Vision Transformer (ViT) – End Evaluation Report

Ayush Chaudhary

240237

---

1. Regularization and Preprocessing Techniques Used

1.1 Data Preprocessing & Augmentation

The following preprocessing and augmentation steps were applied to the training data:

- **RandomCrop(32, padding=4)**
  1. Adds translation invariance by randomly cropping images with padding.
  2. Helps the model generalize better to shifted objects.

- **RandomHorizontalFlip()**
  1. Introduces left–right invariance.
  2. Very effective for natural image datasets like CIFAR-10.

- **ColorJitter(brightness, contrast, saturation, hue)**
  1. Prevents overfitting to specific lighting conditions.

2 Encourages robustness to color variations.

- **ToTensor()**
  1. Converts images to PyTorch tensors and scales pixel values to [0,1].

- **Normalize(mean=(0.5,0.5,0.5), std=(0.5,0.5,0.5))**
  1. Centers data around zero mean.
  2. Improves optimization stability and convergence speed.

Effect:
These augmentations serve as *implicit regularization*, significantly reducing overfitting, which is especially important for ViTs, which lack strong inductive biases compared to CNNs.

1.2 Explicit Regularization Techniques

- **Weight Decay (L2 Regularization)**
  Implemented via **AdamW optimizer** with `weight_decay = 1e-4`.

1. Penalizes large weights.
2. Encourages smoother and more generalizable solutions.

- **Layer Normalization**
  Used inside every Transformer block.

1. Stabilizes training of deep transformer stacks.
2. Reduces internal covariate shift.

Note: Dropout was not explicitly used. Regularization mainly relied on data augmentation and weight decay.

---

2. Hyperparameter Selection and Tuning

This implementation used **manual hyperparameter selection** rather than automated search.

2.1 Key Hyperparameters

Hyperparameter	Value	Reasoning
Patch size	2×2	Generates sufficient tokens (256) for CIFAR-10
Embedding dimension	128	Balance between capacity and computation
Transformer layers	6	Deep enough to learn global context
Attention heads	4	Ensures embed_dim divisible by heads
Learning rate	3×10^{-4}	Standard for AdamW + ViT
Weight decay	1×10^{-4}	Prevents overfitting
Optimizer	AdamW	Better decoupled regularization
Epochs	30	Sufficient for convergence on CIFAR-10

2.2 Tuning Strategy

- Learning rate chosen based on standard ViT practices.
- Patch size and embedding dimension selected to suit small-resolution images.
- Number of heads and layers tuned to avoid excessive computation on Colab GPU.

Effect:
Although no grid/random search was used, the chosen hyperparameters achieved stable training and reasonable accuracy for a ViT trained from scratch.

---

3. Vision Transformer Mechanism (with Code Mapping)

3.1 Patch Embedding

Each image is divided into non-overlapping patches and projected into an embedding space using a convolution layer:

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

1. Converts image → sequence of patch tokens
2. Replaces convolutional feature extraction

3.2 Positional Encoding

- Learnable positional embeddings are added to patch embeddings.
- Necessary because transformers are permutation-invariant.
- Injects spatial information into the token sequence.

3.3 Transformer Encoder Block

Each block consists of:

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

2. Captures global relationships between patches.

3. **MLP (Feed Forward Network)**

4. Learns non-linear representations.

5. **Residual Connections + LayerNorm**

6. Ensures stable gradient flow.

```
x = x + self.attn(self.norm1(x), self.norm1(x), self.norm1(x))[0]
x = x + self.mlp(self.norm2(x))
```

3.4 Classification Head

- A special **[CLS] token** represents the entire image.
- Final linear layer maps embedding → class logits.

```
self.head = nn.Linear(embed_dim, num_classes)
```

Enables end-to-end image classification.

---

4. Results and Observations

- Model successfully converged within 30 epochs.
- Training was stable due to LayerNorm and AdamW.
- Data augmentation played a crucial role in preventing overfitting.
- Performance is competitive given the absence of pretraining.

Key Insight:
Vision Transformers can work well on small datasets *only when strong regularization and augmentation are applied*.

---

5. Conclusion

This project demonstrates:

1. End-to-end implementation of a Vision Transformer from scratch
2. Effective use of preprocessing and regularization
3. Practical understanding of ViT internals and training dynamics

Despite limited hyperparameter tuning, the model achieves solid performance, highlighting the importance of architectural design and data augmentation in transformer-based vision models.

End of Report