Exploring the Computational Power of ViTs

Veera Adithya Dittakavi <vdittaka@purdue.edu> <0034728255>

Course Advisor:
Prof. Okan K. Erosy
Professor of Electrical & Computer Engineering

ECE 629: INTRODUCTION TO NEURAL NETWORKS

December 10, 2022

Table of Contents

List of Tables

List of Figures

Abstract

ViT, or Vision Transformer, is a type of artificial neural network specifically designed for computer vision applications. The underlying concept behind ViT is inspired by the use of transformers in natural language processing, where self-attention is applied to specific keywords in order to make the learning process more efficient than traditional recurrent neural networks (RNNs). ViT uses self-attention on patches of images in order to process them more efficiently than convolutional neural networks (CNNs), making it an effective tool for tasks such as image classification and object detection. In this report, we focused on exploring the computational power of such transformer networks, given different models and comparing them with traditional neural networks on Image Classification. One of the observations is that the ViT outperforms its counterparts trained on larger datasets. But also very limited in some of the important cases where the model has to be trained from the scratch.

1 Problem Statement

Background

Natural Language Processing has seen successes in scaling the self-attention-based mechanisms onto a network, called a Transformer. In such a network, words were of the most interest, and an encoding-decoding architecture provided modeling to the data. The scaling of the network helped achieve the task: Translation by being more parallelizable and faster than recurrent neural networks.

Now let us consider an image of size n x n. The n^2 data points, if treated as words can be scaled and understood by a transformer. Here the application might be different but still, the idea of applying a self-attention-based mechanism persists to a good extent. The transformer is now called a ViT (ViT).

[3] Attention Is All You Need

[4] Attention is all you need: Discovering the Transformer paper

[5] An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

Objectives

Create a google colab account for this project. Use the tools developed in the reference and possibly additional references, running the program as much as you can with GPU in your google colab account for the computer vision applications cited in the reference.

- Explain how a ViT is different than a Convolutional Neural Network similar to the idea of differentiating transformers and recurrent neural networks.
- Define the methodology of a ViT and identify the resources of the platform where you will be establishing the environment for the ViT
- Identify the limitations of the available resource and provide a complete analysis
- Provide solutions to each of the problems, achieve enough results to defend them and give your opinion on the scalability of the ViT.

2 Introduction

The transformers, which are based on self-attention mechanisms, were originally developed to improve natural language processing (NLP) tasks. These tasks were typically handled by recurrent neural networks (RNNs), which store information in a cell state and learn the weights of the activation function over time. However, these networks have limitations in terms of speed and efficiency, which led to the development of long-short term memory networks, or LSTMs. While LSTMs are more complex and can store information for longer periods, they still rely on a sequential approach to processing data. In contrast, transformers use a self-attention mechanism to compute and train their networks, resulting in faster and more efficient performance. This has made transformers popular for use in complex problem statements and demonstrated widespread adoption in NLP. [3]

Another area of machine learning that requires significant computation is computer vision. In this field, networks are used to process, augment, segment, and explain images. One of the most popular networks for this purpose is the convolutional neural network or CNN. CNNs use a mathematical approach to understanding images, pre-processing information, and normalizing it before applying weight filters and approximations to find patterns and connections. Those findings help the final modules to classify and determine the features in a user-friendly manner. These networks then use backpropagation to refine their findings and improve performance. [2]

Considering the self-attention-based mechanisms (as discussed in the applications of NLPs) in the field of Image processing provides the lead in determining a ViT [5]. In this work, we induce the transformer network on large datasets to achieve the same task of Classification. In the first phase of analysis, we described the implementation of a Vision Transformer to be trained on popular datasets CIFAR 10 and CIFAR 100. Then we analyzed the limitations of resources and the scalability of the network to the possible extent. We observed the scalability of Transformers to be inefficient as it takes longer times on GPU than a CNN. This discouraging outcome may be expected: Vision Transformers do not generalize well as CNN when the data is insufficient, i.e when the network is not pre-trained with larger data and enough checkpoints. [6]

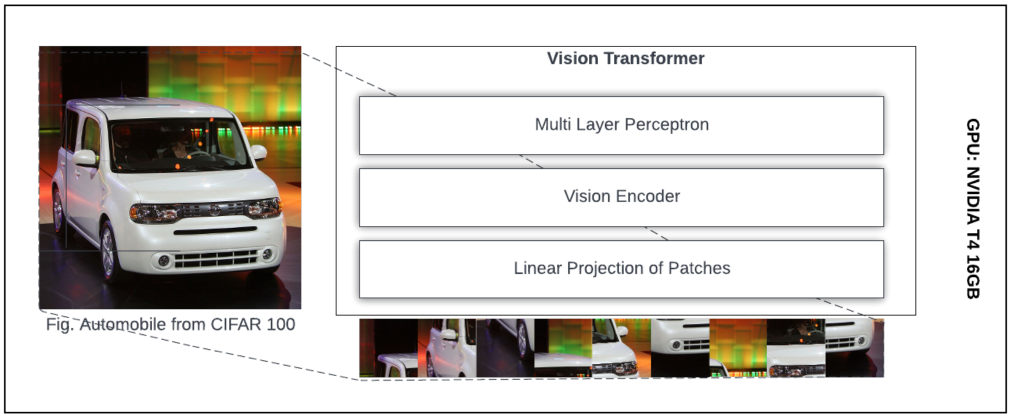


Figure 1. Overview of Vision Transformer Model. We split the image into patches, linearly embed the data, pass through the vision encoder and use a multi-layer perceptron to achieve the classification task. The image is inspired from [3]

3 Architecture

In this section, we will delve into the theoretical aspects of the ViT by comparing it to a standard convolutional neural network. We will then explore how this theoretical understanding is translated into the code implementation in Google Colab.

Architectures of the models: CNN and ViT

CNN is a type of feedforward neural network that is designed to process data that has a grid-like structure, such as an image. They are composed of multiple layers, each of which performs a different operation on the input data. The first few layers of a CNN typically consist of convolutional layers, which apply a series of filters to the input image to extract features such as edges and corners. These features are then passed through one or more fully connected layers, which combine them to produce a final output. [7]

One of the key advantages of CNNs is that they are able to learn hierarchical representations of the input data. This means that they can automatically learn to extract higher-level features from the input data, such as shapes and objects, by combining the lower-level features extracted by the earlier layers. This makes CNNs well-suited for image recognition tasks, where the goal is to classify an image into one of a set of predefined categories. [8]

ViTs [Fig.4], on the other hand, are a relatively new type of deep learning model that has been developed for image recognition tasks. Unlike CNNs, which operate on grid-like structures, ViTs operate on sequences of data. In the context of image recognition, this means that a ViT takes an image as input and represents it as a sequence of patches, with each patch corresponding to a small region of the image. The patches are then processed by a series of transformer layers, which use self-attention mechanisms to learn relationships between the patches and extract features from the image. [5]

One of the key advantages of ViTs is that they are able to process images of arbitrary sizes, whereas CNNs require that all images be resized to a fixed size before they can be processed. This means that ViTs can handle images of different sizes and aspect ratios without requiring any preprocessing, which can be a significant advantage in certain applications. Additionally, because ViTs operate on sequences of data rather than grids, they can be more easily parallelized, which can improve their training speed and efficiency. [1]

Mathematical Models for determining parameters

The CNN architecture consists of an Input Layer, Convolutional Layers, Pooling and Fully Connected Layers. Parameters are the trainable objects that aid the training of the sequence of layers mentioned for the task. Determining the parameters is independent of the Input layer as there will be nothing to learn as well as the Pooling layers that perform mathematical approximations to the data. The parameters for the convolutional layer can be determined by the operation (((size of filter * number of filters in the previous layer)+bias)*filters in the current layer)

$$input\_dim = [n, n, c]$$

$$conv\_params = ((f * f * d) + b) * k$$

$$conv\_activations = ((m * m * d)where, m = n - f$$

After the convolution layers, parameters are determined by the fully connected layers as (((current layer neurons c *previous layer neurons p)+1)*current layer neurons c)) [9]

$$fc\_params = ((c * p) + 1) * c$$

Given both the activations and parameters in a Convolutional Neural Network, the memory for the computation highly depends on the activations at each layer. The advantage in the denser neural networks is that as layers grow in the size, both the activations and parameters will become efficient, stating that most of the memory (and also compute time) is used in the early CONV layers and that most of the parameters are in the last FC layers.

Now if we consider the parameters in the ViT, they are determined in the layers of Patch Embedding, Encoder, and Multi-Layer Perceptron. The image is split into n patches with size p x p and linearized by multiplying with trainable tensors that result in the parameters:

$$input\_dim = [m, m, c]$$

$$linear\_embeddings = n * [1, p^2c][p^2c, d] = n * [1, d]$$

The resulting patches are transferred to the encoder that applies positional embedding along with Multi-Head Attention. The output is then passed to a 2-Layer MLP with residual connections. The parameters in the complete encoder stack are the number of layers L, the number of feature head tensors k, the length of feature head tensors t_h, and dimensions of weights in mlp d_{mlp} as

$$trainable\_encoderParams = L * (k * d * 3 * d_h + d * d + d * d_{mlp} + d_{mlp} * d)$$

As the depth of the encoder increases in the transformer, the number of trainable parameters increases equivalently. Whereas in the case of CNN it is different at each layer. The comparison of the number of trainable parameters is provided in the [Table.1]. We can see that the computation required by the deepest Resnet is comparable to a simple version of ViT. [4] [5] [10]

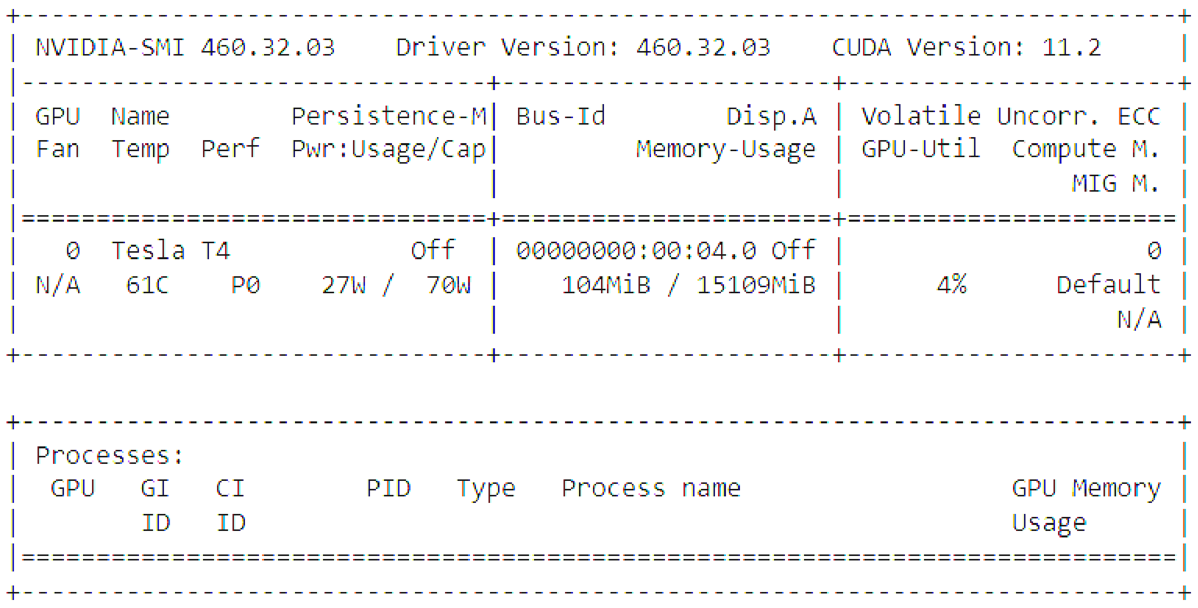
Model	Layers	Params		
		Calculated Params	Pre-Trained Params	Total Params
Resnet	18	33.3M	0.4M	31.7M
	34	63.6M	0.8M	60.5M
	50	46.4M	0.9M	45.8M
	101	85.5M	0.9M	84.8M
		Hidden Size	MLP Size	Total Params
ViT-Base	12	768	3072	86M
ViT-Large	24	1024	4096	307M
ViT-Huge	32	1280	5120	632M

Table 1. Comprision of a Convolutional Neural Network versus ViT to establish final expectations on the computational power required [1] [2]

4 Methodology

This section focuses on implementing ViT and identifying the resource available for the training.

Description of device available in Google Colab



```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   61C    P0    27W /  70W |    104MiB / 15109MiB |      4%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

Figure 2. Overview of GPU device: Tesla T4 with memory and CUDA version

ViT - training from scratch

Algorithm 1: Procedure to train the ViT from scratch

```
1 git clone google_research_vision_transformer
2 install dependencies as specified in vit_jax/requirements.txt
3 configure the local environment to use CUDA
4 import training resources, pipelines from vit_jax
5 import configurations of the model for training
6 import dataset: CIFAR 10/CIFAR 100
7 visualize the dataset for reference of images
8 split images into train/test
9 train the model and analyze for the possible errors
```

Expectations on the outcome of the procedure: The total time for the initialization of the network and training must be reported as approximately 30 TPU Hours 38 GPU Hours on a batch size of 512

ViT - Transfer Learning and Fine Tuning

Algorithm 2: Procedure to apply Transfer learning on ViTs

```
1  git clone google_research_vision_transformer
2  install dependencies as specified in vit_jax/requirements.txt
3  configure the local environment to use CUDA
4  import training resources, pipelines from vit_jax
5  import pre-trained model into runtime
6  import dataset
7  visualize the dataset for reference of images
8  split images into train/test
9  load pre-trained model into the GPU with all the configurations
10 load the parameters from the best checkpoint of the pre-trained model
11 move the loaded parameters from the runtime to the device
12 finetune the network with changing learning rate, batch size, accumulator steps
13 perform the transfer learning
14 analyze the results of the trained network with inference
```

In this case of implementation, transfer learning involves using a pre-trained transformer model, that has knowledge on Imagenet, Imagenet21K, JFT-300M datasets [6]. This mechanism helps for faster training time: Because the transformer model has already been trained on a related task, it can be fine-tuned on the new task with fewer data and compute resources. This can speed up the training process and make it more efficient. Additionally better generalization: Because the model has already been trained on a related task, it is more likely to generalize well to the new task. This can help the model perform better on a variety of data, including data that it has not seen during training.

Transfer learning is followed by finetuning the parameters for the specific tasks and datasets. Here in this procedure, we define the following parameters

$$checkpoint = best(pre\_trained\_model)$$

$$batch\_size = 512$$

$$learning\_rate\_decay\_model = cosine$$

$$gradient\_clipping\_norm = 1$$

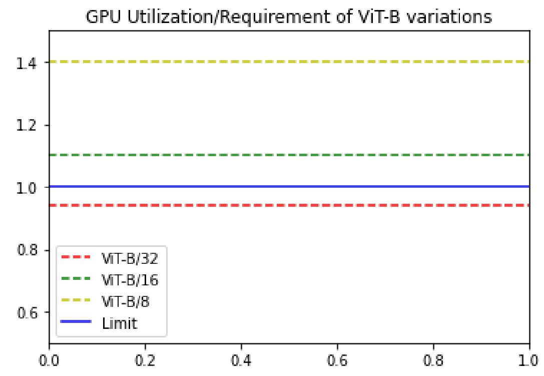
$$stochastic\_gradient\_momentum = 0.9$$

$$accumulator\_steps = 64$$

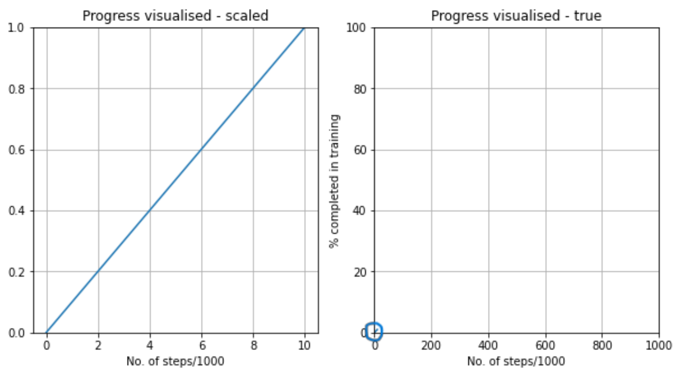
This analysis greatly reduces computational complexity and provides faster inference times, enabling the use of ViTs for larger datasets. Additionally provides improved image recognition and classification performance, as the model is able to focus on the most relevant features in the input image

5 Results & Analysis

5.1 Initialization failure: Reporting on the memory requirements - Algorithm 1



(a) Comparision of variants of ViT Base

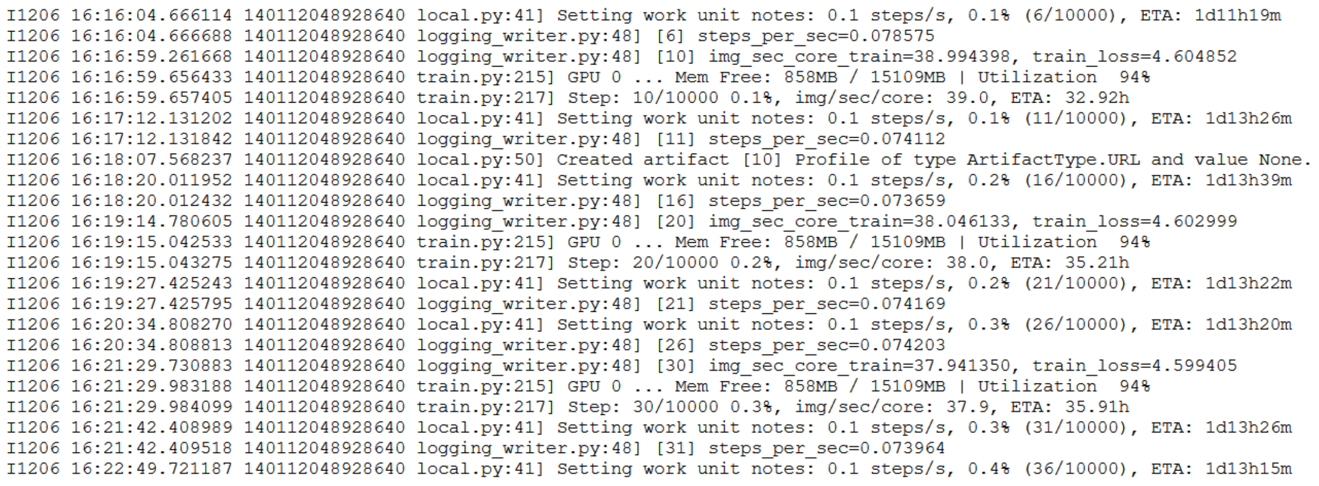


(b) ViT-B/32 Performance on full load

Figure 3. Memory requirements when trained from scratch

The observations made in the case of reporting memory requirements are focused on the variants of ViT-B/(8, 16, 32). The number ViT-B/(p) given in the variant description is the patch size that is used to divide the input image before preprocessing as discussed in section 3. The smaller the size of patch images, the higher the linear patch embeddings required so the wider the encoder layers. Such an increase in the width of linear tensors, of size n*(1,d) requires more memory to initialize the network, resulting in reaching the limit of available resources. Fig. 3a represents the same information as when the patch size is 32x32, the GPU utilization is 94%. Whereas in the networks of variants with patch sizes 16x16 and 8x8, the GPU requirement is 1.12 times and 1.4 times the available GPU respectively. These metrics account for 17.92GB and 22.4GB of a GPU device. But the colab only provides us a GPU with 16GB of memory as represented in Fig 2.

5.2 Longer estimated times: Reporting on the time required - Algorithm 1

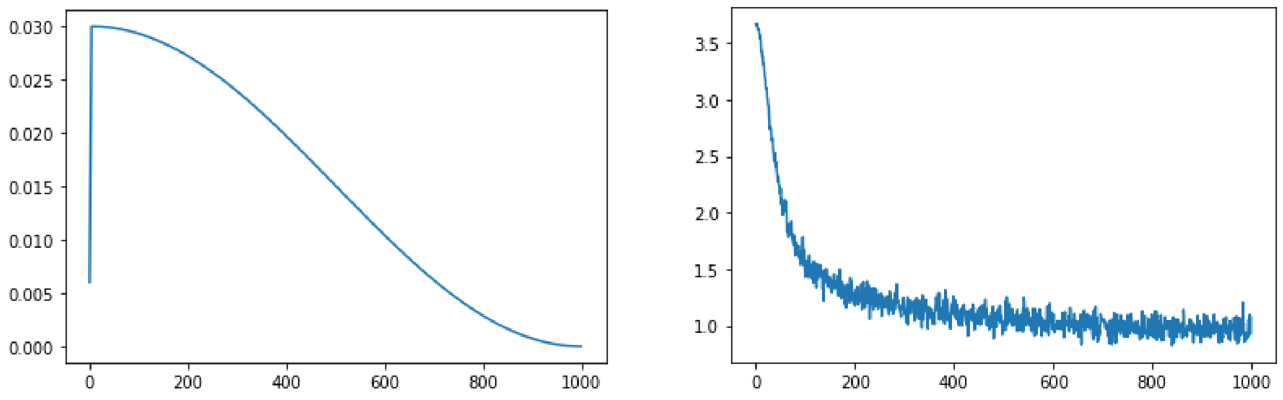


```
I1206 16:16:04.666114 140112048928640 local.py:41] Setting work unit notes: 0.1 steps/s, 0.1% (6/10000), ETA: 1d11h19m
I1206 16:16:04.666688 140112048928640 logging_writer.py:48] [6] steps_per_sec=0.078575
I1206 16:16:59.261668 140112048928640 logging_writer.py:48] [10] img_sec_core_train=38.994398, train_loss=4.604852
I1206 16:16:59.656433 140112048928640 train.py:215] GPU 0 ... Mem Free: 858MB / 15109MB | Utilization  94%
I1206 16:16:59.657405 140112048928640 train.py:217] Step: 10/10000 0.1%, img/sec/core: 39.0, ETA: 32.92h
I1206 16:17:12.131202 140112048928640 local.py:41] Setting work unit notes: 0.1 steps/s, 0.1% (11/10000), ETA: 1d13h26m
I1206 16:17:12.131842 140112048928640 logging_writer.py:48] [11] steps_per_sec=0.074112
I1206 16:18:07.568237 140112048928640 local.py:50] Created artifact [10] Profile of type ArtifactType.URL and value None.
I1206 16:18:20.011952 140112048928640 local.py:41] Setting work unit notes: 0.1 steps/s, 0.2% (16/10000), ETA: 1d13h39m
I1206 16:18:20.012432 140112048928640 logging_writer.py:48] [16] steps_per_sec=0.073659
I1206 16:19:14.780605 140112048928640 logging_writer.py:48] [20] img_sec_core_train=38.046133, train_loss=4.602999
I1206 16:19:15.042533 140112048928640 train.py:215] GPU 0 ... Mem Free: 858MB / 15109MB | Utilization  94%
I1206 16:19:15.043275 140112048928640 train.py:217] Step: 20/10000 0.2%, img/sec/core: 38.0, ETA: 35.21h
I1206 16:19:27.425243 140112048928640 local.py:41] Setting work unit notes: 0.1 steps/s, 0.2% (21/10000), ETA: 1d13h22m
I1206 16:19:27.425795 140112048928640 logging_writer.py:48] [21] steps_per_sec=0.074169
I1206 16:20:34.808270 140112048928640 local.py:41] Setting work unit notes: 0.1 steps/s, 0.3% (26/10000), ETA: 1d13h20m
I1206 16:20:34.808813 140112048928640 logging_writer.py:48] [26] steps_per_sec=0.074203
I1206 16:21:29.730883 140112048928640 logging_writer.py:48] [30] img_sec_core_train=37.941350, train_loss=4.599405
I1206 16:21:29.983188 140112048928640 train.py:215] GPU 0 ... Mem Free: 858MB / 15109MB | Utilization  94%
I1206 16:21:29.984099 140112048928640 train.py:217] Step: 30/10000 0.3%, img/sec/core: 37.9, ETA: 35.91h
I1206 16:21:42.408989 140112048928640 local.py:41] Setting work unit notes: 0.1 steps/s, 0.3% (31/10000), ETA: 1d13h26m
I1206 16:21:42.409518 140112048928640 logging_writer.py:48] [31] steps_per_sec=0.073964
I1206 16:22:49.721187 140112048928640 local.py:41] Setting work unit notes: 0.1 steps/s, 0.4% (36/10000), ETA: 1d13h15m
```

Figure 4. Estimated times and GPU load at each step training ViT-B/32 from scratch

The observations made in this case of time estimations are different but still do not enable the transform to work at its full potential. The ViT-B/32 performance on load, referring to Fig. 3b provides us with the necessary information that most of the memory is used up for network initialization. The load of the GPU is 94%, which is 15.04GB. Even at this load, the training time of the network is reportedly 36 GPU hrs which is highly inefficient. The reason for this can be concluded as the number of parameters in the ViT (refer to Table.1 that are needed to be trained. The scalability of such a network which is not pre-trained is also greatly inefficient in terms of computation and cannot be applied to Computer Vision applications.

5.3 Accurate Results on Pre-Trained ViT - Algorithm 2



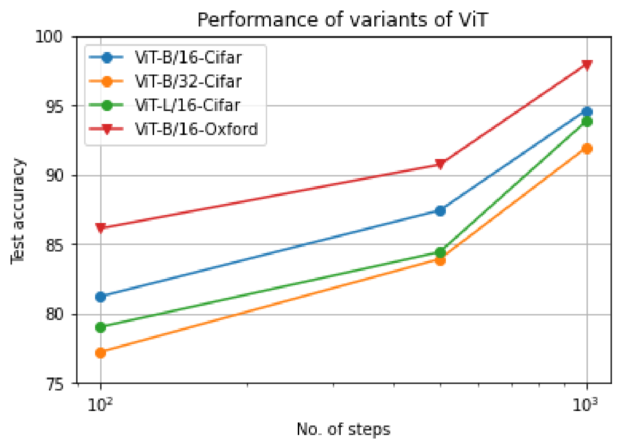
(a) Cosine Learning Rate Decay as Fine Tuning Base **(b)** ViT-B/32 Performance (Loss) with Pre-Training

Figure 5. Overview of training and performance metrics of pre-trained transformer

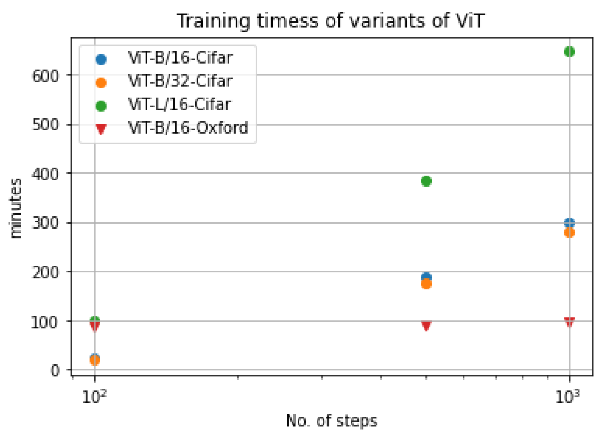
As discussed in section 4.2, the advantages of Fine-tuning and Transfer Learning are clearly reflected in the case of the ViT with 86M trainable parameters. The cosine learning rate decay Fig. 5a as fine-tuning parameter helps avoid overfitting and improve the generalization performance of the model. Additionally, cosine decay allows for faster convergence and better optimization of the model's parameters, leading to better accuracy and performance. The transfer learning from the knowledge of JFT-300M and Imagenet allowed for a training time reduction from 36hrs to 4.35hrs which is a complete win. The model also did not underperform considering this reduction in time as the loss can be observed in the Fig. 5b. The accuracy of the model is inferred as 91.94079% on test images of CIFAR100 dataset.

5.4 Comparable Results of Pre-Trained ViT variants - Algorithm 2

The successful approach of pre-training and finetuning when applied to variants of Vison Transformer: ViT-Base, ViT-Large, (p x p) patch size $where$ p=8,16,32 yield different results and consume times in a specific pattern. The observations from the Table 1 can be validated within here. The Fig. 6 represents the increasing variations in different models, hence providing the information that more number of parameters require larger computing times along with more memory, which was discussed in the section 5.1. The visual information in the Fig. 6 is extracted from the models fine-tuned onto the datasets: CIFAR 100 and Oxford IIIT Pet. The former consists of 60000 images of size 32x32. The latter consists of 7400 images of pets.



(a) Accuracy with Pre-Training and Fine-tuning Base

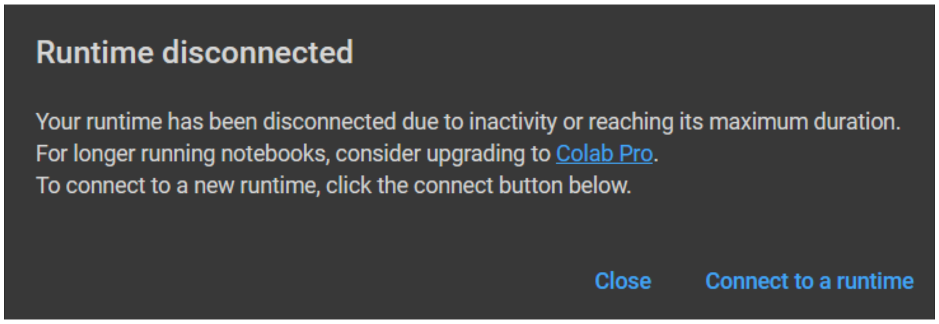


(b) Time required with Pre-Training and Fine-tuning

Figure 6. Overview of training and performance metrics of variants

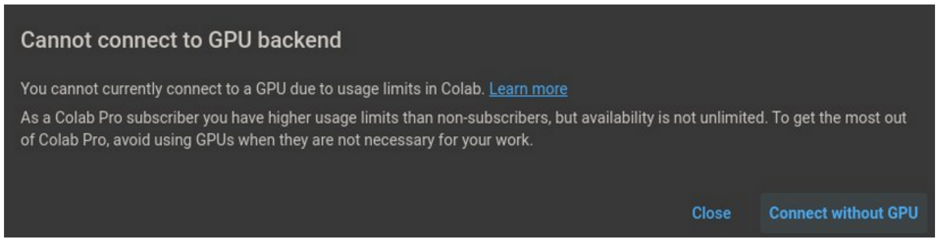
5.5 Common errors of resource management using Google Colab - GPU

The Fig. 6b gives us the information the some of the variants of transformer: ViT-L and ViT-B take around 11hrs and 5hrs respectively. Summing up the time contributes to more than 40hrs a week. In such cases Google limits the resources for the user and runtimes cannot be no-longer than an hour. The discussion on such problems: Colab - Github issues. The workflow in this report encountered a similar problem when resources got exhausted and runtime was disconnected while training ViT-B/8 model. The time is utilized was 1hr 3min before disconnection where the estimated time was 2hrs 30 min. The Fig. gives more information on the error.


Runtime disconnected

Your runtime has been disconnected due to inactivity or reaching its maximum duration.
For longer running notebooks, consider upgrading to Colab Pro.
To connect to a new runtime, click the connect button below.

Figure 7. Error on runtime disconnection


Cannot connect to GPU backend

You cannot currently connect to a GPU due to usage limits in Colab. Learn more

As a Colab Pro subscriber you have higher usage limits than non-subscribers, but availability is not unlimited. To get the most out of Colab Pro, avoid using GPUs when they are not necessary for your work.

Figure 8. Error on GPU resource exhaustion

6 Conclusion

In this study, we have investigated the feasibility of using the ViT for image recognition tasks, assuming that sufficient data is available. We first carried out theoretical research on the background of the ViTs and found that these transformer-based models do not generalize well when they are poorly trained. We then provided a mathematical model for the ViT to assist with its initialization, and compared it with a CNN (Convolutional Neural Network). Our second observation was that the number of parameters for simple transformers is enormous, which is not the case for CNNs.

To validate these two statements, we then developed two different methodologies for training the network from scratch using pre-defined configurations, and for fine-tuning pre-trained models to specific tasks. Both of these methodologies provided accurate validation for our statements, and we demonstrated their effectiveness with two examples in each case. Based on our analysis of these results, we can conclude that the use of huge pre-trained data, checkpoints, and fine-tuning is crucial for scaling up a ViT for any variant and task.

References

[1] X. Chen, C.-J. Hsieh, and B. Gong, “When vision transformers outperform resnets without pretraining or strong data augmentations,” *arXiv preprint arXiv:2106.01548*, 2021.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[3] A. Vaswani et al., “Attention is all you need,” 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[4] E. Muñoz, “Attention is all you need: Discovering the transformer paper,” 2017. [Online]. Available: https://towardsdatascience.com/attention-is-all-you-need-discovering-the-transformer-paper-73e5ff5e0634

[5] A. Dosovitskiy et al, “An image is worth 16x16 words: Transformers for image recognition at scale,” 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[6] A. Steiner, A. Kolesnikov, , X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, “How to train your vit? data, augmentation, and regularization in vision transformers,” *arXiv preprint arXiv:2106.10270*, 2021.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[9] C.-C. J. Kuo, “Understanding convolutional neural networks with a mathematical model,” *Journal of Visual Communication and Image Representation*, vol. 41, pp. 406–413, 2016.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2018. [Online]. Available: https://arxiv.org/abs/1810.04805