# Master Internship Vision and Language: Image Classification with Text Labeling

**Tobias Brandner**
Julius Maximilian University Wuerzburg
tobias.brandner@stud-mail.uni-wuerzburg.de

## Abstract

CLIP, an image-language model, excels in various tasks but struggles with zero/few-shot classification due to prompt sensitivity. We use CoOp, an automated prompt engineering method, integrated with OpenCLIP, the open-source CLIP variant. OpenCLIP backbones perform slightly worse than CLIP counterparts initially, but with a shot value of 16, they show promise when combined with CoOp. We also add support for new models, like Roberta-ViT-B32 and XLM-Roberta-Base-ViT-B32. CoOp combined with OpenCLIP offers potential for enhancing prompt-based classification tasks. Future work may include multilingual dataset applications and addressing overfitting via methods like CoCoOp.

## 1 Introduction

Contrastive Language Image Pretraining (Radford et al., 2021, CLIP) is a comprehensive pre-trained image-language model that can handle a variety of different tasks by merging image and text features in a common space. Using CLIP on tasks with limited available data is already yielding promising results, especially on classification tasks. However, when used as a zero/fewshot classifier, it can be very sensitive to wording, requiring good prompts to achieve good results. Manual development of prompts for a particular task or dataset can be very time consuming. Instead of creating prompts manually, we use Context Optimization (Zhou et al., 2022b, CoOp) to learn them by generating the context words of the prompt through a learnable vector while keeping all other pre-trained parameters fixed. Zhou et al. (2022b) uses the various CLIP models from the original CLIP repository[1]. We want to use CoOp with the open source version of CLIP called OpenCLIP[2], so we can use the same backbone models from the original CLIP, but also additional models trained on other datasets or with a new backbone. For this reason, we have adapted the CoOp repository[3] to be compatible with the new models from OpenCLIP. With a Shot value of 1, the same backbones of OpenCLIP perform slightly worse than their CLIP counterparts and new models perform poorly. Increasing the shot value to 16 seems to fix this behavior and leads to promising results when combining CoOp with OpenCLIP. Our source code is available at https://github.com/BrandnerKasper/MP_CustomCoOp.

## 2 Background

Before we dive in some adaption details and there results, we have to explain some key terms before.

### 2.1 Contrastive Language Image Pretraining (CLIP)

Commonly un-, self- or weak-supervised models are trained with a lot of images to learn concepts for classification. But these models struggle to cover other concepts not presented in their image data, making them unusable for downstreaming on other tasks. CLIP learns visual concepts from natural language supervision, allowing it to learn concepts from raw text. This also allows CLIP to learn from data not originally prepared for machine learning, like a photo on Instagram with a simple caption. The core idea of CLIP is to combine an image encoder, like Resnet 50, 101 or Vision Transformer (ViT), with a text encoder, a transfomer based model.

Residual Network (ResNet) is a convolutional neural network (CNN) with a layer count of, say, 50 or 101. The core idea of ResNet is to use residual blocks to solve the vanishing gradient problem. The vanishing gradient problem occurs when the gradient used to update the weights in the network

[1] https://github.com/openai/CLIP
[2] https://github.com/mlfoundations/open_clip
[3] https://github.com/KaiyangZhou/CoOp

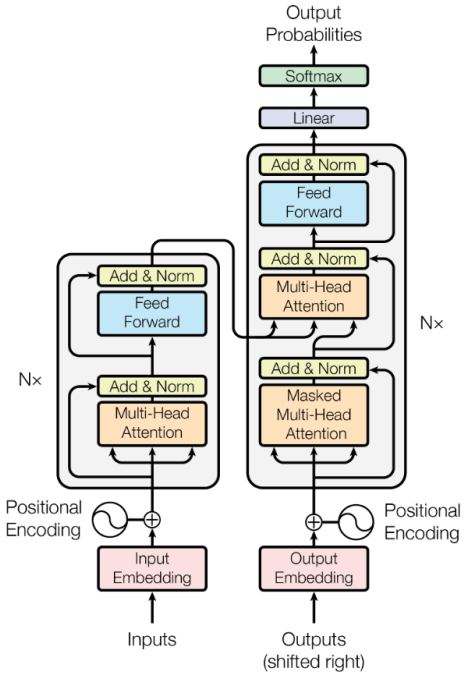
becomes too small during training and therefore vanishes. This phenomenon occurs more frequently as the network becomes deeper. Residual blocks solve this problem by allowing skip connections that allow gradients to bypass one or more layers (He et al., 2016).

The transfomer model is a deep learning architecture known for its ability to efficiently capture long-range dependencies in data through a self-attention mechanism. The self-attention mechanism allows a model to weight the importance of different parts of the input sequence as each element is processed by assigning an attention value that determines its contribution to the output. This is particularly useful when processing sequences of different lengths (e.g. sentences). The Transfomer model uses multiple self-attentional heads (multi-head attention for short) to capture different relationships between elements in terms of attentional patterns. The output of multi-head attention is then used to compute the final output vector weighted by attention. Unlike recurrent neural networks (RNN) or long short term memory (LSTM) blocks, the transfomer model has no idea about the order of elements in its sequence. Therefore, it relies on position encoding. The transfomer uses an encoder-decoder architecture, as seen in 1. The encoder stack processes the input sequence while the decoder stack generates the output sequence, allowing for better results in a translation task, for example (Vaswani et al., 2017).

The transfomer architecture can also be used for image tasks (ViT!), by cutting the image in multiple patches and align them to a sequence to be used as input. ViT models are divided into three categories: Base, Large and Huge, according to the number of layers, multi head attentions and paramters. They are further distinguished by their patch size. For example ViT-B-32 is a vision transformer of the base category with patch size of 32x32 pixels (Khan et al., 2022).

Both, the image as well as the text encoder, are trained from scratch. Given a batch of N (image, text) pairs, CLIP is trained to predict which of the NxN possible (image, text) pairs occured. It does this by learning a multi-model embedding space by training the image/text encoder to maximize the cosine similarity 1 of the image and text embeddings of the N correct pairs and minimizes it on the wrong pairs. Cosine similarity measures the cosine of the angle between two vectors projected in a multi-dimensional space (Prabhakaran).

Figure 1: The architecture of the Transformer model: Left) The encoder - applying the input sequence with position embeddings to the multi-head attention block. Right) The Decoder - using the context vector generated by the encoder as input. Note: In both cases, multiple skip connections can be seen.




$$cos\Theta(\mathbf{A},\mathbf{B}) = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

The whole model is is further optimized by a symmetri cross entropy (SCE) 2.

$$SCE(\mathbf{p},\mathbf{q}) = -\frac{1}{N}\sum_{i=1}^{N}[p_i \log(q_i) + (1-p_i)\log(1-q_i)] \quad (2)$$

Cross-entropy is a measure of the difference between two(binary) or more(multi-class) probability distributions for a given random variable or set of events. Traditional cross-entropy loss, which is commonly used for classification, can lead to biased models in imbalanced datasets. SCE introduces a new parameter, alpha, which controls the weight assigned to each class. It uses a symmetric formulation to balance the contributions of both classes (0 and 1) equally, regardless of the class distribution in the dataset (Wang et al., 2019).

### 2.2 Context Optimization (CoOp)

Zhou et al. (2022b) observed that for pre-trained vision-language models, like CLIP, the text input,

known as prompt, plays a key role in downstream datasets. A prompt can be something simple like *"a photo of a [CLASS]"*. Identifying the right prompts is a non-trivial task, which often takes a lot of time to fine tune. Slight changes in wording can make a huge difference in performance. This process is called prompt engineering, a type of fine-tuning that involves designing input texts so that these models provide the desired and contextually relevant responses. Effective prompt design can significantly impact the performance and behavior of pre-trained language models, making them more adaptable to specific use cases and tasks. The idea of CoOp is to automate prompt engineering, especially for pre-trained vision language models. CoOp models the context words of a prompt with learnable vectors (see 2), which can be initialized either with random values or with pre-trained word embeddings.

Two implementations are supported: a) uniform context - common context for all classes, which works well in most cases. b) class-specific context - learning a specific set of context tokens for each class, whis is more suitable for fine-grained categories.

CoOp performs its own training cycle while keeping all pre-trained parameters of CLIP fixed, the prediction errors of the learnable context vector are minimized using cross entropy. The resulting gradients are fed back through the text encoder to distill knowledge and obtain task-relevant prompts.

$$t = [V]_1[V]_2 \dots [V]_M[CLASS] \tag{3}$$

The context vector t for our prompt can be described like 3 or 4, except that the class token is positioned at the end or in the middle. It can consist of up to M context words, where M is between 1 and 16.

$$t = [V]_1 \dots [V]_{\frac{M}{2}}[CLASS][V]_{\frac{M}{2}+1} \dots [V]_M \tag{4}$$

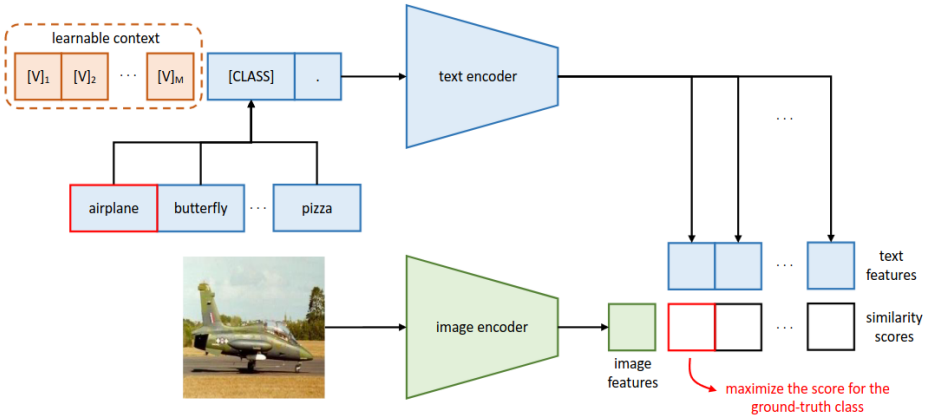
When CoOp's context vector is trained with a few-shot value of 1 or 2 for CLIP, it outperforms already hand-crafted prompts for a given data set. The main difference between zero-shot and few-shot learning lies in the amount of labeled training data available for the model. While no labeled data is available in a zero-shot scenario, so the model must learn new classes and concepts on its own, 1 to 16 labeled examples of new classes or concepts are available in a few-shot scenario. Increasing the value for CLIP to 16 results in an average gain of 15% with CoOp.

## 3 Methodology

As mentioned before, we have adapted the CoOp repository: First we reworked some abstractions, the CoOp code should be called via bash scripts, we reworked and simplified this so that it is easier to debug. Besides the ResNet 50,101 and the ViT-B-16,-32 our main idea was to use the new models Roberta-ViT-B32 and XLM-Roberta-Base-ViT-B32 from the OpenCLIP repository with CoOp. We started by adding the same backbone models (ResNet 50&101, ViT-B-16&32) from the OpenCLIP repository. (OpenCLIP claims very similar behavior of these backbones with the pre-trained tag *'openai'*.) To do this, we included the OpenCLIP repository as a dependency and wrote a function that loads the desired backbone with the correct pretrained tag and injects it into CoOp's codebase. Some minor adjustments also had to be made, as the API between the CLIP and OpenCLIP repositories has some differences. For example, some variables of the CLIP class were renamed or moved, and the text encoder class no longer uses an explicit call to the encoding function, but a call function to use the class instance. Adding support for the new models was a bit more difficult since they use a newer text encoder: The Robust Optimized BERT Pretraining Approach (RoBERTa) builds on Bidirectional Encoder Representations from Transformers (BERT). BERT is designed to capture context and semantics from both directions (left-to-right and right-to-left) within a text sequence. It uses a masked language modeling target during pre-training where it predicts masked words in a sentence (Devlin et al., 2018). RoBERTa is a robust and carefully optimized version of BERT by adjusting key hyperparameters and training data size (Liu et al., 2019). Zhou et al. (2022b) modified the text encoder code of the CLIP code base to pass the prompts as input. Similarly, we modified the hf (hugging face) text encoder of the OpenCLIP codebase to pass the prompts as input embeddings. The difference between Roberta-ViT-B32 and XLM-Roberta-Base-ViT-B32 is in the training data. While Roberta-ViT-B32 is limited to English texts, XLM-Roberta-Base-ViT-B32 is suitable for a multilingual environment.

We summarized our work in a Python script

Figure 2: Overview of CoOp built upon a CLIP's image and text encoder: The idea is to model a prompt's context using a set of learnable vectors, which can be optimized by minimizing the classification loss.




*train.py*, which serves as an argument:

1. the path to the dataset (only Caltech is supported at the moment).
2. the backbone (ResNet 50, Roberta-ViT-B32 etc.)
3. the number of shots (1 to 16)
4. –openai (a tag to decide if we want to use the backbones from the old CLIP or the new OpenCLIP)
5. –pretrained (a tag to load a pre-trained backbone from another dataset; this is only supported in the OpenCLIP repository)

The context word count (n_ctx) is set to 16, the class token position (ctp) is set to the end of the sentence, and the class specific context (csc) is set to false, resulting in a uniform context. (Recommended settings for the Caltech dataset).

We have also added two helper scripts. First, *list_available_models.py*, which lists all backbones and their pre-trained tags supported by this codebase. Second, *eval.py*, which calculates the average accuracy with standard derivation for all generated output files. The output files have an automatic naming scheme starting with clip or open_clip, followed by the backbone name, its pretrained tag, and the shot number. This results in, for example, *open_clip_roberta_base_vit_b32_laion2b_s12b_b32k_1.txt*.

At last, we prepared a jupiter notebook so this code was and can be used on Kaggle[4], a website to run your machine learning code on more suited hardware.

[4] https://www.kaggle.com/

## 4 Evaluation

We evaluated two things: first, how the same backbones of CLIP perform against OpenCLIP, and second, how the two new models Roberta-ViT-B32 and XLM-Roberta-Base-ViT-B32 would perform on the Caltech-101 dataset. The Caltech-101 dataset is a well-known computer vision dataset used for object recognition and classification tasks. It was created by researchers at the California Institute of Technology (Caltech) and contains images of objects belonging to 101 different categories. For our use case, the Caltech101 dataset was used only as a proof of concept because it is relatively small and is commonly used for benchmarking and evaluating the performance of image classification algorithms and computer vision models.

The following results in table 1 were obtained on Kaggle with n_ctx set to 16, ctp set to end, csc set to false, and shot count set to 1.

Two values are shown in the table: the average train accuracy with standard derivation over all epochs and the average test accuracy with standard derivation over the 3 seed runs. Comparing the actual test results of the same backbones between CLIP and OpenCLIP, it is noticeable that the CLIP backbones perform slightly better overall. In one case, with the ResNet 50 backbone, the error rate of the OpenCLIP version does not converge, resulting in a terrible value of $4.5\%$. We don't have a good explanation for this behavior, since ResNet 101 performs quite well and to our knowledge there shouldn't be that much difference between the implementation and hyperparameters of these two versions. The new models also perform very poorly with a test accuracy of $38.4\%$

| Backbone | CLIP Accuracy | | Open CLIP Accuracy | |
|---|---|---|---|---|
| | Train | Test | Train | Test |
| **RN 50** | $87.3\% \pm 9.8\%$ | $83.3\% \pm 3.1\%$ | $3.6\% \pm 3.0\%$ | $4.5\% \pm 0.0\%$ |
| **RN 101** | $88.1\% \pm 8.9\%$ | $85.4\% \pm 3.1\%$ | $86.0\% \pm 20.1\%$ | $80.2\% \pm 1.2\%$ |
| **ViT-B-32** | $90.4\% \pm 7.4\%$ | $87.4\% \pm 0.9\%$ | $75.7\% \pm 27.5\%$ | $73.6\% \pm 5.9\%$ |
| **ViT-B-16** | $90.5\% \pm 7.2\%$ | $89.7\% \pm 1.8\%$ | $87.4\% \pm 16.5\%$ | $86.1\% \pm 2.1\%$ |
| **Roberta-ViT-B32** | / | / | $63.2\% \pm 33.4\%$ | $38.4\% \pm 1.8\%$ |
| **XLM-Roberta-Base-ViT-B32** | / | / | $64.6\% \pm 34.9\%$ | $42.2\% \pm 1.8\%$ |

Table 1: Comparison of the accuracy of the CLIP backbone implementation of CoOp with the Open CLIP implementation of the same backbone in CoOp and with the new Roberta-ViT-B32 and XLM-Roberta-Base-ViT-B32 backbones. All models were trained with a shot value of 1. Two values are displayed per run, the average accuracy with standard derivation for all training and test results. Interesting values are highlighted in color.

for Roberta-ViT-B32 and $42.2\%$ for XLM-Roberta-Base-ViT-B32. It is interesting to note here that the training accuracy is much better with an accuracy of $63.2\%$ for Roberta-ViT-B32 and $64.6\%$ for XLM-Roberta-Base-ViT-B32, which means that the two backbones perform quite well in training but seem to underperform in the test set. Either we made a mistake in the implementation of the custom text encoder by entering the prompts as input, or the two models overfit here. This argument appears to be valid, as the context learned from CoOp does not generalize to other unseen classes within the same dataset, suggesting that CoOp overperforms the base classes observed during training, as Zhou et al. (2022b) suggests. Looking at the training accuracy, we can say that the CLIP backbones seem to learn much faster overall, or with fewer epochs, with standard derivations of under $10\%$. While the OpenCLIP backbones seem to learn much slower with standard derivations up to $35\%$!

We have done some additional runs with interesting findings: For example, changing the pretraining tag for the OpenCLIP-ViT-B-32 from *openai* to *laion2b_s34b_b79k* increases the average accuracy from $73.6\%$ to $85.1\%$, a difference of $12\%$! Increasing the shot value from 1 to 16 on the Roberta-ViT-B32 makes a huge difference, resulting in an average accuracy of $88.6\%$, a difference of over $40\%$! At this point, we can say that an implementation error is unlikely. Roberta-ViT-B-32 appears to abstract very well from the training phase and performs above average on the test set, with an average training accuracy of $70.4\% \pm 18\%$. Finally, we set the shot value for the OpenCLIP ResNet 50 to 16 as well, and surprisingly it achieved an average test accuracy of $88.5\%$ with a training accuracy of $58.4\% \pm 25.5\%$. This is very close to the original CLIP results and far better than our $4.5\%$.

## 5 Conclusion

Using CoOp on CLIP produced very good results, and using CoOp on OpenCLIP produced promising results. When using the same backbones as the original CLIP, the OpenCLIP backbones performed slightly worse than their CLIP counterparts. In particular, ResNet 50 and the new Roberta-ViT-B32 and XLM-Roberta-Base-ViT-B32 models perform significantly worse with a shot value of 1. Most likely due to CoOp not generalizing to other unseen classes within the same dataset and overfiting base classes observed during training. However, our test runs with a shot value of 16 improve these results dramatically with test accuracies of up to $88.6\%$! This makes CoOp in combination with the new models from OpenCLIP a promising prospect for further investigation.

## 6 Future Work

Our original plan was to work with a different data set in a multilingual environment. Therefore, we chose the XLM-Roberta-Base-ViT-B32 as the backbone of OpenCLIP. Unfortunately, due to time constraints, we did not implement support for the multilingual dataset. Looking at our test runs with a shot value of 16, testing XLM-Roberta-Base-ViT-B32 with CoOp on a multilingual dataset would be the obvious next step. Zhou et al. (2022a) has already noted that CoOp overfits in some cases, and proposed a solution called Conditional Context Optimization (CoCoOp) that extends CoOp by learning a lightweight neural network to gener-

ate a conditional token for each image. With this conditional token, CoOp can adapt to cases where it overfits and also achieve better performance in generalizing domains.

## References


Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. 2022. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Selva Prabhakaran. Cosine similarity.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. 2019. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 322–330.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022a. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16816–16825.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022b. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348.