Adapter Module for Computer Vision Transformer

Minjae Lee
Georgia Institute of Technology
mlee745@gatech.edu

Tejas Kandath
Georgia Institute of Technology
tejas-kandath@gatech.edu

Jingyang Ke
Georgia Institute of Technology
jingyang.ke@gatech.edu

Abstract

Since their development in 2020, transformers have taken not only NLP, but also CV world by storm. On many popular, large datasets, Vision Transformers (ViT) have repeatedly outperformed previously considered state-of-the-art models like convolution-based ResNets. Since then, large pre-trained ViT's have become open-source and the pre-trained weights are available to the public for use in downstream tasks; however, the significant cost of fine-tuning remains. Fine-tuning is parameter inefficient as new models are required for every task. Houlsby et al. introduced a solution for NLP transformers in the form of adapter modules that attains similar fine-tuning results while only using training a fraction of the parameters and retaining high degrees of overall parameter sharing. After testing various configurations, hyper-parameters, and training methodologies, we were unsuccessful in replicating the expected results of adapter modules. Further investigation is required before discarding adapter modules for fine-tuning ViT's.

1. Introduction

Adapter modules have found great success within the domain of NLP. However, can it be generalized to other domains of interest within machine learning? We used adapter modules to fine-tune a state-of-the-art, pre-trained computer vision (CV) model for downstream domain-specific tasks using fewer computational resources. At present, state-of-the-art CV models are based on the vision transformer (ViT) architecture. We attempted to replicate the success of adapter modules in downstream fine-tuning using a pre-trained ViT to discover if it has potential for general use within machine learning. The present paradigm for state-of-the-art results and performance is to transfer pre-trained models for downstream tasks. However, the problem remained of having to fine-tune the entire architecture of pre-trained models. By fine-tuning, pre-trained models quickly forget previous tasks and are no longer usable for prior tasks. As a result, for a large suite of tasks, it is necessary to store fine-tuned weights for each, independent task. Additionally, it is computationally expensive to retrain the entire set of weights for very deep models. Houlsby et al [3]. sought to address the problem of fine-tuning pre-trained models such that they are extensible and compact by introducing adapter modules. Adapter modules are simple linear layers that can be inserted into transformer blocks and trained so that all weights of the original, pre-trained network remain fixed, but the new model is fine-tuned for a new task. With adapters, it became possible to fine-tune using a fraction of the parameters for a marginal cost in terms of performance. Houlsby et al. were able to get near state-of-the-art results using their adapter modules on Bert and NLP tasks. The question therefore remains, are adapter modules the general answer for the fine-tuning problem? Can they be applied to domains outside of NLP where transformers are dominant?

2. Methodology

2.1. Design of Adapter

An adapter module is constructed as 2 linear layers. The first down-projects to a bottleneck dimension and the second up-projects back to the dimension of the input to the adapter. The projection layers are functionally linear layers with output dimension of the first layer and input dimension of the second layer being the bottleneck dimension. In between these two linear layers, a nonlinearity is included. Additionally, droupout layers are added after each linear layer for training regularization purposes. Finally, there is a residual connection from the original input that is added to the adapter output. It was found that the residual connection is necessary to prevent over-fitting in later testing.

Each adapter module had a bottleneck of 256 since anything smaller tended to be incapable of learning from the training data. Leaky ReLU was used as the nonlinearity in each adapter layer due to its robustness; however, further investigation is needed to find the optimal nonlinearity. The dropout probability was set to .2 as this was the best value found by the original ViT team as well as the Houlsby et al. Using additional linear layers in between the two projection layers was found to not improve results and was therefore discarded.

2.2. Instantiation of Network

Due to limitations on computational resources, a pre-trained ViT-Base (google vit-base-patch16-224) was used instead of the largest possible pre-trained version. As implemented by Houlsby, we inserted the adapter module serially, prior to the two residual connections in each transformer block. After that, the total structure of each block became (layernorm before) → ViTAttention → Adapter layer → residual → (layernorm after) → ViTIntermediate(Linear + GELU activation) → Adapter layer → ViTOutput(Linear + dropout). The overall structure is similar to the transformer encoder image in Figure 1, but we inserted two adapter layers, and ViT intermediate and ViT output function as the MLP part of the diagram.

All layers of the original ViT model were frozen except for the normalization layers in the ViT transformer blocks, the classifier layer, and the adapter modules. For each dataset, the model was trained until early stopping after the patience of 10 was reached with the validation data. Finally, the model weights are reloaded to the saved checkpoint of the weights that performed best on the validation data and the model is evaluated on the test data for final results.

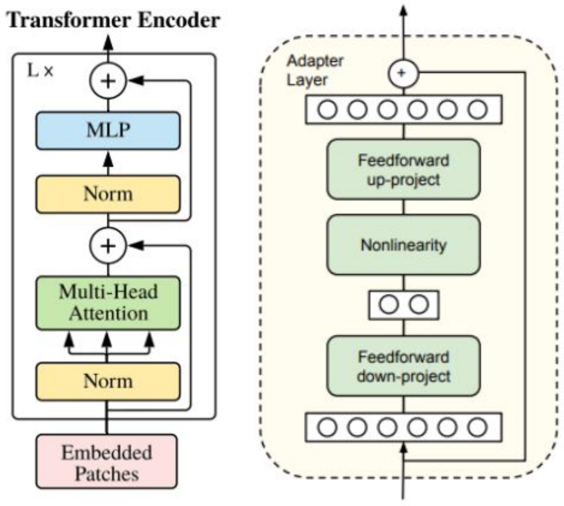

Figure 1. Overall Structure for reference [1, 3]

2.3. Experimental Settings

We optimize using Adam at its default settings except for a fixed learning rate of .0001. Later on we discuss the potential need to tweak other Adam settings such as the weight decay rate. The batch size used was 64 across the board as it was the largest batch size that could be loaded given the constraints we had on training resources. The model was trained using a standard RTX 2080 Ti with 12 gigabytes of memory. Due to time constraints, we were not able to perform a hyper-parameter sweep to find optimal parameters. The parameters chosen were based primarily on Houlsby et al.'s paper and supplemented by the original ViT paper as well. In the future it is necessary to be more judicious about the hyper-parameters.

2.4. Datasets

Each dataset we're using differs in the number of data points by one order of magnitude or so to investigate scaling properties of adapters (i.e., if the model was pre-trained on a very large dataset, will the fine-tuning dataset have to be similarly large?)
Oxford IIIT Pets (7.3k images, 37 classes) : Highly-varied dataset with variety of poses, lighting, and scale. Medium number of classes and small magnitude dataset.
CIFAR-100 (60k images, 100 classes) : Version of CIFAR-10 with greater granularity in terms of classes. Similar image sizes and fairly diverse dataset. Medium magnitude and large number of classes.
PatchCamelyon (330k images, 2 classes) : Images of tissue samples that may include cancer. Classification is binary for whether or not cancer is present. Large magnitude and minimal number of classes.

2.5. Data Preprocessing and Pipeline

The input image size is fixed to 224 by 224 in the pre-trained model. It corresponds to and 7 by 7 grid of patches each 32 by 32 pixels. Due to the fact that image sizes in each of the datasets is highly varied both between them and within them, each image must be fit to 224 by 224 and then normalized. HuggingFace provides a method for that called ViTFeatureExtractor that uses bilinear sampling to downsize or upsize images to the correct size and then normalizes the values of the tensors.

The data pipeline involves loading all of the images in a dataset including the train, validation, and test. Then the data is run through the feature extractor to be normalized. The normalized data is fed to a class that shuffles and batches it, which is then passed to the training loop, the validation accuracy calculator, and the test method with the necessary suite of metrics. Currently, the entirety of the dataset is loaded on to the CPU because the GPU we trained on did not have enough memory to load some of the larger datasets. As a result, during the training, for each batch, the batch is loaded on to the GPU, the model backpropogates and then all of the values associated with the batch on the GPU are deleted for memory management purposes. Significant amounts of time and effort were spent on debugging memory management so that the model could run on larger batch sizes and use the whole training dataset.

3. Results and Discussion

For each dataset, we used the entirety of the training set. Between epochs, the data was shuffled and batched again. After every epoch, the model was run on the validation set. If after an epoch the validation loss did not improve, we added to the early stopping counter. After the counter reached the patience of 10, we stopped the training immediately. Finally, the weights from the best validation loss were reloaded and ran on the test set to get finally test accuracy. For the PatchCamelyon dataset, an additional metric of ROC AUC score was calculated as is standard for the dataset. The metric is for measuring false-positive and false-negative rates compared to true-positive and true-negative rates.

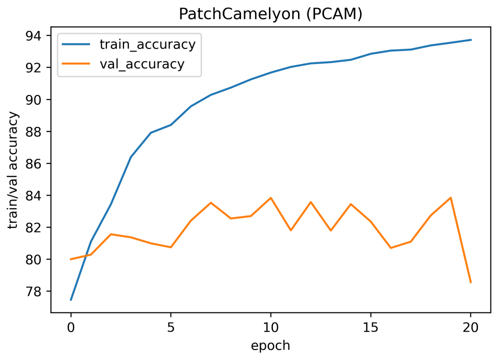

Figure 2. Learning curve for Patch Camelyon Dataset.
Test accuracy 79.18%
Test ROC AUC score 0.7917

Across the board, the results we obtained were underwhelming. Comparing against competitive results for each dataset, we found that we were incapable of replicating or getting near state-of-the-art results. As the dataset magnitude increases, performance appears to increase as well. This is a noted property of ViT's as their lack of inductive bias tends to cause them to underperform on datasets smaller than 1 million data points.

However, it appears as if the number of classes in the data also affects performance significantly. From the first epoch, the PCAM dataset has an accuracy of 80% already (since there are 2 classes, we expect 50% accuracy if simply guessing labels). Testing this on another dataset that had around 1000 training examples and 3 classes, the same

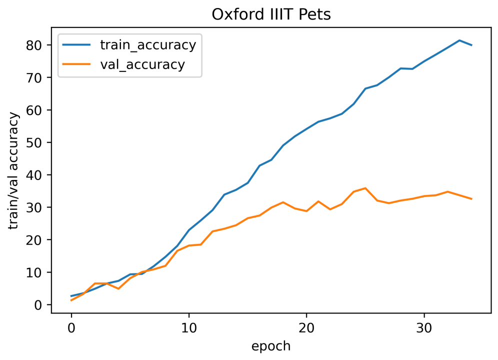

Figure 3. Learning curve for Oxford IIIT Pets Dataset.
Test accuracy 36.20%

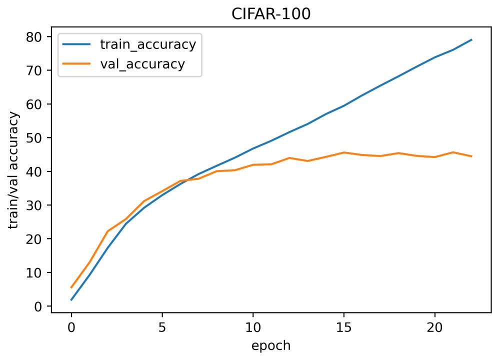

Figure 4. Learning curve for CIFAR-100 Dataset.
Test accuracy 44.59%

property was observed. It appears as if the number of classes has a significant impact on results, more so than the magnitude of the dataset. Nevertheless, across the board, the model overfits the training data almost from the onset. We attempted many regularization techniques to limit overfitting; however, we were never able to successfully train the adapters. Regularization techniques employed included dropout (probabilities of .1, .25. 4), batch normalization, and lowering the learning rate. Finally, the quality and consistency of the datasets also significantly affected the results. The Oxford Pets dataset has no uniform size of images, including incredibly wide or tall images, and as a result, the transformer trained poorly.

Future experimentation is required before discarding the use of adapters in ViT's. Firstly, training paradigms for transformers, and more specifically ViT's, must be established so that poor training methodology can be eliminated as an explanation of the results. Future work should also investigate if embeddings must be retrained and how they should be retrained. The order for where adapter modules are optimally placed in the transformer blocks should be explored further. Finally, data normalization and training regularization techniques should be further investigated conclusively.

4. Attempts at Future Experimentation

Some interesting regularization techniques to attempt in the future includes the effects of gradient clipping at global norm 1 as the original ViT team did, initializing the weights using a normal distribution instead of a uniform distribution. Houlsby et al. found that weights initialized normally centered around 0 allowed the adapter module to initially start as effectively an identity matrix. They also found that if the standard deviation of the distribution they sampled from was too high, $>$ 1e-2, the model failed to train from the start. Additionally, the original ViT team used a high decay rate in Adam, .1, during training to get stable results (it was found that a high decay rate negatively affected performance even during training with further experimentation). Perhaps the decay rate might reduce overfitting in future experiments. Finally, changing the learning rate from a static value to instead warm-up for significant periods before again decaying might improve stability of the momentum vectors in Adam and further reduce overfitting. The original ViT team used a warmup of 10k steps.

After implementing the gradient clipping, weight initialization, and warm-up for 1 epoch (700 steps) we achieved the following results on CIFAR-100. The improvement is abundantly clear, the validation accuracy peaks after only 2 epochs. This demonstrates the findings of Houlsby et al. that adapter modules are capable of rapidly fine-tuning using a fraction of the parameters. Looking at top results all time for CIFAR-100, our test accuracy would be the 23rd best ever achieved (CIFAR) [4], falling only 4% less than the best results the original ViT team achieved. A 4% marginal difference is excellent considering the fact that the original team trained all of the parameters of the ViT.

Running the improved regularization techniques on PCAM, we were only able to get best results after just one epoch, since early stopping kicked in after the second epoch. The train accuracy reached 96% and the validation accuracy was 89.6%. A test accuracy of 88.5% and ROC AUC score of 0.885 is the 10th best result ever achieved on the dataset (PCAM) [2].

From these results, it is clear that improved regularization techniques unlock the potential of adapter modules in ViT's. With only one to two epochs, our model achieved top results all time. By further improving training regimes, it is very likely that adapter modules could become very useful in computer vision tasks and even near state-of-the-art results.

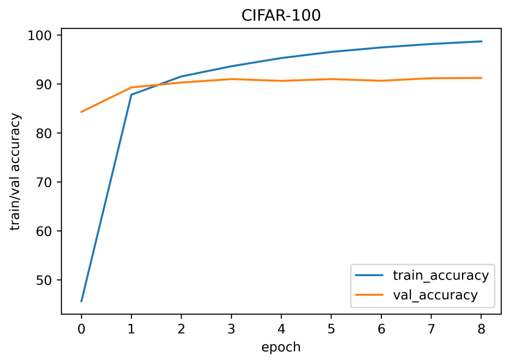

Figure 5. Learning curves for CIFAR-100 using improved regularization methods.
Test accuracy of 90.5%

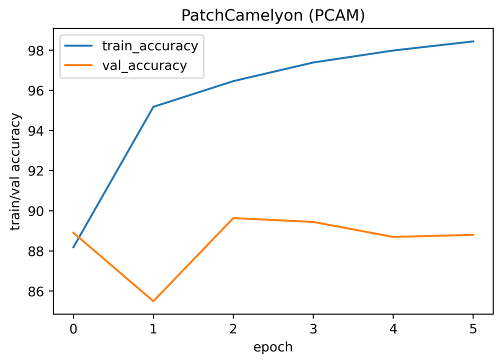

Figure 6. Learning curves for PCAM using improved regularization methods.
Test accuracy of 88.5%
Test ROC AUC score 0.885

5. Contributions

We met with each other in frequently and wrote all of the code synchronously. For clarification, we hereby attach individual contribution table.

Name	Contribution
Tejas Kandath	Suggested initial subject and idea Contributed to design and implementation of the entire model
Minjae Lee	Built adapter layers and model Implemented experiment-related functions Managed write-up, report, and repository
Jingyang Ke	Combined adapter layers into ViT layers Created data process pipeline and test environment Conducted all final experiments

Table 1. Contribution table

References

[1] Nikolas Adaloglou. How the vision transformer (vit) works in 10 minutes: an image is worth 16x16 words. *AISUMMER*, 2021. 2

[2] Grand Challenge Camelyon. https://camelyon16.grand-challenge.org/results/. 2022. 4

[3] Neil Houlsby et al. Parameter-efficient transfer learning for nlp. 2019. 1, 2

[4] Image Classification on CIFAR-100. https://paperswithcode.com/sota/image-classification-on-cifar-100. 2022. 4