Project Report: Vision Transformer (ViT)

1. Regularization & Preprocessing

Vision Transformers (ViTs) lack the inductive bias of CNNs, making them prone to overfitting on small datasets like CIFAR-10. To mitigate this, the following techniques were implemented:

• **Data Augmentation:** Random Crop (padding=4) and Random Horizontal Flip were used to introduce shift invariance and diversity.

• **Normalization:** Input images were normalized using mean (0.4914, 0.4822, 0.4465) and std (0.2023, 0.1994, 0.2010) to stabilize gradients.

• **Weight Decay (1e-2):** Applied via AdamW optimizer to penalize large weights.

• **Dropout (0.1):** Applied after position embeddings and within the encoder to prevent co-adaptation.

• **Pre-Layer Normalization:** Setting *norm_first*=*True* was critical for training stability, ensuring normalization is applied before attention blocks.

2. Hyperparameter Tuning

The architecture was scaled down to a 'Tiny' variant to suit the 32x32 resolution of CIFAR-10.

Hyperparameter	Value	Rationale
Patch Size	4	Creates 64 patches ((32/4)^2), providing sufficient sequence length.
Embed Dimension	128	Reduced from 768 to prevent overfitting.
Num Layers	4	Shallow depth is sufficient for CIFAR-10.
Num Heads	8	Allows attention to 8 different subspaces.
Learning Rate	5e-4	Moderate rate suitable for AdamW.
Epochs	20	Sufficient for loss convergence (~0.85).

3. Mechanism of Vision Transformer

The model processes images as sequences of patches, similar to how NLP models process words.

1. Patch Embedding: A Convolutional layer (kernel=4, stride=4) extracts 4x4 patches and projects them to 128 dimensions.
2. Positional & CLS Embeddings: Learnable position vectors are added to retain spatial information. A 'Class Token' is prepended to aggregate global features.
3. Transformer Encoder: The sequence passes through 4 layers of Multi-Head Self-Attention (MSA) and Feed-Forward Networks (MLP).

Code Implementation (Forward Pass):

```
def forward(self, x):
```

```
# 1. Patch Embedding
x = self.patch_embed(x)
x = x.flatten(2).transpose(1, 2)    # (Batch, 64, 128)

# 2. Add CLS Token & Position Embedding
cls_tokens = self.cls_token.expand(x.shape[0], -1, -1)
x = torch.cat((cls_tokens, x), dim=1)
x = x + self.pos_embedding

# 3. Transformer Layers
x = self.transformer(x)

# 4. Classification
return self.mlp_head(x[:, 0])
```

4. Results & Analysis

The model was trained for 20 epochs. The training loss demonstrated consistent convergence, decreasing from 1.78 to 0.85.

Final Validation Accuracy: 70.64%

This result validates that a pure attention-based architecture can successfully learn spatial features from scratch on CIFAR-10, even without large-scale pre-training.