Queen’s University – Smith Engineering

ELEC 475 – Lab 4 (Project) Fall 2025

Submitted by:

Erhowvosere Otubu – Student Number: 20203052

Mihran Asadullah – Student Number: 20285090

Date Submitted: December 2, 2025

Table of Contents

Introduction 1
Motivation 1
Structure 1
Purpose of Project 2
Methodology 3
Baseline Model Design 3
Image Encoder 3
Text Encoder 3
Project Head 3
Training Design 3
Information Noise-Contrastive Estimation Loss Implementation 4
Modification 1 Design 5
Modification 2 Design 6
Results 6
Baseline Model Quantitative Metrics 6
Modifications 10
Weight Decay & Data Augmentation 10
Unfreezing the Backbone 14
Discussion 18
Conclusion 18
Large Language Model Conversation 19
Appendix 20
References 23

Table of Figures

Figure 1: CLIP neural network architecture (1/2) 2
Figure 2: CLIP neural network architecture (2/2) 2
Figure 3: Loss curve plot for the baseline CLIP model 7
Figure 4: (1/2) Top 5 retrieved images for the baseline model – "A young swimmer walks across surf boards being held" 8

Figure 5: (2/2) Top 5 retrieved images for the baseline model – A table with a plate of pizza and a plate of shrimp on it” 9
Figure 6: Image-to-Text zero-shot classification for the baseline model 9
Figure 7: Loss curve plot for the modified CLIP model with weight decay and data augmentation 11
Figure 8: (1/2) Top 5 retrieved images for the modified model (weight decay and data augmentation) – “The woman is sitting on the motorcycle by the plants” 12
Figure 9: (2/2) Top 5 retrieved images for the modified model (weight decay and data augmentation) – “a dog stretching on a stone railing next to a body of water.” 13
Figure 10: Image-to-Text zero-shot classification for the modified model (weight decay and data augmentation) 13
Figure 11: Loss curve plot for the modified CLIP model with weight decay, data augmentation, and unfrozen layer 14
Figure 12: (1/2) Top 5 retrieved images for the modified model (unfrozen layer) – “A pizza that is sitting on a plate” 16
Figure 13: (2/2) Top 5 retrieved images for the modified model (unfrozen layer) – “A parking meter at the corner of a deck in a garden” 17
Figure 14: Image-to-Text zero-shot classification for the modified model (weight decay, data augmentation, unfrozen layer) 17
Figure 15: Text-to-Image and Image-to-Text Retrieval Scores for baseline model from Visual Studio Code output terminal 20
Figure 16: Text-to-Image and Image-to-Text Retrieval Scores for the modified model with weight decay and data augmentation from Visual Studio Code output terminal 21
Figure 17: Text-to-Image and Image-to-Text Retrieval Scores for the modified model with weight decay, data augmentation, and unfrozen layer from Visual Studio Code output terminal 22

Table of Figures

Table 1: Text-to-Image Retrieval Scores for baseline model 7
Table 2: Image-to-Text Retrieval Scores for baseline model 7
Table 3: Text-to-Image Retrieval Scores for the modified model with weight decay and data augmentation 11
Table 4: Image-to-Text Retrieval Scores for the modified model with weigh decay and data augmentation 11
Table 5: Text-to-Image Retrieval Scores for modified model with weight decay, data augmentation, and unfrozen layer 15
Table 6: Image-To-Text Retrieval Scores for modified model with weigh decay, data augmentation, and unfrozen layer 15

Introduction

Motivation

Contrastive Language-Image Pre-training, also known as CLIP, is an advanced artificial intelligence model with the unique ability to understand and relate both textual descriptions and images [1]. The CLIP framework learns a joint embedding space where images and text captions that describe similar concepts lie close together. By training a model to align these modalities, CLIP enables flexible zero-shot classification, retrieval, and image–text reasoning. This way, the model can understand that an image, for example, a dog and the text "a photo of a dog" are semantically similar. The system can reason about visual concepts using natural language processing, allowing for robust performance on tasks it was not explicitly trained for.

Structure

CLIP is considered a multimodal system because it handles two different types of data simultaneously: vision data and language data. Its architecture consists of two main encoders trained simultaneously.

One of the encoders is the Image Encoder. This encoder processes the input image. Its architectures include ResNet or a Vision Transformer (ViT). This encoder transforms images into embeddings that capture the image’s key features [1]. The other main encoder is the Text Encoder. This encoder processes the associated text caption or prompt. This model converts text into embeddings, dense vectors that capture the meaning of the text [1]. Figure 1 and Figure 2 detail the architecture of the CLIP neural network. It demonstrates the contrastive pre-training that both encoders go through to process and embed their respective data types, creating the dataset classifier from label text, and then the zero-shot prediction it’s able to do.

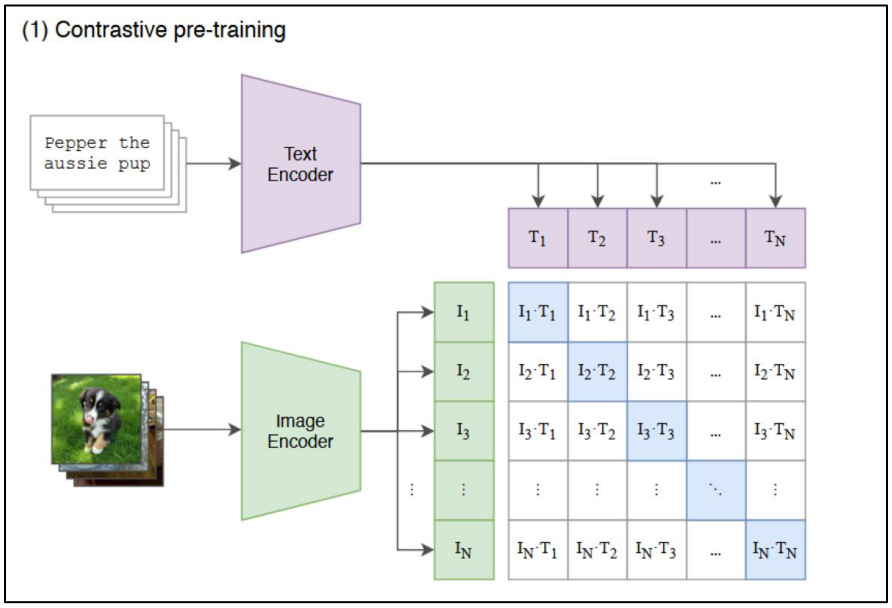

(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$

Figure 1: CLIP neural network architecture (1/2)

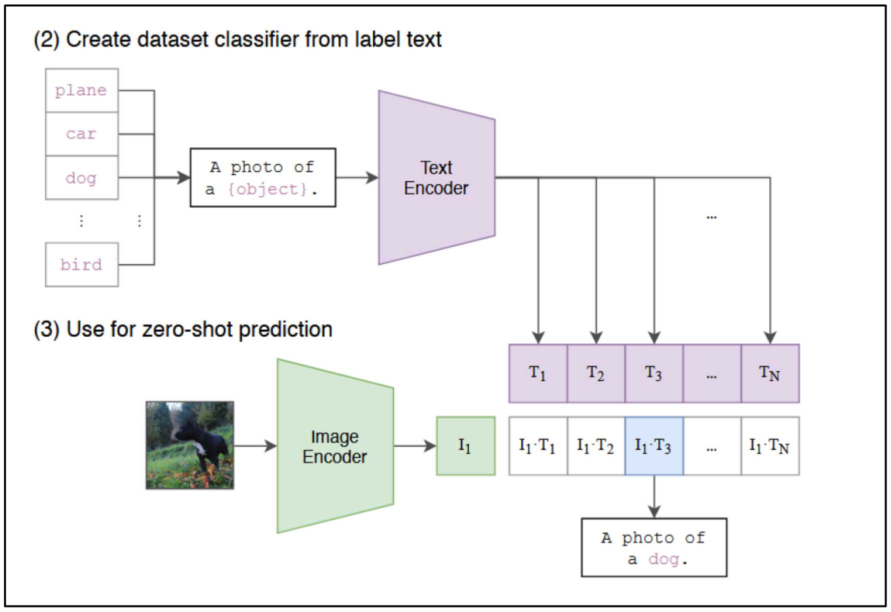

(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
(3) Use for zero-shot prediction
Image Encoder
T_1 T_2 T_3 ... T_N
I_1
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
A photo of a dog.

Figure 2: CLIP neural network architecture (2/2)

Purpose of Project

The purpose of this lab project is to build and train a vision-language retrieval system using the CLIP framework. To be more specific, the group must implement a CLIP-style model and train it to where the text encoder (a pretrained Transformer from OpenAI's CLIP model) remains frozen and the image encoder (made up of a ResNet50 backbone) is fine-tuned to align its output embeddings with those from the text encoder. In addition, the model must be trained on the COCO 2014 dataset, which provides paired image-caption annotations. The contrastive loss will be defined using Information Noise-

Contrastive Estimation (InfoNCE) loss to pull matching image-text pairs together and push non-matching pairs apart. Lastly, to test and evaluate the system's ability to perform Image-to-Text and Text-to-Image retrieval, success will be measured using the Recall metric (i.e., Recall@1, Recall@5, and Recall@10).

Methodology

Baseline Model Design

The core architecture for the group's system is based on the CLIP framework. The model consists of an image and text encoder that maps inputs into a shared 512-dimensional embedding space. Details of the model's design are discussed in further detail in the following sections below.

Image Encoder

For the image encoder, the group implemented a ResNet50 convolutional neural network (CNN) as the backbone. The model was initialized with pre-trained ImageNet weights. The original fully connected classification layer of the ResNet50 was removed. The output from the final Global Average Pooling layer was utilized, which produced a 2048-dimensional feature vector representation of the image. For the baseline experiment, the weights of the ResNet50 CNN were kept frozen.

Text Encoder

A pre-trained Transformer text encoder from OpenAI's *clip-vit-base-patch32* model was utilized for the text encoder. Its role is to take the tokenized captions and process them into a numerical representation. Specifically, the model outputs a single 512-dimensional vector for each caption, which summarizes the entire meaning of the sentence into one set of numbers. To ensure the model had a stable reference point for learning, the team froze the text encoder during training.

Project Head

To map the 2048-dimensional visual features into the shared embedding space, a non-linear projection head was implemented. This project head feature transforms the general visual features extracted by the ResNet50 into a specific embedding space required for comparison with text. It is composed of a Multi-Layer Perceptron (MLP) containing two linear layers separated by a Gaussian Error Linear Unit (GELU) activation function. The first linear layer maintains the dimensionality of the input features, while the second linear layer compresses this representation down to 512 dimensions to match the output size of the text encoder. This architecture allows the model to learn a complex, non-linear transformation that projects generic ResNet visual features into the specific semantic vector space required for accurate text-image comparison.

Training Design

This section covers the training design and considerations for each model modification. The goal of each model redesign was to prevent the overfitting issue and increase the Recall@K metric scores.

Information Noise-Contrastive Estimation Loss Implementation

The goal of the InfoNCE loss in this lab is for the CLIP-model to learn a joint embedding space where semantically related images and texts are "pulled together" (high similarity) and unrelated pairs are "pushed apart" (low similarity). It was created using the **info_nce_loss()** method in the *train.py* file.

The function was defined as:

```
def info_nce_loss(image_features, text_features, temperature=0.07):
```

Where:

- **image_features**: A tensor of shape (N, 512) representing the batch of visual embeddings produced by the image encoder, where N is the batch size.
- **text_features**: A tensor of shape (N, 512) representing the corresponding batch of semantic embeddings produced by the frozen text encoder
- **temperature**: A hyperparameter used to scale the logits

Normalization was defined as the following:

```
    image_features = image_features / image_features.norm(dim=1, keepdim=True)
    text_features = text_features / text_features.norm(dim=1, keepdim=True)
```

Using L2 normalization to normalize the raw vectors to a unit length, the dot product between them becomes the Cosine Similarity. This ensured that the model learns based on the direction (semantic meaning) of the vectors, not their length. The math equation below defines the idea of the normalization process:

$$\hat{u} = \frac{u}{||u||}$$

Once normalized, the code completed a similarity matrix computation by performing a matrix multiplication between the image features and the transposed text features.

```
logits = (image_features @ text_features.t()) / temperature
```

This line performs a matrix multiplication that results in an N x N matrix of logits. The diagonal elements of this matrix correspond to the “positive pairs” (i.e., an image and its ground-truth caption), while the off-diagonal elements represent “negative pairs” (i.e., an image paired with unrelated captions from the same batch). These logits were then scaled by the temperature parameter, which sharpened the probability distribution and forced the model to make more confident predictions by amplifying the difference between high and low scores.

The training objective is treated as a classification problem where the goal is to match the correct image-text pairs within the batch. The ground truth labels were generated using this line:

```
labels = torch.arange(len(logits)).to(DEVICE)
```

The above line acted as the answer key. Using **torch.arange()**, the function creates a sequential list of numbers (e.g., [0, 1, 2, …, N]) that matched values of the batch size. It tells the loss function that the correct match for the i-th image is the i-th text description from the corresponding similarity matrix along the diagonal elements

Lastly, the function calculated the loss into two different directions and returned the averaged result:

```
    loss_i = nn.functional.cross_entropy(logits, labels)
    loss_t = nn.functional.cross_entropy(logits.t(), labels)
    return (loss_i + loss_t) / 2
```

The **loss_i** variable represents the image-to-text direction. It treated the row of similarities as a classification problem where the correct class is the diagonal index. The **loss_t** variable represents the text-to-image direction. It transposed the matrix and solveed the classification problem column-wise. The final loss is the average of both directions. This enforced a stronger constraint, ensuring the relationship holds regardless of whether there is a search for images using text or text using images.

This is the mathematical formulation of the InfoNCE loss implemented. For a batch of N pairs, the loss for the i-th pair is:

$$\mathrm{L}_i = -log\frac{exp(sim(I_i, T_i)/\tau)}{\sum_{j=1}^{N} exp(sim(I_i, T_j)/\tau)}$$

Where:

- $sim(I_i, T_i)$ is the cosine similarity (dot product of normalized vectors)
- τ is the temperature parameter
- The numerator maximizes the score of the correct pair
- The denominator minimizes the score of all incorrect pairs

Modification 1 Design

The first modifications were created to address the overfitting observed from the loss curve plot created during the baseline model's training session. The team implemented regularization and data augmentation techniques. To possibly prevent the model from overfitting, the team used color jittering (**transforms.ColorJitter**). This augmentation randomly adjusted brightness, contrast, and saturation by a factor of 0.2 and hue by a factor of 0.1 for every training image, forcing the model to learn structural and semantic features rather than relying on specific lighting conditions or color histograms. Additionally, the team replaced the original Adam optimizer with the AdamW optimizer, applying a weight decay of 1e-4. This type of regularization penalty discouraged the model weights from growing too large, effectively smoothing the decision boundaries, and further enhancing generalization to unseen validation date.

Modification 2 Design

For the final model modifications, the team decided to freeze the first three layers and unfreeze the fourth and final residual block. By setting these parameters to be trainable, the team allowed the model to adapt its high-level object representations to the specific classes and contexts found in the COCO captions. To support this fine-tuning, the batch size was increased to 64 used the NVIDIA RTX Ada Generation GPU and used a Cosine Annealing Learning Rate Scheduler. This scheduler gradually reduced the learning rate from 1e-4 to 0 over the training session, allowing the model to settle into a more optimal local minimum while fine-tuning the semantic layers.

Results

This section goes over the quantitative and qualitative metrics like the loss curves, Recall@K scores, and visualizations from training the model's baseline and modified adjustments.

To ensure computational feasibility within the constraints of the available hardware at Queen's University's Bain Lab (NVIDIA GeForce RTX 3060 Ti and NVIDIA RTX Ada Generation), all the experiments were conducted using a consistent random subset of the COCO 2014 dataset which was downloaded from KaggleHub using its API. There were about 5 captions associated with each image. This resulted in using approximately 278,325 total training pairs (image and caption), and 135,930 total validation pairs. To accelerate the training process, the text embeddings from the full train and validation sets were pre-computed and cached. However, during the initial setup, the group identified discrepancies where cached entries referenced non-existent image files, so a cleaning script was used to remove non-existent image files, ensuring a stable training pipeline. For model performance at each stage (baseline and modifications), InfoNCE loss was used during training and Recall@K metrics during evaluation.

Baseline Model Quantitative Metrics

Here are the following hyperparameters and hardware used for the baseline training session:

- Epochs = 10
- Batch Size = 32
- Learning Rate = 1e-4
- Optimization: AdamSGD
- Hardware: NVIDIA GeForce RTX 3060 Ti

Overall, training for this session took 208.80 minutes or 3.48 hours. Figure 3 shows the training and validation loss curves over the 10 epochs. As seen in the plot, the training loss shows a consistent and rapid descent from ~0.47 to ~0.11 which indicates that the model successfully learned to map the training images to their captions. However, the validation loss curve decreased initially up until Epoch 3 (~0.38) but started to plateau and slowly rise up again to its initial starting point up until the end of Epoch 10. From this plot, there is clear overfitting. While the model continued to memorize the training data, it stopped generalizing to unseen data after Epoch 3.

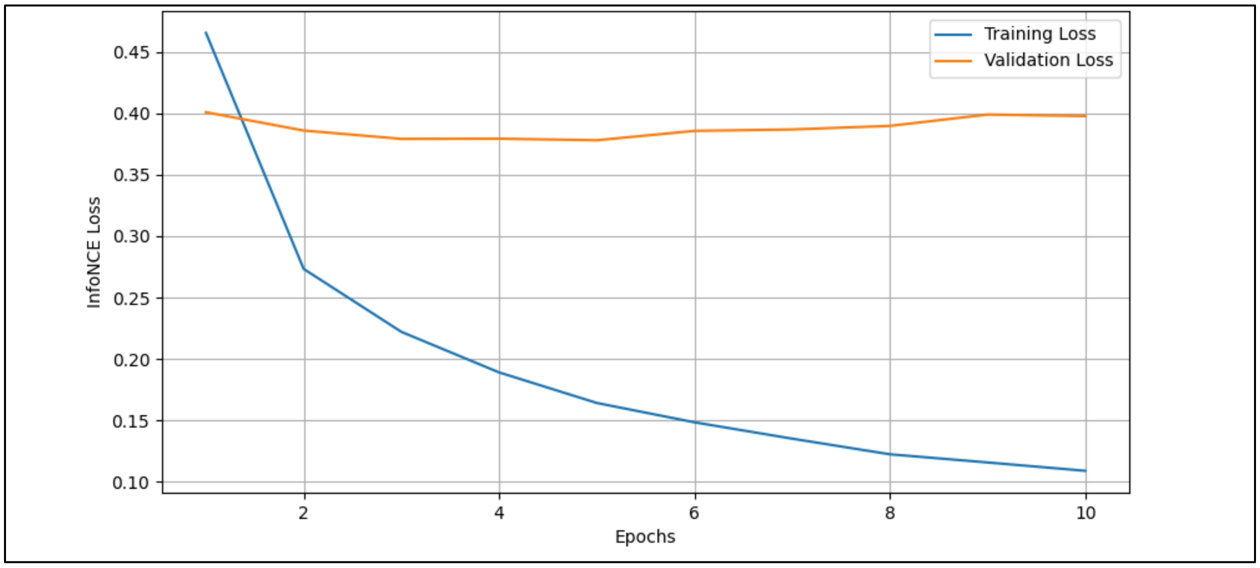


Figure 3: Loss curve plot for the baseline CLIP model

Following training, the baseline model was evaluated using the full validation set of 135,930 pairs. The results are summarized below in Table 1 and Table #.

Table 1: Text-to-Image Retrieval Scores for baseline model

Recall@K	**Score (%)**	**Interpretation**
K = 1	9.52	The correct image was the first result 9.52% of the time
K = 5	16.83	The correct image appeared in the top 5 results about 16.83% of the time
K = 10	23.04	The correct image appeared in the top 10 results about 23.04% of the time

Table 1 summarizes the quantitative performance of the baseline model when evaluated on a random subset of 135,930 validation pairs. These metrics suggest that while the model occasionally struggles to differentiate between semantically similar images at the very top rank, it can place the correct image within the top few results, demonstrating decent retrieval capability.

Table 2: Image-to-Text Retrieval Scores for baseline model

Recall@K	**Score (%)**	**Interpretation**
K = 1	8.27	The correct caption was the first result 8.27% of the time
K = 5	21.65	The correct caption appeared in the top 5 results about 21.65% of the time

K = 10	30.61	The correct image appeared in the top 10 results about 30.61% of the time

Table 2 shows how well the baseline model retrieves the correct caption when given an image. The Recall@1 score indicates the exact caption was the model's top choice only a small portion of the time, while Recall@5 and Recall@10 show that the model is much better at correlating the correct caption somewhere within its top few guesses. Overall, these results indicate that the baseline CLIP model can mostly capture the general meaning of an image but seems to struggle with consistently ranking the correct caption first.

To understand the model's behavior, the group visualized the top 5 retrieved images for specific queries and a zero-shot classification graph for image-to-text classification. Although the top 5 image results are often identical, this is because the COCO dataset structure contains five captions for every unique image, causing the correct image to appear five times in the search index.

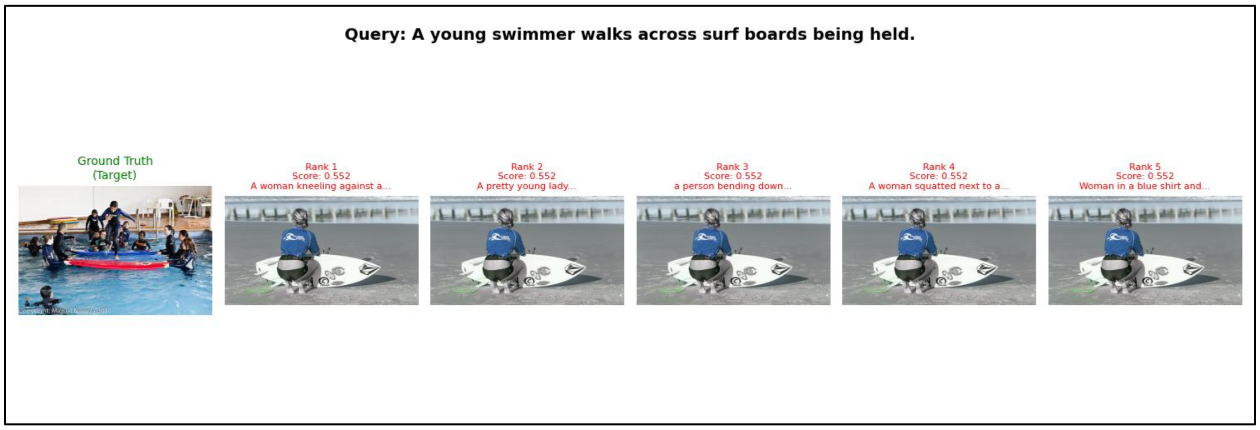


Figure 4: (1/2) Top 5 retrieved images for the baseline model – "A young swimmer walks across surf boards being held"

For Figure 4, the query used was "*A young swimmer walks across surf boards being held.*" The model retrieved an image of a women sitting very closely to a surfboard. In this example, the model successfully attended to the keyword "surfboard" but failed to capture the action ("walks across") or the context (a pool setting vs. the ocean).

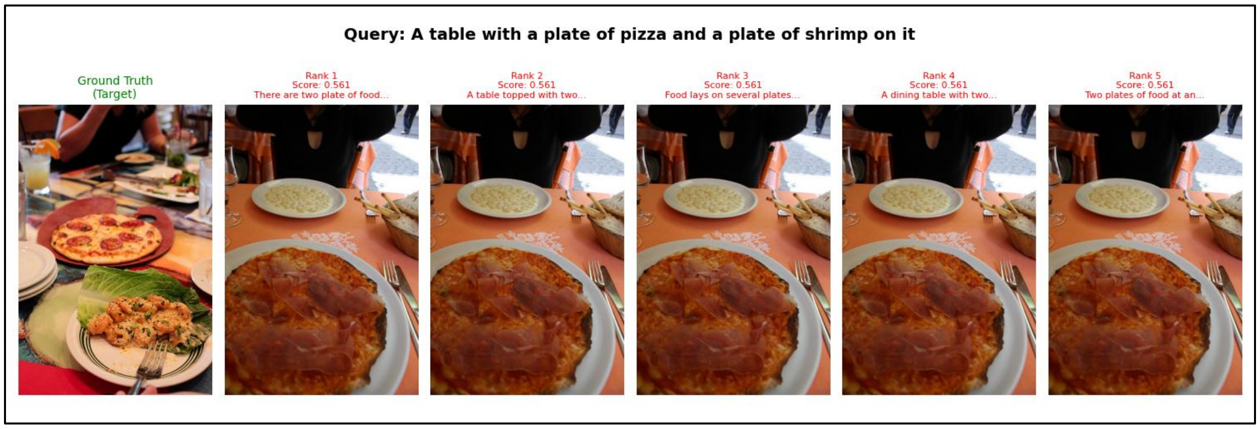

Query: A table with a plate of pizza and a plate of shrimp on it
Ground Truth (Target)
Rank 1 Score: 0.561 There are two plate of food...
Rank 2 Score: 0.561 A table topped with two...
Rank 3 Score: 0.561 Food lays on several plates...
Rank 4 Score: 0.561 A dining table with two...
Rank 5 Score: 0.561 Two plates of food at an...

Figure 5: (2/2) Top 5 retrieved images for the baseline model – A table with a plate of pizza and a plate of shrimp on it"

For Figure 5, the query used was "*A table with a plate of pizza and a plate of shrimp on it.*" The model retrieved an image of a table with pizza. Although it did not match the ground truth image and caption, the retrieval was semantically relevant. The model correctly identified "pizza" and "table", but it missed the specific detail of "shrimp." It's possible that it may have mistaken the other plate in the ranked images as shrimp.

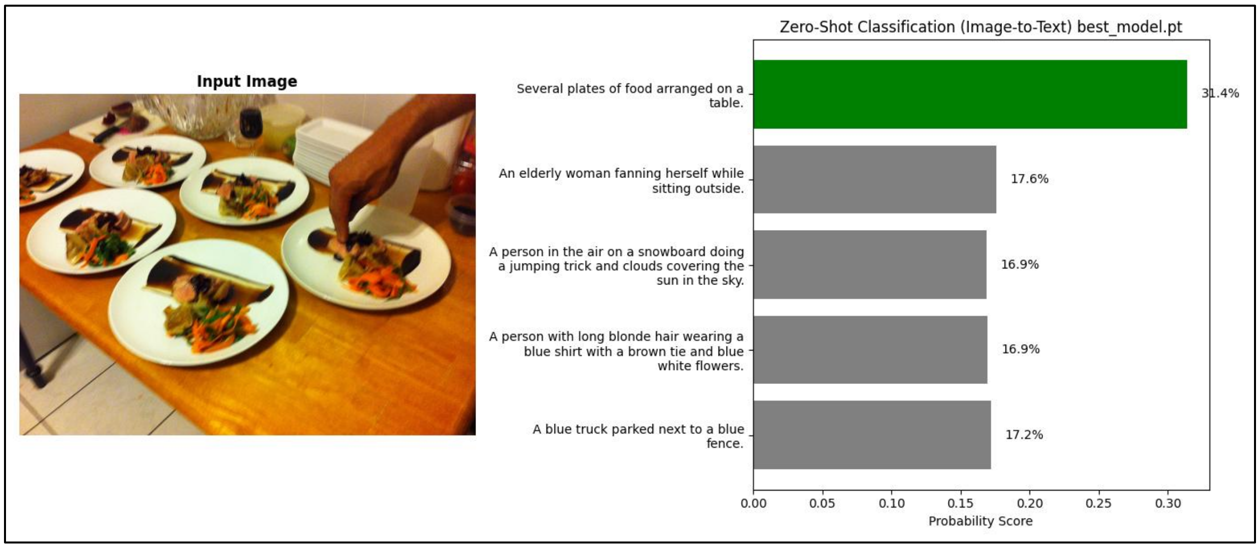


Figure 6: Image-to-Text zero-shot classification for the baseline model

The graph in Figure 6 above displays the zero-shot image-to-text classification results for the baseline model. When shown the image of several plated meals on a table, the model gave the highest score (31.4%) to the correct caption score much lower. This shows that the model can pick up the main idea of the image and clearly prefers the correct description over the unrelated ones. This matches the earlier findings that the model is good at identifying dominant objects in a scene.

Modifications

Weight Decay & Data Augmentation

Here are the following hyperparameters and hardware used for the training session that modified the model. The modifications utilized regularization and data augmentation:

- Epochs = 5
- Batch Size = 32
- Learning Rate = 1e-4
- Optimization: AdamW
- Hardware: NVIDIA GeForce RTX 3060 Ti
- Weight Decay = 1e-4
- Data Augmentation: Color Jitter

Due to overfitting that was discovered during the baseline training session, the group decided to try regularization in the form of weight decay and AdamW optimization. Weight decay is used to prevent overfitting, improve generalization, and stabilize training. Additionally, the AdamW optimizer was used to decouple the weight decay from the gradient-based parameter update. In other words, the weight decay is applied *after* the main parameter update step, ensuring that the regularization is applied consistently across all parameters, regardless of their gradient history. Data augmentation was also introduced to not only prevent overfitting, but to make the model more robust and increase the diversity of the training data since only 20% of the full training image dataset was being used for training. The type of data augmentation used was enabled by the **transforms.ColorJitter** call which altered the brightness, contrast and saturation of every image the model saw. This would force the model to no longer rely on specific colors, but to actually learn shapes and structures. Overall, these three additions were added to possibly prevent overfitting and increase the recall scores from the baseline evaluation.

This training session with the additional modifications took 130.93 minutes or about 2.18 hours. Figure 7 shows the training and validation loss curves over the 5 epochs. As seen in the plot, the training loss curve started a bit higher than before at around 0.48, but gradually made its way down to ~0.21 which indicates that the model still successfully learned to map the training images to their captions. However, the validation loss curve started higher than before at around 0.42 and decreased slightly up until about 0.39 at Epoch 3. It then started to rise again back to 0.42 which indicated a sign of overfitting again.

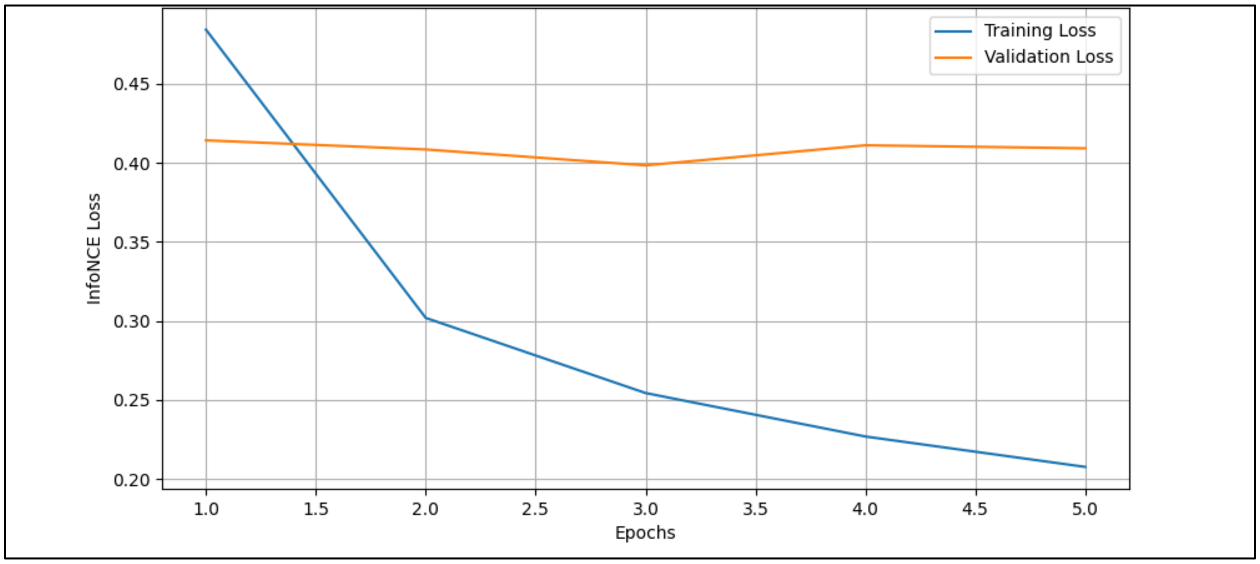


Figure 7: Loss curve plot for the modified CLIP model with weight decay and data augmentation

Following training, the baseline model was evaluated using the full validation set of 135,930 pairs. The results are summarized below in Table 3 and Table 4.

Table 3: Text-to-Image Retrieval Scores for the modified model with weight decay and data augmentation

Recall@K	**Score (%)**	**Interpretation**
K = 1	8.97	The correct image was the first result 8.97% of the time (0.55% decrease from the baseline model)
K = 5	13.89	The correct image appeared in the top 5 results 13.89% of the time (2.94% decrease from the baseline model)
K = 10	19.64	The correct image appeared in the top 10 results 19.64% of the time (3.4% decrease from the baseline model)

As shown in the table, the Recall@K scores dropped slightly compared to the baseline. This suggests that while weight decay and color jittering reduce overfitting, they also make the model slightly less effective at accurately ranking the correct image. The model still retrieved relevant results however the added augmentation likely made it harder to the model to be able to capture some of the finer visual features needed for stronger text-to-image retrieval.

Table 4: Image-to-Text Retrieval Scores for the modified model with weigh decay and data augmentation

Recall@K	**Score (%)**	**Interpretation**

K = 1	8.06	The correct caption was the first result 8.06% of the time (0.21% decrease from the baseline model)
K = 5	21.33	The correct caption appeared in the top 5 results 21.33% of the time (0.32% decrease from the baseline model)
K = 10	30.05	The correct image appeared in the top 10 results almost 30.05% of the time (0.56% decrease from the baseline model)

Table 4 shows how well the modified model retrieves the correct caption when given an image. All three Recall@K scores dropped slightly compared to the baseline, meaning the added weight decay and color jittering didn't help the model in the image-to-text direction. The model still finds the right caption around 30% of the time within its top 10 guesses but it is still a bit less confident overall. This suggests that while regularization helped reduce overfitting, it also made the model slightly worse at ranking the exact matching caption.

To understand the updated model's behavior, the group visualized the top 5 retrieved images for specific queries and a zero-shot classification graph for image-to-text classification.

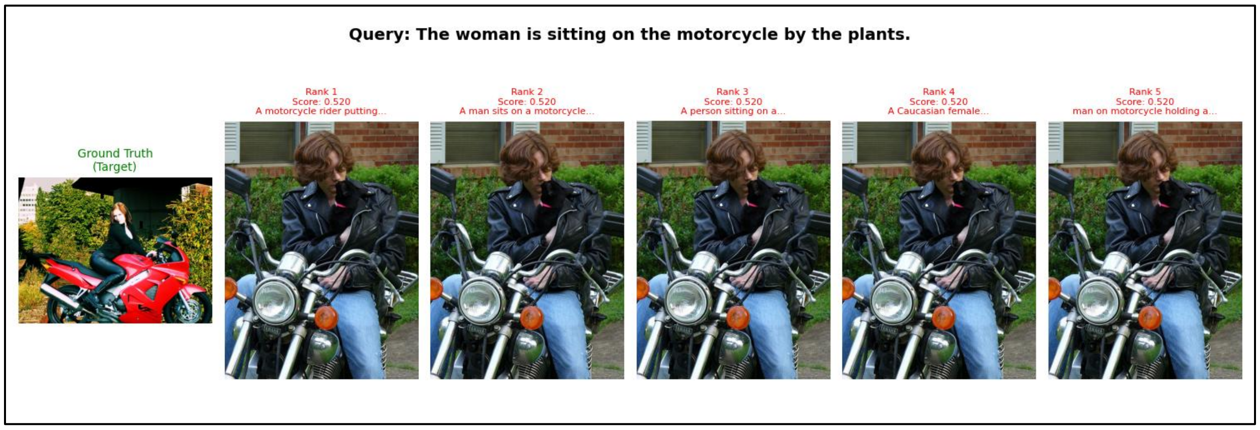

Query: The woman is sitting on the motorcycle by the plants.
Ground Truth (Target)
Rank 1 Score: 0.520 A motorcycle rider putting...
Rank 2 Score: 0.520 A man sits on a motorcycle...
Rank 3 Score: 0.520 A person sitting on a...
Rank 4 Score: 0.520 A Caucasian female...
Rank 5 Score: 0.520 man on motorcycle holding a...

Figure 8: (1/2) Top 5 retrieved images for the modified model (weight decay and data augmentation) – "The woman is sitting on the motorcycle by the plants"

For Figure 8, the query used was "*The woman is sitting on the motorcycle by the plants.*" The modified model consistently retrieved images of a woman sitting on a motorcycle which indicates an improved alignment with the core subject described in the query. However, all images depict a different woman with a different motorcycle that from the ground truth. This suggests that although the model was able to effectively capture the main objects, it struggled to utilize contextual details. The result highlights that even with added weight decay and augmentation, the model still prioritizes dominant visual key words rather than finer contextual details.

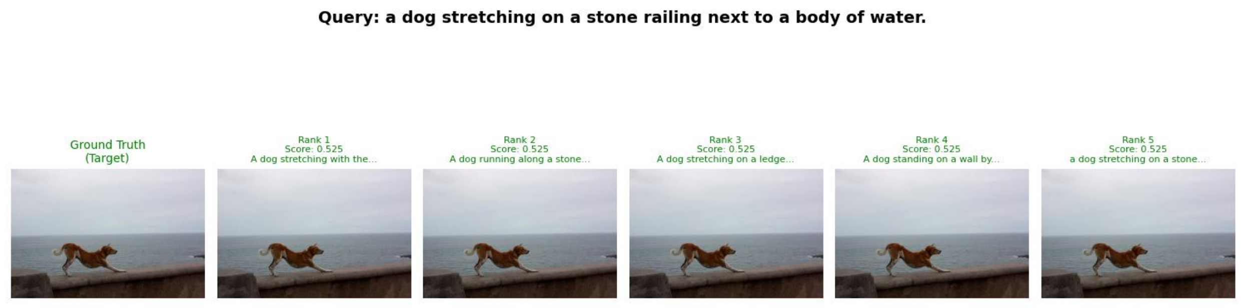

Query: a dog stretching on a stone railing next to a body of water.
Ground Truth (Target)
Rank 1 Score: 0.525 A dog stretching with the...
Rank 2 Score: 0.525 A dog running along a stone...
Rank 3 Score: 0.525 A dog stretching on a ledge...
Rank 4 Score: 0.525 A dog standing on a wall by...
Rank 5 Score: 0.525 a dog stretching on a stone...

Figure 9: (2/2) Top 5 retrieved images for the modified model (weight decay and data augmentation) – "a dog stretching on a stone railing next to a body of water."

For Figure 9, the query used was "*A dog stretching on a stone railing next to a body of water".* The modified model performed perfectly on this example, retrieving 5 images that were identical to the ground truth target, but with its various respective captions. Each retrieved image contains the correct subject (dog), correct action (stretching) and correct setting (stone railing beside water). However, the model spotted the correct image and caption on the fifth try. This suggests that the model was able to identify both the primary object and contextual cues. Compared to other tests, this case demonstrates a scenario in which the modifications to the model yielded a more desirable result.

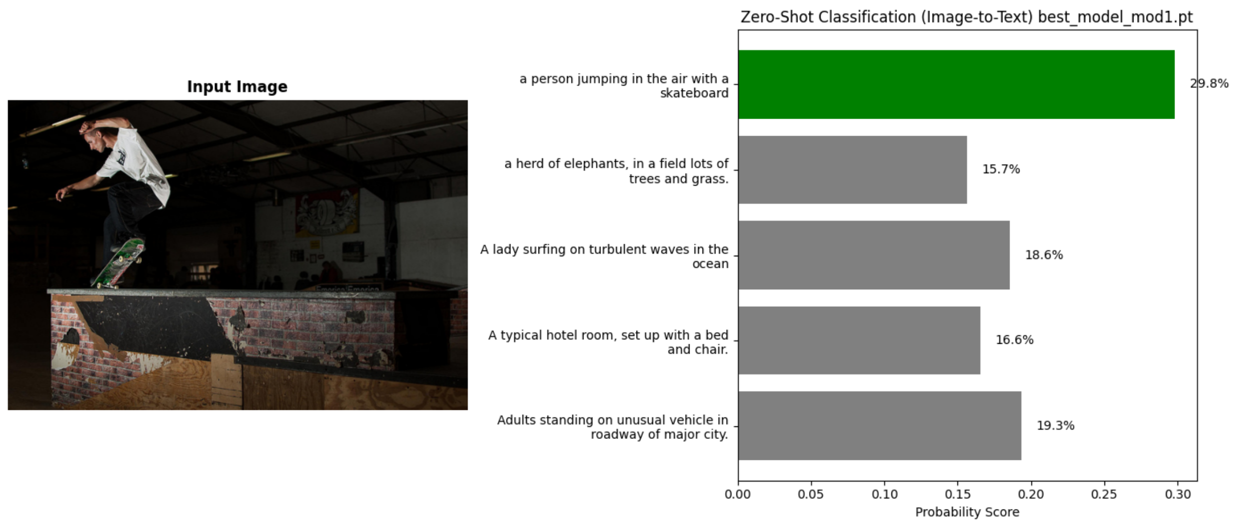


Figure 10: Image-to-Text zero-shot classification for the modified model (weight decay and data augmentation)

Figure 10 shows that the modified model with weight decay and data augmentation assigns the highest probability to the caption describing "*a person jumping in the air with a skateboard*", which matches the content of the photo. Other captions score much lower which shows that the weight decay and augmentation aid in the model's ability to identify the main action and object in the scene. However, the confidence distribution is much less even.

Unfreezing the Backbone

Here are the following hyperparameters and hardware used for the training session that modified the model. The modifications utilized reused the same regularization and data augmentation from the previous modified model with an addition of unfreezing a layer:

- Epochs = 10
- Batch Size = 64
- Learning Rate = 1e-4
- Optimization: AdamW
- Hardware: NVIDIA RTX Ada Generation
- Weight Decay = 1e-4
- Data Augmentation: Color Jitter

This training session with the additional modifications took 157.66 minutes or about 2.62 hours. Figure 11 shows the training and validation loss curves over 10 epochs. The training loss decreased steadily from approximately 0.80 to 0.15, demonstrating that only unfreezing the final ResNet50 block allowed the model to continue learning meaningful visual representations. While the plot still showed signs of overfitting like the previous trials, the validation curve from this session did not start to rise again until the fourth epoch, which indicates it's increase in learning the dataset. Overall, the plot indicates that the unfreezing of the backbone helped the model refine high-level features, but some overfitting still remains within the model.

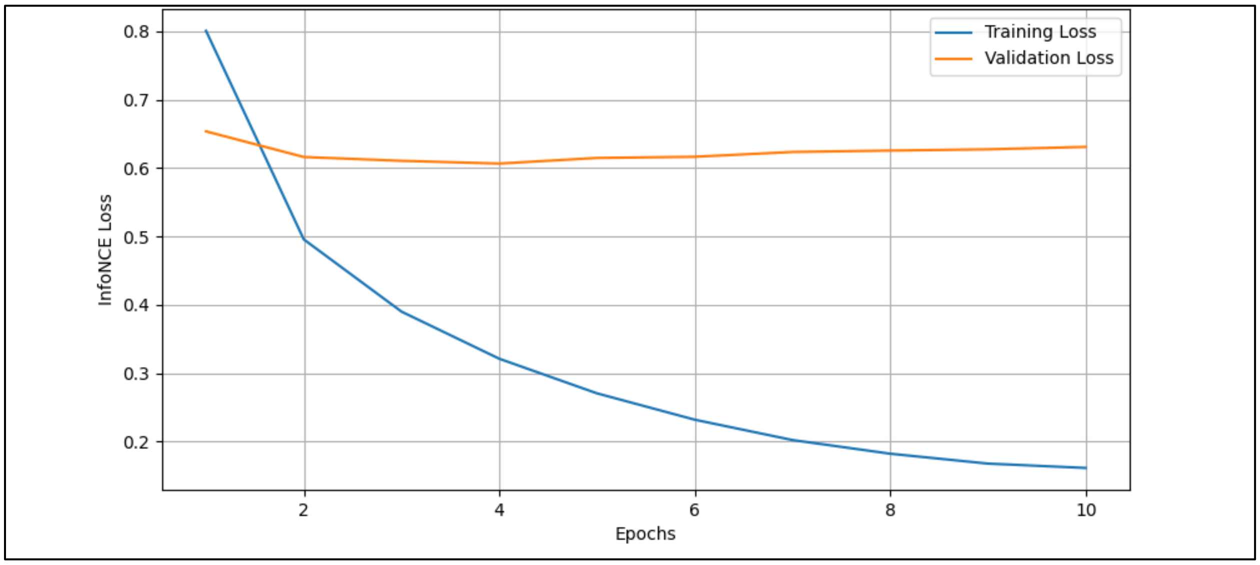


Figure 11: Loss curve plot for the modified CLIP model with weight decay, data augmentation, and unfrozen layer

Following training, the baseline model was evaluated using the full validation set of 135,930 pairs. The results are summarized below in Table 5 and Table 6.

Table 5: Text-to-Image Retrieval Scores for modified model with weight decay, data augmentation, and unfrozen layer

Recall@K	Score (%)	Interpretation
K = 1	10.63	The correct image was the first result 10.63% of the time (1.11% increase from the baseline model and 1.66% increase from the first modified version of the CLIP model)
K = 5	17.36	The correct image appeared in the top 5 results about 17.36% of the time (0.53% increase from the baseline model and 3.47% decrease from the first modified version of the CLIP model)
K = 10	23.76	The correct image appeared in the top 10 results 23.76% of the time (0.72% increase from the baseline model and 4.12% decrease from the first modified version of the CLIP model)

Table 5 show the highest results that were obtained during the whole training and modification session. Unfreezing layers allowed the model to fine-tune its feature extraction more effectively for exact matches. The group was able to successfully fine-tune the model by increasing its original Recall@1 score from 9.50% to 10.63%. It shows that the unfreezing layers can significantly sharpen the model's confidence in its top choice while also remaining relatively stable compared to the augmentation-based improvements.

Table 6: Image-To-Text Retrieval Scores for modified model with weigh decay, data augmentation, and unfrozen layer

Recall@K	Score (%)	Interpretation
K = 1	9.90	The correct caption was the first result 9.90% of the time (0.38% increase from the baseline model and 1.84% increase from the first modified version of the CLIP model)
K = 5	24.93	The correct caption appeared in the top 5 results 24.93% of the time (3.28% increase from the baseline model and 3.6% decrease from the first modified version of the CLIP model)
K = 10	34.80	The correct caption appeared in the top 10 results about 34.80%

		of the time (4.16% increase from the baseline model and 4.75% decrease from the first modified version of the CLIP model)

Table 6 shows how well the final model retrieves the correct caption when given an image. All three Recall@K scores are higher than the baseline, meaning unfreezing the last ResNet50 block helped the model understand images better. Recall@1 rises to 9.90% showing the model is more confident in picking the correct caption as its top choice. The Recall@5 and Recall@10 also increase, which means the correct caption appears more often within the top few predictions. These results show that fine turning the backbone helped the model match images to captions more accurately.

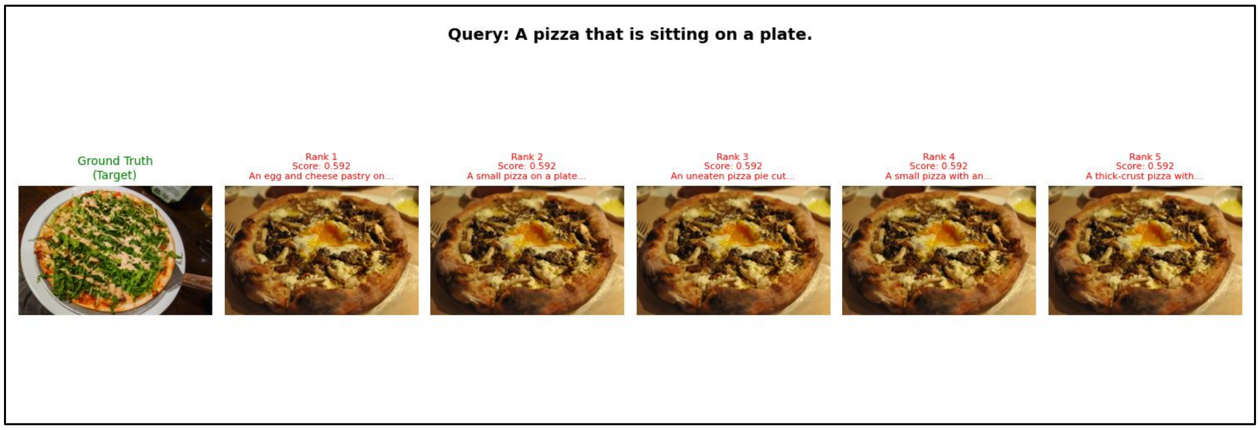

Query: A pizza that is sitting on a plate.
Ground Truth (Target)
Rank 1 Score: 0.592 An egg and cheese pastry on...
Rank 2 Score: 0.592 A small pizza on a plate...
Rank 3 Score: 0.592 An uneaten pizza pie cut...
Rank 4 Score: 0.592 A small pizza with an...
Rank 5 Score: 0.592 A thick-crust pizza with...

Figure 12: (1/2) Top 5 retrieved images for the modified model (unfrozen layer) – "A pizza that is sitting on a plate"

In Figure 12, the query was "*a pizza that is sitting on the plate*". The unfrozen-layer model retrieved five images that all contain pizza on plates, showing strong alignment with the core object described in the prompt. While the retrieved images differ from the ground truth image in many areas, the model consistently matched the main concept with high accuracy. This indicates the freezing layer improved the model's ability to capture key categorical features. However, the model is still not sensitive to finer contextual or stylistic variations in food images.

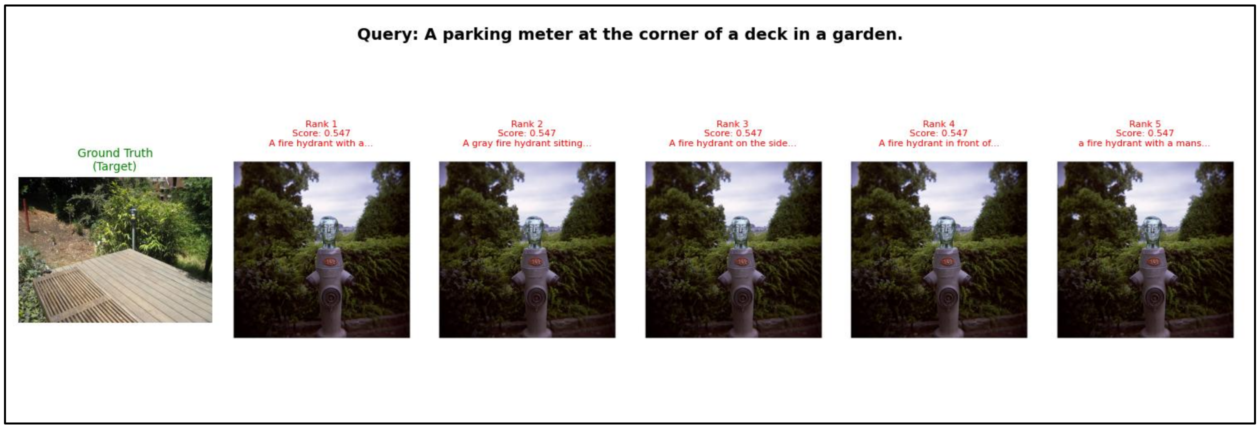

Query: A parking meter at the corner of a deck in a garden.
Ground Truth (Target)
Rank 1 Score: 0.547 A fire hydrant with a...
Rank 2 Score: 0.547 A gray fire hydrant sitting...
Rank 3 Score: 0.547 A fire hydrant on the side...
Rank 4 Score: 0.547 A fire hydrant in front of...
Rank 5 Score: 0.547 a fire hydrant with a mans...

Figure 13: (2/2) Top 5 retrieved images for the modified model (unfrozen layer) – "A parking meter at the corner of a deck in a garden"

In Figure 13, the query was *"A parking meter at the corner of a deck in a garden."* The model successfully identified the primary object, the parking meter, but struggled with the more unusual prompt provided. The retrieved images capture the meter itself but failed to include elements such as the deck or gardening setting. These results higlighted a limitation of the model, although freezing the backbone helped with object-level recognition, it does not fully resolve issues in retrieving images with more uncommon multi-aspect scene descriptions. The model seemed to remain more biased towards single object descriptions.

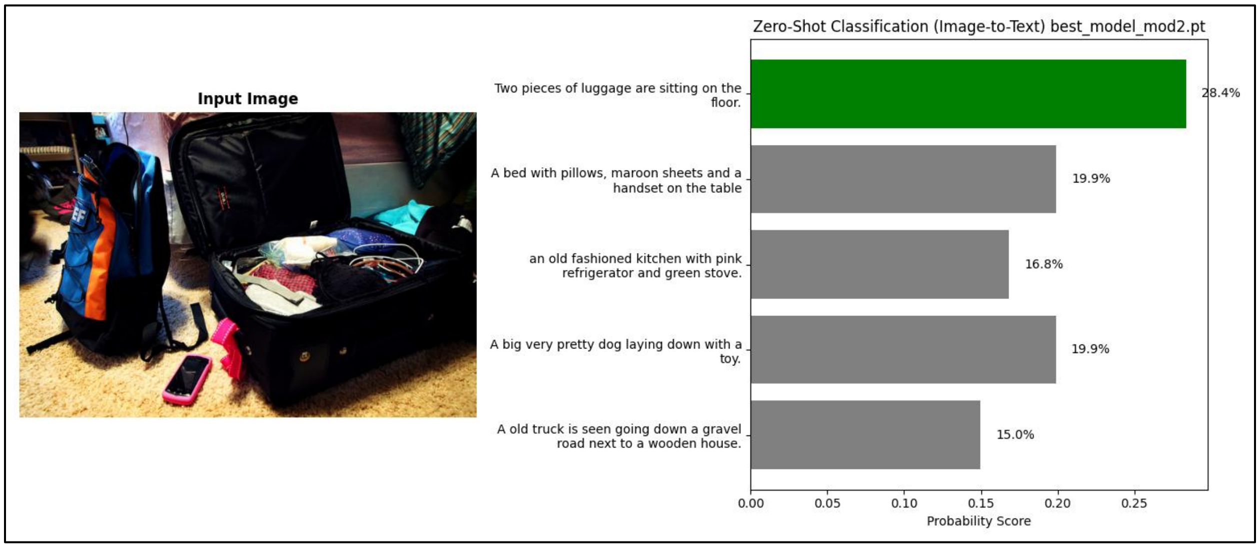


Figure 14: Image-to-Text zero-shot classification for the modified model (weight decay, data augmentation, unfrozen layer)

The model correctly gives the highest probability to the caption "*two pieces of luggage are sitting on the floor"* which matches the main objects in the image. The other captions score much lower in comparison showing that the unfrozen model can reliably identify the key items in the scene. However, it is evident that it still focuses mainly on the most obvious objects and doesn't recognize or acknowledge smaller details.

Discussion

Across all three training stages, the results showed steady improvements in retrieval performance, along with clear patterns in how the model understands images and captions. The baseline model could reliably identify the main object or subject in most prompts, but it often ignored important details like actions, background elements, or secondary objects. This explains why many of the retrieved images were “close enough” but not exact ground-truth matches. The loss curves also showed early overfitting, with validation loss flattening and then increasing after the third epoch.

When weight decay and data augmentation were added, the model was slightly worse than before. Even so, the model continued to choose relevant images and remained consistent in identifying dominant objects. The zero-shot classification examples showed the same pattern, the modified model successfully picked the correct caption as the top prediction, but the probability distribution became more spread out, suggesting an overall lower confidence.

The biggest improvement came from unfreezing the final ResNet50 block, which allowed the image encoder to learn higher level features. This boosted the Recall@1 scores for both retrieval directions, with the largest improvement seen in text-to-image retrieval. The model became better at recognizing the main items in an image and aligning them with caption descriptions. The zero-shot classification graphs reinforced this, when the model was shown an image, it consistently ranked the correct caption highest with a clearer margin than in earlier versions. However, this model still struggled with queries that required understanding several details at once, such as combining “parking meter,” “deck,” and “garden,” showing that the model improved at recognizing objects but not full scenes.

Overall, all three versions of the model behaved the same way, they worked well when the image had one clear object but struggled when the scene had many details. Weight decay and data augmentation helped a small amount when combined with the ResNet50 backbone, making the largest difference. The zero-shot classification results matched this pattern, they showed that the model can usually spot the main idea of an image, but its accuracy and confidence improve the most when the higher-level visual features are fine-tuned.

Conclusion

In conclusion, the group successfully implemented and fine-tuned a contrastive learning model based on the CLIP neural network that yielded higher text-to-image and image-to-text recall scores from its original baseline scores. Most importantly, the Recall@1 score for text-to-image was increased from 9.52% to 10.63%, and the Recall@1 score for image-to-text was increased from 8.27% to 9.90% through various ways of modifications.

During training for all the models, the models suffered from signs of overfitting based off the loss plots produced. The group noticed each model would show signs of learning as the training loss curve went down alongside the validation loss curve, but typically by the third epoch, the validation loss curve would start to rise again. The group thought implementing weight decay and data augmentation would cause the recall scores to increase, but it had the opposite effect. The modified model found it a bit more difficult to perform text-to-image and image-to-text classifications. It appears using regularization and the specific kind of data augmentation (color jittering) in this part of the experiment hindered the model's ability to learn. However, the group understands from class that regularization and data augmentation should help the model perform better than its baseline results. A different type or types of data augmentation could be implemented for further improvement to help increase the overall recall scores. The final modified model included the previous modifications, froze the first 3 layers and unfroze ResNet50's fourth and final group of convolutional blocks. Unfreezing this stage of the ResNet50 architecture that is responsible for extracting high-level semantic concepts resulted in the modified model yielding its highest results in the Recall@1 score for text-to-image of 10.63% and the Recall@1 score for image-to-text of 9.90%, demonstrating stronger retrieval capabilities.

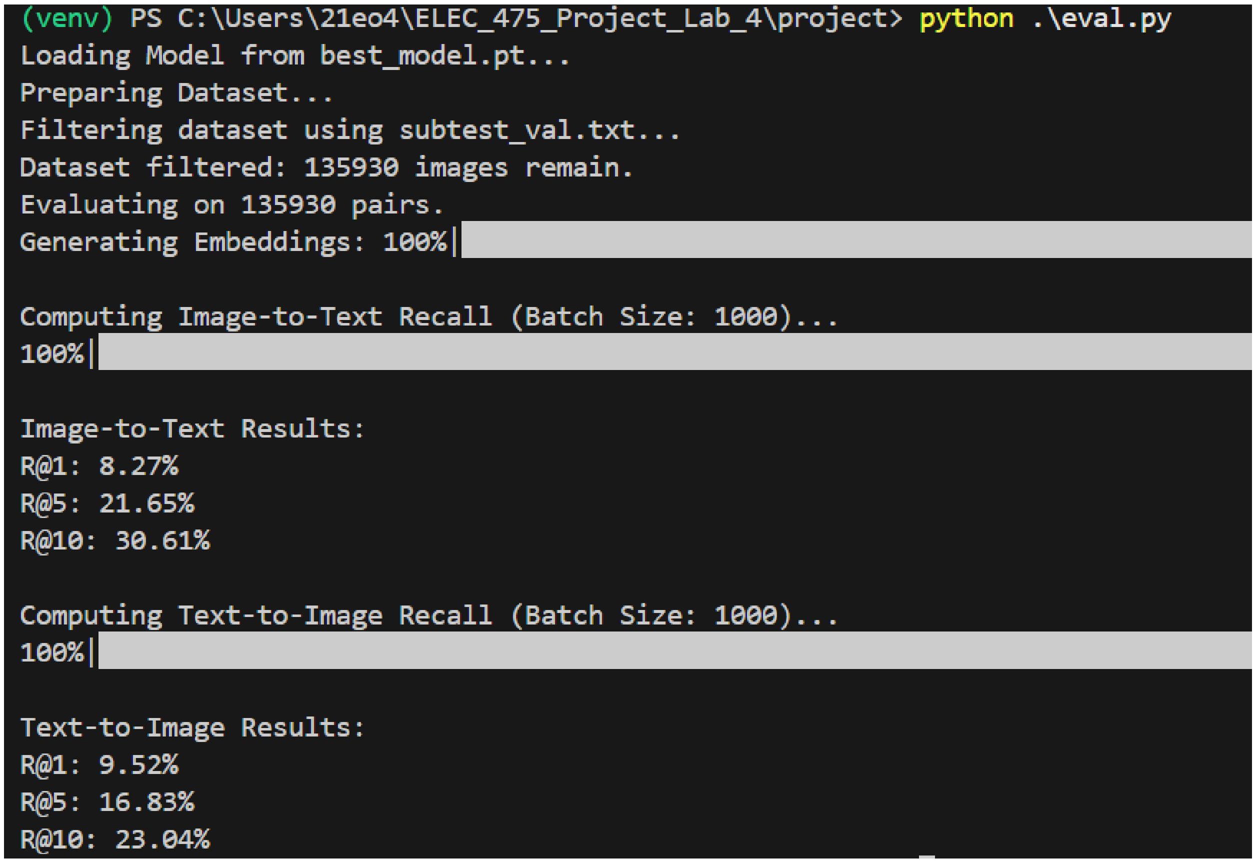
While the modified model achieved strong performance, several improvements could have further enhanced the results. For one, the batch sizes used during training could have been increased on better GPU hardware and more time. Contrastive Learning benefits from larger batch sizes as they enhance performance by providing more negative samples to better distinguish between similar and dissimilar data [2]. The max batch size used was 64, but it was limited by the hardware constraints of the NVIDIA RTX Ada Generation GPU. Increasing the batch size, using a couple more epochs for the modifications, and having access to better hardware could have likely increased our overall recall scores. Secondly, while color jittering appeared to be effective, incorporating other types of data augmentation like rotating, flipping, and cropping could have forced the model to recognize objects from different scales and perspectives. These were a few ideas the team thought of that could have helped improve the model's performance even more.

Large Language Model Conversation

The Large Language Model (LLM) used throughout this project was Google's Gemini and was treated as a collaborative tool. It assisted with conceptual understanding, code debugging, and result interpretation. The team made sure to intervene when the code and other outputs generated didn't make sense or align with the goals of the project. For example, the team asked Gemini to help with the definition of the InfoNCE loss and to also explain line by line what the particular function was doing to get a better understanding of the loss function. The LLM also helped with explaining why the models were overfitting. It could analyze the loss curve figures and suggest ways of improving its performance. The group did not just take the LLM's outputs and suggestions at face value. Computer Vision concepts, definitions and ideas were researched to verify the LLM's claims. Instead of letting the LLM do all the work, the team would offer their own suggestions and challenge some of the results gathered from the code. Performance suggestions by the LLM were treated as hypotheses. The team validated the suggestions by running the experiments and observing the actual impact on the loss curves and Recall@K metrics. In general, no output was ever trusted by default, every code snippet was executed, debugged, and edited if necessary.

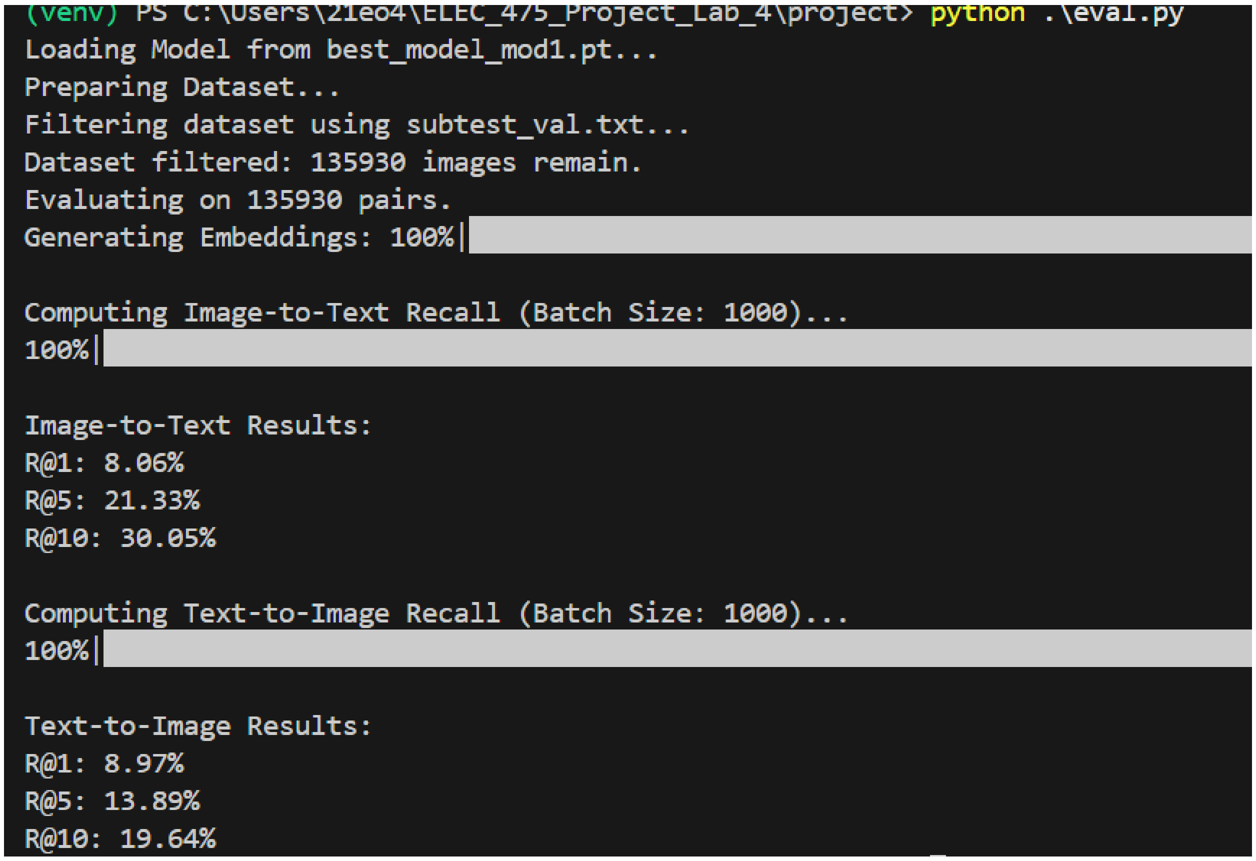
Link to conversation with Gemini: https://gemini.google.com/share/8a81616ae78c

Appendix



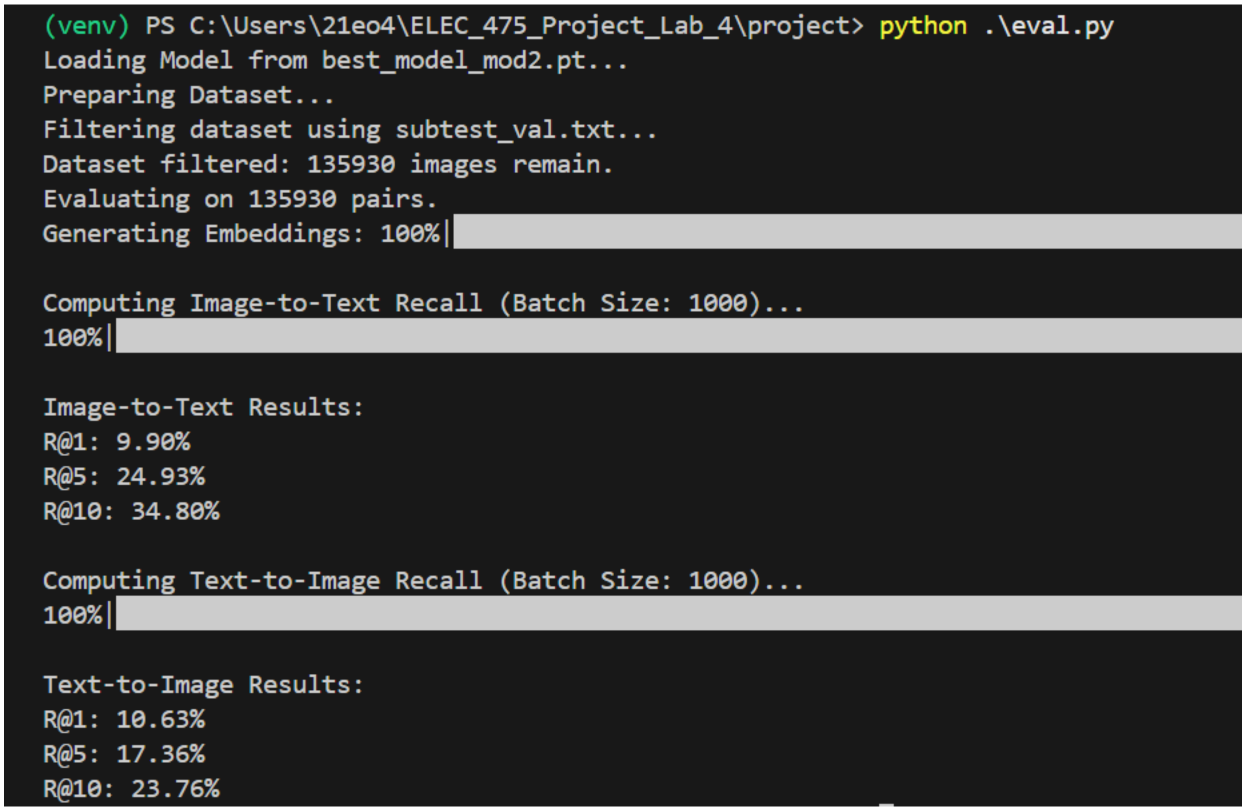
```
(venv) PS C:\Users\21eo4\ELEC_475_Project_Lab_4\project> python .\eval.py
Loading Model from best_model.pt...
Preparing Dataset...
Filtering dataset using subtest_val.txt...
Dataset filtered: 135930 images remain.
Evaluating on 135930 pairs.
Generating Embeddings: 100%|

Computing Image-to-Text Recall (Batch Size: 1000)...
100%|

Image-to-Text Results:
R@1: 8.27%
R@5: 21.65%
R@10: 30.61%

Computing Text-to-Image Recall (Batch Size: 1000)...
100%|

Text-to-Image Results:
R@1: 9.52%
R@5: 16.83%
R@10: 23.04%
```

Figure 15: Text-to-Image and Image-to-Text Retrieval Scores for baseline model from Visual Studio Code output terminal



```
(venv) PS C:\Users\21eo4\ELEC_475_Project_Lab_4\project> python .\eval.py
Loading Model from best_model_mod1.pt...
Preparing Dataset...
Filtering dataset using subtest_val.txt...
Dataset filtered: 135930 images remain.
Evaluating on 135930 pairs.
Generating Embeddings: 100%|

Computing Image-to-Text Recall (Batch Size: 1000)...
100%|

Image-to-Text Results:
R@1: 8.06%
R@5: 21.33%
R@10: 30.05%

Computing Text-to-Image Recall (Batch Size: 1000)...
100%|

Text-to-Image Results:
R@1: 8.97%
R@5: 13.89%
R@10: 19.64%
```

Figure 16: Text-to-Image and Image-to-Text Retrieval Scores for the modified model with weight decay and data augmentation from Visual Studio Code output terminal



```
(venv) PS C:\Users\21eo4\ELEC_475_Project_Lab_4\project> python .\eval.py
Loading Model from best_model_mod2.pt...
Preparing Dataset...
Filtering dataset using subtest_val.txt...
Dataset filtered: 135930 images remain.
Evaluating on 135930 pairs.
Generating Embeddings: 100%|

Computing Image-to-Text Recall (Batch Size: 1000)...
100%|

Image-to-Text Results:
R@1: 9.90%
R@5: 24.93%
R@10: 34.80%

Computing Text-to-Image Recall (Batch Size: 1000)...
100%|

Text-to-Image Results:
R@1: 10.63%
R@5: 17.36%
R@10: 23.76%
```

Figure 17: Text-to-Image and Image-to-Text Retrieval Scores for the modified model with weight decay, data augmentation, and unfrozen layer from Visual Studio Code output terminal

References

[1] GeeksForGeeks, "CLIP (Contrastive Language-Image Pretraining)," 26 November 2025. [Online]. Available: https://www.geeksforgeeks.org/deep-learning/clip-contrastive-language-image-pretraining/.

[2] Z. Cheng, H. Zhang, K. Li, S. Leng, Z. Hu, F. Wu, D. Zhao, X. Li and L. Bing, "Breaking the Memory Barrier: Near Infinite Batch Size Scaling for Contrastive Loss," 22 October 2024. [Online]. Available: https://arxiv.org/html/2410.17243v1#:~:text=Specifically%2C%20larger%20batches%20provide%20a,et%20al.%2C%202021)%20.&text=Despite%20the%20above%20benefits%2C%20scaling,et%20al.%2C%202020)%20.. [Accessed 28 November 2025].