An Exploration on How Clip Handles Out-of-Distribution Problem with Finetuning

Leqi Wang [1] HanFu Chen [1]

Abstract

Clip model has shown its power ever since it was released in 2021, and its excellent performance on traditional image classification tasks is one part of it. However, as the original paper of Clip model and our test has both shown, when Clip meets with the test data that are out of the possibility distribution of its original training dataset, even though it's extraordinarily large, the model may still exhibit poor test accuracy. In correspondence to it, we try to dig in the understanding ability of model Clip. And through the technique of finetuning, we research how Clip model responds when it meets with those datasets who have a new distribution of information compared to the original training datasets. Finally, we use Clip to challenge a harder classification task, from this experience we are inspired by new ideas of using models like Clip to construct "World Knowledge", and manually implement a technique of finetuning Clip without altering its parameters.

1. Introduction

Clip was released by OpenAI in 2021, it has an image encoder and a text encoder separately. When used for image classification, each of the target is represented by a prompt like "this is a photo of (target)" with text encoder, and the image will be encoded by the image encoder and then be computed the cosine distance with different target prompts using simple vector multiply for prediction, as shown in the Fig 1.

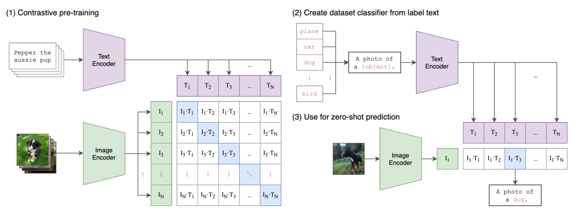
(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
(3) Use for zero-shot prediction
A photo of a dog.

Figure 1. Figure 1 in Clip's paper

There has been lots of different research about it, such as its transfer learning ability, while we choose to focus on a rather simple one: the traditional image classification task. The OpenAI used tons of training data of images and its corresponding caption, of which mostly are collected from Internet, as materials for the model's pretraining, hoping for its achieving a powerful zero-shot inference ability, and the result published by the Clip model is rather interesting.

As the Fig 2 shows, compared with ResNet50 with linear probe finetuned, Clip did some great job on datasets such as StanfordCars and Food101. However, when faced with some rather simple datasets, such as the MNIST, which is a hand-written number detection dataset, the model's prediction accuracy drops severely, so does it when Clip met with Flower102.

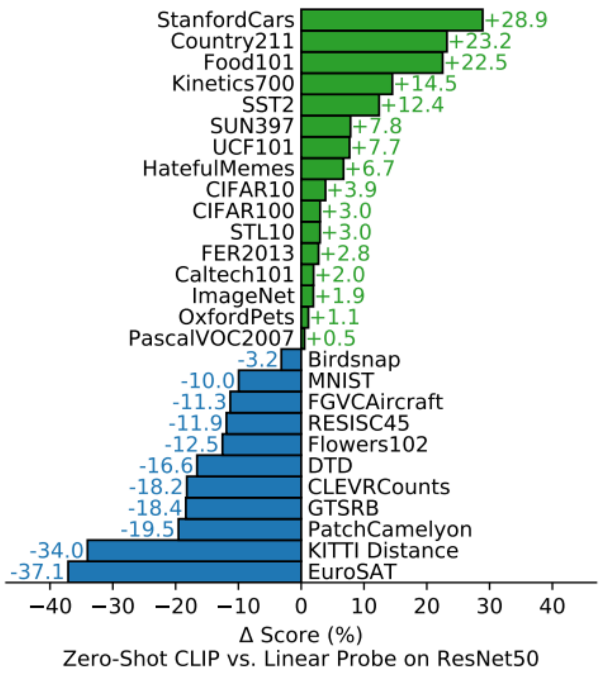

Figure 2. Figure 5 in Clip's paper

Table 1. Prediction accuracy on different datasets

DATA SET	CLIP	RESNET50
FOOD101	80.7	61.87
CIFAR100	61.8	61.67
MNIST	32.63	94.54
FLOWERS102	51.9	77.96
ANIMALS90	86	95.28

We propose a possible explanation to this phenomenon.As shown in the Chapter 2.2 in the Clip's paper, OpenAI used the (image,text) pair collected from the Internet as the training data, thus making it self-constrained to some probability distributions, which makes the model respond actively to certain datasets that might be common on the Internet, while perform badly on certain datasets that might be out of the original probability distribution. Our test result, as shown in Table 1 where the columns represents the prediction accuracy, also exhibits such characteristics that Clip does poor zero-shot inference on datasets like MNIST, Flower102 and Animal90, since there might not be so much pure handwritten numbers or specialized knowledge about plants or animals in the form of (image,text) pair on the Internet.

Here we name such problem as **Out-of-Distribution Problem**. And how shall we deal with it?Well, we might should try not to push so hard on Clip model, since even as human, we still take some time to learn when we meet with knowledge out of our daily experience. If we consider each Out-of-Distribution dataset as a **domain of symbol**, then the model should be using "few-shot learning" instead of "zero-shot learning" to differentiate this domain of symbol from others, and also capture the inner relationships of it.

To explore the possibility of achieving this goal, we take three steps in this paper:

- First,we examine how the prediciton result goes as we finetune different parts of the model, in order to see the inner understanding of Clip, we use a technique called "Grad-cam" to capture the model's attention on the picture.
- Second, we will try on a harder dataset called "hateful memes" which has a far more complex and larger semantic information, and we will try two different ways to see if we can guide the model to place the attention on the right place.
- Finally, we will implement a finetuning technique which requires no parameter-alternation to be made, and the mechanism of how this technique works may better illustrate our opinion about the transition between different symbol domains.

2. Coarse-grained finetuning

In this section we will discuss the first step we take on exploring the Clip.

2.1. Finetuning result

We will finetune the different parts of the model, including the image-encoder, text-encoder, the linear projection of the image-encoder and the linear projection of the text-encoder, on the basis of the training data from MNIST, Flower102 and Animal90 in order to reach a higher prediction accuracy. We use Adam as our optimizer algorithm ,and for the sample number per class we choose for each dataset, the MNIST's is [1,5,10,15,20,25,30,40,60], the FLOWERS102's is [1,2,4,6,8], and the ANIMAL90's is [1,5,10,15,20,25].

And the result is shown in the Table 2,3,4:

Table 2. Finetuning result on MNIST

FINETUNED PART									
TEXT ENCODER	57.8	80.8	82.1	84.9	83.8	87.6	88.0	89.7	92.5
IMAGE ENCODER	62.4	75.8	75.6	78.7	82.4	84.5	85.8	84.9	89.6
TEXT PROJECT	35.9	36.2	37.1	37.9	39.3	39.9	40.1	40.3	40.4
IMAGE PROJECT	37.5	36.9	38.2	39.0	40.0	40.6	40.5	40.6	40.8

Table 3. Finetuning result on Flowers102

FINETUNED PART					
TEXT ENCODER	53.0	54.1	57.0	56.9	56.5
IMAGE ENCODER	61.6	55.2	55.8	62.0	63.0
TEXT PROJECT	63.0	63.2	63.1	63.3	63.2
IMAGE PROJECT	63.2	63.1	63.2	63.2	63.1

Table 4. Finetuning result on Animals90

FINETUNED PART						
TEXT ENCODER	61.0	83.6	82.0	79.7	83.1	85.0
IMAGE ENCODER	56.8	74.4	72.1	76.5	78.0	79.7
TEXT PROJECT	93.6	93.7	93.6	93.7	93.6	93.7
IMAGE PROJECT	93.8	93.6	93.6	93.7	93.7	93.7

We can see that as the number of samples goes up, no matter what part of the model you finetune, the prediction accuracy all goes up with it. The projection layers performs the worst for the MNIST, but take a little bit lead over the encoder for the ANIMAL90 and FLOWERS 102. As for the reason, we think it may be related with that the pretrained CLIP might have seen lots of pictures like those in ANIMAL90 and FLOWERS 102, so it doesn't take the whole encoder to be retrained, training the Linear layer is all it needs, while the MNIST dataset has a much more rare possibility distribution on the INTERNET, so it may take the whole encoder to be retrained and get used to its ”domain of symbol”. We will do more serious research on this phenomenon in the future work.

2.2. Grad-cam attention

In order to see whether the encoder has actually captured the meaning of symbol, we use a technique called **Grad-cam** (https://github.com/jacobgil/pytorch-grad-cam) to see which part of a picture contributes the most when the model assigns this picture to a certain category. As we change the size of finetuning datasets, we see that the attention (the buler part of the image) does move into someplace where the picture’s characteristic lies, like the animal’s face or the curve of the number. The result is shown in the Fig3.

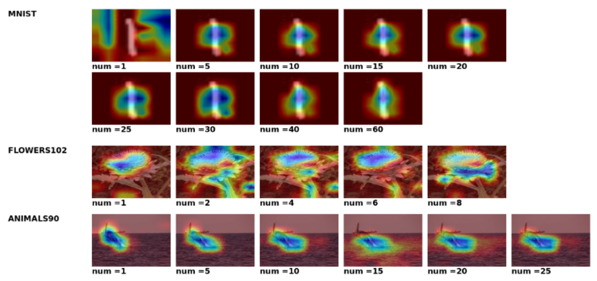

Figure 3.

As for the example in MNIST and ANIMAL 90, the variation of attention may converge as we increase the number of samples, as well as their prediction accuracy. And if we set the sample-number smaller, like from one to five, we can see the attention moves in a much more obvious way, as shown in the Fig 4.

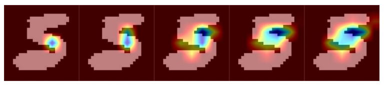

Figure 4.

And we have also found that if you force the model to predict a wrong target, the attention would be distracted and diffused, as shown in the Fig 5, we make the model predict ”one” on the basis of the image input of number ”five”.

2.3. A harder classification task

We then move our focus to a much harder image classification task: the Hateful Memes dataset, which is constructed by MetaAI (https://ai.meta.com/tools/hatefulmemes/). Each item of the dataset is an image with a caption on it, and

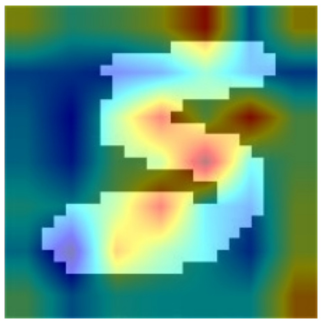

Figure 5.

the model have to combine the information of text and image to decide it’s hateful or not, which sounds quite fit the characteristics of model Clip whose image-encoder and text-encoder are separately trained.

For finetuning, we make text-encoder generate two prompts for training, one with ”this photo is hateful”, the other with ”this photo is not hateful”, setting the batch-size as two, training a positive sample together with a negative sample at a time, the other setting remain fixed as we discussed in chapter 2.1. However, our finetuning result doesn’t turn out well as we thought, the highest accuracy doesn’t surpass 0.52, which is close to the possibility of random guessing.

Even when we try to insert the caption into the prompt, such as ”this photo says {caption}, it is (not) hateful”, the result still doesn’t get any better.

In order to dig the reason for that, we generate two grad-cam attention pictures.The first one is picture exhibiting the head of a goat, with the caption of ”safe sex, ISIS style”, which is hateful because it aims at defaming Arabian people as being terrorists and having sex with goats. We choose the whole part of image-encoder and text-encoder as the target in the remaining finetuning jobs since the dataset is complex so that there’s a lot to learn for the model.

The Fig 6 is result of finetuning visual encoder as we increase the sample number from 1 to 60, while the Fig 7 is the result of text encoder, we can see that the former one seems to only place attention on the head of goat, while the latter one places attention on both the head of the goat and the text ”ISIS”, which seems to make more sense because it’s these two elements combined that makes this meme hateful.

This result enlightened us about how the different encoders interacts.The image encoder maps information into a visual semantic space, while the text encoder maps information into a text semantic space. When you finetune the text-encoder, you are actually drawing certain distribution of the text semantic space towards the certain distribution of the visual semantic space, make them closer in distance,

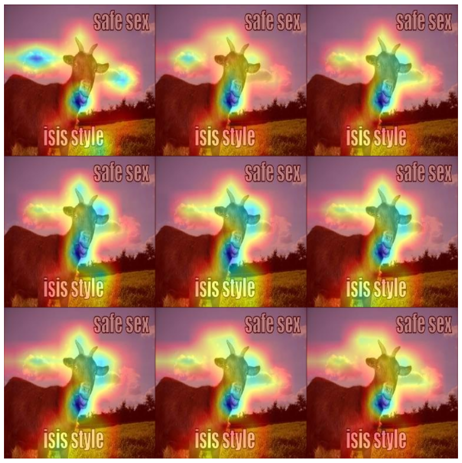

Figure 6.

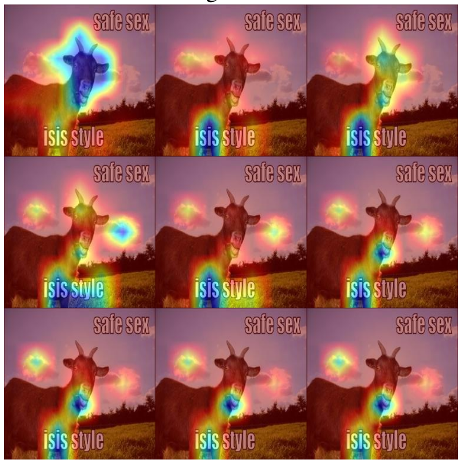

Figure 7.

while the finetuning of image-encoder moves in the reverse direction.

However, the result doesn't turns out well when we generate the grad-cam picture of another hateful example, whose visual information is Hitler raising his fists while text information is ”gas yourself”, which aims at making fun of Jewish people's miserable history during WW2 for being killed by poisonous gas. As shown in Fig 8, after we finetune the text encoder, the model lose attention of the picture and get diffused everywhere.

These two examples exhibit different performance on the attention map, as for the reason, we think that the straightforwardness of the first example make it easier for the model to focus attention, while the second example is not that straightforward, because it actually contains an implicit semantic link: lots of jewish people were killed by gas —— Hitler killed lots of jewish people —— thus combining ”gas” with ”Hitler” in the picture might be hateful. However, the semantic elements of ”jewish people” doesn't exist in this

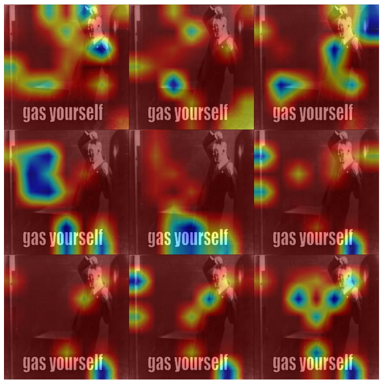

Figure 8.

meme, breaking the logic chain, thus making it harder for the model to focus attention.

In order to help the model focus right, as we have discussed, we may need to find out a way to fill the broken chain, and we make two attempts in the following two sections.

2.3.1. Simple alignment between picture and text

For the first attempt, we allocate 10 pictures of Hitler from the Internet, as shown in Fig 9, each paired with text ”Hitler,hateful”, mixed with other non-hateful data, and then thrown into the model to finetune the text encoder, the result is Fig 10. We can see that as the sample numbers goes from 2 to 8, the attention was gradually drawn to the head of Hitler instead of diffusing everywhere, which means the model starts to take the visual prompt of Hitler as a trigger for making decision.

However, this method still has its flaw, cause it only takes visual information into consideration but ignores the semantic elements of ”gas” and ”jewish people”, in the next attempt, we will try to fix this.

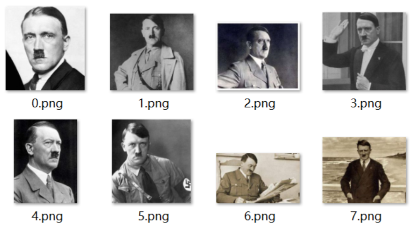

Figure 9.

2.3.2. Text encoder self-learning

In the second attempt, we will strictly follow the logic link mentioned in the chapter 2.3: lots of jewish people were killed by gas —— Hitler killed lots of jewish people ——

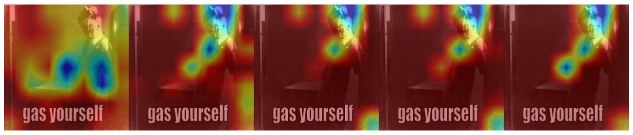

Figure 10.

thus combining ”gas” with ”Hitler” in the picture might be hateful.

We first follow the operation we did in the 2.3.1 except that we replace the text with ”hitler, killed many jews”, this operation is intended to simulate the second node on the logic chain, building up the alignment between picture and caption.

Then we load another Clip model and freeze its grad, therefore we can use it as a benchmark to justify the former model’s text encoder. We let the **frozen** encoder generate two pairs of text: (”hateful”,”positive”) and (”hateful”,”happy”), where ”happy” and ”positive” is a pair of synonyms, in order to help model differentiate word ”hateful” from some delightful meanings. And we also let the **unfrozen** encoder generate text of ”hitler killed lots of jews by gas”, and train it to finally categorize the text as ”hateful”, in corresponding to the first node of the logic chain. The training pipeline is illustrated in Fig 11.

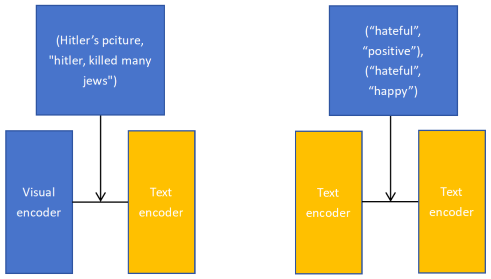
(Hitler's pciture, "hitler, killed many jews")
Visual encoder
Text encoder
("hateful", "positive"), ("hateful", "happy")
Text encoder
Text encoder

Figure 11.

After rounds of training, the model yields the right prediction, and the result of grad-cam is presented in Fig 12. As the sample number of (image,text) pair increases from 0 to 8, also with the cooperation of text encoder’s self-supervised learning, the model gradually locks its attention on the word ”gas”, which implies that the model gradually understands the semantic meaning of ”gas”, and that not only the image-text alignment works, but also does the text-text alignment.

Though we haven’t found out a better way than the grad-cam attention to research the understanding ability of Clip, and are also constrained to the ability to find large enough auxiliary training data, the above method of training may still bring inspiration to us that the release of Clip may overthrow the traditional way of classification task.

As in the real world, we don’t always get knowledge that are directly mapped to each other, such as the image classification task where each image is directly linked with a category. Instead, we often pick up randomly related knowledge and link them together with our own logic ability, eventually leading us to gain the so called **world knowledge**. Since the Clip architecture has a separate construction of visual and text encoder, we might not need to be bothered by constructing a strongly mapped training dataset anymore, instead, we can use technique like Web Crawler to collect discretely linked information on the Internet, like what our job has done, and feed it into model like Clip, hoping it would self-construct the **world knowledge**.

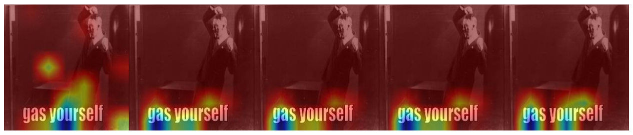

Figure 12.

3. Finetuning without altering parameters

Remember when we were in chapter 1, we discussed about the concept of **domain of symbol**, and we hope through finetuning, model can transfer from the old domain of symbol to a newer one, thus helping it perform better on image classification task of a new dataset. And after that we use the technique called ”grad-cam” to get a look of the transition of symbol domain by attention map.

Here we are going to take a much more direct way to comply the concept of **domain of symbol**, we implement a technique called Tip-Adapter(https://github.com/gaopengcuhk/Tip-Adapter), which does alternation on the input information instead of the parameters, as shown in the Fig 13. It combines the information of input image, the text prompts and the image samples of all classes, which self-contains the knowledge of this dataset, and then generate the probability logits by a simple activation function and matrix multiply. Using Tip-Adapter, the zero-shot predicition accuracy on MNIST with 5 samples each class reaches 70%, far beyond 32% with usual zero-shot inference. The finetune result on MNIST, ANIMAL90 and Flowers102 are plotted in Fig 14, and the grad-cam attention map by Tip-Adapter with sample number from 1 to 16 is exhibited in Fig 15.

Our implementation of Tip-Adapter could be found in https://github.com/Danny-1-8/ClipProgram/blob/main/tipadapter.ipynb.

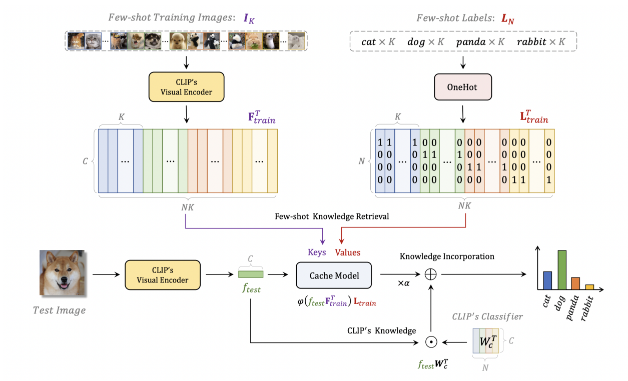

Figure 13. Figure 1 in the Tip-Adapter's paper

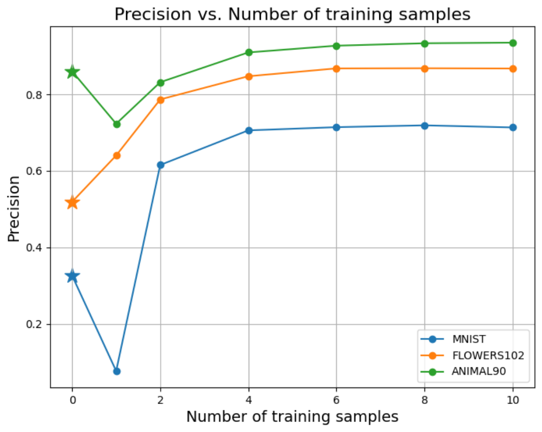

Figure 14.

4. Conclusion

In the discussion of chapter 2, model Clip exhibits powerful ability of adaption when met with relatively simple but out-of-distribution dataset like MNIST, ANIMAL90 and FLOWERS102 by coarsely finetuning certain part of the parameters.

However, when met with highly semantic datasets like Hateful-memes, even finetuning couldn't improve its performance because of its lack for so-called **world knowledge**. Fortunately, the free structure of model Clip bring us opportunities to finetune the model in an unusual way, following the rule of logic link and drawing the attention of the model to the right place. If we got more time and resource, we would research on how to apply this method to a broader range of data samples. Moreover, we would do serious ablation study over the result Clip produced.

In the end, we manually implement the Tip-Adapter pipeline,

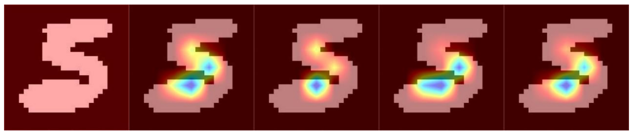

Figure 15.

which strictly complies with the principle of ”domain of symbol” we brought up with in the Chapter 1, and it shows that Clip performs well when we intentionally integrate the knowledge of symbol domain into the input information.

I think the value of our work is, we not only exhibit our ability to build finetuning pipelines, to master relevant research toolkits like Grad-cam and to reproduce the code implementation of relevant paper works, but also to state our idea about how the characteristics of Clip's architecture could bring a fresh air to the traditional classification task or even more challenging work like constructing the ”World Knowledge”, and also our understanding about the feature space embedded by the different encoders that guides our work.

I have to admit that there's still lot of questions remain to be discussed about this work, and we will take it as a fresh start and inspiration for our future research career.