End Evaluation Report: Vision Transformer Based Image Classification on CIFAR-10

Kanhaiya Kumar(240515)
Supervised by Prof. Rajesh Hegde ,EEA

February 7, 2026

1 Introduction

Vision Transformers (ViTs) extend the transformer architecture, originally developed for natural language processing, to vision tasks by modeling images as sequences of patch embeddings. Unlike convolutional neural networks, Vision Transformers rely on self-attention mechanisms to capture global relationships across an image. In this work, a Vision Transformer classifier is implemented from scratch and evaluated on the CIFAR-10 dataset using PyTorch.

2 Dataset Description

The CIFAR-10 dataset is a standard benchmark for image classification. It consists of 60,000 RGB images of size 32×32, divided into 50,000 training images and 10,000 test images. The dataset contains 10 object classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class is uniformly represented.

3 Preprocessing and Regularization Techniques

3.1 Data Preprocessing

3.1.1 Image Resizing

The original CIFAR-10 images of size 32×32 were resized to 128×128 before being fed to the Vision Transformer.

Effect: Increasing image resolution increases the number of patches extracted from each image, allowing finer spatial information to be preserved. However, it also increases computational cost and attention complexity.

3.1.2 Normalization

Each image channel was normalized using a mean of 0.5 and standard deviation of 0.5:

$$x_{norm} = \frac{x - 0.5}{0.5}$$

Effect: Normalization ensures numerical stability, improves gradient flow, and accelerates convergence during training.

3.1.3 Data Augmentation

Random horizontal flipping was applied to training images.

Effect: Data augmentation increases data diversity and reduces overfitting, which is particularly important for Vision Transformers trained on relatively small datasets.

3.2 Regularization Techniques

3.2.1 Dropout

Dropout with probability 0.1 was applied after positional embeddings and within the feed-forward layers of transformer blocks.

Effect: Dropout prevents co-adaptation of neurons and improves the model's generalization capability.

3.2.2 Weight Decay

L2 regularization was applied through the AdamW optimizer.

Effect: Weight decay penalizes large weights, leading to smoother decision boundaries and more stable training.

4 Hyperparameter Tuning

Multiple hyperparameters were tuned to balance computational efficiency and classification performance.

Hyperparameter	**Final Value**
Image size	128×128
Patch size	8×8
Embedding dimension	256
Transformer depth	4
Number of attention heads	4
Dropout	0.1
Optimizer	AdamW
Loss function	Cross-Entropy
Epochs	15

Table 1: Final hyperparameter configuration

Observation: Increasing image size significantly increases the number of patches ($(128/8)^2 = 256$). With a shallow transformer depth of 4 layers, the model exhibits limited capacity to model long-range dependencies, resulting in moderate test accuracy.

5 Vision Transformer Mechanism

5.1 Patch Embedding

Each input image is divided into non-overlapping patches of size 8×8. These patches are flattened and projected into a 256-dimensional embedding space using a convolutional layer.

```
self.projection = nn.Conv2d(
    in_channels, embed_dim,
    kernel_size=patch_size,
    stride=patch_size
)
```

5.2 Positional Embedding

Since transformers are permutation invariant, learnable positional embeddings are added to preserve spatial order:

$$z_0 = x_{patch} + E_{pos}$$

```
x = x + self.pos_embed
```

5.3 Class Token

A learnable classification token (`[CLS]`) is prepended to the patch sequence. The final representation of this token is used for classification.

```
cls_tokens = self.cls_token.expand(B, -1, -1)
x = torch.cat((cls_tokens, x), dim=1)
```

5.4 Transformer Encoder

Each transformer encoder block consists of:

- Multi-Head Self-Attention
- Feed-Forward MLP
- Layer Normalization
- Residual Connections

The self-attention mechanism is defined as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

5.5 Classification Head

After passing through the transformer encoder layers, the output corresponding to the `[CLS]` token is normalized and passed through a linear layer to produce class logits.

```
output = self.head(x[:, 0])
```

Metric	Value
Training accuracy	∼80–85%
Test accuracy	∼65–70%
Epochs	15

Table 2: Final model performance

6 Results

The model was trained for 15 epochs. Training accuracy increased steadily, while test accuracy saturated around 65–70%.

7 Discussion

The results indicate that Vision Transformers require sufficient model depth to effectively process a large number of patch tokens. While higher image resolution improves spatial detail, shallow architectures struggle to capture long-range dependencies, leading to underfitting. This highlights the importance of balancing input resolution and transformer depth.

8 Conclusion

A Vision Transformer-based classifier was implemented and evaluated on the CIFAR-10 dataset. Through preprocessing, regularization, and hyperparameter tuning, the model achieved moderate classification performance. The study demonstrates that while Vision Transformers are powerful, careful architectural design is necessary when applying them to small-scale datasets.