# Loss Comparison for Image Retrival Challenge

Matteo Massari, Andrea Battaglia, Tommaso Cestari

June 2025

## Abstract

We present a comparative analysis of loss functions for face image retrieval using the CLIP ViT-B/32 backbone. We evaluated contrastive loss (in a Siamese setup), CLIP loss, triplet loss, and ArcFace. After systematic tuning of hyperparameters such as batch size, encoder freezing, and regularization, ArcFace emerged as the best-performing loss, achieving superior accuracy through full fine-tuning. Triplet loss also showed strong generalization, while contrastive loss underperformed due to overfitting. Our findings highlight the importance of choosing both a suitable pre-trained model and a task-appropriate loss function when adapting vision transformers to image retrieval

## 1 Introduction

### 1.1 Task Description

In this competition, we confront an image retrieval task: given a query image, our goal was to identify the ten most similar images from a predefined gallery, specifically focusing on face retrieval. Initially, we were provided with a minimal "proof-of-concept" dataset to verify our implementation and evaluate our codes. On the hackathon day, we received the complete training dataset along with a submission utility to compute accuracy.

Our predictions were evaluated based on similarity using the following metrics: Top-1 accuracy worth 600 points for correctly retrieving the matching identity as the first image, Top-5 accuracy worth 300 points for having at least one correct match within the top five retrieved images, and Top-10 accuracy worth 100 points for at least one correct match within the top ten retrieved images, resulting in a maximum possible score of 1000 points.

### 1.2 Overview of Approaches

Initially, the provided dataset suggested a correlation with the animal domain, prompting our initial models to focus on identifying animals belonging to the same species using publicly available datasets. We experimented primarily with some pre-trained convolutional neural networks (CNNs), such as ResNet50 and EfficientNet50, exploring different approaches. However, these initial attempts yielded unsatisfactory results. Consequently, we reevaluated our approach to the task and decided to leverage transformer-based pre-trained models, particularly

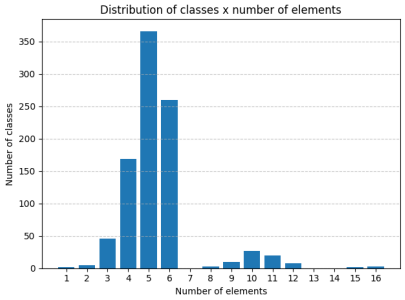


Figure 1: Class distribution in the original dataset

CLIP-ViT-B/32 (CLIP). As reported by Luu 2021, CLIP's remarkable ability to capture fine-grained facial details originates from its multimodal pretraining on vast amounts of vision–language data. This extensive training enables the model to learn joint embeddings that encode subtle variations in expression, illumination, and head orientation. In the single-shot framework proposed by the authors, the model attains 75 % accuracy with a 20% false positive rate, thereby outperforming traditional methods such as VGG-Face and ArcFace in both robustness and efficiency under computationally constrained conditions.

**Dataset** In terms of data, to comply with the main requirement of the generalization rule (Rota 2025), the dataset must satisfy the IID (independent and identically distributed) assumption. Therefore, we first assessed several metrics to analyze the class distribution within the dataset.

As shown in Figure 1, the dataset is highly imbalanced across several classes. To address this issue and create a more balanced class distribution, we applied a data augmentation strategy based on the methodology proposed in A. Mumuni and F. Mumuni 2022. Specifically, we increased the number of samples in each class by a factor of 10 2. This augmentation allowed for a more reliable split between training and validation sets, while maintaining class representativeness and supporting generalization.

**Training loop** The training procedure was designed to ensure stability, reproducibility, and robustness. the Training was configured using a selected optimizer (either stochastic gradient descent, Adam, or AdamW) initialized with the specified learning rate, momentum (if applicable), and weight decay. Logging of hyperparameters and training metrics was performed through an external tracking platform (WandB Biewald 2020). If resuming from a previous training session, the procedure automatically identified and

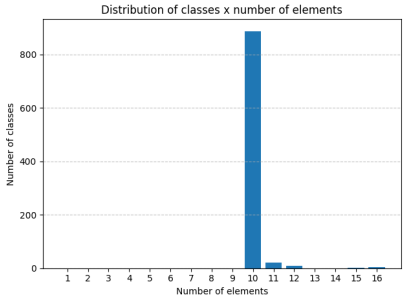


Figure 2: Class distribution after data augmentation

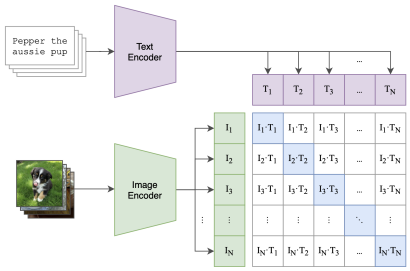


Figure 3: Original ViT-B/32 architecture

loaded the most recent checkpoint, restoring both the model and optimizer states, and resumed training from the corresponding epoch. Otherwise, training was initiated from scratch. Prior to the first epoch, the model was evaluated on the validation set to obtain a baseline loss value for early stopping. Training then proceeded for a predefined number of epochs, during which the model parameters were iteratively updated over mini-batches of training data. Each iteration consisted of a forward pass, loss computation, backpropagation, and an optimization step. After each epoch, the model was evaluated on the validation set in inference mode to assess generalization performance. Model weights were saved at the end of each epoch, both as complete checkpoints and as standalone parameters for downstream evaluation. Early stopping was implemented by monitoring the validation loss; if no improvement was observed over 5 consecutive epochs, training was halted to prevent overfitting. Throughout training, key metrics including training loss, validation loss, and learning rate were continuously logged for subsequent analysis.

Building on these insights, we focused on optimizing the CLIP-ViT configuration to find the best synergy between architecture and training objective, aiming to improve classification performance.

### 1.3 Summary of Results

| Loss | Frz | B_s | Ep | W_d | Drp | Score |
|---|---|---|---|---|---|---|
| Triplet | T | 64 | 45 | 1e-4 | 0.0 | 586.53 |
| CLIP | T | 64 | 21 | 1e-4 | 0.3 | 546.53 |
| ArcFace | F | 128 | 38 | 1e-4 | 0.3 | **616.98** |

Table 1: Compact comparison of loss functions and training settings.

## 2 Models Considered

We considered two architectural paradigms: a vanilla (single-branch) network and a Siamese network. In both cases, the pre-trained CLIP ViT-B/32 visual encoder was employed. A custom projection head was subsequently added to the output of this encoder, effectively mapping its features into a new embedding space with a dimension of 512. Each model variant was then paired with a specific loss function, as illustrated in Figure 1.

| Model | Loss Function |
|---|---|
| ViT-B/32-siamese | Contrastive Loss |
| ViT-B/32 | CLIP Loss |
| ViT-B/32 | Cross-Entropy + ArcFace |
| ViT-B/32 | Triplet Loss |

Table 2: Summary of the tested models and associated loss functions.

**Clip-ViT-B/32**

CLIP from Radford et al. 2021 (Contrastive Language–Image Pre-training) is a model that learns to associate images and text through large-scale natural language supervision. It is trained on 400 million image–text pairs collected from the internet using a contrastive learning objective that brings the embeddings of aligned image-text pairs closer together while pushing apart unaligned pairs within each batch. The model architecture consists of two independent encoders: a visual encoder, implemented as a Vision Transformer (ViT) (Dosovitskiy et al. 2020), and a text encoder based on a 12-layer Transformer with multi-head self-attention.

Each encoder maps its input into a fixed-dimensional space, and both outputs are projected via learned linear transformations into a shared multimodal embedding space, followed by L2 normalization. Training optimizes a symmetric cross-entropy loss over cosine similarity scores between all image-text combinations in the batch. After pre-training, CLIP can perform zero-shot classification by encoding candidate label descriptions as natural language prompts (e.g., "a photo of a cat"), computing their embeddings, and selecting the one with highest cosine similarity to the input image. This allows CLIP to generalize to a wide variety of visual recognition tasks without task-specific fine-tuning, achieving performance competitive with fully supervised baselines on numerous benchmarks.

In our cases we have used the pre-trained weight, and fine-tuned the model in several way. The two main architecture where the CLIP model trained with 3 different losses, and a Siamese network trained with a contrastive loss.

**Siamese Network**

A Siamese Neural Network is a neural architecture designed

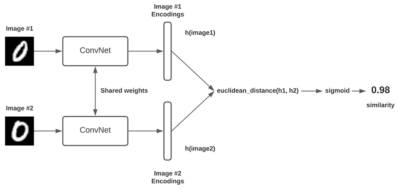


Figure 4: Original Siamese Network

to learn a similarity function between pairs of inputs. It is trained on a verification task, learning to determine whether two inputs belong to the same class. The architecture consists of two identical subnetworks with shared weights (fig4), which process the input samples in parallel. Each subnetwork extracts a feature representation of its input, and these embeddings are then compared using a distance function — in our case, cosine distance — to produce a similarity score.

The model is trained using contrastive loss. In the original formulation by Koch, Zemel, and Salakhutdinov 2015, the network uses a weighted L1 distance between the learned feature vectors, followed by a sigmoidal activation to produce a probability of similarity. Once trained, the network can be used to classify a novel input by comparing it to a single example from each candidate class and selecting the one with the highest similarity score.

We selected this model for its ability to learn highly discriminative visual features that generalize well across classes, enabling robust performance even under limited training conditions. Architecturally, we re-implemented a SimCLIP-inspired framework following the approach proposed by Huertas-Tato et al. 2024, replacing the original CNN backbone with the CLIP ViT-B/32 visual encoder. In our setup, only the visual branch is used, and the model is trained to assess the similarity (cosine distance) between pairs of images within a shared embedding space. To construct these image pairs, we randomly sampled 50% of the dataset using a fixed seed (seed = 1) and subsequently balanced the classes to ensure an equal distribution of similar and dissimilar pairs. The loss used is the contrastive loss.

**Contrastive Loss**

The contrastive loss (Chopra, Hadsell, and LeCun 2005) is a commonly used objective function for learning cross-modal embeddings. It is particularly suitable for tasks that involve comparing the similarity or dissimilarity between pairs of samples in a learned embedding space.

Let $D$ denote the distance between the embeddings of two inputs, and let $y \in \{0, 1\}$ be the label indicating their semantic relationship:

- $y = 0$: the inputs are **similar**
- $y = 1$: the inputs are **dissimilar**

The contrastive loss is defined as:

$$L = (1 - y) \cdot \frac{1}{2} D^2 + y \cdot \frac{1}{2} \left[\max(0, m - D)\right]^2$$

where $m > 0$ is a predefined margin that determines the minimum acceptable distance between dissimilar embeddings.

**Triplet Loss**

Triplet loss introduced by (Balntas et al. 2016) is a metric learning loss function designed to learn feature embeddings where similar inputs are mapped closer together and dissimilar inputs are pushed apart. It operates on triplets of samples: an anchor (a), a positive sample (p) from the same class, and a negative sample (n) from a different class. The formulation that we have used is the Margin Ranking Loss, formularized as

$$\mathcal{L}(\delta^+, \delta^-) = \max\left(0, \mu + \delta^+ - \delta^-\right)$$

Where:

- distance between the anchor and positive: $\delta^+ = \left(f(\mathbf{a}) - f(\mathbf{p})\right)^2$
- distance between the anchor and negative: $\delta^- = \left(f(\mathbf{a}) - f(\mathbf{n})\right\|^2$

This enforces that the positive pair is closer than the negative pair by at least margin $\mu$. No gradient is propagated if this condition is already satisfied. To generate training triplets, we adopt a random sampling strategy, where each triplet (a,p,n) is formed by randomly selecting an anchor and a corresponding positive sample from the same class, along with a randomly chosen negative sample from a different class. While this approach is computationally efficient and easy to implement, it often results in easy triplets where the distance between the anchor and the negative is already much larger than that between the anchor and the positive.

**Cross-Entropy Loss**

Cross-entropy loss is a widely used objective function in classification tasks, designed to measure the difference between the predicted class probabilities and the true class labels. Given an input sample $\mathbf{x}$, a model outputs a probability distribution over $C$ classes via a softmax layer:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

where $z_i$ is the logit (raw output) for class $i$, and $\hat{y}_i$ is the predicted probability for that class.

Given a true label $y \in \{1, \ldots, C\}$, the cross-entropy loss is defined as:

$$\mathcal{L}_{\text{CE}} = -\log(\hat{y}_y)$$

This encourages the model to assign high probability to the correct class, penalizing confident but incorrect predictions more heavily. The loss is differentiable and works well with gradient-based optimization methods. Although we did not use this type of loss directly, several of the loss functions we employed use it.

**Additive Angular Margin Loss**

The Additive Angular Margin Loss introduced by Deng et al. 2019 can be interpreted as a modification of the target logit prior to the application of the softmax function and, subsequently, the standard Cross-Entropy loss. In practice, the implementation consists of the following steps:

- Normalize both features and class weights

- Compute the cosine similarities
- Apply the additive angular margin $m$ to the target logit
- Rescale all logits by a fixed scalar $s$
- Pass the modified logits to the CrossEntropyLoss

This procedure can be formalized as follows:

$$\mathcal{L}_{\text{ArcFace}} = -\log\left(\frac{e^{s\cdot\cos(\theta_{y_i}+m)}}{e^{s\cdot\cos(\theta_{y_i}+m)}+\sum_{j\neq y_i} e^{s\cdot\cos(\theta_j)}}\right) \quad (1)$$

where $\theta_j = \arccos(\hat{\mathbf{W}}_j^\top \hat{\mathbf{x}}_i)$, with both the feature vector $\hat{\mathbf{x}}_i$ and the class weights $\hat{\mathbf{W}}_j$ being $\ell_2$-normalized, and $s$ is a fixed scaling factor that controls the strength of the softmax response.

**Clip-Loss**

Our implementation (ViT-B/32 + CLIP Loss) adopts the original loss function used to train CLIP. This loss is based on a symmetric contrastive learning framework, which jointly trains both the image and text encoders to produce aligned embeddings within a shared representation space. For each batch, the model computes cosine similarities between all image-text pairs, forming a similarity matrix in which each row corresponds to an image and each column to a text description.

This contrastive objective encourages the model to maximize similarity scores along the diagonal (correct pairings) while minimizing off-diagonal similarities (incorrect pairings). The similarities are scaled by a learnable temperature parameter $\tau$ that helps control the "sharpness" of the similarity distribution.

The loss function treats each correct image-text pair as a positive example and all other pairs in the batch as negatives. It applies cross-entropy loss in two symmetric directions:

- Image-to-text: For each image, treat its corresponding text as the correct answer among all texts in the batch
- Text-to-image: For each text, treat its corresponding image as the correct answer among all images in the batch

The cross-entropy loss internally converts the scaled similarities into probability distributions (via softmax) and then maximizes the probability assigned to the correct pairing while minimizing probabilities for incorrect pairs. The final loss is the average of these two directional losses, creating a symmetric training objective that pulls correct pairs together and pushes incorrect pairs apart in the embedding space. The clip loss is:

$$L_{CLIP} = \frac{1}{2}\left[\text{CE}(S, y, \text{axis}=0) + \text{CE}(S, y, \text{axis}=1)\right]$$

where

$$\text{CE}(S, y, \text{axis}=0) = -\frac{1}{N}\sum_{i=0}^{N-1}\log\frac{e^{S[y[i],i]}}{\sum_{j=0}^{N-1} e^{S[j,i]}}$$

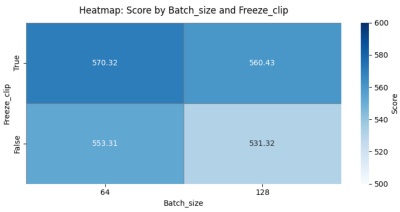


Figure 5: Hyperparameter Grid Search Results

is the image-to-text loss function and

$$\text{CE}(S, y, \text{axis}=1) = -\frac{1}{N}\sum_{i=0}^{N-1}\log\frac{e^{S[i,y[i]]}}{\sum_{j=0}^{N-1} e^{S[i,j]}}$$

is the text-to-image loss function

# 3 Evaluation

## 3.1 Quantitative Analysis

Even without fine-tuning, CLIP immediately achieved a score of approximately 345 points, significantly outperforming the CNN-based models. This confirmed CLIP's superior generalization capabilities, especially in contexts with high intra-class variability and limited representative data.

### 3.1.1 Training Setup and Epoch Selection

We configured the training pipeline using a learning rate of $10^{-4}$ with the Adam optimizer. Early stopping was not applied, allowing us to observe the full training dynamics up to 90 epochs. Selecting an appropriate number of epochs is a crucial step common to all loss functions. For each, the optimal training duration was defined by monitoring both training and validation loss across epochs. In this section, we illustrate the procedure using triplet loss as a representative example.

### 3.1.2 Freeze and Batch sizes

We performed a systematic hyperparameter optimization, applied to all loss functions. Here, we present the results obtained with triplet loss as an illustrative example. The hyperparameter optimized were the batch size and whether to freeze the CLIP visual encoder during training. Other parameters were kept fixed: dropout at 0.3, weight decay at $10^{-4}$, and margin at 0.2. As previously discussed, triplets were randomly sampled at each epoch. We tested two batch sizes (64 and 128) and two backbone configurations: full fine-tuning (Freeze_clip = False) and frozen encoder (Freeze_clip = True). The results, expressed in terms of final evaluation score, are shown in figure 5

Freezing the CLIP backbone consistently led to better performance across both batch sizes. This suggests that the pre-trained features from CLIP are already well-suited for this task, and further fine-tuning may introduce noise

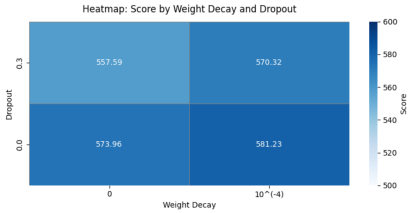


Figure 6: Regularization Ablation Results

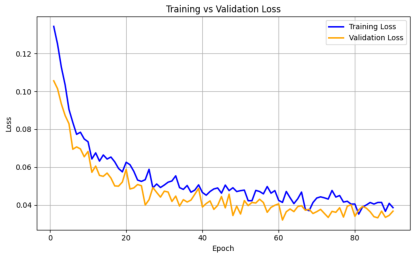


Figure 7: Training vs Validation Loss

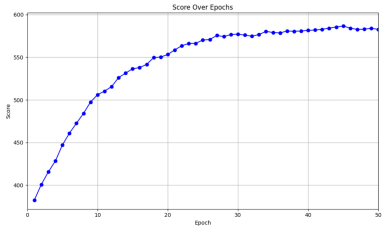


Figure 8: Score Over Epochs with Triplet Loss

or lead to suboptimal solutions, particularly given the limited dataset size. Freezing the encoder allows the optimization to focus on the projection layers directly involved in the triplet loss. Batch size also influenced performance. Smaller batches (64) outperformed larger ones (128), achieving the highest score of 570.32 with Freeze_clip = True. Smaller batches likely introduce greater variability in sampled triplets, improving the exploration of the embedding space and enhancing generalization. In conclusion, the optimal configuration identified by using triplet loss consists of batch size 64, frozen CLIP backbone, dropout 0.3, weight decay $10^{-4}$, and triplet margin 0.2. This setup was adopted for subsequent ablation studies.

### 3.1.3 Regularization Ablation

After selecting the optimal batch size and CLIP freezing configuration, we analyzed the impact of regularization techniques on model performance. As in previous stages, this ablation study was performed across all loss functions, but here we present the results for triplet loss as a representative case. We evaluated two regularization mechanisms: dropout and weight decay, using the previously optimized setup (batch size 64, frozen CLIP, 35 epochs). Dropout was tested at 0 and 0.3, while weight decay was set to either 0 or $10^{-4}$. The results are summarized in 6

The results reveal a clear trend. Weight decay consistently improved performance, with $10^{-4}$ yielding the highest score of 581.23, achieved without dropout. This suggests weight decay effectively controls weight magnitudes and improves generalization without relying on dropout. Conversely, applying dropout at 0.3 degraded performance in all cases. Introducing random noise into the activations likely disrupts the embedding space learned by the model, which is particularly sensitive in metric learning tasks like triplet loss. Unlike classification tasks where dropout may prevent co-adaptation, stable activations here favor consistent embedding structures. Based on these findings, we selected weight decay of $10^{-4}$ and dropout of 0 as the optimal regularization setup for subsequent experiments.

### 3.1.4 Loss function comparison

**Triplet loss** After selecting the optimal hyperparameters, specifically, batch size of 64, frozen CLIP backbone, weight decay of $10^{-4}$, no dropout, and a margin of 0.2, we evaluated the final training dynamics of the triplet loss as shown in figure 7

The training loss decreases sharply during the initial epochs and then more gradually. The validation loss follows a similar trend but stabilizes earlier, remaining consistently lower than the training loss for most of the training process. This behavior, while counterintuitive, is explained by the random sampling of triplets at each training epoch. During training, the model continually encounters varying combinations of easy and difficult triplets, often including hard violations of the margin, resulting in higher average training loss. Conversely, validation was computed on a fixed set of triplets sampled once, which may include fewer hard examples, leading to lower average loss values even when generalization no longer improves. Around epoch 35, the validation loss stabilizes, suggesting the model has captured most of the generalizable structure. Meanwhile, training loss keeps decreasing slowly as easier triplets are optimized.

We analyzed the score evolution across epochs to assess convergence under the optimized configuration (Figure 8).

The plot shows that the score improves rapidly during the early training phases. The most significant gains are observed within the first 20–25 epochs, after which the growth rate begins to slow down. The model continues to improve slightly beyond this point, but the score progressively stabilizes, reaching a plateau around epoch 35, achiving the convergence, with no meaningful improvements in subsequent training. Under these conditions, the triplet loss successfully leads the model towards an optimal embedding configuration capable of discriminating between identities in the retrieval task.

**Siamese Network with Contrastive Loss** For the contrastive loss experiments, we adopted a configuration similar to the one previously optimized for the triplet loss. The batch size was kept at 64, weight decay at $10^{-4}$, dropout set to 0.3, and the margin for the contrastive loss kept consistent with standard practice. Differently from the previous setup, the CLIP backbone was not frozen in this case, allowing full fine-tuning of the entire network. The training

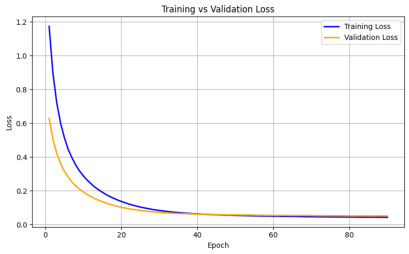


Figure 9: Training vs Validation Loss for Contrastive Loss

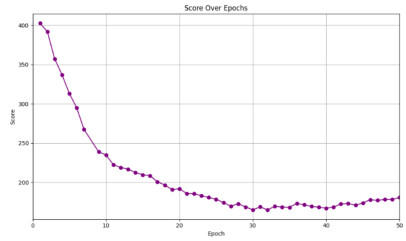


Figure 10: Score Over Epochs for Contrastive Loss

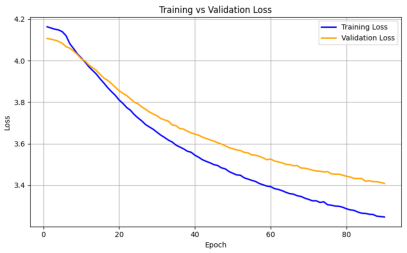


Figure 11: Training vs Validation Loss for CLIP Loss

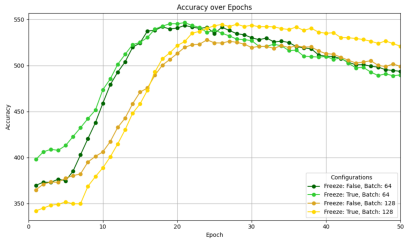


Figure 12: Score Over Epochs for CLIP Loss

dynamics of the contrastive loss are shown in Figure 9, which illustrates the evolution of both training and validation loss across epochs.

Both training and validation loss curves exhibit a rapid decrease during the early training phases, indicating effective initial learning. The two losses progressively converge and stabilize around epoch 30, suggesting that the model reaches a form of convergence in terms of minimizing the contrastive loss objective.

However, the evolution of the evaluation score, displayed in Figure 10, reveals a different behavior in terms of generalization.

While the score starts at a reasonably high value during the initial epochs, it consistently declines as training progresses. This indicates that, despite minimizing the loss, the model's ability to correctly retrieve images on the test set degrades over time. After approximately 30 epochs, the score stabilizes at a significantly lower value compared to its initial performance, suggesting a phenomenon of overfitting to spurious features that do not generalize well to unseen data. This divergence between loss minimization and score degradation highlights the sensitivity of contrastive loss in this setting, especially when full fine-tuning is applied on top of a pre-trained backbone. In this case, the model likely overwrites some of the valuable pre-trained embeddings, leading to a deterioration in retrieval performance despite an apparently successful optimization from the loss perspective.

**CLIP Loss** For the CLIP loss experiments, we maintained the same general hyperparameter configuration adopted in the previous tests. The training was performed using a learning rate of $10^{-4)}$, batch sizes of 64 and 128, weight decay of $10^{-4}$, dropout of 0.3, and with experiments both freezing and not freezing the CLIP backbone. The evolution of training and validation loss for CLIP loss is shown in Figure 11.

Both loss curves exhibit regular and stable convergence throughout training. The losses decrease smoothly, and both training and validation curves continue to follow a downward trend across all epochs, indicating that the model is consistently optimizing the loss function without clear signs of overfitting in terms of the loss value itself. However, when analyzing the actual retrieval performance over epochs, a more complex behavior emerges. The score evolution across different configurations is presented in Figure 12.

The results show that smaller batch sizes lead to faster learning in the early stages. In particular, with batch size 64, the model reaches its peak score earlier, around epoch 20, whereas with batch size 128 the peak occurs later, around epoch 25. Nevertheless, a significant difference with respect to the triplet loss is observed: after reaching the maximum score, the performance progressively deteriorates as training continues. This decline suggests that prolonged training lead to overfitting or a form of drift in the learned embedding space, even though the loss continues to decrease. Regarding the freezing of the CLIP backbone, the results indicate a slight advantage when freezing is applied. For both batch sizes, configurations with frozen CLIP weights consistently outperform their non-frozen counterparts throughout most of the training process. The best configuration was obtained with batch size 64 and frozen CLIP , reaching a maximum score of 546.53 before the degradation phase.

**Cross Entropy + ArcFace** For the ArcFace (Additive Angular Margin Softmax) experiments, we maintained the same training configuration used in the previous loss functions: learning rate of $10^{-4}$, weight decay of $10^{-4}$, dropout 0.3, and batch sizes of 64 and 128. Both frozen and unfrozen versions of the CLIP backbone were tested. The evolution of training and validation loss during ArcFace training is shown in 13.

Both training and validation losses decrease steadily throughout the training process. The curves show smooth

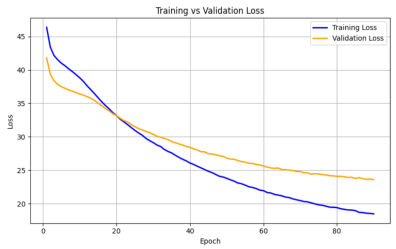


Figure 13: Training vs Validation Loss for ArcFace

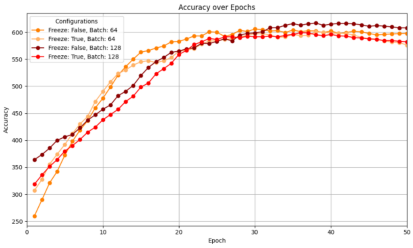


Figure 14: Score Over Epochs for ArcFace

convergence behavior without any sudden divergence between training and validation losses. The stability of both curves suggests that the model consistently learns useful representations for the classification objective throughout the entire training process. The corresponding retrieval performance, expressed as score over epochs, is presented in Figure 14.

From these results, we observe learning speed is correlated with the batch size. With batch size 64, the model achieves its peak performance slightly earlier than with batch size 128. However, unlike previous loss functions, here the configurations using batch size 128 consistently reach higher maximum scores and maintain stronger overall performance throughout training. Additionally, freezing the CLIP backbone appears to be less beneficial in this setting. The unfrozen configurations generally outperform the frozen ones, suggesting that ArcFace benefits from fine-tuning the full model and allows the network to better adapt the pre-trained representations to the specifics of the face retrieval task. The best performance was achieved with the configuration using batch size 128 and no freezing, which reached a maximum score of 616.98.

#### 3.1.5 Summary and Comparative Analysis

In this section, we summarize and compare the results obtained across all tested loss functions, using the optimal hyperparameter configurations identified for each. The evolution of the evaluation score over epochs for the best-performing configurations of each loss function is shown in Figure 15.

The plot clearly highlights the different learning dynamics of each loss. The triplet loss and ArcFace loss both show strong and stable performance, while clip loss reaches its peak earlier but starts degrading with continued training. In contrast, the siamese contrastive loss shows poor generalization, with the score rapidly decreasing after initial training,

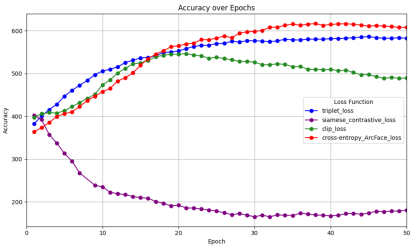


Figure 15: Score Over Epochs for the Best Configurations

confirming its unsuitability for this specific retrieval task.

The final best scores and the corresponding hyperparameter configurations are summarized in Table 3

| Loss | Frz | B_s | Ep | W_d | Drp | Score |
|---|---|---|---|---|---|---|
| Triplet | T | 64 | 45 | 1e-4 | 0.0 | 586.53 |
| CLIP | T | 64 | 21 | 1e-4 | 0.3 | 546.53 |
| ArcFace | F | 128 | 38 | 1e-4 | 0.3 | **616.98** |

Table 3: Compact comparison of loss functions and training settings.

Among all models tested, the best overall performance was achieved by the ArcFace loss, with a maximum score of 616.98, obtained using batch size 128, no freezing of the CLIP backbone, weight decay of $10^{-4}$, and dropout 0.3. This confirms the strong capacity of ArcFace to effectively leverage both the pre-trained features and the fine-tuning process for this face retrieval task.

### 3.2 Qualitative Analysis

As shown in Figure 16, the retrieval results obtained with ArcFace demonstrate its effectiveness in capturing meaningful visual similarities. In contrast, the Siamese network fails to retrieve semantically related images, as illustrated in the corresponding worst-case example.


Figure 16: Qualitative example of the performances of the models.

# 4 Final Discussion

The main takeaway from this project is the critical role that model selection plays for the face image retrieval addressed in this work. Our early experiments with CNN-based architectures such as ResNet50 and EfficientNet—despite extensive tuning and use of metric learning losses—consistently produced poor results, with scores plateauing below 50. In contrast, switching to a transformer-based backbone pre-trained with multimodal supervision (CLIP ViT-B/32) yielded a noticeable leap in performance, even without fine-tuning. This finding underscores that, in low-data and high-variance regimes like face retrieval, choosing a suitable pre-trained representation is far more impactful than any isolated design choice. Building on the CLIP backbone, we explored four different loss functions—triplet loss, CLIP loss, ArcFace, and contrastive loss in a Siamese setting—each evaluated under carefully controlled training conditions. Among them, ArcFace achieved the best retrieval performance, reaching a final score of 616.98 when combined with full fine-tuning and large batch sizes. Triplet loss also performed robustly, especially with a frozen encoder and moderate regularization. CLIP loss demonstrated early gains but exhibited performance degradation over time, likely due to embedding drift. The Siamese contrastive loss, while conceptually simple, failed to generalize and led to steady drops in retrieval score, reinforcing its limitations in this setting. While this study covered a range of loss functions and training setups, only a limited subset of possible hyperparameter combinations were tested. Other values for learning rates, margins, or projection architectures might lead to better performance. We also did not experiment with techniques like hard negative mining or alternative regularization beyond dropout and weight decay. Finally, all experiments were done with a fixed image resolution and embedding size. This leaves room to explore other directions that could lead to improved retrieval performance.

The code can be found on GitHub: `https://github.com/CrispyCocaBoy/Loss_Comparison_for_Image_Retrieval_Challenge`

# 5 Workload Table and Team Contributions

**Matteo Massari**
– ViT-B/32 Siamese (Contrastive Loss)
– ArcFace implementation
- Data Augmentation
– Writing of Chapters 1–2-Abstract- Qualitative analysis

**Andrea Battaglia**
– Base ViT-B/32 implementation
– Triplet Loss variant
– Quantitative evaluation

**Tommaso Cestari**
– CLIP Loss implementation
– Writing of Chapter 4
– Final editing and proofreading

# References


Balntas, Vassileios et al. (2016). "Learning Local Feature Descriptors with Triplets and Shallow Convolutional Neural Networks". In: *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 119.1–119.11. DOI: 10.5244/C.30.119.

Biewald, Lukas (2020). *Experiment Tracking with Weights and Biases*. Software available from wandb.com. URL: https://www.wandb.com/.

Chopra, Sumit, Raia Hadsell, and Yann LeCun (2005). "Learning a Similarity Metric Discriminatively, with Application to Face Verification". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 539–546.

Deng, Jiankang et al. (2019). "ArcFace: Additive Angular Margin Loss for Deep Face Recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4690–4699.

Dosovitskiy, Alexey et al. (2020). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929. arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

Huertas-Tato, Javier et al. (2024). "A CLIP-based Siamese Approach for Meme Classification". In: *arXiv preprint arXiv:2409.05772*. Available at https://arxiv.org/abs/2409.05772.

Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov (2015). "Siamese neural networks for one-shot image recognition". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. Available at: https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf. JMLR Workshop and Conference Proceedings.

Luu, Nhan T. (2021). "CLIP: Unreasonable Potential in Single-Shot Face Recognition". In: *arXiv preprint arXiv:2108.04889*. URL: https://arxiv.org/abs/2108.04889.

Mumuni, Alhassan and Fuseini Mumuni (2022). "Data augmentation: A comprehensive survey of modern approaches". In: *Array* 16, p. 100258. DOI: 10.1016/j.array.2022.100258.

Radford, Alec et al. (2021). "Learning Transferable Visual Models From Natural Language Supervision". In: *CoRR* abs/2103.00020. arXiv: 2103.00020. URL: https://arxiv.org/abs/2103.00020.

Rota, Paolo (2025). *Generalization with Neural Networks*. Appunti di lezione, Spring 2025, Generalization.pdf.