Vision Transformer on Small Datasets: Investigating Techniques for Improved Accuracy and Reduced Training Time

Junwen Yu junwenyu@umich.edu
Hanlin Bi hanlinbi@umich.edu
Zihao Ye zihaoye@umich.edu
Ziang Li ziangli@umich.edu

1. Introduction

1.1. Problem Definition

In this project, we explore methods of applying vision transformer architecture (ViT) - an approach different from the traditional CNN - to address the image classification task. In addition to implementing the original ViT architecture, we make several modifications, attempting to improve the model's accuracy and performance on small datasets.

1.2. Approach

We start from implementing the ViT model described in *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale* [3]. We refer to an existing codebase (*Vision Transformer from Scratch* [6]) as our code skeleton, and implement the details according to the ideas described in the paper. We mainly researched follow-up works on vision transformer, implemented add-ons and measured accuracy and training time.

1.3. Motivation

Although the transformer architecture has great potential, it typically has lower accuracy than CNN when the dataset and model scale are small. Moreover, the initial vision transformer [3] applies huge training resources and transfer learning. Therefore, we aim to improve its accuracy on small datasets, and accelerate the speed of training.

2. Background

2.1. Related Work

In the field of computer vision (CV), convolutional neural networks (CNNs) have been quite popular, mostly because of its high accuracy, simplicity, and efficiency. Although transformer structure has been dominant in natural language processing (NLP), it may not be a good fit for CV tasks, due to its lack of properties like the inductive bias that CNNs naturally own. Nonetheless, its outstanding advantage of large receptive fields still indicate its great potential.

The Transformer architecture was originally proposed in the paper *attention is all you need* [8]. It introduces the attention machanism, which is able to deal with dependencies between tokens in a long range and utilize parallel computation.

Before ViT was proposed, there had been many works trying to apply the attention machanism to address CV tasks. For example, in *Exploring Self-attention for Image Recognition* [9], two different sorts of self-attention are evaluated. However, few of these previous works were able to leverage the potential of vanilla transformer, as in *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale* [3]. In this paper, images are divided into patches and embedded, then sent to a pure transformer architecture. It demonstrates the great potential of transformer architecture in CV tasks.

There have been many followups that aim to improve the efficiency of ViT. For example, several original authors of ViT proposed *Better plain ViT baselines for ImageNet-1k* [1], which shows methods to greatly improve the performance. Also, there are modifications that focus on improving the accuracy on small datasets, like *Vision Transformer for Small-Size Datasets* [5].

2.2. Required Backgrounds

To fully grasp the concepts and methodologies discussed in this project, readers should have a foundational understanding of both machine learning and computer vision. Key concepts include neural networks, specifically convolutional neural networks (CNNs) and the principles of deep learning. Familiarity with the transformer architecture, is also crucial as it forms the basis for the Vision Transformer (ViT) architecture.

3. Methodology

In the following subsections, we describe our work and implementations of the basic ViT structure, as well as several enhancements.

3.1. Vision Transformer

Our project starts with implementing the Vision Transformer described in the paper *An Image is Worth 16x16*

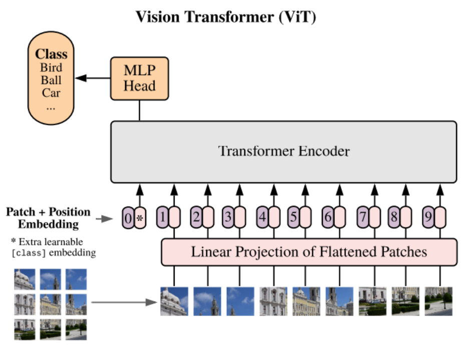
Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
Linear Projection of Flattened Patches

Figure 1. Model overview, cited from the paper

Words: Transformers for Image Recognition at Scale [3]. We refer to the structure of an existing implementation *Vision Transformer from Scratch* [6]. Specifically, the class hierarchy (skeleton.py), training, and data loading parts are reused, and the implementation details for each layer is our own work.

The overall structure can be described by the figure from the paper.

3.1.1 Patch Embedding

As described in the paper, the first step is to transform the input images to patches, so that they can match the input format of the transformer encoder. Specifically, we need to convolve the each image into several patches, then flatten the resulting 2D array and use a linear layer to project them into the latent vector size D. In practice, we use a *nn.Conv2d* layer from torch and several matrix transformations to achieve this.

This step can be described like this:

$$\mathbf{x} \in \mathbb{R}^{H\times W\times C} \rightarrow \mathbf{x}_p \in \mathbb{R}^{N\times D}$$

, where $\mathbf{x}$ is the input image and $\mathbf{x}_p$ represents the patches.

3.1.2 CLS and Position Embedding

Next, we embed a learnable CLS token to the beginning of the patches. This token will eventually encode the necessary information for prediction and will be passed to the final classifier. Then we add position embedding to each patch, so that the transformer encoder can better utilize the positional information. In the basic implementation, we follow the paper and use learnable embeddings.

$$\mathbf{z}_0 = [\mathbf{x}_{class}; \mathbf{x}_p^1; \mathbf{x}_p^2; \cdots; \mathbf{x}_p^N] + E_{pos}$$

3.1.3 Transformer Encoder

The transformer encoder's structure can be described by this figure from the paper:

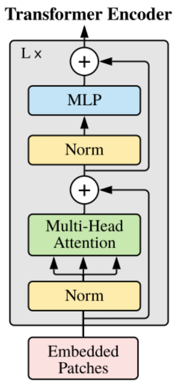

Figure 2. Transformer encoder overview, cited from the paper

The transformer encoder in Vit is essetially composed of identical blocks, each containing norm layers, multi-head attention block, and an MLP block. It also utilizes skip connections, which are simply additions, to mitigate gradient vanishing or exploding.

In our implementation, we use layers from the torch library to build the blocks. Specifically, we use *nn.LayerNorm* as a Norm, *nn.MultiheadAttention* as the multi-head attention block, and *nn.Linear* and *nn.GELU* to build the MLP block.

3.1.4 Classifier

Finally, we extract the CLS token and pass it to a classifier. The paper uses an MLP head for pre-training. Since our model and dataset are both small, we directly use a fully-connected layer as the classifier.

3.2. Enhancements

3.2.1 Shifted Patch Tokenizer Embedding

Under the context of training with small dataset, we want to use as much detail as possible for model complexity.

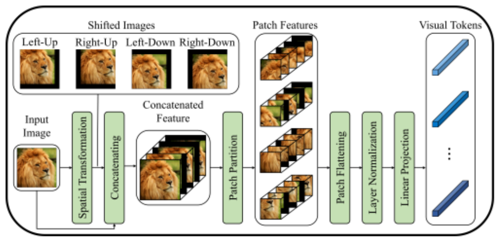

Figure 3. Architecture for Shifted Patch Tokenization

Thus, in order to improve the diversity of our dataset, we need to apply data augmentations on the training data. Instead of using traditional patching embedding, we chose one of the most robust approaches, Shifted Patch Tokenization (SPT) [2], for model to capture more varied and comprehensive features from limited data.

The forward method applies predefined shifts from top, bottom, left, and right in to the input image. We chose the circular padding because circular padding maintains the spatial continuity of the image. When the image is shifted, circular padding allows the pixels that go out of one boundary to come back in from the opposite side, creating a seamless transition. [2] This method ensures that all parts of the image are used effectively, avoiding the introduction misleading edges for the model, increasing the robustness and maintaining the integrity for the image content. [5] Then, the shifted images are concatenated with the original image and then processed through a normalization and linear transformation pipeline. This technique effectively increases the dataset's diversity without needing additional data, improving the model's ability to generalize and recognize features across different spatial configurations. This approach perfectly aligns with our data limited setting.

3.2.2 Local Self Attention

Since we focus on the model for small datasets, we want to make optimizations on the model corresponding to this property. The motivation is to have two layers: the local one that captures fine-grained details within a certain window, and a global one that captures broader dependencies.

Inspiring from the paper, we decided to implement the LocalSelfAttention algorithm and embed it into our own transformer network. In our costom built LocalSelfAttention module, the forward method begins by normalizing the input tensor block x using layer normalization to stabilize the training dynamics. The tensor is then passed to the prepare qkv method, which uses a linear transformation followed by rearrangement to split it into queries (q), keys (k), and values (v). We took the same mapping from the implementation from the paper [5] for qkv attention head split. After that, a dot product is calculated between queries and keys, scaled by a learnable parameter temperature, and a diagonal mask is applied to avoid self-attention, setting these values to negative infinity to diversify feature learning. The softmax function is applied to the resulting matrix to obtain attention weights, which are then regularized via dropout. These weights are used to compute the weighted sum of values, which is then rearranged to combine features from all heads. At the end of the forward pass, it passes through a linear output layer to produce the final output.

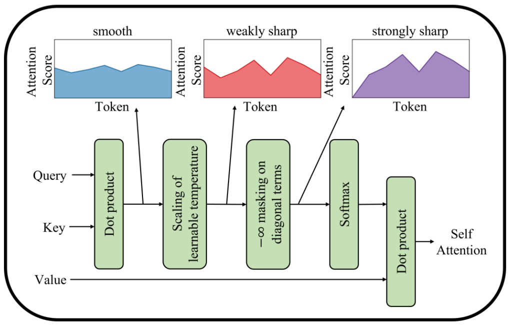

Figure 4. Architecture for Local Self Attention

Then we utilized the residual connection between this layer and the global attention block layer in our encoder. By doing this, sequence of operations efficiently captures both local and global dependencies within the data, enhancing the model's ability to focus on relevant features in the context of small dataset.

3.2.3 Encoder: Patch Merger

To improve the performance of encoder, we inspired from the idea of patch merger. It aims to reduce the number of tokens or patches processed by the network by selectively merging them based on learned attention mechanisms. [7]This approach involves using a set of learned queries to compute attention scores with respect to the input patches, patches are then combined according to these attention scores, effectively reducing the sequence length and focusing the model's capacity on the most significant areas of the input.

On the base of this optimization approach, we enhance the merger structure with our work: The scale is initialized based on the dimension of the input, but as a learnable parameter. This allows the model to adjust the scaling of the similarity scores dynamically during the training process. This learnable scale gives the model flexibility to find an optimal scaling factor for the dot-product attention. This adaptation can help in balancing the magnitude of gradients during back propagation, potentially leading to faster convergence and improved stability.

Secondly, A bias term is added to the similarity scores before applying softmax. This term allows adjustment of the raw dot-product values, providing an additional degree of freedom in the attention mechanism. The bias can help manage the dynamic range of the attention scores, preventing extremely large or small values that might lead to numerical instability or gradient issues. It allows the model to effectively learn to ignore or pay more attention

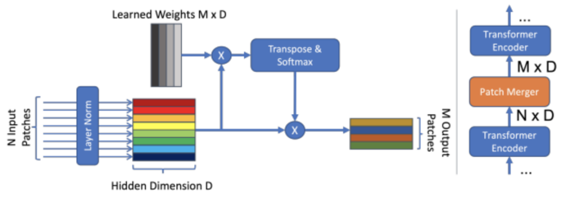

Figure 5. Patch Merger Architecture

to certain patches if necessary, enhancing the flexibility and capability of the model to focus on relevant features.

Finally, a Dropout is applied to the attention weights after the softmax function. This introduces randomness in the attention mechanism during training, effectively regularizing the model. This can be particularly beneficial in preventing overfitting, especially when dealing with small or less diverse datasets.

In addition, we use another learnable parameter to control the temperature of the softmax operation. This temperature adjusts the sharpness of the distribution produced by the softmax. A higher temperature leads to a more uniform distribution (soft attention), which can be useful in early stages of training or tasks requiring broader context. A lower temperature results in a more focused attention (hard attention), which might be useful for tasks requiring fine-grained detail. [7]

3.3. Model Configuration and Testing Automation

To enhance compatibility and streamline result evaluation, we have implemented several improvements. Drawing inspiration from the structured approach described in [6], we have made the model fully configurable via a single JSON file. This single model file is designed to handle all scenarios; it reads the configuration file at runtime and activates specified add-ons accordingly.

Given that the model supports easy configuration across different types, we have also fully automated the testing process. This automation significantly alleviates our workload, especially as training transformer models is typically time-consuming.

4. Results

We tested our ViT with various enhancements on the CIFAR-10 dataset. We use accuracy to measure model performance. During training, time per epoch is also recorded.

For training configuration and implementation of the basic ViT, we use parameters as follows:

batch size	epochs	learning rate
256	100	0.01

patch size	embed dimension	attention layers
4	48	4

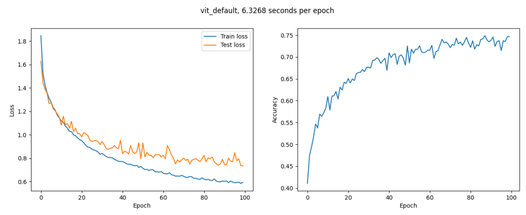

Figure 6. Results for Initial ViT

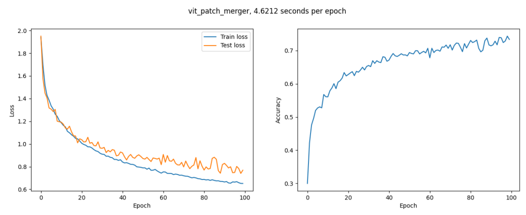

Figure 7. Results for Applying Patch Merger

4.1. Initial ViT

It can be observed that both train loss and test loss are steadily decreasing, while model accuracy is steadily increasing. Convergence is achieved and the final accuracy is around 0.75. We conducted further hyper-parameter tuning but did not reach a better result given total model complexity constrained.

This is a slightly worse result compared to CNNs, and is consistent with the conclusions mentioned in the original ViT paper [3].

4.2. ViT with Patch Merger

As the expectations mentioned in the Encoder: Patch Merger section, our modified Vision Transformer with the optimized Patch Merger should demonstrates a significant improvement in training efficiency. The result clearly supports our expectation: the training time per epoch for the ViT with the optimized Patch Merger is recorded at 4.6212 seconds/epoch, compared to the 6.3268 seconds/epoch for the default ViT setup. This represents a reduction of approximately 27% in the time required per training epoch. This efficiency gain is primarily attributed to the effective reduction in sequence length facilitated by the Patch Merger, which decreases the computational load associated with the transformer's self-attention mechanism. The optimizations introduced, including learnable scaling factors, bias adjustments in attention computations, and

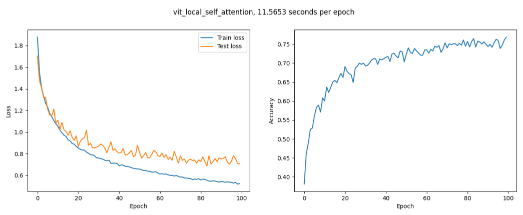

Figure 8. Results for Applying LSA

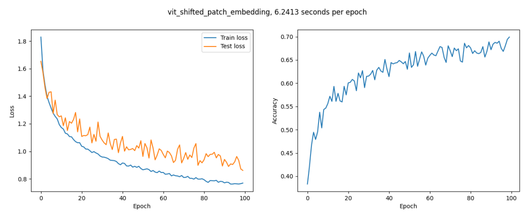

Figure 9. Results for Applying SPT

the regularization effects of dropout, contribute to faster convergence and more stable training dynamics.

Regarding model accuracy, the Vision Transformer equipped with our optimized Patch Merger maintains a performance level that is approximately equivalent to that of the default model (around 0.75). The ability of the Patch Merger to focus the model's attention on the most informative parts of the input without significant loss in overall accuracy is a testament to the success of the implemented modifications.

4.3. ViT with Local Self Attention (LSA)

There are several benefit with Local Self Attention mentioned in the paper [5]. By focusing on local patches rather than the entire input, local self-attention improves the extraction of finer details and textures, crucial for tasks like image segmentation and object detection. Additionally, local self-attention acts as a form of regularization, limiting the scope of attention to prevent overfitting and improve the model's generalization capabilities. It also offers flexibility by allowing adjustments in the attention mechanism to our classification task, thereby enhancing the model's adaptability and effectiveness in understanding spatial relationships and structures within the data. As we observed, the model finally reached 76 percent of accuracy. This is a 1 percent performance boost comparing to the 75 percent of accuracy of the plain ViT we initially ran. In addition, the accuracy is still varying heavily at epoch 100, and we expect that model hasn't converge at that point. So reducing learning rate for SGD to move in smaller step and add more epoch of training might further improve the accuracy of our model. But overall, the improvement aligns with the research paper findings, and we expect this is due to the regularization enhancement that LSA provides, since it prevents overfitting at later training stage.

4.4. ViT with Shifted Patch Tokenizer (SPT)

As we observed, there no significant accuracy improvement comparing with the result from plain ViT. However, we observed a jumping between accuracy at the later stage. This is likely due to the improved data diversity by the data augmentation done by SPT. Since the data feature became

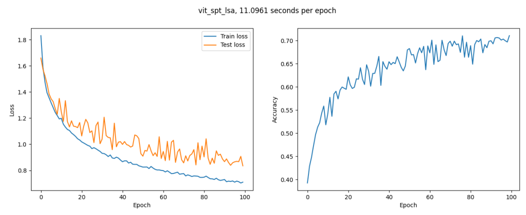

Figure 10. Results for Applying LSA and SPT

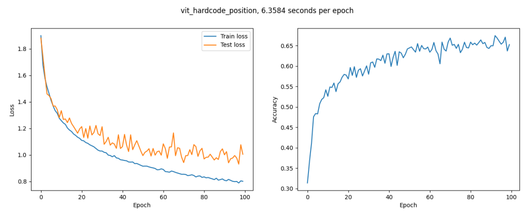

Figure 11. Results for Hardcode Positioning

more complex, the model is able to gain more variety of data and thus reduce the possibility of overfitting at later stage. Thus, I expect that lowering learning rate can make a better convergence in more epochs just the improvement we can make for tuning LSA ViT.

4.5. ViT with LSA and SPT combined

As we observed, the combined version of LSA and SPT can reach the performance of 77 percent of accuracy, which is 2 percent higher than the plain ViT accuracy. Same as what we found earlier, we observed a jumping between accuracy at the later stage. And we expect that lowering the learning rate will result into a better model convergence in more epochs of training.

4.6. ViT with a Different Positional Embedding

The original ViT paper uses learnable positional encodings. In this project, we also tried to use a hardcoded (i.e., non-learnable) sin and cos positional encoding based on the index of each patch, which was also mentioned in the paper. We expect this to achieve similar results, as described in the paper. However, it turns out that the accuracy decreases greatly.

We think this is related to the size of the dataset and the model. Since the model has only a few layers, and the number of patches is relatively small, there may not be much useful information within the positional encoding. In addition, we trained for a relatively large number of epochs, whereas the dataset is small, so there might be an overfit, in which case the learnable positional encoding may have provided some hidden information that are specifically derived from the small model and datasets. Therefore, the hard-coded encoding may work better on larger models that have a much stronger ability of generalization.

5. Conclusion

5.1. Summary and Conclusion

In this study, we explored the application and enhancement of Vision Transformer (ViT) architecture for image classification on small datasets. Our approach involved implementing the base ViT model as described in prominent literature and extending its capabilities through strategic modifications aimed at enhancing accuracy and training efficiency.

Through the course of our research, we successfully adapted the ViT architecture to better suit small-scale datasets by incorporating several novel components, including Shifted Patch Tokenizer Embedding (SPT) for improved input diversity, Local Self Attention (LSA) to capture fine-grained details, and a Patch Merger to reduce computational overhead by focusing on significant image features. These enhancements were rigorously tested against the CIFAR-10 dataset.

5.2. Future Work

5.2.1 Masked Autoencoder

A great advantage of ViT over CNN is that it can naturally adopt existing methods from transformer. In recent years, as models grow larger, the limited size of labeled datasets likely become the bottleneck. One way to mitigate this is to use *self-supervised learning*, where the model can be trained on unlabeled datasets, thus being capable of using much more data.

One way of achieving this is to randomly mask out part of the image, and train the model through reconstructing. The paper *masked autoencoders are scalable vision learners* [4] describes an effective way to apply this with vision transformer (and potentially other structures) as the backbone. It performs the reconstructing task on large, unlabeled datasets, and then fine-tune the model on labeled small datasets, achieving promising results.

5.3. Attention Visualization

We could plot the model's attention maps to some test images and examine what feature the model has learned. This can be easily done by plotting the attention outputs (which has already been used for back propagation) right before the final linear layer.

6. Appendix

See attached.

References

[1] Lucas Beyer, Xiaohua Zhai, and Alexander Kolesnikov. Better plain vit baselines for imagenet-1k, 2022. 1

[2] Lei Cheng, Ruslan Khalitov, Tong Yu, and Zhirong Yang. Classification of long sequential data using circular dilated convolutional neural networks, 2022. 3

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 1, 2, 4

[4] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. 6

[5] Seung Hoon Lee, Seunghyun Lee, and Byung Cheol Song. Vision transformer for small-size datasets, 2021. 1, 3, 5

[6] Tin Nguyen and Jose Benitez. Vision transformer from scratch. https://github.com/tintn/vision-transformer-from-scratch, 2023. 1, 2, 4

[7] Cedric Renggli, André Susano Pinto, Neil Houlsby, Basil Mustafa, Joan Puigcerver, and Carlos Riquelme. Learning to merge tokens in vision transformers, 2022. 3, 4

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 1

[9] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition, 2020. 1