Re-implementation of Vision Transformer (ViT)

Xiaoyang Song Han Liu
University of Michigan, Ann Arbor
{xysong, lhalice}@umich.edu

Abstract

Inspired by the idea of applying transformer architecture on computer vision tasks, we re-implemented the vision transformer and compared its performance with the benchmark `CNN` *and* `Res-Net` *architectures. Due to GPU limitations, we modified the model architecture and instead conducted small scale experiments. Finally, we did an in-depth ablation study to analyze the reasons of performance gap between our re-implemented* `ViT` *and benchmark models, and identified the effectiveness of the hybrid model.*

1. Introduction

Being aware of the computational inefficiency of the existing Recurrent Neural Network (`RNN`), the transformer is first introduced by Vaswani *et al.*[7] for natural language processing (`NLP`) tasks. Discarding the recurrent structures and convolutions entirely and based solely on attention mechanisms, the transformer architecture significantly improves the computational efficiency and becomes the state of the art method in many `NLP` tasks.

Even though transformers' application in computer vision is limited, many new attempts including the Vision Transformer (`ViT`) attain excellent results compared to the classical convolutional networks and the `ResNet`. In this project, we re-implemented the `ViT` model that is proposed by Dosovitskiy *et al.*[4]. However, due to the hardware and dataset limitation, we choose not to reproduce the results from their paper but instead do small scale experiments to compare the performance of our model with benchmark architecture `VGG` and `ResNet`. However, because of the low model complexity which is restricted by GPU limitation, we find a big performance gap between our model and the benchmark model. Then, we did a series of ablation experiments to explore the potential reasons for this gap.

Even though our re-implemented model does not show the superiority of `ViT` over other model, through our experiments, we are confident that pre-trained `ViT` with enough model complexity will be a promising model for image classification tasks.

2. Related Work

The idea of transformer can also be adopted to image classification tasks. In the past few years, several approaches have been experimented to apply transformers architecture in the context of image classification. However, naively applying transformer on images has quadratic cost in the number of pixels on time and memory because the self-attention mechanisms require each pixel to attend to every other pixel. To address this issue, Parmar *et al.*[5] restrict the self-attention mechanism to only attend to local neighborhoods instead of globally, while the introduction of sparse transformers by Child *et al.*[1] reduce the complexity to $O(n\sqrt{n})$ by separating the full attention computation into several faster attention operations to achieve scalable approximations for image processing.

The model that is proposed by Cordonnier *et al.*[2] is most related to the `ViT` model that we re-implemented. Their model extracts 2×2 image patches and applies full self-attention on them. However, the vision transformer, which is proposed by Dosovitskiy *et al.*[4], takes a step further and demonstrates that large vanilla transformers with pre-training are competitive with state-of-the-art `CNN` and `ResNet`. In addition, `ViT` handles the limitation of Cordonnier *et al.*[2] on small-resolution images and expands the application to medium-resolution images as well. However, the `ViT` is a relatively new architecture that is just published on June 2021 so there are not much related works and are still many things that need to be explored.

3. Method

The architecture of `ViT` is complex. Unlike in original transformer architecture where both encoders and decoders are used, `ViT` only requires a series of encoders. In general, `ViT` consists of three blocks: feature preprocessing blocks, transformer encoders, and the classification head.

3.1. Feature pre-processing

3.1.1 Patch embedding

While the traditional transformer model for `NLP` task takes in a sequence of words as input, `ViT` takes in a sequence

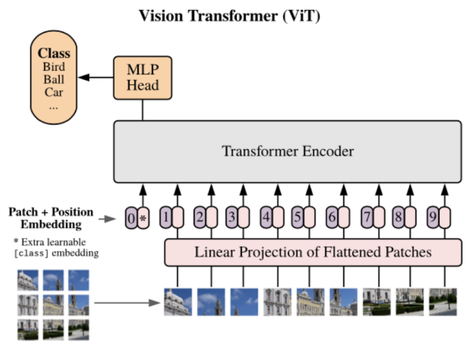
Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
Linear Projection of Flattened Patches

Figure 1. Vision transformer (ViT) architecture

of image patches. To handle this, a $2D$ image of size $H \times W \times C$ are divided into N different square patches of size $P \times P \times C$. All N image patches are flattened into vectors of size $P^2C \times 1$, respectively.

Then a sequence of flattened image patches (i.e. P^2C by 1 vectors) is obtained. Each of the vector in the sequence will then be linearly projected to a D-dimensional feature vector through a fully-connected layer, where D is called the latent vector size. This step is called **patch embedding**.

3.1.2 Class embedding

The ViT is mainly used for classification problem; therefore, other than a sequence of D by 1 feature vectors, an additional vector that represents the categories of the image is required. This is achieved by prepending a learnable embedding to the sequence of feature vectors, which is similar to the CLS tokens that are used in BERT[3] model. After this **class embedding**, a sequence of $(N + 1)$ D-dimensional vectors are obtained.

3.1.3 Positional Embedding

The transformer model also requires some positional information, and this is achieved through adding **positional embedding** to the sequence of feature vectors. To do this, we can simply use an 1D embedding: adding a learnable D by 1 vector to each vector in the embedded sequence.

After these three embedding procedures, an image of dimension $H \times W \times C$ is embedded as a sequence of $(N+1)$ D by 1 feature vectors, and this sequence will be the input to the following transformer encoders.

3.2. Transformer Encoder

The transformer encoder mainly consists of three blocks: layer normalization (LN), multi-head self-attention(MSA), and the multi-layer perceptron (MLP). Among these three building blocks, there are two residual connections, which boost the model performance.

3.2.1 Single self-attention (SA) block

A self-attention block receives a sequence of vectors as input and output a representation of it through attention mechanism. In ViT, denote the input for transformer encoder as $\mathbf{X} \in \mathbb{R}^{(N+1)\times D}$, with columns $\mathbf{x}_i \in \mathbb{R}^{D\times 1}, \forall i \in [1, N+1]$ and $i \in \mathbb{Z}$. Then, in a single self attention block, for each $\mathbf{x}_i$, three corresponding linear projections $\mathbf{q}_i$, $\mathbf{k}_i$ and $\mathbf{v}_i \in \mathbb{R}^{D_h \times 1}$ are computed, where $D_h = \frac{D}{k}$, where k is the number of heads that we will discuss in the next section. Writing in compact matrix form, for the input sequence $\mathbf{X}$, it is linearly projected to three different matrix:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q \in \mathbb{R}^{(N+1)\times D_h} \quad (1)$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}_K \in \mathbb{R}^{(N+1)\times D_h} \quad (2)$$

$$\mathbf{V} = \mathbf{X}\mathbf{W}_V \in \mathbb{R}^{(N+1)\times D_h} \quad (3)$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are all learnable parameters in the transformer model. After the linear projection step, the weight matrix $\mathbf{A}$ is computed in the following way:

$$\mathbf{A} = \texttt{Softmax}(\mathbf{Q}\mathbf{K}^\top / \sqrt{D_h}) \in \mathbb{R}^{(N+1)\times(N+1)} \quad (4)$$

Note that each entry A_{ij} represents how important or relevant $\mathbf{x}_j$ is when considering $\mathbf{x}_i$. This idea is the core of attention mechanism.

3.2.2 Multi-head self-attention (MSA) block

In the transformer encoder, to seek a better performance, instead of using only a single self-attention block, we can feed the input sequence into k different SA blocks and then concatenate their results. After concatenation, we linearly projected the result to obtain the final output tensor. The formula is given below:

$$\texttt{MSA}(\mathbf{X}) = \left[\texttt{SA}_1(\mathbf{X}); \texttt{SA}_2(\mathbf{X}); ...; \texttt{SA}_k(\mathbf{X})\right]\mathbf{U} \quad (5)$$

where $\mathbf{U} \in \mathbb{R}^{D\times D}$ is the weight matrix associated with the final fully-connected layer, which is a set of learnable parameters. Note that for each self-attention block, we will have a distinct set of $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$, which will be learned during the training stage.

3.2.3 Multi-layer perceptron (MLP) block

The MLP block consists of several fully-connected layers with non-linear activation functions in between. In our implementation, we use two linear layers with a GELU() activation function. A dropout regularization is applied after the first linear layer. As for dimensionality, the input and output dimensions are designed to be identical (both are $(N + 1)$ by D tensors); however, the hidden size of the MLP is a hyperparameter that needs to be tuned.

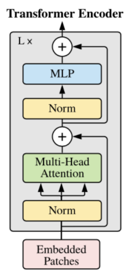

Figure 2. Illustration of transformer encoders

3.2.4 Layer normalization (`LN`) block

Layer normalization block is used inside the encoder to process the input for `MSA` and `MLP`. However, this normalization layer is different from traditional normalization. In `ViT`, there are two sets of parameters (weights and bias) associated with the `LN` block. Fortunately, in Pytorch, we can directly utilize the built-in implementation from `nn.LayerNorm`.

According to the Figure. 2, the main structure of an encoder is a `MLP` block after `MSA` with two `LN` before them, respectively. In addition, we can see that the residual connections add the raw input (i.e. unnormalized input) of `MSA` and `MLP` to their output, respectively. In `ViT`, instead of using only one transformer encoder, a series of them are used to extract more features and representations.

3.3. `MLP` classification head

After a series of transformer encoders, a final classification head is needed for supervised learning problem. In `ViT`, only the left-most vector of the output sequence is considered: it will be feed into a linear layer followed by a `Softmax` function to obtain a vector of probabilities.

3.4. Simplified `ViT` architecture

3.4.1 Implementation

In our project, we implemented the `ViT` without using any external resources from the internet. We follow the original paper rigorously without any modifications; however, some of the architecture specifications are different.

3.4.2 Customized model architecture

`ViT` is extremely memory-intensive because of its complex internal structure: the original paper's `ViT` model has a minimum of 86M parameters. Unfortunately, we found that Colab Pro's GPU memory is insufficient to hold such a large model. Thus, we instead use the following architecture:

Table 1. Customized Model Architecture

Model	D	**`MLP` size**	**`MSA` heads**	**# of encoders**
`S-ViT`	128	512	4	4
`S-ViT`$^+$	256	1024	4	4

Note that this model has a complexity that is much lower than the original `ViT` model. For simplicity, we denote our simplified `ViT` as `S-ViT`. Unfortunately, through testing, we found that `S-ViT`$^+$ is the most complex model that we can run on Colab and it is used for our ablation studies which we will introduce later.

4. Experiments

4.1. Overview

4.1.1 Dataset

In the original paper by Dosovitskiy *et al.*[4], they conducted their transfer-learning-based experiments mainly on the JFT dataset, which is introduced in 2017 by Sun *et al.*[6] and is an extremely large dataset that is privately owned by Google. Because we do not have the access to it, we based our experiments on `CIFAR-10` and `ImageNet`.

4.1.2 Experiments objectives

Due to the GPU limitation (especially Colab's memory limitation) that we mentioned in section 3.4.1, we are unable to conduct the same transfer-learning-based experiments that are done in the original paper, which requires intensive pre-training on extremely large dataset. Therefore, we decided not to reproduce the results from the paper; instead, we will do a small scale experiment and evaluate the performance of the simplified `ViT` that we implemented by comparing its performance on `CIFAR-10` and `ImageNet` with some baseline models: `VGG` and `ResNet`. In addition, we will do an in-depth ablation studies in order to understand how each section of the `ViT` contributes to the performance.

4.1.3 Training hyperparameters

We trained our `S-ViT` that is specified in section 3.4.2 for 100 epochs with the following hyperparameters:

1. Learning rate (η): we use a learning rate scheduler to dynamically decrease η during the training stage. Through experiments, we choose an initial learning rate of 0.1 and a decay rate of 0.5 with step size 10.
2. Batch size (B): we choose a batch size of 512, which can give us a stable approximation while exploring the advantages of randomness as well.
3. Loss function: for traditional classification problem, we stick with the `CrossEntropyLoss`.

4. Optimizer: `SGD` and `Adam` are both experimented in our studies. We finally choose the plain `SGD` optimizer without momentum.

4.2. Experimental results

In this section, we will present our experimental results while analyzing the potential reasons of the observed performance gap through a series of ablation experiments.

4.2.1 Comparison with the benchmark

We directly used the `VGG-16` and `ResNet-18` from the Pytorch library, and in our experiment no pre-trained versions are used in order to be consistent with our `S-ViT`, where we do not do pre-training. The experimental results are provided below:

Table 2. Experimental Results

Model	Dataset	Accuracy
`VGG`	`CIFAR-10`	83.58%
`ResNet`	`CIFAR-10`	86.44%
`S-ViT`	`CIFAR-10`	54.23%
`VGG`	`ImageNet`	76.93%
`ResNet`	`ImageNet`	81.54%
`S-ViT`	`ImageNet`	51.41%

According to the table, we found that our `S-ViT` does not achieve comparable performance as the benchmark `VGG` and `ResNet` architectures. However, we are not surprised by this result because of the limited model complexity of `S-ViT` given that our model complexity is only about $\frac{1}{10}$ of that of the original paper, and we do not do large scale pre-training. This is acceptable to us since our goal is not to reproduce the results of the paper; instead, we will do an in-depth analysis and ablation studies in the next section.

4.3. Ablation studies and discussion

We identify that there are several possible reasons that our `S-ViT` does not have good performance, among which the model complexity is the most important factor.

4.3.1 `S-ViT` model complexity

We discover that the internal structure for `ViT` is complex which makes it prone to overfitting. However, through numerical experiments, the problem associates with our `S-ViT` is most likely underfitting. We examine this hypothesis by two ways: increasing model complexity and reducing regularization.

1. We modify the latent vector size D and the `MLP` hidden size of our `S-ViT` by doubling them, respectively. The modified architecture is given by $\texttt{S-ViT}^{+}$ in section 3.4.2. Similarly, we trained $\texttt{S-ViT}^{+}$ on the `CIFAR-10` with the same training hyperparameters. The experimental result is given below:

Table 3. Comparison of `S-ViT` and $\texttt{S-ViT}^{+}$

Model	Dataset	Accuracy
`S-ViT`	`CIFAR-10`	54.23%
$\texttt{S-ViT}^{+}$	`CIFAR-10`	57.71%

We can observe that there is about 3% performance gain when we increase the model complexities.

2. In addition, we reduce the dropout regularization rate in order to observe the influence of the regularization. The experimental results on the `CIFAR-10` dataset are provided below:

Table 4. Ablation Studies Results on Dropout Rate

Model	Dropout Rate	Accuracy
`S-ViT`	0.00	60.47%
`S-ViT`	0.05	56.82%
`S-ViT`	0.10	54.23%
`S-ViT`	0.25	46.18%
`S-ViT`	0.50	41.39%

We can see clearly that for our `S-ViT`, as we gradually remove the regularization, the model performance increases steadily, which confirms our hypothesis and expectation that our model has low complexity which makes it underfit.

4.3.2 Hybrid architecture

As proposed by Dosovitskiy *et al.*[4], we can also combine our `S-ViT` with the classical `CNN` architecture. Instead of directly doing the patch embedding of the image, we can apply a `CNN` to the image and then do a patch embedding on the feature map. To check whether this idea works, we conduct a small scale experiment on the `CIFAR-10` dataset. As for the convolution layer, we set its kernel size and stride equal to the patch size (P) that we specified before, and the output channels to D. Then we train this hybrid model for 100 epochs, and the experimental result is provided below:

Table 5. Hybrid Model Performance

Model	Accuracy
Hybrid `S-ViT`	57.10%

We observe that the hybrid structure does improve the performance of our model; however, the accuracy is still not satisfied because the overall model complexity is too low.

4.3.3 Discussion

The main limitation of our project is the GPU resource and the access to the JFT dataset. Even though we do not see a comparable performance of our model with the benchmark, we successfully reimplement it and discover how each part of the `S-ViT` influence the model performance through a series of in-depth experiments.

5. Conclusions

In our project, we implemented the vision transformer following the architecture mentioned in the original paper[4]. In addition, due to many limitations that we confronted in this project, we modified our goals to compare our implementation with some benchmark architectures and did in-depth ablation studies, instead of trying to reproduce the paper's results. Even though our simplified `ViT` does not achieve satisfied accuracy due to underfitting issue, we discover that there are several ways that we can try to improve its performance; unfortunately, we are unable to do that due to hardware limitation. However, we do find that the intuition behind vision transformer is practical through our experiments and believe that the `ViT` architecture will be a good direction to explore for future work.

References

[1] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[2] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *arXiv preprint arXiv:1911.03584*, 2019.

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

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[5] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[6] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.