Transformers for image recognition

Ahmad HARKOUS

An Vinh Lala

Valentin Dauzere-Peres

https://arxiv.org/pdf/2010.11929

Dec 2024

Data Set

- **CIFAR-10 dataset**: 60,000 32x32 color images across 10 classes (e.g., airplanes, automobiles, birds)

- **Training Data:** A subset of 20% of the training set was used

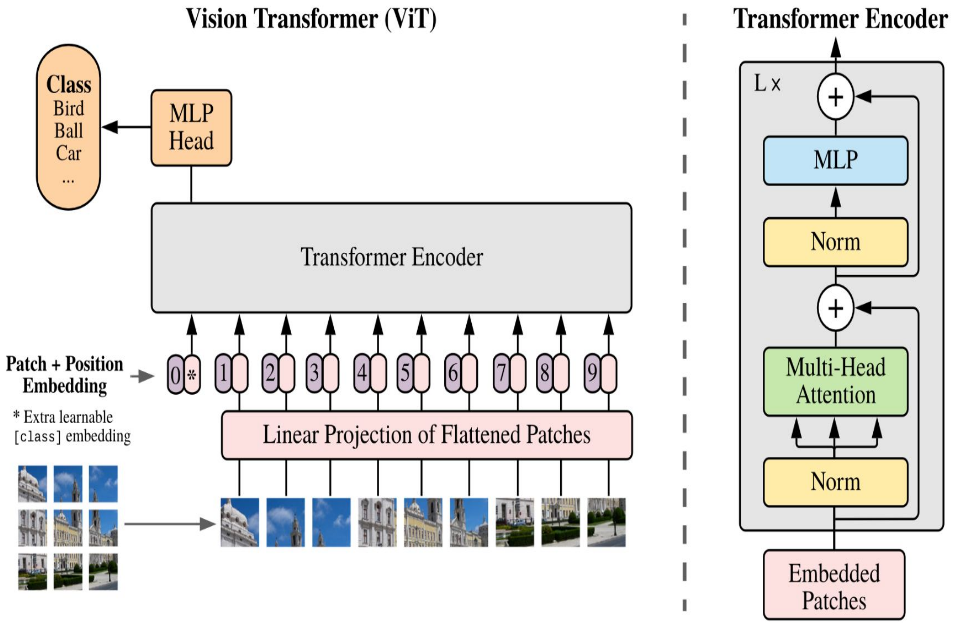
- **Test Data:** The full 10,000-image test set was retained

Model Architecture Overview

1. **Patch Embeddings**

2. **Transformer Encoder**

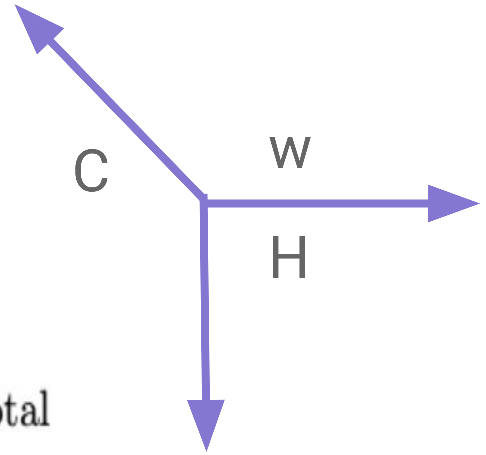
3. **Classification Head**



https://arxiv.org/pdf/2010.11929

1 > Patch Embeddings: 3D -> 1D

> Original 3D vector:



> Dividing the Image into Patches

Given an input image I with dimensions $H \times W \times C$, where:

- H: Height of the image.
- W: Width of the image.
- C: Number of channels (e.g., 3 for RGB).

The image is divided into non-overlapping patches of size $P \times P$. The total number of patches N is calculated as:

$$N = \frac{H \times W}{P^2}$$

> Flattening Each Patch

Each patch of size $P \times P \times C$ is flattened into a 1D vector of size:

$$\text{Patch Size} = P^2 \cdot C$$

1 > Patch Embeddings: 1D -> D-dimensional

> Linear Projection

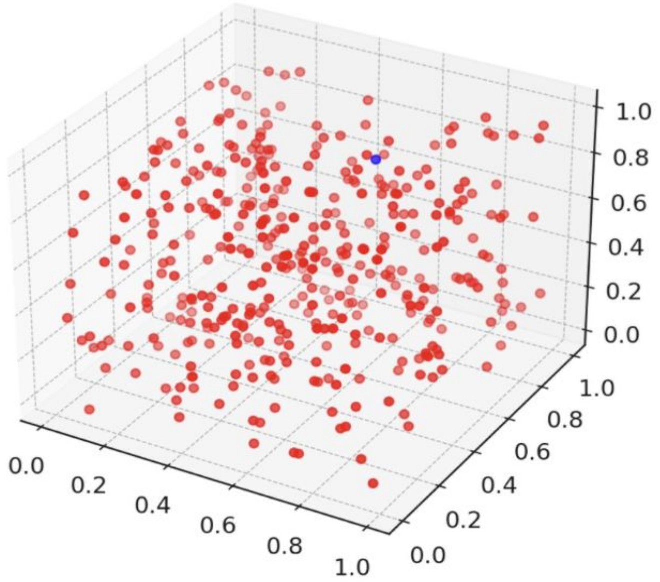
Each flattened patch is linearly projected into a D-dimensional embedding space:

$$\mathbf{e}_p = \mathbf{W} \cdot \mathbf{x}_p + \mathbf{b}$$

where:

- $\mathbf{x}_p$: Flattened patch (of size $P^2 \cdot C$).
- $\mathbf{W}$: Learnable weight matrix of size $D \times (P^2 \cdot C)$.
- $\mathbf{b}$: Bias term.

The result is an embedding $\mathbf{e}_p$ of size D for each patch.



The embeddings of all N patches are concatenated to form a sequence:

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N]$$

1 > Patch Embeddings: D-dimensional + CLS + Position

> Prepending the [CLS] Token

$$\mathbf{E}_{\text{input}} = [[\text{CLS}], \mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N]$$

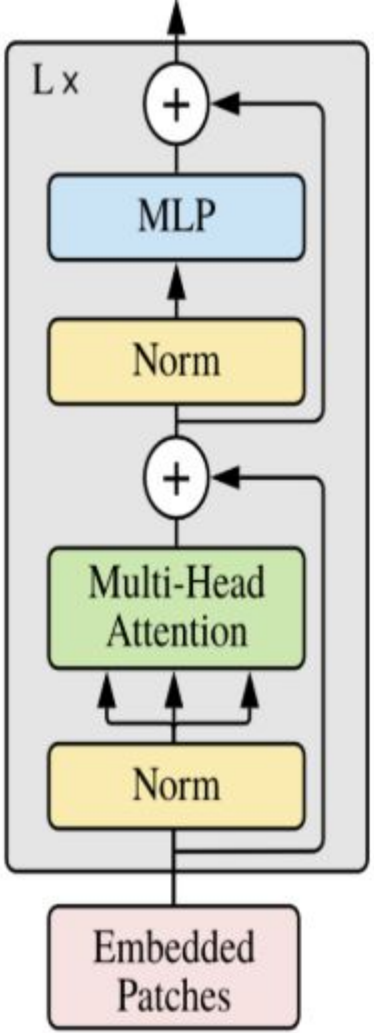
where $\mathbf{E}_{\text{input}}$ is of size $(N+1) \times D$.

> Adding Positional Encoding

$$\mathbf{E}_{\text{input}} = \mathbf{E}_{\text{input}} + \mathbf{P}$$

where $\mathbf{P}$ is the positional encoding matrix of size $(N+1) \times D$.

2 > Transformer Encoder



1. **Multi-Head Self-Attention (MHSA)**

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{D_k}}\right) \cdot V$$

2. **Feed-Forward Networks (FFN)**

$$\text{FFN}(x) = \text{GELU}(x \cdot \mathbf{W}_1 + \mathbf{b}_1) \cdot \mathbf{W}_2 + \mathbf{b}_2$$

Layer normalization (LN) is applied before every block, and residual connections after every block of both the MHSA and FFN(MLP).

3 > Classification Head

$$\mathbf{z}_{\text{CLS}} = \text{Transformer Output}[\text{CLS}]$$

$$\text{Logits} = \mathbf{z}_{\text{CLS}} \cdot \mathbf{W}_{\text{head}} + \mathbf{b}_{\text{head}}$$

MLP HEAD

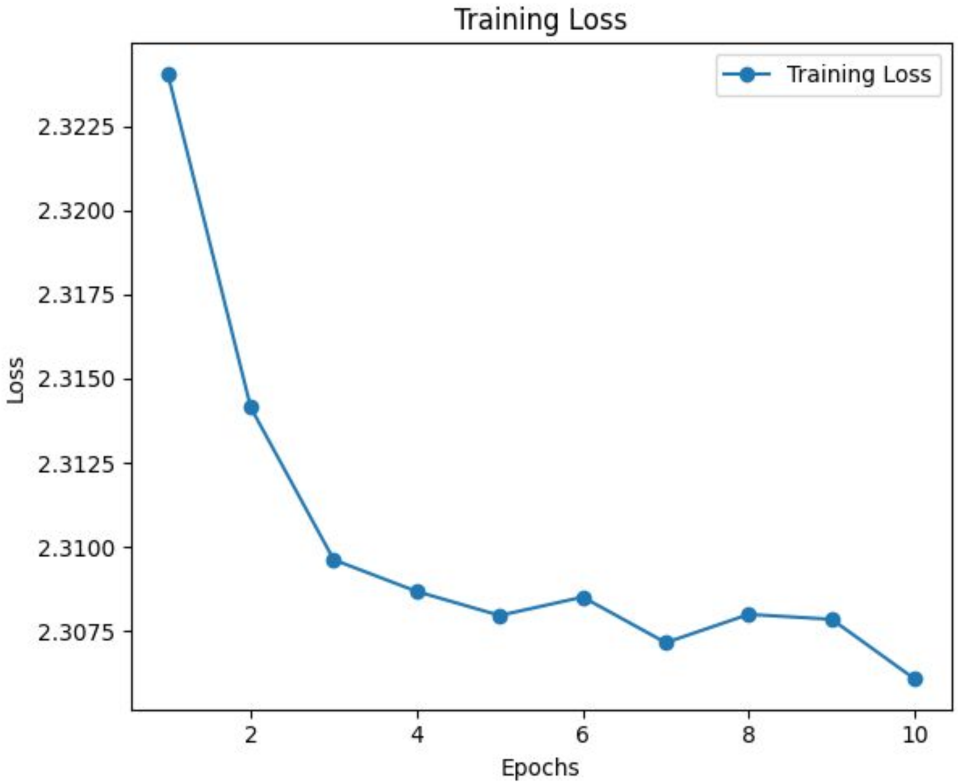
final output is a vector of size num_classes, containing the predicted class logits.

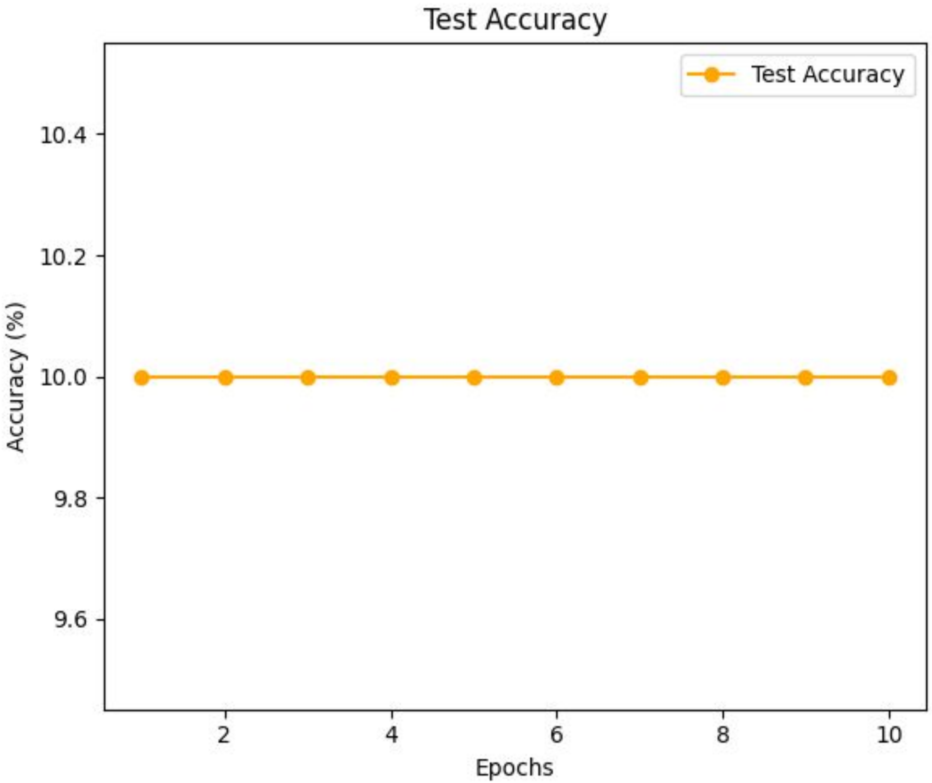
$$\mathbf{x} = \left[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\right]$$

Simplifications

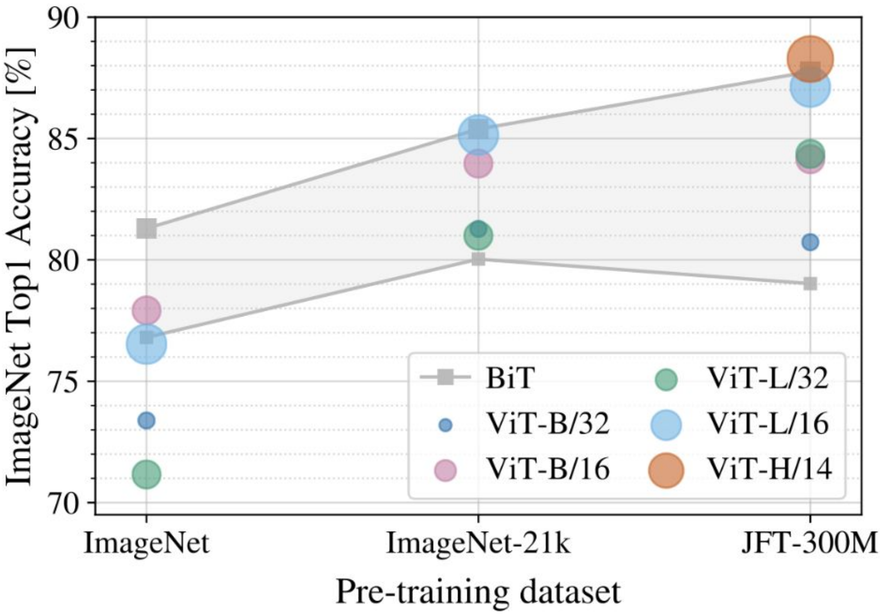
- Using fewer Transformer layers (6 layers instead of the original 12)
- Reducing embedding dimensions D to 64
- Limiting self-attention heads to 4
- Fixed learning rate and fewer epochs for tuning,

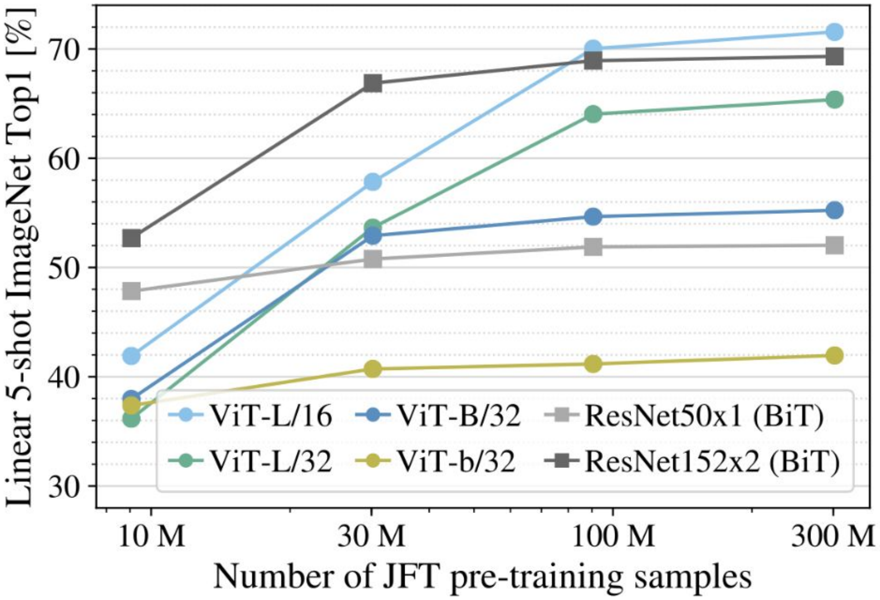
Performances





Article Performances





https://arxiv.org/pdf/2010.11929

Questions ?