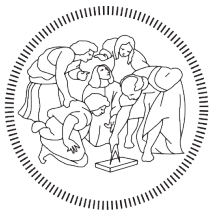


POLITECNICO
MILANO 1863

School of Industrial and Information Engineering

058623 - ADVANCED TOPICS IN DEEP LEARNING:
THE RISE OF TRANSFORMERS

CLIP Project

Erik Galler
Politecnico di Milano
erik.galler@mail.polimi.it

Kristian Walseth Krøgenes
Politecnico di Milano
kristianwalseth.kroegenes@mail.polimi.it

July, 2023

Abstract

This project aims to implement the CLIP model for image and text retrieval tasks. The simplified ROCO dataset, consisting of medical images and corresponding captions, will be used for training. The key steps involve preprocessing the data, designing and implementing the image and text encoders, defining a contrastive loss function, and training the model using gradient descent. Once trained, our CLIP model will enable image retrieval given a text query and text retrieval given an image query.

Contents

1 Introduction

The ability to understand and extract meaningful information from images and text is a fundamental challenge in the field of artificial intelligence. Traditional approaches often treat images and text as separate modalities, limiting their ability to capture the rich semantic relationships that exist between the two. The Contrastive Language-Image Pretraining (CLIP) model, introduced by Radford et al. (2021), addresses this limitation by jointly training image and text encoders in a shared embedding space. By learning to align images and their corresponding captions, CLIP enables tasks such as image retrieval given a text query or text retrieval given an image query.

In this paper, we present the implementation of the CLIP model using PyTorch and apply it to the domain of medical image and text retrieval. We leverage a simplified version of the ROCO dataset, which consists of thousands of medical images accompanied by relevant captions. Our primary goal is to develop and experiment with retrieving relevant images given textual queries or retrieving relevant captions given an image query.

In the following sections, we will describe the methodology and architecture of our CLIP implementation, detail the preprocessing steps and training process, present experimental results, and discuss the implications of our findings.

2 Related work

Our implementation of the CLIP model is based on the official OpenAI research publication on CLIP [4], which provides a comprehensive overview of the model's architecture, training methodology, and applications. This publication serves as a key reference, offering valuable insights and guiding our implementation process. Additionally, we draw inspiration from the tutorial "Simple Implementation of OpenAI CLIP Model" [6]. This tutorial provides practical guidance for implementing the CLIP model, making it a valuable resource for understanding the model's architecture and its application to image-text tasks. Furthermore, the work by Radford et al. [5] presents an important contribution to the field with their approach to learning transferable visual models from natural language supervision, which aligns with the core principles of CLIP.

3 Baseline Model

3.1 Architecture

We decided to make a simple Baseline model for the purpose of being able to incrementally develop and improve the CLIP model in a clear manner. Although this has some drawbacks like convergence not necessarily being at a global maxima, we prefer this strategy as it is efficient and we don't have to many resources.

Our baseline model uses inspiration from [6] and follows a simple architecture where the initial captions are tokenized and embedded with a pretrained *distilbert-base-uncased* model. The image on the other hand is encoded with a *resnet50* model. Before calculating the loss, we also passed the image and text encodings through a simple 2-layered neural network to end up with matching dimensions. We also started of with a constant temperature.

Here are the different parameters that where used:

```
batch_size = 64
validation_ratio = 0.2
head_lr = 1e-3
image_encoder_lr = 1e-4
text_encoder_lr = 1e-5
weight_decay = 1e-3
patience = 1
factor = 0.8
epochs = 20
temperature = 1.0
```

```
model_name = 'resnet50'
image_embedding = 2048
text_encoder_model = "distilbert-base-uncased"
text_embedding = 768
text_tokenizer = "distilbert-base-uncased"
max_length = 200
num_projection_layers = 1
size = 224 # image size
projection_dim = 256
dropout = 0.1
similarity_threshold = 0.8
```

3.2 Metrics

The aim of this project is to perform image retrieval, i.e. to find the most related image in the collection to the text query (or vice versa, find the caption in the dataset which is closest to a query image). For that reason we will mainly use the loss and the accuracy to evaluate our model. Both metrics are evaluated for each batch, and the updated metrics are plotted. Meaning the accuracy for a random model would be $\frac{1}{64} = 1.56\%$

Since there is not necessarily only one correct caption that should be matched with a picture (or vice versa), the cosine similarity score is used with a threshold of 0.8 to decide if a prediction is correct. We used the *bert-base-nli-mean-tokens* SentenceTransformer to create the semantic embedding of the captions.

3.3 Resource based compromises

Since none of us have a GPU available, and we don't have access to any clusters, we decided to use the standard free subscription of Google Collaboration. This resulted us in having to downscale some of the parameters to be able to train the model in an efficient manner. We decided to develop and test our model on 10000 images, rather than the full 80000+ images provided, and with a batch size of 64 to not run out of memory. In the original paper a much bigger batch size was used [5], but this will not be possible for us.

3.4 Results

Below you can see the results of our baseline model. Looking at the loss, our model starts to overfit after approximately 10 epochs, and reaches its highest validation accuracy of 22.7%. Continuing the development of the model with 15 epochs therefore looks sufficient, and will approximately take 2.5 hours per run.

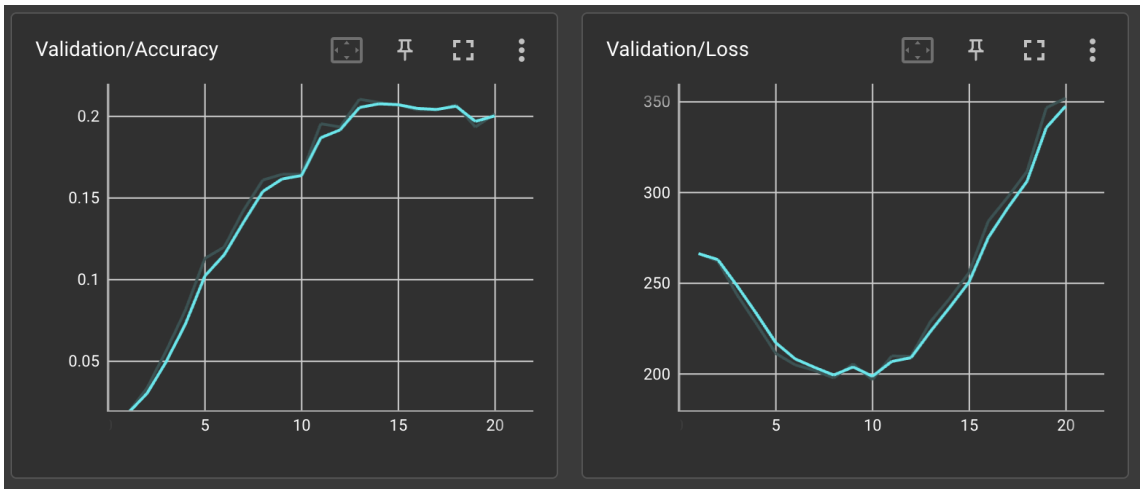


Figure 1: Baseline model

Figure 2: Baseline epoch 13

4 Pre-Processing

4.1 Cleaning Data

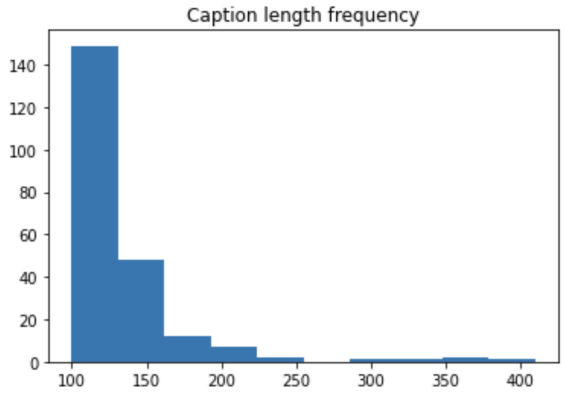


Figure 3: Histograms of captions with $\geq$ 100 words

After performing a exploratory data analysis we found out that there where captions with over 400 words. These captions often described more about the surrounding situation, e.g. "a women whom was in a car accident with a previous history of alcoholism", rather than the contents of the corresponding image.

As one can observe in 4, when removing long captions ($\geq$ 100 words), the accuracy is slightly increased by a few percent (purple graph).

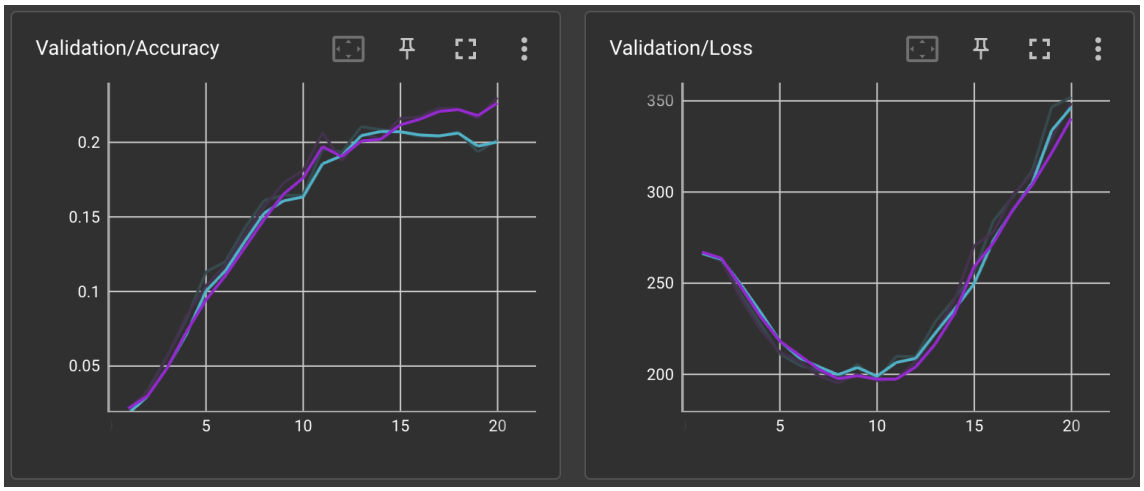


Figure 4: Removed long captions

4.2 Concept Encodings

Each image is associated with a set of concept labels from a set of over 8 thousand classes, we decided to try to encode these labels to their corresponding image caption. We tried different string augmentation, but ended up with the following format:

The concepts of this image is angiogram, Embolization, Therapeutic, tooth filling and Pseudoaneurysm. The caption of the medical image is: "film after glue embolization show no filling in the pseudoaneurysm and the glue angiocatheter".

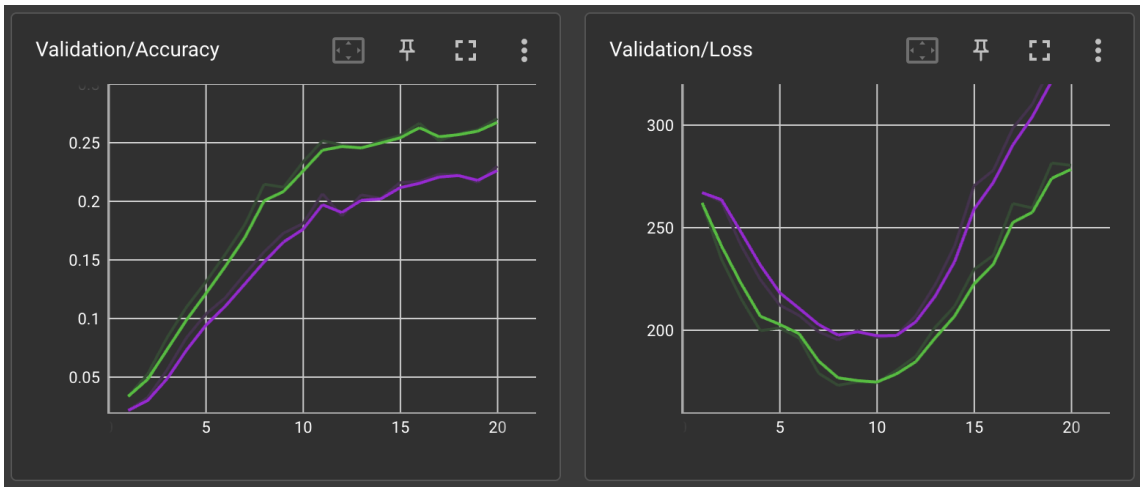


Figure 5: Concept Encodings

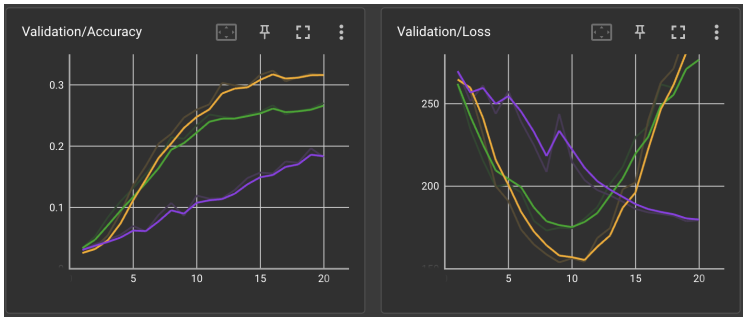
As one can see in Figure 5 this resulted in an increase of approximately 5%. It is important to note that the cosine similarity was naturally computed with the unaugmented captions.

5 Image Encoder

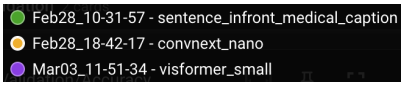
5.1 Model

Our baseline model uses the *resnet50* model as the image encoder, like in the original paper (they also tried different ViT models) [5]. We decided to try different state of the art models (including

ViT models) to see if we could improve our results. Unfortunately we had some issues with Google Collaboration running out of memory when testing some of the bigger and more advanced models limiting our testing. After a lot of trial and error we found the *ConvNeXT nano* model from timm to be the most successful [3].



(a) Image Encoders



(b) Image Encoder Legends

Judging from the results the *ConvNeXT nano* model performed ca. 5% better than the *resnet 50* model. But the model also takes 20% longer to finish 20 epochs. We will continue using this model in feature iterations since this is a pretty significant increase in accuracy.

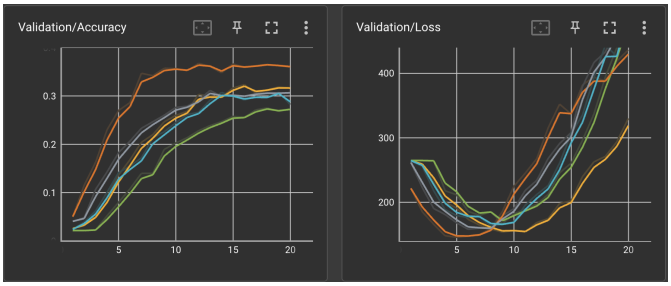
The ViT model on the other hand performed poorly, it both had a substantially longer training time, and had 8% lower accuracy. Looking at the validation loss one can argue that its accuracy has not converged yet and judging from the slope, it could overtake the two models. But since this is computationally too expensive, we decided not to pursue this inquiry.

5.2 Pre-training on Concepts

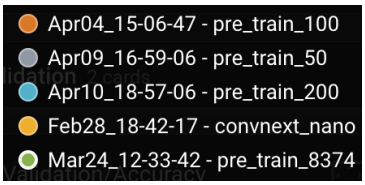
Until this point we have used pre-trained models from timm, and fine-tuned the models on our new set of medical images. But we wanted to specialise our pre-trained model to our medical situation by pre-training the models ourselves. Since pre-training is very time consuming (and did not give us any promising results), we decided to do further pre-training on already pre-trained models.

Since every image has correspondingly x number of concepts associated to it, pre-training the image encoder as a multi-label classifier makes sense. We did this by using a binary cross-entropy loss function, and a sigmoid activation function. During training we consecutively calculated the F1 score of the validation set to evaluate if the pre-training was done. Afterwards we simply remove/reset the classification layers, and our image encoder is good to go.

But there are 8374 classes, and many of the classes are barely used in the pre-training yielding bad results. We therefore tried to reduce the number of classes by choosing those with the highest frequency.



(a) Pre-training



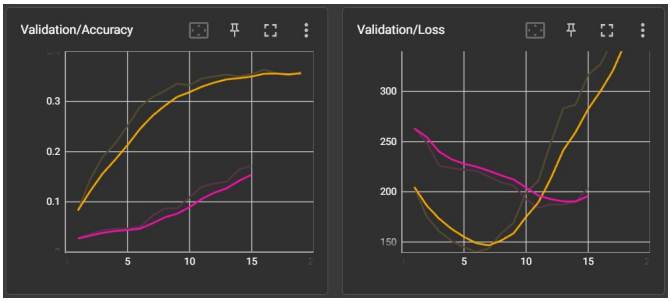
(b) Pre-training legends

Looking at the result, most of the pre-trained models performed likewise or worse, except the model pre-trained on 100 concept. This model both converged quicker, and also had an increase of about 5% in accuracy. We will continue using this model.

6 Text Encoder

6.1 Model

Our baseline model uses the *DistilBERT* model as the text encoder. We tried different state of the art models to see if we could improve our results. Unfortunately, like the Image Encoder models, we had some issues with Google Collaboration running out of memory when testing some of the bigger and more advanced models limiting our testing. After different approaches, we ended up comparing the *DistilBERT* model and the *T5-small* model from the Hugging Face library. It turned out that the *DistilBERT* model gave us the best results for the Text Encoder.



(a) Text Encoders

(b) Text Encoder Legends

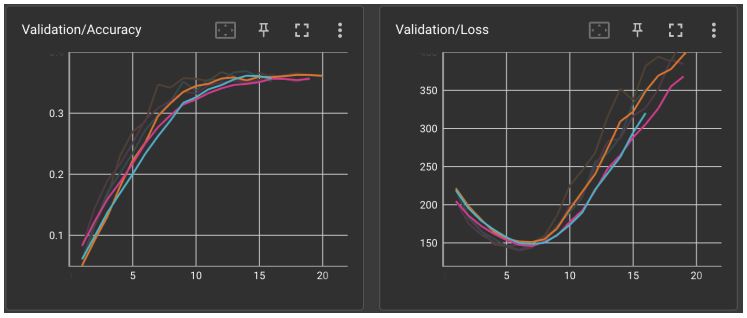
Judging from the results the *DistilBERT* model performed ca. 20% better than the *T5-small* model. The *T5-small* model was never close to compete with the *DistilBERT* model with neither the accuacy or the loss.

The two models had approximately the same running time. Even though the *T5-small* model stopped at epoch 16, because of our Google Colab's memory shortage, it was clear that it would not do any better.

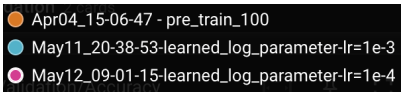
7 Loss Function

7.1 Learnable temperature parameter

Until now we have simple set the learnable temperature parameter $\tau = 1$. But in the paper this is set to be a learnable parameter which is directly optimized during training as a log-parameterized multiplicative scalar to avoid turning as a hyper-parameter[5]. We set the initial value to be 0.07 [7].



(a) Learnable Temperature Graphs



(b) Learnable Temperature Legends

As you can see above, we tried with different learning rates, but it did not make a significant difference. Since the literature suggested that this would be a better approach, we did not see a reason to not continue with it. We chose the learning rate with $1e-4$.

7.2 CyClip

We also wanted to optimise the loss function, and found a resent paper addressing this[2].

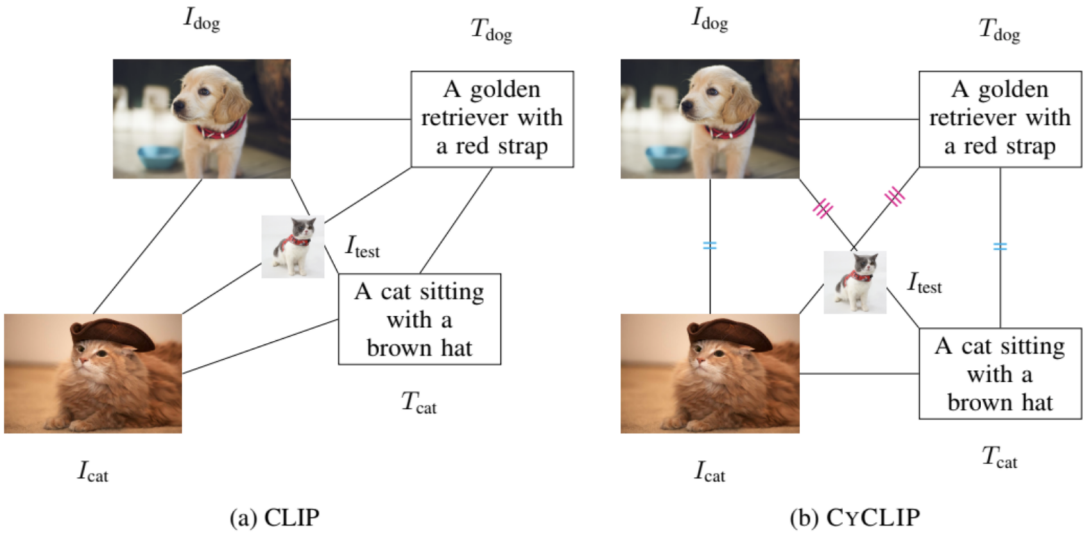


Figure 10: CyClip

As described in the paper, CyCLIP imposes an additional geometric structure on the learned representations. Specifically, given two image-text pairs, we augment the contrastive learning objective with two symmetrization terms. The first term provides for in-modal consistency by encouraging the distance between the two image embeddings to be close to the distance between the corresponding text embeddings. The second term for the cross-modal consistency that encourages the distance between the image and text embedding from the first and second pairs respectively to be close to the distance between the text and image embeddings from the first and second pairs respectively.

(1) The **cross-modal consistency** regularizer reduces the gap in the similarity scores between the embeddings of all the mismatched image-text pairs in a batch, two at a time:

$$\mathcal{L}_{C\text{-}Cyclic} = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{N}(\langle I_j^e, T_k^e\rangle - \langle I_k^e, T_j^e\rangle)^2 \tag{1}$$

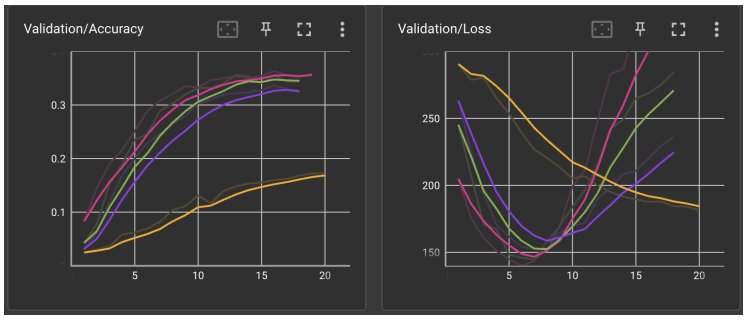
(2) The **in-modal consistency** regularizer reduces the gap in the similarity scores between the embeddings of all combinations of image pairs and their corresponding text pairs in a batch:

$$\mathcal{L}_{I\text{-}Cyclic} = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{N}(\langle I_j^e, I_k^e\rangle - \langle T_k^e, T_j^e\rangle)^2 \tag{2}$$

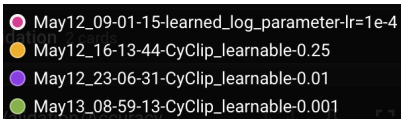
Hence, our overall loss for CyCLIP is given as:

$$\mathcal{L}_{CyCLIP} = \mathcal{L}_{CLIP} + \lambda_1\mathcal{L}_{C\text{-}Cyclic} + \lambda_2\mathcal{L}_{I\text{-}Cyclic} \tag{3}$$

We implemented the loss function as in the corresponding repository [1], but unfortunately this did not show any significant improvements. We tried with different λ values, but the results only improved as λ converged to 0. One thing to note is that the CyCLIP loss did not explode like the CLIP loss. Meaning that it might give us a better result if we gave it more time.



(a) CyClip Plots

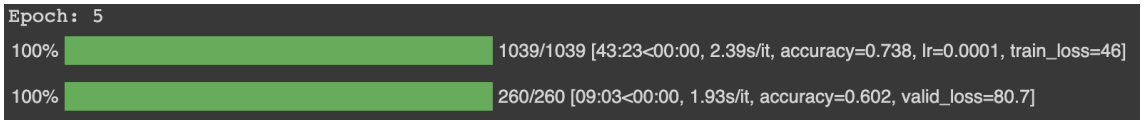


(b) CyClip Legends

Another explanation is that the semantic difference between the two images, and the two captions (in-modal) are unproportional in the given dataset. All the images and captions are very specific medical images/captions, whilst the data used in [2] has a more semantically nuanced theme rendering this loss function more appropriate. This also applies for the cross-modal consistency.

8 Final Results

Finally, we trained our model on the full dataset, and got the following results after 5 epochs. The accuracy was still increasing, but because of our limited resources we did not manage to train the model any further beyond this point.



(a) Full dataset results

In conclusion, our study demonstrated that the application of the CLIP model to medical image and text retrieval yields promising results. However, the results also indicate that the design and implementation of the model require careful consideration of various factors such as dataset characteristics, model architecture, and resource constraints. The enhancements made throughout our study led to noticeable improvements in the model's performance, resulting in a lightweight model achieving 36% accuracy trained on 10.0000 samples, and 60% accuracy on 80.000+ samples. This suggests that further research in this direction can yield even more refined models capable of efficiently and accurately retrieving medical images and their corresponding text.

References

[1] Shashank Goel and Hritik Bansal. Cyclip — official pytorch implementation, February 2023. URL: https://github.com/goel-shashank/CyCLIP.

[2] Shashank Goel, Hritik Bansal, Sumit Bhatia, Ryan Rossi, Vishwa Vinay, and Aditya Grover. Cyclip: Cyclic contrastive language-image pretraining. *Advances in Neural Information Processing Systems*, 35:6704–6719, 2022.

[3] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[4] OpenAI. Clip: Connecting text and images.

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. URL: https://proceedings.mlr.press/v139/radford21a.html.

[6] M. Moein Shariatnia. Simple CLIP, April 2021. doi:10.5281/zenodo.6845731.

[7] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.