Revolutionizing Image Caption Fine-Tuning with Adapters: A Parameter Efficient Approach

Chijie An ##1

Abstract

The popularity of text-to-image models has led to discussions on reversing the typical direction of a generative text-to-image model. Instead of generating an image from a text prompt, our goal is to generate a reliable prediction of the text prompt given a generated image. AI painting has been an influential tool in the creative field, and similarly, the image-to-prompt field could establish a strong connection between computer vision and natural language processing, creating a new field of interest. However, the training and improvement of this kind of model require high model complexity, consume significant computational power, and demand high-quality labeled training datasets. These preconditions make it challenging to train the model with limited resources and investment.

In this paper, we address this problem by first implementing classic fine-tuning methods on a classic BLIP model to successfully fine-tune it on a highly correlated dataset. Next, we added an additional adaptor, a residual neural network, after the feed-forward neural network in the image and prompt encoder in the BLIP model to improve the output results. Our approach improved the fine-tuning results while consuming less computational power.

1 Introduction

Image-to-text models are composed of an image-encoder and text-decoder, while the attention architecture [1] and the BERT encoder [2], is crucial for an efficient and accurate training model. Currently the attention model, as a neural network architecture, has been applied to a wide range of natural language processing tasks. Self attention layers in encoder and cross attention layers in training allows the model to effectively put attention to the most important parts of the input sequence while ignoring irrelevant information. BERT, a pre-training language model that uses a Transformer architecture with a bidirectional encoder to learn contextual information about words, allows for better understanding of semantics and syntax in natural language. BERT can be fine-tuned for specific natural language processing tasks.

Due to the lack of high-quality datasets and computational power, we started by fine-tuning a classic BLIP model for our image-to-text task. The BLIP model uses attention and cross-attention networks with feed-forward layers that share parameters from loss functions. It employs the BERT module for text encoding and raises the Image-Text Matching Loss. The decoder is trained with a language modeling loss to generate captions given images. During our classic fine-tune, we only fine-tuned the LM and locked parameters in the image encoder, as shown in Figure.1. However, the loose dataset we used resulted in the degeneration problem, as seen in Figure 2 and 3, where the generated text was even less readable after fine-tuning. To address this, we changed to a more correlated dataset and successfully fine-tuned the model. To address the high consumption problem, we fine-tuned only an adapter after the self-attention and feed-forward layers. This adapter fine-tuning improved the BLIP model under limited dataset conditions.

In this paper, we will first introduce dataset and model choice. Then, the result and discussion would be stated.

2 Dataset

Two types of datasets were used for different tasks in this project.

The classic fine-tune used a small sample dataset of football players, consisting of six highly correlated images, to enable comparison with a loose dataset.

In contrast, the adapter fine-tune used the

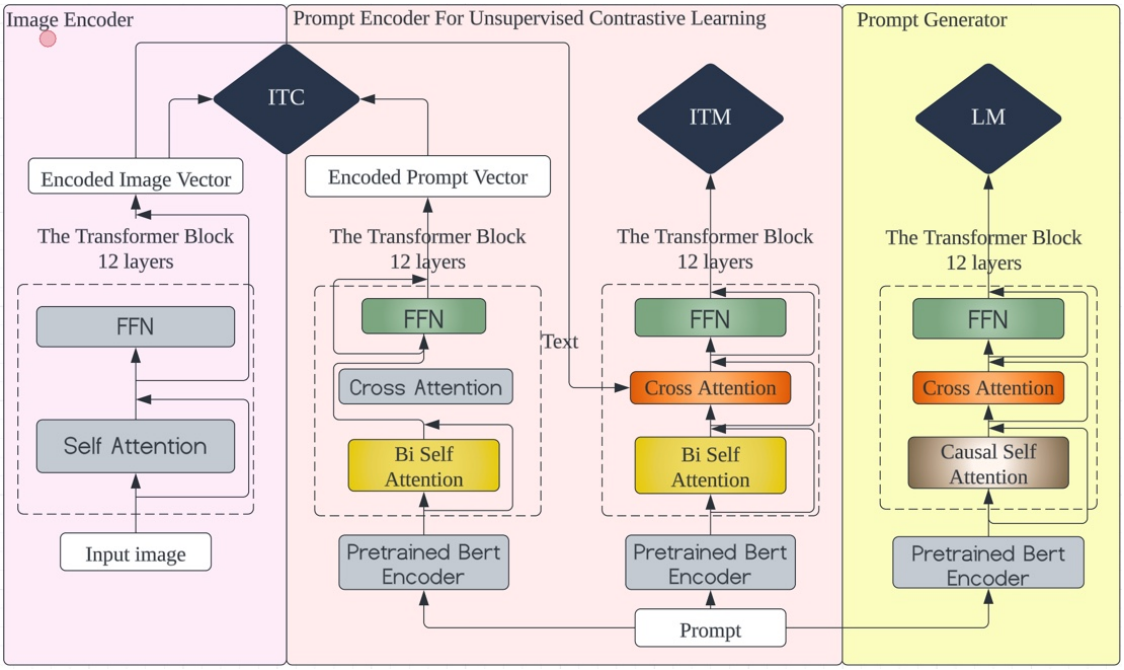

Figure 1: Grey Blocks: We freeze the parameters
Blocks with same color: Share parameters

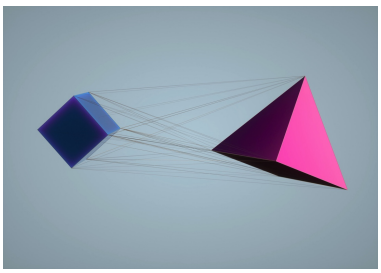

Figure 2: *0 iteration (original model)
a kite with a pink and blue kite
*10 iteration (classic model, loose dataset)
the the triangular is a triangular, and the triangular is a triangular, and the triangular is a triangular, and

Figure 3: *0 iteration (original model)
a woman drinking from a cup
*10 iteration (classic model, loose dataset)
The woman's face is covered with a cap and a jacket, and the name tag is over her name tag

Winoground dataset, which contains 800 images sourced from Facebook with topics mostly related to people, pets, and daily objects. The loose text-prompt dataset often requires significant computational power and a cleaning process. Since a standard hand-curated and labeled image-to-prompt dataset was not easily available, Winoground was used instead. Expert annotators carefully curated this dataset and labeled it with a rich set of fine-grained tags to assist in analyzing model performance. The images in this dataset

appear in pairs, and each pair has two captions that contain the same set of words/morphemes but in a different order.

3 Model Choices, Design and Training

3.1 BLIP model structure

We fine-tune on BLIP model [3] for our research. BLIP mainly consists of an Image Encoder and a Prompt Encoder. The Image Encoder is trained using a contrastive learning strategy, and the decoder is trained using a language modeling loss.

The Image Encoder is composed of 12 transformer blocks, each containing a self-attention layer and a feedforward neural network layer. It is trained using contrastive learning together with the Prompt Encoder. On the other hand, the Prompt Encoder takes the prompt as input and converts it into embeddings using a pretrained BERT encoder. It also consists of 12 transformer blocks, each containing a bidirectional self-attention layer, a cross-attention layer, and a feedforward neural network layer. The parameters are shared and also learnt through contrastive learning with the Image Encoder.

The loss functions used are the ITC loss (contrastive loss) and the ITM loss (matching loss). Detailed mathematical analysis is conducted on these loss functions to ensure optimal performance.

The Decoder is based on a BERT decoder and has 12 transformer layers. Each layer contains a causal self-attention layer, which does not share parameters with the other encoders, and a cross-attention layer and a feed-forward neural network, which do share parameters with the other encoders. The loss function used in the Decoder is the language modeling loss (LM), which is based on cross-entropy loss and matches the generated word with the ground truth word.

Each of the attention layers and FFN layers in the transformer blocks are connected using the residue connection, in order to reduce the risk of vanishing gradient.

3.1.1 The Image Text Contrastive Loss

Let v_i denote the encoded vector of the i th image and u_j denote the embedded vector for the j th prompt, the constractive loss is given by:

$$l_i^{v\to u} = -\log \frac{exp(< v_i, u_i > /\tau)}{\sum_{k=1}^{N} exp(< v_i, u_k > /\tau)} \tag{1}$$

$$l_i^{u\to v} = -log \frac{exp(< v_i, u_i > /\tau)}{\sum_{k=1}^{N} exp(< v_i, u_k > /\tau)} \tag{2}$$

$$L_{contrast} = \frac{1}{N}\sum_{i=1}^{N}(\lambda l_i^{v\to u} + (1-\lambda) l_i^{u\to v}) \tag{3}$$

Here, τ, which is called the temperature and λ are hyper parameters. $l_i^{u\to v}$ and $l^{v\to u}$ are respectively the contrastive error from prompt to image, and from image to prompt. $< v_i, u_j >$ is called the cosine similarity between the encoded prompt and image, given by:

$$< v_i, u_j >= \frac{v_i^T u_j}{||v_i||||u_k||} \tag{4}$$

The consine similarity is a measure of the distance of two verctors after L_2 normalization. Applying L_2 normalization to a vector, for instance, let v_i and u_j be n dimensional vectors, we have:

$$\sum_{k=1}^{n}(v_i^k)^2 = \sum_{k=1}^{n}(u_j^k)^2 = 1 \tag{5}$$

$$\begin{aligned} d_{l_2} &= \sum_{k=1}^{n}(v_i^k - u_j^k)^2 \\ &= \sum_{k=1}^{n}(v_i^k)^2 - 2\sum_{k=1}^{n} v_i^k u_j^k + \sum_{k=1}^{n}(u_j^k)^2 \\ &= 2(1 - \frac{\sum_{k=1}^{n} v_i^k u_j^k}{\sqrt{\sum_{k=1}^{n}(v_i^k)^2}\sqrt{\sum_{k=1}^{n}(u_j^k)^2}}) \\ &= 2(1- < v_i, u_j >) \end{aligned} \tag{6}$$

Hence, we have shown that maximizing the consine similarity of two vectors is actually the same as minimizing their distance under the l_2 normalization. It needs to be emphasized that, in this constractive loss,

our goal is to minimize the distance between the vector encoded by the same image and corresponding prompt, as well as maximizing the distance between the vectors generated by different prompts and images, for we put their summation on the denominator in equation (1) and (2).

3.1.2 The Image Text Matching Loss

The image text matching loss is similar to the Image Text Constrastive loss, but it more emphasis on enlarging the distance between positive and negative samples. It uses the dot product simlarity to calculate the similarity between different vectors, which is given by:

$$sim(v_i, u_j) = v_i u_j = \sum_{k=1}^{n} v_i^k u_j^k \tag{7}$$

And the loss function is given by:

$$L_{match} = \frac{1}{2N} \sum_{i=1}^{n} \sum_{j=1}^{n} [y_{ij} log(1 - sim(v_i, u_j)) + (1 - y_{ij}) log\ max(0, sim(v_i, u_j) - \alpha))] \tag{8}$$

In this loss function, α is the boundary parameter in the loss function, which is used to adjust the distance difference between positive and negative samples.y_{ij} represents whether the vectors v_i, u_j represent the image and its corresponding prompt. $y_{ij} = 1$ if the image matches with the prompt while $y_{ij} = 0$ when they don't match. As a result, when $y_{ij} = 1$, which means v_i, u_j represent the encoded vector of the same image and prompt. Then, the larger the dot similarity is for these two vectors, the smaller the loss is, thus we can enlarge the similarities of the vectors generated by the image and corresponding prompt. Also, for the case when $y_{ij} = 0$, then larger the similarity between the vectors v_i, u_j is, the larger the loss function is, so by minimizing the loss, we can minimize the similarity between the vectors that are generated by different images and prompts as well.

3.1.3 Language Modeling Loss

We often use cross entropy loss as the language modeling loss. The loss function is as below:

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} y_{ij} log(p_{ij}) \tag{9}$$

Here, N represents the total length of our vocabulary table, and y_{ij} is the indicator of the j^{th} word in the i^{th} sample. And p_{ij} is the corresponding probability predicted by the model of the j^{th} word in the i^{th} sample.

3.2 Fine-tuning Approaches

3.2.1 The Basic Fine-tuning Task

In the basic fine-tuning, data are trained based on the BLIP pretrained model, using the Winoground dataset. In terms of programming structure, we locked the parameters in the Image Encoder and only trained the Prompt Decoder part of the model. We also employed a residue connection between the attention layers and FFN inside the transformer blocks to improve the performance of the model. In this way, only parameters in the prompt decoder was fin-tuned with a support from the residue connection. See the Figure.1 .

3.2.2 Problem encountered in the basic fine-tuning task and Solutions

The basic fine-tuning task required significant computational resources and memory, which led to long training times and inefficiencies. To train the model for 50 iterations, we had to use a TPU with 35G of system RAM from Google Colab, and it took 7 minutes to complete the training. The high resource consumption made the training process less efficient. At last, we trained and stored 6 models in addition to the originally inserted pretrained model, which are models fine-tuned for 5,10,20,30,40,50 epochs respectively. Then we evaluated the performance of each model on the testing set by comparing their testing loss (LM loss), as well as comparing the generated prompts.

During this process, the major discovery we encountered was a severe text degeneration problem in the results. Figure 2 and 3 illustrate this problem after training for only 10 epochs. After training for 20 epochs and above, the text degeneration problem become more severe and the prompts appear to be repeating almost the same sentence for all the testing samples. For instance, after training for 40 epochs, 90 percents of the testing cases are all captioned by a single sentence 'the confident doctor's stethoscope is around their neck, and their name tag is over their jacket', which is a typical representative case of text degeneration. And Figure.4 and Figure.5 show the training, validation and testing loss respectively. It needs to be mentioned that, by viewing the training, validation and testing curves, we observe that the text degeneration problem can not be classified as a conventional overfitting problem, for as the training loss reduces along with the increase of training epochs, the testing and validation loss reduces monotonely as well.

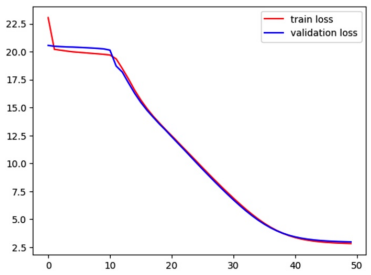

Figure 4: Basic BLIP Fine-tuning Training and Validation Loss

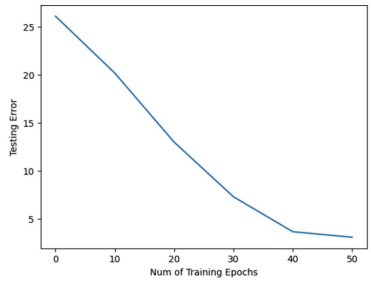

Figure 5: Basic BLIP Fine-tuning Testing Loss

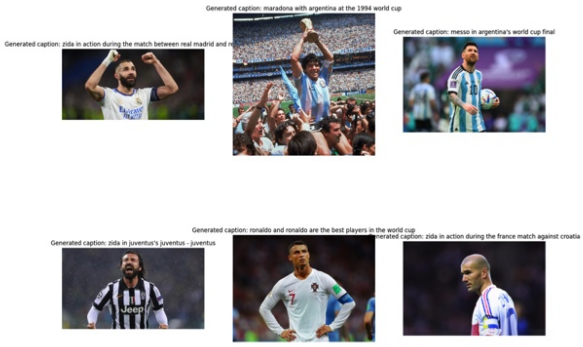

Figure 6: The Precise Caption of the Model After Fine-tuning for 10 Epochs

To address the text degeneration problem, we explored two solutions. First, we used a dataset consisting of highly correlated images and prompts, which resulted in a positive fine-tune training with more accurate results. This can be seen in Figure 6. In the figure, we have the prompts generated by the model fine-tuned for 10 epochs, and as it is shown in Figure.4, the model can successfully capture most of the information provided by the image, although contain some mistakes in terms of footballer's names and the specific year. Then, after the further training of 50 epochs, we found that the model can perfectly caption the image as the training loss is reduced to a very low level.

However, this solution was restricted to small tasks. This isn't a universal solution for general datasets, which are more likely to contain uncorrelated images. Based on the adaptor model, we revised the model, which would be explained in 3.4. Adaptor model could solve the degeneration problem in the loose dataset while reducing the computational consumption as well.

3.3 Adapter

3.3.1 Adapter model

Adapter Training [4] is a method that enables more efficient use of pre-trained models by selectively adding new parameters to the pre-trained model for each new task, instead of fine-tuning the entire model for

each task. This approach reduces the computational resources and training time required for each task, while still achieving state-of-the-art performance on various NLP benchmarks.

3.3.2 Adding adapter

In our model, we add adapters after the feed-forward neural network in each transformer block in BLIP, as shown in Figure 7. A total of 48 adapters are added (12 in the image encoder transformer block, 36 in the parallel prompt encoder and decoder transformer blocks). Each of them is a residual neural network which contains two linear layers, a normalization layer and an activation function. The input dimension and the output dimension are the same, all of (,768). During training, all parameters in the pre-trained model, except those in the adapter, are frozen to ensure computational and parameter efficiency. We also trained and stored 6 models apart from the originally inserted pre-trained model. These 6 models are trained for 5, 10, 20, 30, 40, 50 epochs respectively. The training, validation and testing loss(LM loss) are shown in Figure.8 and Figure.9.

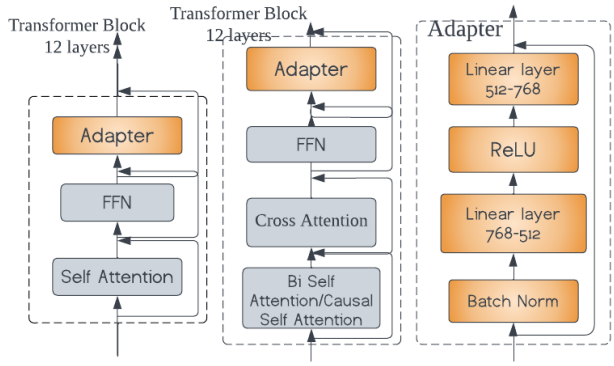

Figure 7: From left to right: The transformer block in image encoder added with an adapter, the transformer block in text encoder and decoder added with an adapter, the structure of the adapter.

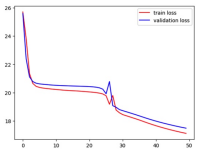

Figure 8: Training and Validation Loss After Adding Adapter

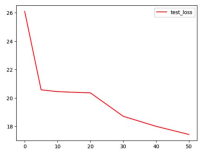

Figure 9: Testing Loss after Adding Adapter

4 Result

By comparing the training results of the model in the basic fine-tune and the model added adapters, we discovered some interesting results in the following few perspectives.

4.1 LOSS

Compared to the model with adapters, the training and testing loss of the previous model are obviously lower. The lowest training and validation error of the previous model after fine-tuning 50 epochs are all around 2.5 and the testing error was reduced to 3.08. However, for the adapted model, the training and validation error are still over 16 after training for 50 epochs, and the final testing error is 17.42. This phenomenon is not hard to explain, for in the adapted model, apart from the added adapters, the parameters are all frozen, so the total number of trainable parameters is much smaller than the original model, thus, the model complexity is lower and it tends to have higher bias and loss.

4.2 Training Consumption

Our adapted model show the basic fine-tuning model has 161,323,580 trainable parameters, whereas with the adapter, the number is significantly reduced to

20,033,280. Moreover, the adapter approach saves memory as it was trained for 50 epochs only using a V100 GPU with a maximum GPU memory of 15G, and the training time was only 42 seconds. In contrast, we trained of the BLIP model without adapter using TPU provided by Google Colab for more than seven minutes to finish the same number of training epochs.

4.3 Reduce of Text Degeneration

Additionally, the adapter approach effectively avoids the problem of text degeneration, ensuring that the generated prompts even after 40 epochs of training are not degenerated. We show in the following example:

And as apart form the loss function, which is the LM loss, the main metric to evaluate the model performance to measure the caption is to compare it to the ground truth caption and the caption provided by the pretrained BLIP model. Further collection of the testing results are presented in the attatched codes.

In conclusion, after 5 iterations, due to text degeneration problem, the captions generated by the model without adapter become unreadable. Meanwhile, for the model with adapter, even after training for 40 epochs, the model still can produce valid sentences.

4.4 Comparison with Unfinetuned BLIP Model

We conducted a comparison between the generated texts from the adapted model and the original model. The adapted model was trained for 5 or 10 epochs as we found that the models trained for more epochs are influenced by the text degeneration effects to some extent, resulting in a worse performance than the non-adapted model. After performing an artificial evaluation, we noticed that both the adapted model outperforms the non-adapted model on more test cases. Overall, both models showed a decent performance on human evaluation, and the adapted model had a lower loss than the original model due to its training. We provide some examples in Figures 10 and 11.

Figure 10: Unfinetuned BLIP model: a man and woman are playing with a cardboard box
Adapted BLIP model (5 epochs): a man and woman moving into their new home
The unfinetuned model only captured the exterior meaning of the image, but our adapted model deduced the high possibility that the man and woman are moving to the new home by the suggestion from their facial expression and surrounding packing boxes.

5 Discussion

5.1 Project Summary and Discoveries

One interesting observation from our studies is that in some occasion, the loss isn't the only metric that determines the performance of a model. For instance, in our studies, the LM loss, which measures the generation accuracy by word, reduces to a lower level in the model without adding adapter, however, the text degeneration problem is actually higher in the lower loss case. This implies that when measuring the performance of a language generation model, apart from the general metrics like generation accuracy, likelihood, and distance between generated text and ground truth, other factors such as the readability of sentences and their inner logic also play a significant role. To evaluate these factors, we need to conduct human evaluations.
Also, the impressive performance of the adapted model can be considered as the major discovery of this project. In the evaluation metric of the LM loss,

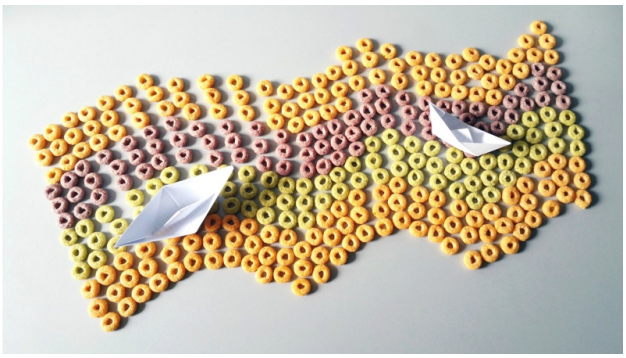

Figure 11: Unfinetuned BLIP model:a white paper boat
Adapted BLIP model (10 epochs): a bunch of cereals with a paper airplane
The unfintuned model only recognized the paper boat in the image, but the 10-epochs finetuned model recognized more objects, such as the cereals. Although it misrecognized the paper boat as paper airplane

the model outperforms the original pretrained BLIP model. And by artificial evaluation, the generated text by the adapted model outperforms the finetuned model without adding an adapter. Moreover, the adapted model also performs well in the sense of computational and parameter efficiency. With only 1/8 of the parameters as the basic finetuned model, the adapted model has higher performance on the accuracy of generating prompt compared to the ground truth, as well as a descent performance on avoiding the problem of text degeneration.

5.2 Limitations

1. Due to the limit of computational power and the resource of dataset, the current model is only trained on a image-text dataset from facebook which contains only 800 samples.
2. Due to time constraints, the evaluation metrics for this project only included the LM loss defined in the model and an artificial evaluation procedure. Incorporating additional evaluation methods for text generation tasks could result in more convincing results
3. In this project, we only tried to add one kind of adapter, which is the two layer residue neural network to the original model at some specific places. We haven't explored the ideas of applying other kinds of adapters and do some further adaptations to the model structure.

5.3 Further Directions

1. Try to conduct the research using more diverse datasets.
2. Apply more model evaluation methods for the text generating tasks to make the result more convincing.
3. Explore some further adaptations to the model structure.
4. Try to find other methods of dealling with the problem of text degeneration, such as adding random noise to the model or reduce the unnecessary complexity of the model.

6 Related Work

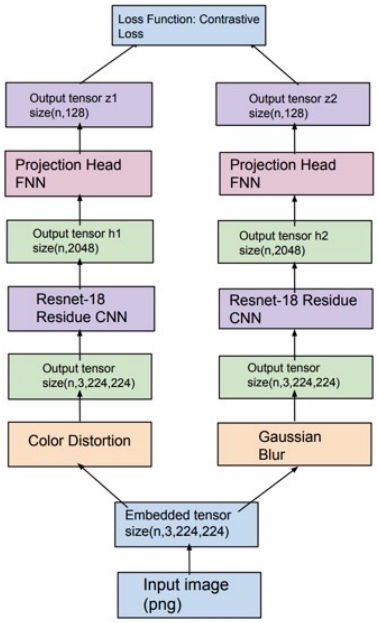

Figure 12: Framework of SimCLR Self-Supervised Contrastive Learning Image Encoder. The Neural networks marked with same color share the same parameters.

6.1 SimCLR Self-Supervised Image Encoder

In order to further explore the underlying structure of self-supervised learning models and contrastive learning image encoders, we build a SimCLR [5]model from scratch. The related code attached in the in the assignments. The loss function of SimCLR model is also a contrastive loss:

$$y = -\frac{1}{2N}\sum_{i=1}^{2N}\sum_{j=1}^{2N} y_{ij} log\sigma \tag{10}$$

Where

$$\sigma = \frac{exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N}[k \neq i]exp(s_{ik}/\tau) + [k \neq j]exp(s_{jk}/\tau)} \tag{11}$$

The idea of this loss function is similar to the contrastive loss mentioned before in the image encoder in the BLIP model. The main structure of the model is shown in Figure.12 The main frame work of the model is to first use two methods to do data augumentation: Gaussian Blur and Color Distortion. Then, we feed the output of data augumentation. Then, we feed the output in the Res-18 residue network and the feed forward projector, which projects the vector to a lower dimension. Then, we use contrastive loss to calculate the similarity of the result vectors, and our training objective is to maximize the distance between the vectors generated by different images and minimize the distance between the vectors generated by the same vector.

7 Reference

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 6000–6010.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] X. Li, Z. Gan, Y. Cheng, Y.-C. Wu, J. Liu, L. Carin, and L. Li, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," *arXiv preprint arXiv:2004.02149*, 2020.

[4] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," *arXiv preprint arXiv:1902.00751*, 2019.

[5] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv preprint arXiv:2002.05709*, 2020.

8 Related Code

Related Code