Transformers in Computer Vision and Hyperparameter exploration on ResNet

Efficient Deep Learning

GILLARD Antonino

2026/02/17

Paper Introduction

An Image is Worth 16×16 Words : Transformers for Image Recognition at Scale

Dosovitskiy et al., 2021

Context

- Image classification is mostly done by CNN
- Transformers : NLP tasks
- Attempts at using self attention mechanisms in image classification do not scale

ResNet like networks are still state of the art for image classification

Main Idea : Treat Images like texts in Transformers

Transformers in NLP

- Texts are split into **tokens** : 2-4 character long pieces
- Tokens are passed through an embedding (incl. positions)
- Pass them through transformers and then output

Vision Transformer (ViT)

- Split images in 16x16 patches
- Patches are flattened then passed through an embedding (incl. 2D position in the image)
- Pass them through transformers and then classification

ViT representation

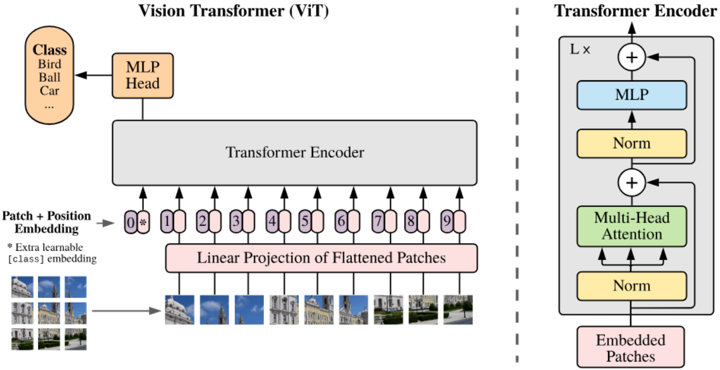


Figure 1: ViT Model Overview

Performance Benchmarks

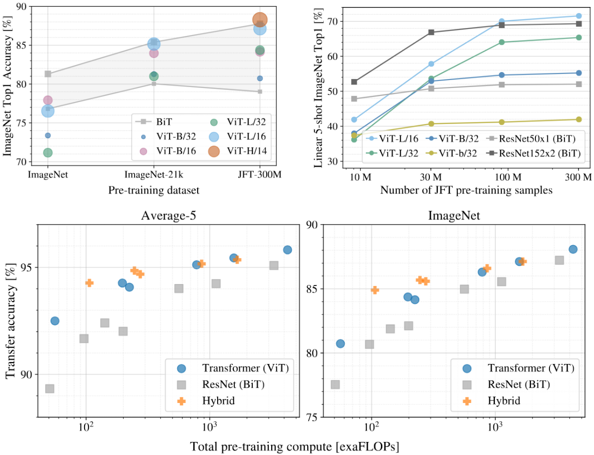


Figure 2: Model evalutations

Paper Conclusion

Model performance

- Reaches state of the art performance when trained on large datasets
- CNN models such as BiT (ResNet) still outperform when trained on smaller datasets

Good potential for image transformers but they are still inefficient.

There is still some work to be done ...

The paper only focuses on one task : **Image Classification**. Although results are promising in that field, work still has to be done on other tasks (e.g. Detection)

Hyperparameter exploration

ResNet18

- 18-layer CNN
- Residual connections
- Image classification

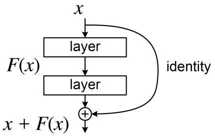


Figure 3: ResNet architecture

Focus of Hyperparameter exploration strategy : **Learning Rate**

Learning Rate	0.1	0.01	0.001
Testing Loss	0.67	0.54	0.43

Table 1: First runs by modifying LR. ResNet18 model with SGD optimizer (weight decay : 5×10^{-4}) and CosineAnnealing of period 20

Optimizer and Scheduler Configurations

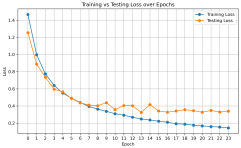
Weight Decay	LR	Loss Function	Epoch
5×10^{-6}	10^{-3}	Cross Entropy	≈ 25

Table 2: Common parameters across all trained models

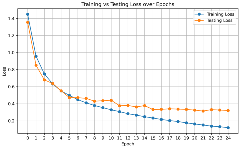
Optimizer	LR	Momentum	Scheduler	Patience
SGD	10^{-3}	0.9	CosineAnnealingLR	N/A
SGD	10^{-3}	0.9	ReduceLROnPlateau	5
Adam	10^{-3}	N/A	ReduceLROnPlateau	5

Table 3: Differences between trained models

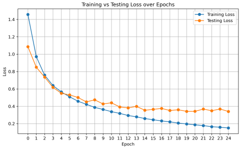
Training Loss Evolution



(a) Adam



(b) Cosine (SGD)



(c) ReduceLR (SGD)

Figure 4: Training loss per epoch for different optimizers and schedulers.

- Start to see some overfitting
- However, test accuracy went up during the plateau (+3%)

Model Accuracy

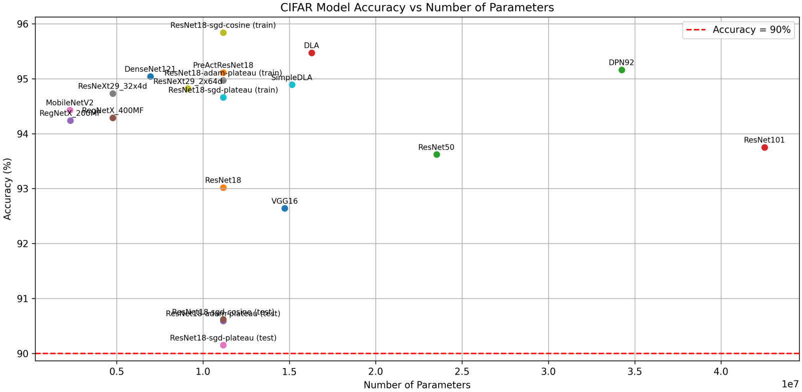


Figure 5: Accuracy (%) of each model per number of parameters