# When Fine-tuning always Beats Training from Scratch: Strategies to Adopt in Vison Transformer Training

**Kuangshi Ai, Jiaying He**
KUAI6409, JIHE7032
kuai6409@uni.sydney.edu.au, jihe7032@uni.sydney.edu.au

## Abstract

In this assignment, we propose ideas to improve the training efficiency of ViT, and discuss training costs under different scenarios as well as the strategies for ViT training. Our research is conducted along two branch lines: training from scratch & fine-tuning pre-trained models. For the former part, we come up with Rotated Patch Tokenization to increase the locality inductive bias of ViT. This method is proven effective on small datasets, but soon encounter bottleneck on larger datasets. For the latter part, we fine-tune pre-trained models on target datasets, with significant increase in accuracy. Interestingly, our findings in these two branch lines have a subtle connection: with the growth of data size, "Attention is all you need", whereas any architecture modification towards CNN, aiming at increasing "receptive field", has more impressive preformance on smaller datasets. At last, based on our experiment findings, we discuss strategies for training ViT. Refer to this GitHub repository for our code implementation: https://github.com/KuangshiAi/vision_transformer

## 1 Introduction

Recently, Vision Transformer (ViT) has achieved outstanding performance in various vision tasks, and still remains state-of-the-art in image classification. However, a widely known restriction is that the method requires a large enough dataset to achieve ideal performance, which challenges the computational power. Moreover, With the models and datasets growing, there is still no sign of saturating performance. [1] Therefore, casting our sight on this area may hopefully bring wonderful results.

When looking at the past data pre-processing method, we noticed some inefficiency. The data image is uniformly cut into several square tokens, which may prevent attention from being focused on the desired object. Compared with convolution neural networks traditionally adopted in CV, the Vision Transformer's weaker inductive bias results in smaller receptive field and lost of local information. Small receptive field cause ViT to tokenize with too few pixels, even not sufficient enough to embed the spatial relationship with adjacent pixels.

To capture more local spacial information during image embedding, we propose Rotated Patch Tokenization (RPT) and utilize learnable positional embedding. But they are only proven effective when training from scratch on smaller datasets, like CIFAR10 and CIFAR100. Besides, ViT is not as effective as ResNet on the same training setting. However, when we try to fine-tune on smaller datasets with models pre-trained on much larger datasets, Vit shows its super efficiency and astonishing high accuracy.

When it comes to training ViT, the strategies to adopt actually vary a lot with users' need. Generally speaking, there are three types of cost to be considered: pre-train cost, practitioner cost (fine-tuning cost on target dataset), and deployment cost (inference cost of trained model). Besides, training from scratch for research purpose is also an important application scenario. There are already many

Research Report for COMP5329 (USYD 2023), Sydney, Australia.

proposed methods and excellent researches that give insightful ideas on training ViT. We conduct experiments on the base of former research, and further conclude viable instruction manual for training ViT.

## 2 Scope of the study

### 2.1 Background

With the rapid development of deep learning in computer vision field, in many practical application scenarios, it has been quite common to pre-train large models on large datasets and use their parameters to initialize parameters of models trained on a broad variety of other tasks, which is also known as fine-tuning. That seems to be how transformers work well in today's deep learning field, just as what we know about the groundbreaking large language model: GPT (Generative Pre-trained Transformer). However, it could take tons of time, calculation resources and money to train such a large model. It is estimated that it costs millions of U.S. dollar to train GPT-3, let alone more advanced GPT-4. When looking at the overall computational and sample cost of both pre-training and fine-tuning, "pre-train cost" typically dominates overall costs.

Nevertheless, for most of the cases, researchers can have easy access to pre-trained model, or in the worst case, the model can be trained once in a while. Under this circumstance, it is the budget required for adapting and fine-tuning the model that becomes the main bottleneck. Therefore, we would concentrate on the time and compute spent on finding an optimal pre-trained model and fine-tuning it, which is also known as "practitioner cost".

In daily and industrial application scenarios, things become a bit different. The whole training cost is no longer crucial, as the inference cost of the trained model will finally amortize all other costs as time goes by. Consider the case of ChatGPT again, billions of people send requests to the server daily, the inference cost will sum up and far exceed the cost in training the model itself. In a word, especially for large scale model, "deployment cost" is the main consideration. In this study, we will evaluate the "deployment cost" of ViT by the number of images the trained model can process per second.

Besides, with limited compute resources we have and limited time, the cost of training from scratch on relatively smaller datasets is also where our study focuses on. In image classification task, the original study on ViT published in 2021 [1] is still the state-of-the-art. The main cause is that the dataset they used, JFT-300M with 300M images, is still private now. The pre-training on such a huge one also requires lots of compute and time. Comparatively, ImageNet-21k, the biggest dataset where most researchers pre-train their models on, only contains 13M images. Therefore, from research perspective, training from scratch on relatively smaller datasets will give us insightful findings on the inner property of the the ViT model. The Rotated Patch Tokenization (RPT) we propose and the learnable positional embedding we utilize are also proven effective improvement here.

### 2.2 Related work

Recently, researchers have proposed several data-efficient ViTs to reduce the dependence of ViT on large scale pre-training. Some of them with effective optimization methods even outperforms ResNets on relatively smaller datasets without pre-training or strong data augmentation [2], which completely clears CNN type network out of SOTA list of image classification. Besides, many strategies in machine learning are applied to further perfect the original ViT model. For example, DeiT [3] improved the efficiency of ViTs by adopting the existing data augmentation and regularization strategies pre-existing for convolutional neural network, one variant of transfer learning: "distillation" is used to guide DeiT learning from a well-trained ResNets with 40 times more parameters. T2T [4] used a tokenization method that flattened overlapping patches to learn local information around a token better. Also, there are researchers focus on image embedding just as we do, their idea of patch tokenization and locality self attention [5] are very intuitive and easy to implement, which inspired us to propose Rotated Patch Tokenization (RPT).

Most of previous studies aim at improve Vision Transformer's weaker inductive bias. Other than that, researchers from Google present us with hands-on instructions on how to train a ViT with data augmentation and regularization [6]. The pre-trained model with grid-search hyperparameters are also

the ones that we mainly use in our experiment part. In the following part, we will evaluate compute versus performance trade-offs in recent studies and propose our own insights on ViT training.

# 3 Proposed Method

## 3.1 Overview

Basically, we propose our methods by putting ourselves in two different shoes of training ViT: training from scratch and transfer learning.

For training from scratch, we come up with the idea of Rotated Patch Tokenization to enlarge receptive field and capture more local spacial information. For transfer learning, we fine-tune famous models pretrained on much bigger datasets to adapt to our tasks.

## 3.2 Rotated Patch Tokenization

### 3.2.1 Basic Concepts

Before looking into Rotated Patch Tokenization in detail, we first breifly review the basic concepts in standard ViT.

Assume $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ represents an input image with height H, width W and C channels. Vision Transformer splits the input image into N non-overlapping patches and sort these patches to derive a sequence of vectors. This can be expressed as formula(1):

$$\mathcal{P}(\mathbf{x}) \quad = \quad \left[\mathbf{x}_p^1; \mathbf{x}_p^2; \ldots; \mathbf{x}_p^N\right] \tag{1}$$

where $\mathbf{x}_p^i \in \mathbb{R}^{P^2 \cdot C}$, P and $N = HW/P^2$ respectively stands for the i-th patch, the patch size and the number of patches.

Then we linearly project the derived vectors onto the space of the hidden dimension of the transformer encoder, or in other words, we tokenize the patch embeddings. This is defined by:

$$\mathcal{T}(\mathbf{x}) \quad = \quad \mathcal{P}(\mathbf{x})\boldsymbol{W} \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{(P^2 \cdot C) \times d}$ is the projection matrix and d is the hidden dimension of the transformer encoder.

The receptive field size of these tokens can be calculated by[7]:

$$r_{token} = s * (r_{trans} - 1) + k \tag{3}$$

where $r_{token}$ and $r_{trans}$ represents the receptive field size of the tokens and the transformer encoder. s and k stands for the stride and the kernel size of the convolutional layer.

As the transformer encoder won't affect the receptive field, $r_{trans}$ = 1. Thus, $r_{token}$ equals the kernel size. Here the kernel size is the same as the patch size of ViT.

Up to now we have demonstrated that the receptive field size of a standard ViT equals its patch size. But as we know some famous CNN models such as ResNet, have a much larger receptive field compared to their patch size.

### 3.2.2 Method Principles

To address the problem of small receptive field for ViT, we first rotate the input image clockwise and anticlockwise for some random angle between $(\frac{\pi}{20}, \frac{\pi}{14})$. We combine them with four shifted images derived by spatially shifted by half the patch size in four diagonal directions and gain a series of rotated images(for brevity, we regard those shifted ones also as rotated images). For convenience, we

define this kind of strategy as $\mathcal{R}$. Next, the rotated images are cropped to the same size as the input image and then concatenated with the input. Then we divide the concatenated features into patches and flatten them just as we do in standard ViT. Next, we apply layer normalization(LN) and linear projection. This process can be summarized as

$$\mathcal{R}(\mathbf{x}) = \mathrm{LN}\left(\mathcal{P}\left(\left[\mathbf{X}\mathbf{R}^1\mathbf{R}^2\dots\mathbf{R}^{N_\mathcal{R}}\right]\right)\right)\boldsymbol{W}_\mathcal{R} \tag{4}$$

where $\mathbf{R}^i \in \mathbb{R}^{H\times W\times C}$ represents the i-th rotated image under the strategy $\mathcal{R}$ and $\boldsymbol{W}_\mathcal{R} \in \mathbb{R}^{(P^2\cdot C\cdot(N_\mathcal{R}+1))\times d}$ stands for the linear projection matrix. $N_\mathcal{R}$ and d respectively indicates the number of rotated images and the hidden dimension of the transformer encoder.

Now we apply the RPT to the patch embedding layer. We connect a class token which records information of the entire image to visual tokens and then add positional embeddings. This can be formulated as:

$$\mathrm{R}'(\mathbf{x}) \quad = \quad \begin{cases} [\mathbf{x}_{cls}; \mathrm{R}(\mathbf{x})] + \boldsymbol{POS} & \text{if } \mathbf{x}_{cls} \text{ exist} \\ \mathrm{R}(\mathbf{x}) + \boldsymbol{POS} & \text{otherwise} \end{cases} \tag{5}$$

where $\mathbf{x}_{cls} \in \mathbb{R}^d$ is the class token and $\boldsymbol{POS} \in \mathbb{R}^{(N+1)\times d}$ encodes the positional information. We also let $\boldsymbol{POS}$ be a learnable parameter.

Obviously, through Rotated Patch Tokenization, we have embedded richer spatial information into our tokens, increasing the receptive field size of tokens.

### 3.3 Upstream Pre-train and Downstream Fine-tune

This section describes basic concepts in the second part of our experiment. Transformers are widely known for their astonishing ability when they are pre-trained on large datasets and fine-tuned on a specific sub task. However, upstream pre-training can be quite expensive, making hyperparameter selection and ablation study almost impossible for individuals and researchers in universities. It is estimated that in 2018, pre-training BERT [8] model only once takes 3 days and thousands of dollar, let alone larger transformers like GPTs today. However, big tech companies with sufficient computing resources, like Google and Facebook, have provided us with pre-trained ViTs with various configuration settings.

The upstream performance of these pre-trained models are transparent for us. As practitioners, our main appeal is to find the best pre-trained model for our downstream sub task, and effectively fine-tune it using smaller dataset. Several questions are raised then: how to choose the favorable ViT model size? How to find the parameter checkpoint with optimal hyperparameter settings? And how to effectively fine-tune the chosen model? We will answer these questions later based on our experiments conducted on small-sized and mid-sized datasets.

## 4 Experiment

### 4.1 Environment and Dataset

The proposed method Rotated Patch Tokenization (RPT) is implemented in Tensorflow [9]. And we use a JAX/Flax codebase for fine-tuning pre-trained models on our selected datasets. The pre-trained models, which are pre-trained on ImageNet and ImageNet-21k datasets, are from GitHub repository of Google Research [10]. Inference speed measurements were obtained on GPUs offered by Google Colab, with batch size 256.

For training from scratch and fine-tuning pre-trained large models, we use four relatively small size datasets: Tensorflow Flowers, CIFAR10, CIFAR100 and Tiny-ImageNet [11]. The former three ones are accessed through the TensorFlow Datasets library [12], the latter one is accessed from GitHub public repository. Generally speaking, our dataset selection covers typical image classification datasets ranging from small-sized ones to mid-sized ones. And the pre-trained models we adopt are trained on large scale image datasets: ImageNet and ImageNet-21k.

### 4.2 Metrics and Training Regime

We report both top-1 accuracy for all datasets as our main metric. The datasets were divided into training set and test set, with 5:1 split ratio for CIFAR10 & CIFAR100, and 10:1 split ratio for Tensorflow Flowers and Tiny-ImageNet. Throughput measurements were obtained with NVIDIA A100 GPU in this section.

With limited compute resources offered by Google Colab, we did not manage to do hyper-parameter selection on validation set ourselves, but turned to the recommended setting for ViT-Tiny given by Google researchers [6] from their ablation studies. As for data augmentations, which significantly improve model performance on small scale datasets, we applied horizontal random flip, random rotation and random zoom. To maintain the consistence of results, we preprocess all raw datasets this way before input them into the models. In the meantime, we used AdamW as the optimizer, with initial learning rate 0.001, and cosine learning rate decay was used. Weight decays were set to 0.0001 (prefer augmentation to regularization), batch size to 256, and warm up epoch percentage to 10%. When comparing the performance of our proposed method on different datasets, we trained our models from scratch for 50 epochs.

### 4.3 Quantitative Results

#### 4.3.1 Training from Scratch

This section presents the experimental results of our proposed RPT method. All results are obtained by training from scratch. Throughput and Parameter number are measured on NVIDIA A100 GPU in Tensorflow Flowers dataset.

Table 1: Top-1 accuracy comparison of different models, all trained from scratch for 50 epochs. The throughput is measured on 224px resolution.

| Model | Throughput (images/sec) | Params (M) | TF_FLOWERS | CIFAR10 | CIFAR100 | T-ImageNet |
|---|---|---|---|---|---|---|
| ResNet 56 | 798 | 0.8 | 76.67 | 83.41 | 54.16 | 33.38 |
| ViT-Ti | 1600 | 2.58 | 73.02 | 81.56 | 52.35 | 32.41 |
| SL-ViT | 1264 | 2.7 | 67.57 | 82.94 | 55.71 | 34.45 |
| DeiT-Ti | 238 | 5 | 72.21 | - | - | - |
| Rot-ViT | 1333 | 2.6 | 67.3 | 81.96 | 54.75 | 34.22 |

Firstly, as shown in Table 1, our proposed method could improve the performance of ViT by merely 2% on CIFAR100 and Tiny-ImageNet, while not significant improvement compared with ViT-Ti on CIFAR10, and perform even worth in TF_Flowers. And one interesting fact we found is that ResNet 56, the CNN type model, exhibits exactly the opposite properties to ViT type models. ResNet 56 performs best on TF_Flowers and CIFAR10, but beaten by improved ViT models on CIFAR100 and Tiny-ImageNet. The improvements brought by our proposed method also show similar attributes as that of ResNet 56: more effective on smaller datasets, but soon encounter bottleneck on larger scale datasets.

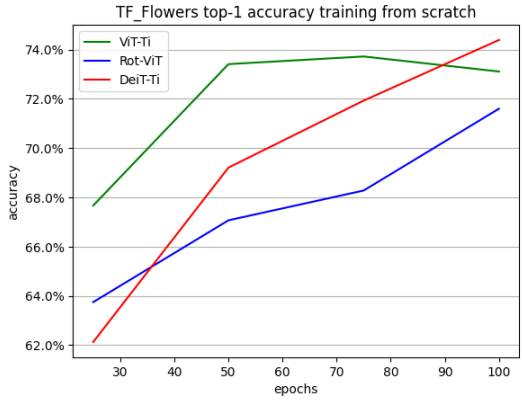
One main benefits that our model and many other improved models based on ViT bring is stronger generalization ability, or using the term in CNN, bigger "receptive field". For example, DeiT improves the accuracy of small-sized ViT models by distillation from much larger ResNets, which capture local positional information better than ViTs inherently. However, as the size of dataset improves, "Attention is all you need". The strong global features catching ability enables transformers to achieve good enough performance. This is also partly why pre-trained large models with less advanced architecture always perform better after fine-tuning, which we will go into details in next section.

Table 2: Throughput (images/sec) comparison of original ViT-Ti and our model on different datasets

| Model | TF_FLOWERS | CIFAR10 | CIFAR100 | T-ImageNet |
|---|---|---|---|---|
| ViT-Ti | 1676 | 3605 | 2162 | 3454 |
| Rot-ViT | 1333 | 3282 | 1954 | 3240 |

Another thing worth mentioning is that Rotated Patch Tokenization improves the model accuracy only with small overhead of inference latency. As it is shown in Table 2, when RPT used in dividing image patches and token embedding, it only causes latency overhead of 20.5%, 8.95%, 9.62% and 6.20% on TF_Flowers, CIFAR10, CIFAR100 and Tiny-ImageNet, respectively. Considering the image resolutions in these datasets are 224px, 32px, 32px and 64px, the results are in line with our expectations, as higher resolution causes more compute burden for RPT. Lower latency overhead on Tiny-ImageNet is an exception, in this case, GPU manages to process data in parallel more efficiently with 10 times bigger training set.

Figure 1: Improvements aiming at inductive bias problem significantly enhance the generalization ability of ViT models on small datasets.




What's more, Figure 1 shows us how our proposed method and DeiT contribute to training from scratch on small datasets (take TF_Flowers as an example). On the smallest dataset that we use, we find that original ViT model converges much earlier than our model and DeiT, overfitting comes soon after that. This supports again that RPT and distillation significantly enhances the generalization ability of ViT. Other than that, as Andreas Stet et al. [6] point out, adding the right amount of regularization and image augmentation can also lead to improvement in generalization ability, with similar gains as increasing the dataset size by an order of magnitude.

#### 4.3.2 Fine-tuning Pre-trained Models

This section presents experimental results of fine-tuning large scale pre-trained ViT model [10] on TF_Flowers, CIFAR10, CIFAR100 and Tiny-Imagenet. These datasets contains 3,600, 50,000, 50,000, and 100,000 samples for fine-tuning, respectively. Throughput measurements were obtained with NVIDIA V100 GPU in this section. In table 3, we give the configuration of available ViT models offered by Google research.

Table 3: Configuration of ViT models

| Model | Layers | Width | MLP | Heads | Params (M) |
|---|---|---|---|---|---|
| ViT-Ti [3] | 12 | 192 | 768 | 3 | 5.8 |
| ViT-S [3] | 12 | 384 | 1536 | 6 | 22.2 |
| ViT-B [1] | 12 | 768 | 3072 | 12 | 86 |
| ViT-L [1] | 24 | 1024 | 4096 | 16 | 307 |

We download and fine-tune four different pre-trained models: ViT-B_16, ViT-B_16s, ViT-B_32 and R50+ViT-B_16. The number after "_" denotes patch size of this ViT, and "R50+ViT-B_16" is a ResNet+ViT hybrid model. Besides, "ViT-B_16s" was pre-trained on both ImageNet 21k and ImageNet 1k, while other models were only pre-trained on ImageNet 21k. Table 4 and Table 5 gives the top-1 accuracy before fine-tuning and after fine-tuning, respectively. All models were fine-tuned for 100 epochs. The throughput may not be accurate enough, as Google Colab sometimes designates compute resources randomly during our training.

Table 4: Top-1 accuracy before fine-tuning, which is almost random guess.

| Model | TF_FLOWERS | CIFAR10 | CIFAR100 | T-ImageNet |
|---|---|---|---|---|
| ViT-B_16 | 0.2246 | 0.1006 | 0.0103 | 0.0007 |
| ViT-B_16s | 0.2246 | 0.1006 | 0.0103 | 0.0007 |
| ViT-B_32 | 0.2246 | 0.1006 | 0.0103 | 0.0007 |
| R50+ViT-B_16 | 0.2246 | 0.1006 | 0.0103 | 0.0007 |

Table 5: Top-1 accuracy after fine-tuning. The throughput measurements were obtained by averaging all instant inference speed of a single model.

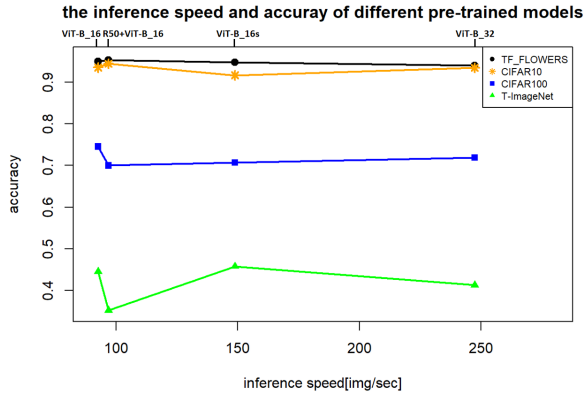
| Model | Throughput (images/sec) | TF_FLOWERS | CIFAR10 | CIFAR100 | T-ImageNet |
|---|---|---|---|---|---|
| ViT-B_16 | 92.59 | 0.9492 | 0.9351 | 0.7447 | 0.4452 |
| ViT-B_16s | 148.84 | 0.9473 | 0.9163 | 0.7064 | 0.4580 |
| ViT-B_32 | 247.34 | 0.9395 | 0.9343 | 0.7181 | 0.4128 |
| R50+ViT-B_16 | 96.88 | 0.9531 | 0.9441 | 0.6994 | 0.3524 |

As shown in Table 4, before fine-tuning, none of these pre-trained model shows any difference than models with randomly generated parameters. However, after very efficient fine-tuning process, they all get pretty nice accuracy. On V100 GPU, the whole process only takes several minutes. Correspondingly, training smaller models from scratch on the same dataset takes more than half an hour, while the final performance is much worse.

One interesting thing we find is that pre-trained ResNet+ViT hybrid model does not perform as well as other pure ViT models on mid-sized dataset, like Tiny-ImageNet. This is in accordance with our finding in training from scratch. With the growth of data size, the advantage of transformer emerges, which applies to both training from scratch and fine-tuning on target dataset. To conclude, any improvement aiming at increasing "receptive field" or data augmentation works and only works on small-sized datasets.

The above discovers are more explicitly reflected in Figure 2, in which we sort different pre-trained models by the inference speed. Besides, it is quite obvious that the larger the pre-trained model is, the more images can be processed in one second. However, the figure doesn't reflect a significant connection between the inference speed and the classification accuracy.

Figure 2: The inference speed and accuracy of different pre-trained models




### 4.4 Strategies to Adopt in ViT Training

Basically speaking, considering the "practitioner cost", for most practical purposes, transferring a pre-trained model is both more cost-efficient and leads to better results. But training from scratch also gives us insights into model architecture and training properties of ViT type models. In this section, we will analyse compute versus performance trade-offs, and draw a conclusion on strategies for ViT training.

As for selecting optimal pre-trained model, Andreas Stet et al. suggest in their research that model with best upstream accuracy (on pre-training validation set) is typically the one that does best on downstream task [6]. But it may not be the case in small-sized, specialized tasks, like our TF_Flowers dataset. With limited compute spent on fine-tuning and limited samples, bigger pre-trained model with more complex architecture may fail to compete with smaller model with less parameter. So our suggestion is, when your task is in a specialized vision sub field, like processing satellite image or classification of biological species, choose smaller pre-trained ViT if possible. Besides, when the number of samples you have for fine-tuning is limited, hybrid models may help.

Our experimental results also tells what to do if higher inference speed is needed in the application scenario. As shown in Figure 2, the larger a model is, the slower its inference speed is. And also, increasing patch size could increase the throughput efficiently, with acceptable drop in downstream accuracy. Rather than decreasing the model size to obtain higher inference speed, increasing the patch size is a preferred choice.

## 5 Conclusion

In this assignment, we've explored how to improve the efficiency and lower the cost of training ViT and further conclude instruction manual for practical operation, whether you are training from scratch or would like to fine-tune pre-trained models. For training from scratch, we come up with Rotated Patch Tokenization to increase the receptive field size, which turns out to be only effective on smaller datasets. For fine-tuning pre-trained models, we fine-tune 4 different pre-trained models (ViT-B_16, ViT-B_16s, ViT-B_32, R50+ViT-B_16) on 4 different datasets (TF_FLOWERS, CIFAR10, CIFAR100, T-ImageNet). While seeing transfer learning performs better in accuracy, we also notice that pre-trained ResNet+ViT hybrid model does not perform as well as other pure ViT models on mid-sized dataset, like Tiny-ImageNet. This finding is consistent with our discovery in training from scratch: increasing "receptive field" or data augmentation is only effective for smaller datasets while the transformer and the principle of attention is more competitive for large datasets. Based on all these valuable discoveries, in the last section we propose a practical guide for training ViT.

## References

[1] 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, 2020.

[2] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pretraining or strong data augmentations. *CoRR*, abs/2106.01548, 2021.

[3] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers amp; distillation through attention, 2020.

[4] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis E. H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *CoRR*, abs/2101.11986, 2021.

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

[6] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *CoRR*, abs/2106.10270, 2021.

[7] A. Araujo, W. Norris, and J. Sim. Computing receptive fields of convolutional neural networks. *Distill*, 4(11), 2019.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[9] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[10] Google Research. Vision transformer and mlp-mixer architectures, 2022. Software available from tensorflow.org.

[11] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge, 2015.

[12] Google. Tensorflow datasets, a collection of ready-to-use datasets.