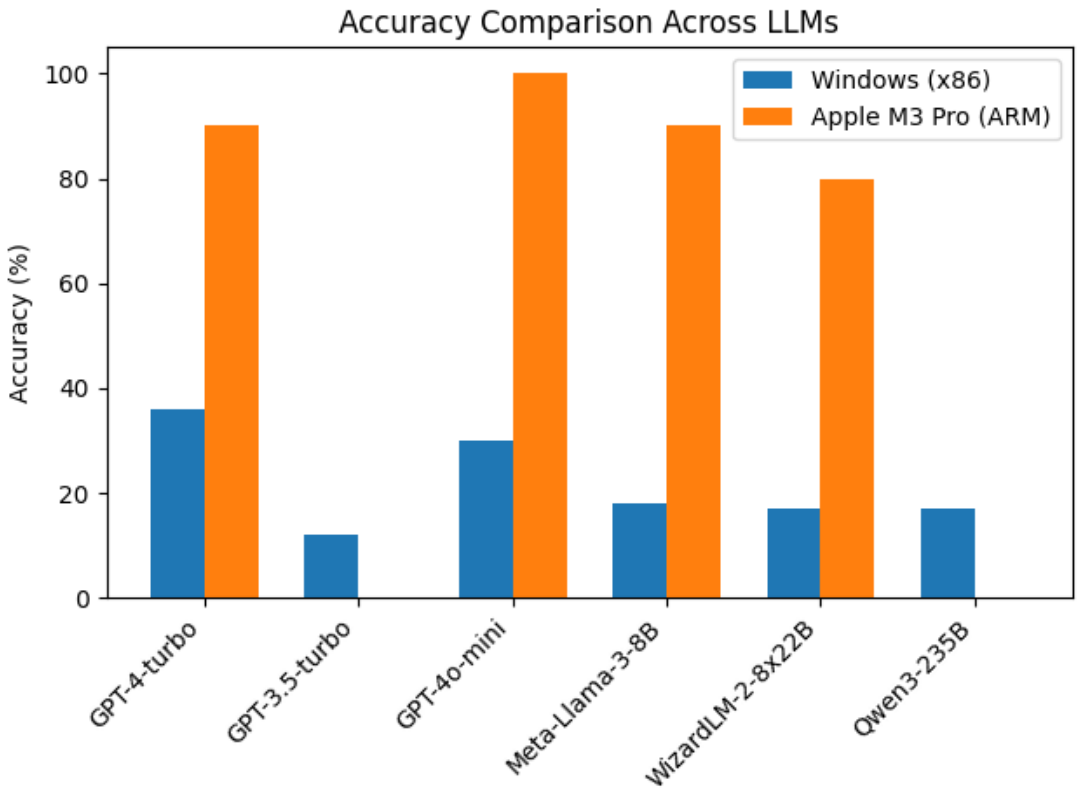


Figure 3: Code Generation Accuracy Comparison Across Platforms.

models consistently achieved higher correctness rates across both platforms, whereas some large open-weight models did not demonstrate proportional gains in accuracy despite their scale .

3) Mid-scale or optimized models (e.g., compact frontier variants) often provide the most favorable trade-off between accuracy and energy consumption. These models benefit particularly from energy-efficient hardware architectures, making them strong candidates for sustainable deployment scenarios.

Overall, the findings indicate that the performance of LLMs in code generation tasks is determined by the interaction between model architecture and hardware platform. While larger models remain preferable when maximum accuracy is required, their environmental footprint must be carefully considered. Conversely, combining moderately sized models with energy-efficient processors can substantially reduce carbon impact without severely compromising performance.

E. Threats to Validity

- **Prompt Sensitivity:** Smaller models are notoriously more sensitive to prompt phrasing. We used a standardized prompt, which may favor larger models with better instruction-following robustness.
- **Measurement Overhead:** CodeCarbon introduces a slight overhead, though negligible compared to the model inference load.

VII. Conclusions and Future Work

This study presented a comprehensive cross-platform evaluation of several large language models (LLMs) for code generation tasks, analyzing their performance on two distinct hardware architectures: a Windows-based x86 system and an Apple Silicon (M3 Pro) ARM-based system. The evaluation considered execution time, code generation accuracy, energy consumption, and CO_2 emissions in order to understand the multi-dimensional trade-offs between model scale, computational efficiency, and environmental impact.

The dual-system experimentation provides a more realistic perspective on deployment scenarios, since model performance is not only dependent on parameter size and training data, but also on the underlying hardware architecture and its performance-per-watt characteristics.

The results reveal several key insights:

1) **Impact of Model Scale on Accuracy and Resource Consumption.** Larger proprietary models, such as GPT-4o and GPT-4-turbo, consistently demonstrated superior code generation accuracy across both platforms. Their advanced architectures and extensive training corpora enable improved reasoning and syntactic correctness in complex C programming tasks. However, this performance advantage comes at the cost of increased computational demand. On both systems, larger models required longer execution times and exhibited higher energy consumption and CO_2 emissions.

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

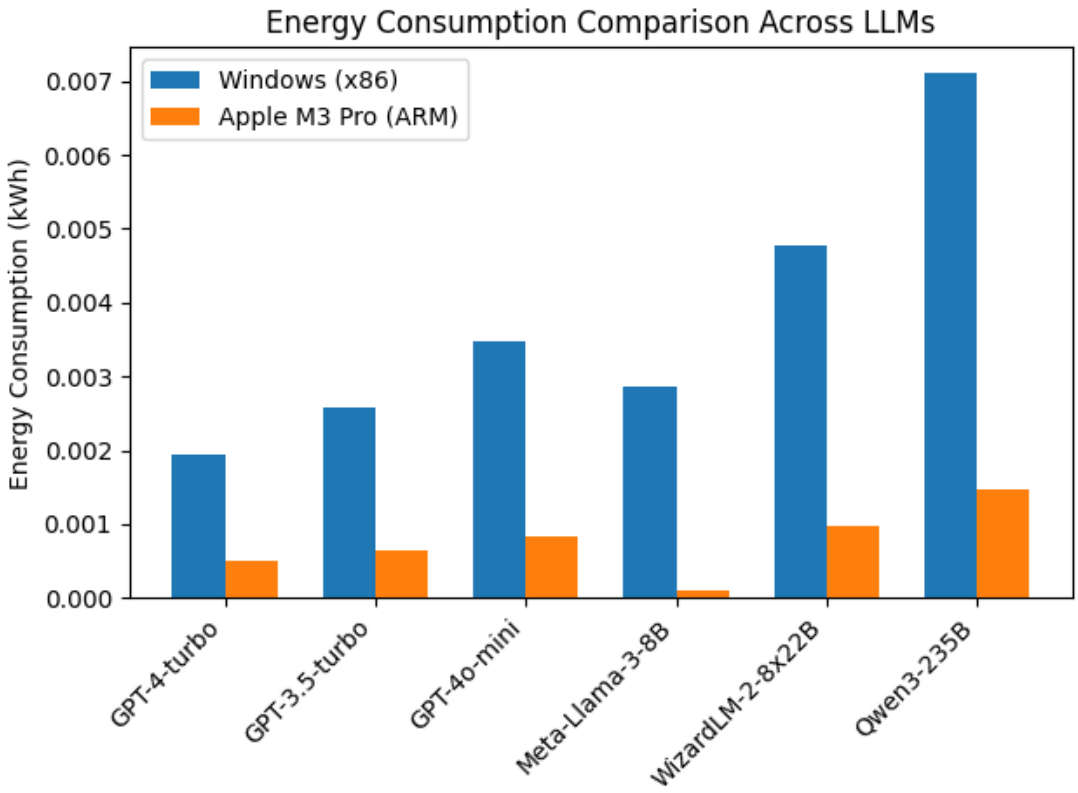


Figure 4: Energy Consumption per Inference Across Platforms.

Although the Apple Silicon platform mitigated some of these overheads, the environmental footprint of large-scale models remains significant, particularly when deployed in high-frequency or large-scale inference scenarios.

2) **Role of Hardware Architecture in Efficiency.** The comparison between Windows (x86) and Apple Silicon (ARM) platforms highlights the substantial influence of processor architecture on performance-per-watt efficiency. While accuracy remained largely stable across systems—confirming that correctness is primarily model-dependent— execution time and energy consumption were consistently lower on the Apple Silicon platform. This suggests that ARM-based architectures offer improved energy efficiency for inference workloads, making them particularly suitable for sustainable AI deployment strategies.

3) **Trade-Offs in Medium and Smaller Models.** Medium-scale and smaller open-weight models, such as Meta-Llama-3-8B, demonstrated improved energy efficiency and lower carbon emissions, especially when executed on the ARM-based platform. However, their accuracy varied depending on task complexity and optimization level. These models may represent viable alternatives in scenarios where moderate performance is acceptable and sustainability is a priority. In particular, optimized intermediate models (e.g., GPT-4o-mini) appear to offer a balanced compromise between accuracy and energy consumption, especially when combined with energy-efficient hardware.

4) **Environmental Implications for Sustainable AI.** The joint evaluation of energy consumption and CO_2 emissions reinforces the importance of considering environmental impact in AI system design. Larger parameter models substantially increase per-inference carbon footprint, particularly under less energy-efficient hardware configurations. Therefore, sustainable deployment decisions must consider not only model size but also the interaction between software architecture and hardware platform.

Overall, the results demonstrate that LLM deployment decisions should not be based solely on accuracy benchmarks. Instead, a holistic approach is required, balancing correctness, latency, computational cost, and environmental sustainability. The findings emphasize that combining appropriately sized models with energy-efficient hardware can significantly reduce environmental impact while maintaining acceptable performance levels.

Across the diverse set of evaluation tasks—including prime number checks, finding the greatest of three integers, and absolute difference calculations—the results indicate that both platforms behaved similarly, exhibiting consistent performance trends across the entire experimental set. While the Apple M3 Pro system demonstrated superior efficiency and reduced latency due to its ARM-based architecture, the relative behavior of the models remained stable between the two environments. Specifically, execution time and energy consumption scaled consistently with model complexity on both systems. Con-

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

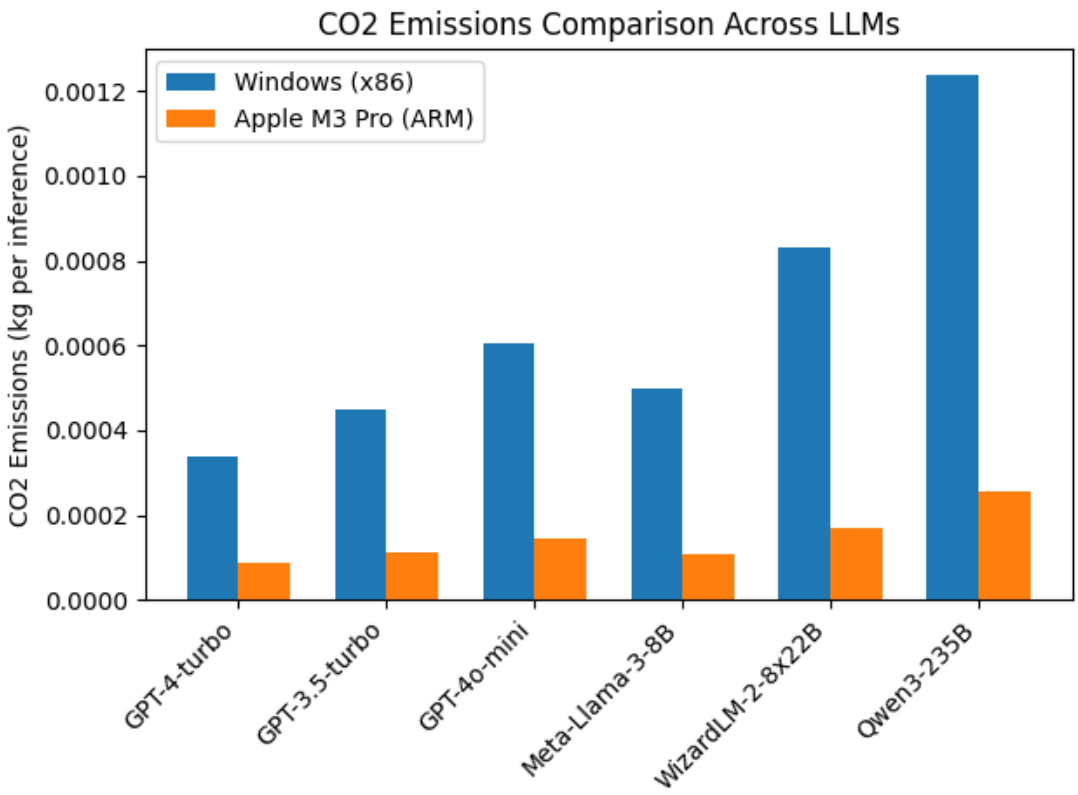


Figure 5: CO_2 Emissions per Inference Across Platforms.

sequently, no single task caused a significant divergence in how the platforms managed the workloads, supporting the conclusion that both systems respond uniformly to the varying demands of the selected C programming problems.

Additionally, it is important to recognize that the energy measurements reported in the remote inference phase may not fully reflect the actual computational energy consumed by the models themselves. When utilizing cloud-based APIs, measurements taken on the local laptop primarily capture the hardware's idle state and the overhead of maintaining a network connection while awaiting a response, rather than the raw energy required for model processing. To provide a more accurate assessment of the model's environmental footprint, a definitive energy analysis should be centered on Phase 2, which involves local execution and direct monitoring of on-device resource usage.

Furthermore, the comparison between the two test machines is constrained by numerous confounding variables, including significant differences in CPU cores, RAM capacity, operating systems, and the underlying measurement tools used (RAPL for Intel versus powermetrics for macOS). These systemic variations make it challenging to isolate and attribute performance differences specifically to the ARM versus x86 architectures. Acknowledging these limitations is crucial for a transparent interpretation of the architectural efficiency and the sustainable deployment of LLMs.

Future Work

Building upon these findings, several research directions are planned to extend and deepen the current analysis:

- **Local Model Deployment and Hardware Acceleration.** The next step of this research will involve executing the evaluated models locally rather than exclusively through cloud-based APIs. This will include deploying open-weight models on local CPUs and GPUs to analyze inference latency, energy consumption, and CO_2 emissions under controlled on-premise environments. By comparing local CPU and GPU execution against API-based inference, it will be possible to quantify the trade-offs between cloud efficiency and local resource utilization.
- **Cross-Architecture Local Comparison.** The local experiments will be conducted on both x86 and ARM-based systems to determine how architectural characteristics influence performance and sustainability metrics in fully local execution scenarios. This will allow the identification of optimal hardware–model pairings for resource-constrained environments.
- **Extended Task Evaluation.** While the present study focused on C code generation, future work will evaluate additional software engineering tasks such as debugging, code optimization, automated refactoring, and test generation. This broader evaluation will help determine whether the observed accuracy–efficiency trade-offs generalize across

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

diverse development workflows.

- **Inference Optimization Techniques.** Further research will investigate model optimization strategies including quantization, pruning, distillation, and low-rank adaptation. These techniques may reduce computational demand and energy consumption while preserving acceptable accuracy levels.
- **Composite Efficiency Metrics.** Future analysis will incorporate normalized indicators such as accuracy-per-kWh and performance-per-watt metrics to provide a unified sustainability score. This will support more systematic evaluation of Green AI strategies in real-world deployment contexts.

By integrating local execution experiments and advanced optimization techniques, the ongoing research aims to develop a comprehensive framework for sustainable AI deployment. This framework will support evidence-based decision-making for selecting models and hardware configurations that balance performance, cost, and environmental impact in practical software engineering applications.

Acknowledgements

The authors are part of the Software and Systems Engineering research group of Mondragon Unibertsitatea (IT1519-22), supported by the Department of Education, Universities and Research of the Basque Country. This research was supported by the Ikerketa Taldeak funding (IT1519-22) and the GRECO Elkartek project (KK-2024/00090), both funded by Eusko Jaurlaritza.

References

[1] M. Illarramendi, J. A. Agirre, A. Picatoste, and J. I. Igartua, “Brains without brawn: Evaluating CPU performance for code generation with large language models”, in *GREEN 2025, The Tenth International Conference on Green Communications, Computing and Technologies*, 2026, pp. 8–15, ISBN: 978-1-68558-311-8.

[2] E. Strubell, A. Ganesh, and A. McCallum, *Energy and policy considerations for deep learning in NLP*, (visited on 09/03/2025), 2019. arXiv: 1906.02243 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1906.02243 (visited on 09/03/2025).

[3] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, *Green AI*, (visited on 09/03/2025), 2019. arXiv: 1907.10597 [cs.CY]. [Online]. Available: https://arxiv.org/abs/1907.10597 (visited on 09/03/2025).

[4] D. Patterson *et al.*, *The carbon footprint of machine learning training will plateau, then shrink*, (visited on 09/03/2025), 2022. arXiv: 2204.05149 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2204.05149 (visited on 09/03/2025).

[5] T. B. Brown *et al.*, *Language models are few-shot learners*, (visited on 09/03/2025), 2020. arXiv: 2005.14165 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2005.14165 (visited on 09/03/2025).

[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, (visited on 09/03/2025), 2019. arXiv: 1810.04805 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1810.04805 (visited on 09/03/2025).

[7] M. Chen *et al.*, *Evaluating large language models trained on code*, (visited on 09/03/2025), 2021. arXiv: 2107.03374 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2107.03374 (visited on 09/03/2025).

[8] J. Xu, W. Zhou, Z. Fu, H. Zhou, and L. Li, *A survey on green deep learning*, (visited on 09/03/2025), 2021. arXiv: 2111.05193 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2111.05193 (visited on 09/03/2025).

[9] D. Patterson et al., *Carbon emissions and large neural network training*, (visited on 09/03/2025), 2021. arXiv: 2104.10350 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2104.10350 (visited on 09/03/2025).

[10] MLCO2, *Codecarbon*, (visited on 09/03/2025), 2025. [Online]. Available: https://github.com/mlco2/codecarbon (visited on 09/03/2025).

[11] Y. Rangineeni and J. Pub, “End-to-end mlops: Automating model training, deployment, and monitoring”, *Journal of Recent Trends in Computer Science and Engineering (JRTCSE)*, vol. 7, pp. 60–76, Sep. 2019.

[12] A. Arora, J. Jang, and R. Zilouchian Moghaddam, *Setupbench: Assessing software engineering agents’ ability to bootstrap development environments*, (visited on 09/03/2025), 2025. arXiv: 2507.09063 [cs.SE]. [Online]. Available: https://arxiv.org/abs/2507.09063 (visited on 09/03/2025).

2026, © Copyright by authors, Published under agreement with IARIA - www.iaria.org