Vision Transformer (ViT) Classifier on CIFAR-10

Mayank Agarwal 240635

February 7, 2026

Abstract

This report details the implementation and analysis of a Vision Transformer (ViT) trained from scratch on the CIFAR-10 dataset. Unlike Convolutional Neural Networks (CNNs), which rely on inductive biases like translation invariance and locality, ViT interprets images as sequences of patches, relying on self-attention mechanisms to capture global dependencies. This report explores the fundamental architecture of ViT, the specific preprocessing and regularization techniques required to train it on small datasets, and the hyperparameter tuning necessary for convergence on 32×32 images.

1 Introduction to Vision Transformers

1.1 What is a Vision Transformer?

The Vision Transformer (ViT) is an architecture that applies the standard Transformer encoder, originally designed for Natural Language Processing (NLP), directly to images with minimal modifications. Instead of processing pixels via sliding convolutional kernels, ViT splits an image into fixed-size patches, flattens them, and projects them into a sequence of embeddings.

1.2 Pros and Cons

- **Pros:**
  - **Global Context:** Through Self-Attention, every patch can attend to every other patch from the very first layer, capturing long-range dependencies that CNNs only capture in deeper layers.
  - **Scalability:** ViTs scale incredibly well with data and compute, often outperforming CNNs on massive datasets (e.g., JFT-300M, ImageNet-21k).
- **Cons:**
  - **Lack of Inductive Bias:** ViTs lack the inherent assumptions of locality (neighboring pixels are related) and translation invariance (an object is the same regardless of position) found in CNNs.
  - **Data Hunger:** Because they must learn these spatial relationships from scratch, they typically require more data or stronger regularization to generalize well.

2 Methodology and Architecture

The implementation follows the standard ViT architecture but adapted for the smaller resolution of CIFAR-10.

2.1 Core Mechanism

The processing pipeline consists of four distinct stages:

1. **Patch Embedding:** The input image $x \in \mathbb{R}^{H \times W \times C}$ is divided into patches of size $P \times P$. For CIFAR-10 (32×32) and patch size $P = 4$, we obtain $N = (32/4)^2 = 64$ patches. These are flattened and projected to an embedding dimension D.

2. **Positional Embedding:** Since the transformer is permutation invariant (it sees the patches as a bag of vectors), learnable positional embeddings are added to the patch embeddings to retain spatial information.

3. **Transformer Encoder:** The sequence passes through L layers of Transformer blocks. Each block contains:

   - Layer Normalization (LN)
   - Multi-Head Self-Attention (MSA)
   - Multi-Layer Perceptron (MLP)
   - Residual Connections

4. **Classification Head:** A learnable `[CLS]` token is prepended to the sequence. The final state of this token serves as the global image representation used for classification.

2.2 Code Implementation: The Transformer Block

The following PyTorch code illustrates the core repeating unit of the ViT architecture used in this experiment:

```
class Block(nn.Module):
    def __init__(self, dim, num_heads, mlp_ratio=4.,
                 qkv_bias=False, drop=0., attn_drop=0.):
        super().__init__()
        self.norm1 = nn.LayerNorm(dim)
        self.attn = Attention(dim, num_heads=num_heads,
                              qkv_bias=qkv_bias,
                              attn_drop=attn_drop,
                              proj_drop=drop)
        self.norm2 = nn.LayerNorm(dim)
        self.mlp = MLP(dim, int(dim * mlp_ratio), dim, drop=drop)

    def forward(self, x):
        # Residual connections around Attention and MLP
        x = x + self.attn(self.norm1(x))
        x = x + self.mlp(self.norm2(x))
        return x
```

Listing 1: Transformer Block Implementation

3 Regularization and Preprocessing

One of the most critical aspects of training ViT on small datasets like CIFAR-10 is preventing overfitting. Due to the lack of inductive bias, the model can easily memorize the training data without generalizing.

3.1 Preprocessing Techniques

- **Normalization:** Images are normalized using CIFAR-10 specific statistics:

$$\mu = (0.4914, 0.4822, 0.4465), \quad \sigma = (0.2023, 0.1994, 0.2010)$$

  This ensures inputs have zero mean and unit variance, which is crucial for the stability of the gradients in the Transformer layers.

- **Resolution Preservation:** Unlike standard ViT implementations that upsample small images to 224×224, this implementation maintains the native 32×32 resolution to reduce computational overhead, relying on a smaller patch size to compensate (see Section 4).

3.2 Regularization Strategies

To simulate the robustness provided by CNNs, strong regularization is applied:

1. **Data Augmentation:**

   - `RandomCrop`: The images are padded by 4 pixels and randomly cropped back to 32×32.
   - `RandomHorizontalFlip`: Applied with a probability of $p = 0.5$.

   Effect: This forces the model to learn translation invariance (recognizing a car whether it is centered or shifted) and orientation robustness manually.

2. **Dropout:**

   - Applied within the Attention mechanism (`attn_drop`) and after the MLP projection (`drop`).
   - Rate: 0.1.

   Effect: Prevents co-adaptation of neurons, forcing the network to learn redundant and robust features.

3. **Weight Decay:**

   - Applied via the AdamW optimizer with a value of $\lambda = 0.05$.

   Effect: This is significantly higher than typical CNN weight decay (usually $1e-4$), reflecting the need to heavily penalize complexity in the high-capacity Transformer model.

4 Hyperparameter Tuning

The performance of ViT is highly sensitive to hyperparameters. The following specific configurations were chosen to adapt the ViT architecture.

Hyperparameter	**Value**	**Rationale**
Patch Size	4×4	Critical for 32×32 inputs. Size 16 would yield only 4 patches.
Embed Dimension	192	Reduced from 768 to prevent overfitting on small data.
Depth (Layers)	6	Reduced from 12 to lower capacity and training time.
Num Heads	8	Ensures diversity in attention representation.
Learning Rate	3×10^{-3}	Higher initial LR required for Transformers.
Optimizer	AdamW	Handles weight decay explicitly, crucial for ViT.
Scheduler	Cosine	Smooth decay helps convergence to flatter minima.

Table 1: Hyperparameter Configuration for CIFAR-10 ViT

4.1 Significance of Patch Size

The choice of `patch_size=4` is the most significant architectural decision. The sequence length N is determined by $N = (H \times W)/P^2$.

- If $P = 16$ (standard): $N = (32 \times 32)/256 = 4$. This sequence is too short for the attention mechanism to learn meaningful relationships.
- If $P = 4$ (ours): $N = (32 \times 32)/16 = 64$. This provides a rich sequence length comparable to sentence lengths in NLP tasks.

5 Experimental Results

The model was trained for roughly 100 epochs.

5.1 Training Progression

The following observations are drawn from the training logs:

- **Early Epochs (1-10):** The model starts with random guessing ($\sim 10\%$) and rapidly learns low-level features, reaching $\sim 25\%$ accuracy.
- **Mid Training (Epoch 30-50):** The accuracy climbs to $\sim 60 - 65\%$. The loss decreases consistently, and the gap between training and test accuracy stays stable, indicating the regularization (Dropout/Data Aug) is working to keep the generalization gap in check.
- **Late Training (Epoch 80+):** The model stabilizes around **65-67%** test accuracy.

5.2 Performance Graph

The loss curves (Training vs. Validation) show that while the training loss continues to drop (reaching near 0.7), the test loss plateaus around 0.95. This is a classic signature of the capacity limits of training a Transformer from scratch on only 50,000 images without pre-training.

6 Conclusion

This report demonstrated the feasibility of training a Vision Transformer from scratch on CIFAR-10. While ViTs generally excel with massive pre-training, we successfully achieved convergence by:

1. Drastically reducing the patch size to maintain sequence length.
2. Reducing the model depth and embedding dimension.
3. Applying aggressive regularization (High Weight Decay, Data Augmentation) to compensate for the lack of inductive biases.

Future improvements could involve pre-training to further boost accuracy beyond the 67% baseline achieved here.