

Intelligent Photo Retrieval System Using Text and Image Queries

Akbota Tangirbergenova

BSc Computer Science

Nazarbayev University

Astana, Kazakhstan

akbota.tangirbergenova@nu.edu.kz

Danial Sakhpantayev

BSc Computer Science

Nazarbayev University

Astana, Kazakhstan

danial.sakhpantayev@nu.edu.kz

Alisher-Polat Kurmanayev

BSc Computer Science

Nazarbayev University

Astana, Kazakhstan

alisherpolat.kurmanayev@nu.edu.kz

Dinmukhamed Tolegenov

BSc Computer Science

Nazarbayev University

Astana, Kazakhstan

dinmukhamed.tolegenov@nu.edu.kz

Abstract—In the modern world, taking a photo has become accessible to everyone and having thousands of photos on your phone is common nowadays. That’s why, finding a certain image can be frustrating, time-consuming, and unsuccessful. To address this issue, we propose the intelligent photo retrieval system that helps to find an image with text and image queries and works offline. The system pipeline integrates lightweight multimodal models for text-to-image processing and image-to-image processing, deployed via Flutter-based frontend. Through various tests, it was concluded that lightweight models can be sufficient for offline intelligent photo retrieval mobile system.

I. INTRODUCTION

A. Problem Statement and Motivation

Modern smartphones create large, unstructured photo collections, making it hard to navigate. Manual browsing does not scale with time. Existing cloud solutions like Google Photos depend on the Internet connection, cloud storage size and also raise privacy concerns. In contrast, existing offline gallery apps do not require internet connection and only fill your phone’s storage, but their searching algorithm relies on weak metadata, like timestamps and location, and there is no semantic understanding in such algorithms. The core aim of this project is to implement fast, accurate semantic photo retrieval without internet access.

B. Proposed Solution Overview

Proposed solution features 3 key components:

- Text-to-image search using embeddings
- Image-to-image search
- Natural language queries using metadata filtering and named entity recognition.

Our design focus was on making the system work fast under the mobile constraints. For these purposes, we switched to lightweight model *mobileclip-s2*, but ultimately excluded face clustering feature because of the significant memory usage

and processing overhead. On resource-constrained hardware, such as average mobile phone, such heavy algorithms like face clustering lead to increased latency and decreased throughput, which is detrimental for the system’s requirement of offline real-time operation. As a result, it was concluded to prioritize consistent performance on mobile devices over additional image grouping features.

C. Report Organization

The following ideas will be expanded in the next sections:

- Section II: constraints (technical, ethical, deployment limits).
- Section III: background (retrieval models, clustering, mobile systems).
- Section IV: system design (architecture and modules).
- Section V: implementation decisions and challenges.
- Section VI: evaluation (performance, accuracy, usability).
- Section VII: conclusions and future improvements.

II. IDENTIFICATION OF CONSTRAINTS

A. Technical and Economic Constraints

The project was developed within a constrained resources environment. The constraints included no dedicated funding for a designated GPU server and cloud infrastructure. This directly shaped the most critical architectural project decision of this semester – shifting towards on-device, offline inference. Previously, we used to process users’ requests on the deployed backend, but as the system grew, the operational costs of maintaining the project rose dramatically. As a result, we moved our system towards the use of light-weight, mobile-compatible models. The original CLIP was replaced with Mobile-CLIP, and the face clustering was dropped from the implementation, as keeping the face detection, recognition, alongside the MobileCLIP on the mobile device RAM, seemed not to be possible.

B. Ethical, Social, and Sustainability Considerations

1) *Ethical and Social Considerations*: The system handles deeply personal data – private photos of places and faces – and therefore carries significant ethical responsibilities regardless of its legal standing. This consideration directly affected the system design, having on the initial stage of the project only the storage of the embeddings instead of images, and further on moving towards the full offline processing, protecting the users' data from potential leakage. Additionally, as the project utilizes the model pretrained on the large internet data, there might be possible biases in its queries, and it might not perform equally across all demographics, image types, or query languages. Users whose photos might fall out of the usual data distribution may experience lower retrieval quality. That is an acknowledged and unresolved limitation.

2) *Environmental Considerations*: ML systems tend to utilize an enormous amount of computing power, resulting in carbon emissions and water waste, so it is important to keep the development as efficient as possible. The shift to fully offline processing removes the dependence on the server's processing, which usually consumes an enormous amount of power, making the system more environmentally efficient at scale compared to cloud-based alternatives.

C. Impact on Design and Deployment

III. BACKGROUND AND RELATED WORK

A. Text-to-Image Processing

Text to image search in retrieval systems focuses mainly on natural language processing. Earlier solutions included manually assigning keywords and tags to each photo and word, which required all images to be annotated in advance and user to use only specific words that are recognized by the system. Such methods were very limited and could not capture larger semantic meanings, and was strictly deterministic. Any deviation from the tag or more diverse natural language description breaks the system.

One major step forward came with vision-language models such as CLIP, which utilized shared embedding space both for images and text [1]. This approach is very different, as it encodes every text query and all images from the dataset into high-dimensional vector within the same vector space. Vectors that lie close to each other have close semantic meanings, and this property of embeddings allows us to search for similar image using cosine similarity. Such models are trained on large amounts of text-image pairs, which makes it capture more semantic relationships between visual images and raw text.

Due to our project's computing constraints, lightweight versions of such solutions are especially important. MobileCLIP is lightweight model that manages to do the same at a smaller size, while maintaining acceptable levels of performance [2]. This makes it practical to run such system on-device without the need in internet connection or cloud compute. MobileCLIP2 further improves its performance by strengthening its training. The retrieval quality is higher, too.

B. Image-to-Image Processing

One of the ways to implement image-based processing is to use vision transformers. There are several high accuracy models that can be used to implement efficient image retrieval system.

DINOv2 model was developed by self-supervised learning [3]. The model extracts visual features without predefined labels, but it uses student-teacher technique to train efficiently. While DINOv2 can be used only for image-to-image processing, there are multimodal models, such as CLIP and SigLIP that would be efficient to integrate to the system that processes both text and image queries.

CLIP (Contrastive Language-Image Pre-training) model uses semantic-based image encoder to implement image-to-image processing [4]. Input image encoder and output image encoder creates shared embedding space and detects similar images through cosine similarity between datapoints. The further changes to this model has been made to increase accuracy. Changing the softmax function to sigmoid function, SigLIP (Sigmoid Loss for Language-Image Pre-training) model was proposed [5].

Even though, the models above tend to provide high accuracy, they also require large memory space and computational power. However, developing a mobile app, memory constraints and hardware constraints must be taken into account. Therefore, lightweight image processing models should be also considered. MobileCLIP model family is one of the optimal solutions for latency constraints. The lighter model was developed from larger CLIP models by reinforced learning [6]. Furthermore, MobileCLIP family models were advanced into MobileCLIP2 family models, providing state-of-the-art accuracy under low latency [2].

C. Offline Mobile Systems Architecture

Running ML models directly on the mobile device is usually done with several established frameworks. TensorFlow Lite (TFLite) is the most widely used on-device inference runtime for Android, designed specifically for low-latency inference on mobile hardware. Another commonly used framework is the use of ONNX Runtime, which is a cross-platform inference engine. Both frameworks require the models to be converted from their original format (PyTorch and TensorFlow) into a mobile-compatible format before deployment. In our system, we proceed with the ONNX-based models, as we utilized a native Kotlin Android layer responsible for all ML inference.

IV. PROJECT APPROACH

A. Overall System Architecture

The final architecture of the photo retrieval system is a fully on-device, offline application capable of processing the user's images from the personal gallery. The overall system can be seen on the Figure 1.

At a high level, the system consists of 3 core layers:

- 1) The UI and the application orchestration layer
- 2) The native ML inference layer

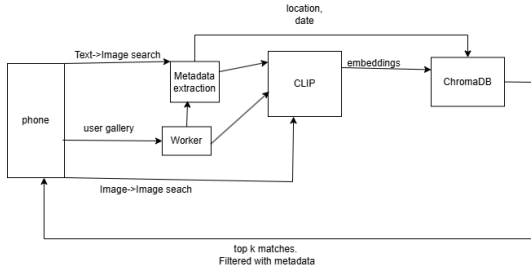


Fig. 1. System Overview

3) The local storage and retrieval layer

The **UI and application orchestration layer** is implemented using Flutter and is responsible for the user interaction, state management, and high-level control of the application's workflow. It handles actions query input(text, image) and is responsible for communication with ML layer for the image search.

The **ML inference layer** is implemented with Kotlin and its weights are imported in the ONNX format. This layer is responsible for the model loading, input preprocessing, and the inference execution. This layer also includes the tokenization for text queries, as well as the normalization of the inputs. A key requirement for this layer is keeping consistent input for the user queries, ensuring that both texts and images are generated within the same semantic representation.

The **local storage and retrieval layer** manages data persistence and efficient similarity search of the embeddings. It stores the precomputed image embeddings along with associated metadata and provides nearest-neighbor search for the fast retrieval. This layer enables the system to operate fully offline by relying on a local vector index.

Together these layers form a cohesive architecture where the UI orchestrates other system components for the efficient use.

B. Text-to-Image Processing

Our implementation of text-to-image search first turns the text query into a high-dimensional embedding, which is then compared against all image embeddings stored in the vector database. The system then retrieves top-k results within some threshold, which is set in the system configuration. This makes it possible to search the photo gallery using text description, not exact filenames, timestamps, or manually assigned tags.

Latter however is still possible due to hybrid search using metadata filtering and NER extraction. When the user enters the query, we run a NER process, which automatically finds all "Location" and "Time/Date" tags, and then uses it during the search. It allows for custom queries, such as "Cat photos in Astana taken last year". Separate NER handling function extracts "Astana" and "last year" as 2 entities. Then, a deterministic search is performed using a custom handler function that maps relative timestamps to date metadata, and city names to GPS location.

To make the system more practical - all image embeddings are created in advance during the first application launch. It takes some time to process all photos, but the result is very fast retrieval in real time with low latency.

C. Image-to-Image Processing

The purpose of image-to-image processing is to retrieve semantically and visually similar images to the query image. To meet all of the mobile constraints and maintain high accuracy at the same time, the MobileCLIP2-S2 model was integrated into the mobile app.

When the app is downloaded, the feature vectors of the user's images are processed by the image encoder of the MobileCLIP2-S2 model and the embedding vectors of local images are generated. Extracting the feature vectors of the query images, the image encoder generates the embedding vector of the query image. All of the embeddings are stored locally on the user's phone in the ChromaDB vector database. After this, cosine similarity between the query embedding vector and user's image embedding vector is calculated. What it means is that the higher the cosine similarity score, the more similarities the images have [2].

D. System Development and Software Engineering

The frontend of the application is developed using **Flutter**, which provides a cross-platform framework for building responsiveness and a visually consistent user interface. Flutter is used to implement all the user interaction components, including the query input, image search, and retrieval.

State management is handled within a Flutter layer to coordinate user interactions and system workflows, such as indexing initiation, search handling, and rendering retrieval outputs.

The application communicates with the native Android layer via platform channels. This mechanism allows Flutter to invoke native Kotlin functionalities, in particular the ML inference pipeline, and similarity search within the vector space.

For data persistence, the system utilizes the local, on-device storage to store the embeddings for the images from the user's library, along with the associated metadata. This local database enables efficient retrieval of corresponding images.

Overall, the system development approach in our system emphasizes the separation of tasks into specially designated components.

E. Third-party Components and Integration

The system depends on a number of third-party components, frameworks, and libraries for its functioning. Additionally, the system utilizes the pre-trained models exported in ONNX format. **Flutter** is primarily used for mobile UI development and system orchestration, while the native **Android Kotlin layer** is used specifically for ML-associated operations, such as model loading and inference.

For the multimodal functioning of the system, the application utilizes the pretrained MobileCLIP2-S2 model, which

provides a shared embedding space for semantically similar images and texts. The model was selected and adapted specifically for mobile development.

Overall, these third-party components enable the system to reliably function for the users' utilization.

F. Team Coordination and Workflows

The team has used several tools to implement the project. To provide accurate evaluation metrics, all models were tested in the same environment, in Google Colab, utilizing an NVIDIA T4 Tensor Core GPU. Github was used to assign tasks and keep the track of previously done, active, and future tasks.

In the first part of the semester, the ML engineers of our team have focused on model evaluation and selection, while the software developer constructed the interface of the mobile app. When the models were selected, they were passed to the software developer, who then integrated it to the system, ensuring that the vector database and mobile interface work smoothly.

V. PROJECT EXECUTION

A. Design Decisions and Timeline

Over the course of the previous semester, we proceeded with the development of a server-side processing application, which included a Flutter UI communicating via Rest API with FastAPI backend. The system was clearly separated into a interface and a specially designated processing pipeline.

However, this semester we decided to move on with the on-device processing due to the reasons discussed above. This decision required numerous design changes. First, the model was changed to a more lightweight version and converted into a mobile-compatible format. Currently, all the processing and computations are being done on-device, but we do still have a clear separation of UI from the computations, as we decided to utilize Kotlin-based processing specifically for the ML inference.

Additionally, another aspect of the design change in our pipeline this semester was the removal of the face clustering pipeline for stable system performance.

B. Collaborative Development and Leadership Roles

Our team consists of 4 members, and the work was contributed across all members. Each member actively participated in the project implementation and documentation. Akbota Tangirbergenova was the leader of the group and was responsible for operational processes, including communication with the advisor, scheduling meetings, following deadlines, distributing tasks equally, and ensuring documentations are completed. In addition, she participated in the model selection processes for image-based image retrieval, as well as literature review and model evaluation. Alisher-Polat Kurmanayev was mainly responsible for model selection to implement text-based image retrieval. It includes literature review, model testing and evaluation. Danial Sakhpantayev made key decisions on model selection for both image retrieval processes,

participated in model testing and evaluation, and was responsible for software engineering. Dinmukhamed Tolegenov played a significant role in the mobile app development and ensured the smooth system functionality. All decisions regarding the project were discussed collectively, bringing various viewpoints on each problem. Conducting regular meetings with the advisor and within the team, the team members shared the observations, insights, and suggestions, thus, clear communication led to efficient problem-solving.

VI. EVALUATION

A. Text-To-Image Models Evaluation

Used COCO-2017 Dataset has a list of tags in each image. We randomly sampled a set tags as text prompts for search query, and then evaluated how these models perform. We also ran a series of manual complex queries, and all MobileCLIP models performed well, always retrieving the most relevant results.

MobileCLIP2-S2 was chosen as the final model because it offered the best trade-off between accuracy, model size, and runtime. MobileCLIP2-B achieved slightly better retrieval results, but MobileCLIP2-S2 was a bit smaller and faster with only a slight dip in performance.

TABLE I
TEXT-TO-IMAGE RETRIEVAL PERFORMANCE OF EVALUATED MODELS

Model	PR_AUC	R@1	R@5	R@10	Size (MB)	Time (s)
MobileCLIP2-B	0.7210	0.776	0.951	0.986	571.35	9.42
MobileCLIP2-S2	0.7084	0.761	0.946	0.989	378.58	6.98
MobileCLIP-B	0.7041	0.754	0.949	0.987	571.35	7.55
MobileCLIP2-S0	0.6768	0.728	0.938	0.981	285.47	4.86
MobileCLIP-S2	0.6529	0.719	0.934	0.978	378.58	6.73
MobileCLIP-S1	0.6415	0.708	0.928	0.975	324.13	5.84
convnext_base_w	0.6127	0.691	0.921	0.972	684.30	13.21
RN50	0.4316	0.552	0.861	0.938	389.13	7.03

B. Image-to-Image Models Evaluation

During the evaluation, our primary goal is to find the optimal tradeoff between model's inference time, accuracy, and memory. For this reason, our goal was to get record of each metric to compare the models between each other. 25000 images of the MS COCO 2017 dataset was utilized to conduct the tests [7]. As it can be seen in Table II, several models from MobileCLIP and MobileCLIP2 family models were chosen for evaluation.

To calculate the accuracy, the metrics Precision-Recall Area Under Curve (PR_AUC) and Recall@K (where K=1, 5, 10) were applicable to this case. At the same time, Model Size and Total Time of Inference were recorded.

From Table II, it is possible to highlight that the MobileCLIP2-S2 is the most optimal one since it is significantly faster and lighter than MobileCLIP2-B, however, it does not lose a lot in PR_AUC and Recall metrics. That's why, it was the model chosen for Image-to-Image Processing implementation.

TABLE II
MODEL EVALUATION METRICS AND PERFORMANCE TRADE-OFFS

Model	PR_AUC	R@1	R@5	R@10	Size (MB)	Time (s)
MobileCLIP2-B	0.7460	0.800	0.960	0.988	571.35	10.14
MobileCLIP2-S2	0.7305	0.781	0.954	0.992	378.58	7.52
MobileCLIP-B	0.7328	0.778	0.958	0.989	571.35	8.04
MobileCLIP2-S0	0.6970	0.749	0.946	0.984	285.47	5.18
MobileCLIP-S2	0.6691	0.740	0.942	0.980	378.58	7.21
MobileCLIP-S1	0.6535	0.733	0.935	0.979	324.13	6.25
convnext_base_w	0.6442	0.727	0.934	0.978	684.30	14.09
RN50	0.4774	0.587	0.877	0.946	389.13	7.50

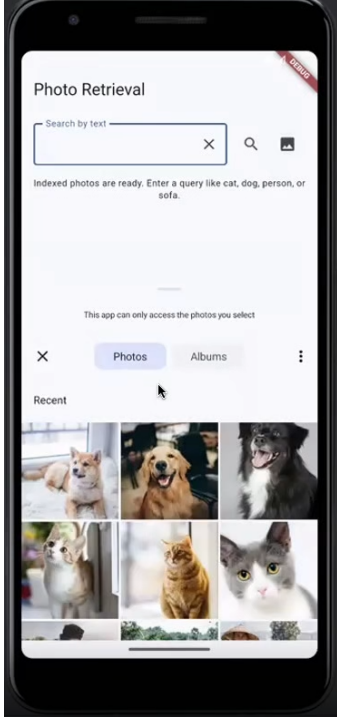


Fig. 2. Image search UI

C. Software Engineering Evaluation

The software engineering evaluation of the project includes the evaluation of the system for its user experience and the system's reliability.

The UI was made easy to use for any user. The image 2 illustrates the user's prompt to search for the images from his gallery. The possible options for the query include the image he can select, as well as a typed natural language query.

The retrieved elements also appear to be easily accessible and in very coherent scrollable format, as illustrated on the Figure 3.

Additionally, despite running the pipeline entirely on-device, the model demonstrates latency and maintains responsive performance.

VII. CONCLUSION AND FUTURE WORK

A. Summary of Contributions

Over the course of these two semesters, we managed to develop a robust mobile application with on-device processing for the retrieval of relevant images from the user's personal

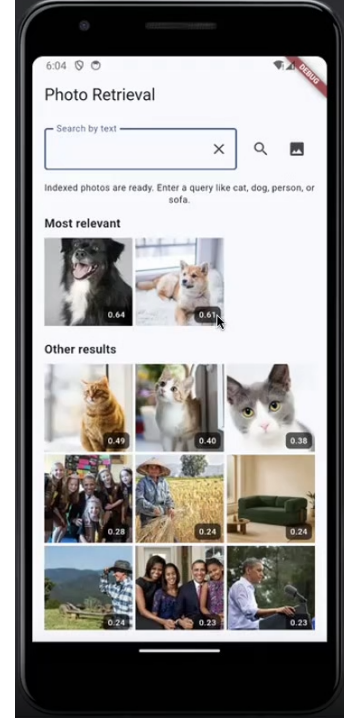


Fig. 3. The retrieved images

library. The system is capable of processing both image and textual inputs and returning semantically similar results, an increasingly valuable function in recent times as personal image collections grow immensely, making manual organization and search impractical.

A key contribution of this work is an implementation of a fully offline privacy-preserving retrieval system. By performing all of the computations on-device, the system ensures no data leakage to third parties and preserves the privacy of the user's personal image collection.

B. Limitations and Future Enhancements

The limitations of the current implementation mostly are connected to the performance optimization under the resource-constrained mobile environment. In particular, these constraints resulted in the removal of the face-clustering module in order to maintain system stability and reduce computational overhead. Future work of this project will include both performance and functionality improvements. One important direction is the introduction of incremental and more optimized background indexing, as this would further improve the scalability and the processing speed of the application. From a functionality perspective, several extensions can be considered. The lightweight version of face-clustering could be reintroduced by integrating an ultra-lightweight edge model specifically for faces. Another features which could be introduced include on-device fine-tuning and video retrieval support.

REFERENCES

- [1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," *arXiv preprint arXiv:2103.00020*, 2021.
- [2] F. Faghri, P. K. A. Vasu, C. Koc, V. Shankar, A. T. Toshev, O. Tuzel, and H. Pouransari, "Mobileclip2: Improving multi-modal reinforced training," *Transactions on Machine Learning Research*, 2025, accepted by TMLR. [Online]. Available: <https://openreview.net/forum?id=WeF9zolng8>
- [3] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Sbai, P. Bojanowski *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.
- [4] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021. [Online]. Available: <https://arxiv.org/abs/2103.00020>
- [5] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Sigmoid loss for language image pre-training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 11 975–11 986.
- [6] P. K. A. Vasu, H. Pouransari, F. Faghri, R. Vemulapalli, and O. Tuzel, "Mobileclip: Fast image-text models through multi-modal reinforced training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 15 963–15 974.
- [7] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.