Image Classification with Vision Transformers: A Theoretical Exploration

Introduction

This project implements a **Vision Transformer (ViT)** model for image classification. The ViT is a novel architecture that applies transformer models, originally designed for natural language processing, to image data. The model is trained on a dataset of images and classifies them into different categories. The project covers data preprocessing, model training, and evaluation, offering insights into the strengths and limitations of Vision Transformers compared to Convolutional Neural Networks (CNNs).

Vision Transformer Architecture

Patch Embeddings

Unlike CNNs, which operate on the full image, the ViT divides an image $x \in \mathbb{R}^{H \times W \times C}$ into fixed-size patches. Each patch is flattened and projected into a feature vector. Let:

- H and W be the height and width of the image.
- C be the number of channels.
- P be the patch size (e.g., $P = 16$).

The image is split into N patches, where:

$$N = \left(\frac{H}{P}\right) \times \left(\frac{W}{P}\right)$$

Each patch x_p is flattened into a vector $x_p \in \mathbb{R}^{P^2 \cdot C}$ and linearly projected into an embedding:

$$z_p = W_p x_p + b_p$$

where W_p and b_p are learnable parameters.

Position Embeddings

Since transformers do not inherently capture positional information, **position embeddings** E_{pos} are added to the patch embeddings:

$$z_0 = [z_{\text{cls}}; z_1 + E_1; z_2 + E_2; \ldots; z_N + E_N]$$

where z_{cls} is a special `[CLS]` token used for classification.

Transformer Encoder

The patch embeddings are passed through a series of **Transformer Encoder** layers. Each layer consists of:

- **Multi-Head Self-Attention (MHSA)**:

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

  where:

  - $Q = W_Q z$ (queries),
  - $K = W_K z$ (keys),
  - $V = W_V z$ (values),
  - d_k is the dimension of the keys.

- **Feedforward Network (FFN)**:

$$\text{FFN}(z) = \text{ReLU}(W_1 z + b_1)W_2 + b_2$$

Each encoder layer can be represented as:

$$z' = \text{LayerNorm}(z + \text{MHSA}(z)), \quad z'' = \text{LayerNorm}(z' + \text{FFN}(z'))$$

Classification Head

The final representation of the `[CLS]` token z_{cls} is passed through a linear layer to produce the class logits:

$$y = W_{\text{cls}} z_{\text{cls}} + b_{\text{cls}}$$

Training

Data Preprocessing

The images were preprocessed with the following steps:

- **Resizing**: All images resized to 224×224.
- **Normalization**: Pixel values normalized to $[0, 1]$ or standardized using ImageNet statistics.
- **Augmentation**: Random cropping, flipping, and color jittering to improve generalization.

Loss Function

The model was trained using the **Cross Entropy Loss**:

$$L = -\sum_{i=1}^{C} g_i \log(p_i)$$

where:

- g_i is the ground-truth label (one-hot encoded).
- p_i is the predicted probability for class i.

Optimization

- **Optimizer**: Adam
- **Learning Rate**: $\eta = 0.0001$
- **Batch Size**: 32
- **Epochs**: 50

Evaluation

The model performance was evaluated using:

- **Accuracy**:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

- **Confusion Matrix**: Visualizing true positives, false positives, and false negatives.

Insights and Comparisons

Data Dependency

The Vision Transformer requires large datasets to perform well. With smaller datasets, the model tends to underperform compared to CNNs due to the lack of inductive biases (e.g., spatial locality and translational invariance).

Efficiency

- **On Small Datasets**: CNNs are more efficient and achieve better accuracy.
- **On Large Datasets**: The Vision Transformer demonstrates superior performance by leveraging global self-attention to capture long-range dependencies.

Conclusion

This project provided a theoretical and practical exploration of the Vision Transformer for image classification. While ViTs are powerful models for large datasets, they are less efficient than CNNs when working with limited data. Understanding this trade-off is crucial when selecting models for real-world applications.