Mid-Evaluation Report: Vision Transformer

Aayushman Tripathi
240025

1. Project Overview

The primary objective of this project is to construct a Vision Transformer (ViT) architecture from scratch. Rather than relying solely on high-level abstractions provided by deep learning frameworks, this project follows a "first principles" approach. The development roadmap involves building the foundational mathematical engines (Linear Algebra, Calculus), constructing standard neural networks (MLP/CNN), optimizing them, and implementing the specific components of the Transformer architecture (Self-Attention) alongside sequence modeling fundamentals (RNNs).

This mid-evaluation report covers the progress from **Assignment 1 to Assignment 5**.

2. Weekly Progress Timeline-

Week 1: Foundations & Data Pipeline Design

Objective: To establish a robust codebase using Python and NumPy, focusing on vectorization and data preprocessing pipelines.

- **Object-Oriented Design:** Developed a modular DataSample class to encapsulate feature vectors and labels. Implemented methods for in-place normalization (min_max_norm) to scale features to the [0, 1] range.
- **Advanced Vectorization:** Leveraged NumPy broadcasting to perform complex matrix operations without explicit loops. This included implementing the mathematical core of the **Softmax function** (exponentiation and row-wise normalization), a critical component for the attention mechanism.
- **Stratified Sampling:** Designed a custom data splitting algorithm for the "StudentsPerformance" dataset. Unlike random splitting, this approach sorted data by class before systematic sampling, ensuring that training and testing distributions remained statistically identical.

Week 2: Neural Network from Scratch (MLP)

Objective: To demystify deep learning by building a Multilayer Perceptron (MLP) entirely from scratch using only NumPy.

- **Architecture:** Constructed a 2-layer neural network (784 Input -> 128 Hidden ->10 Output) to classify handwritten digits from the **MNIST** dataset.
- **Mathematical Derivation:** Manually derived and implemented the **Backpropagation** algorithm. This involved calculating the partial derivatives (dZ, dW, db) for the chain rule to update weights via Gradient Descent.
- **Pipeline:** Implemented the full training loop, including One-Hot Encoding for labels and random weight initialization scaled by 0.001 to prevent exploding gradients.

- **Outcome:** The model achieved **93.08% training accuracy** and **91.31% test accuracy** after 1,000 iterations, proving the correctness of the manual implementation.

Week 3: Theoretical Foundations & Evaluation

Objective: To master the theoretical concepts governing model optimization and rigorous evaluation.

- **Metric Analysis: Moved beyond simple accuracy to implement a suite of evaluation metrics including Precision, Recall, and F1-Score. Analyzed the Confusion Matrix to understand specific class-level errors.**
- **Optimization Theory: Conducted a comparative analysis of optimization algorithms:**
    - **Batch vs. Stochastic Gradient Descent: Analyzed trade-offs between stability and speed.**
    - **Advanced Optimizers: Studied the mechanics of Momentum, RMSProp, and Adam optimizers.**
- **Regularization: Investigated techniques to mitigate overfitting, specifically analyzing Dropout (random neuron deactivation) and L1/L2 Regularization.**

Week 4: Deep Learning Optimization (Batch Normalization)

Objective: To implement modern optimization techniques required for deeper networks, specifically focusing on Batch Normalization within a CNN context.

- **Mathematical Logic:** Derived the equations for Batch Normalization, addressing "Internal Covariate Shift" by normalizing layer inputs to a stable distribution.
- **Custom Implementation:** Engineered a custom Keras layer wrapper (BatchNormalizedLayer) that systematically applies normalization before activation functions.
- **Experimentation (CIFAR-10):**
    - Designed a VGG-style Convolutional Neural Network (CNN) as a baseline.
    - Integrated the custom Batch Normalization layer into the architecture.
    - **Result:** The normalized model demonstrated significantly faster convergence and stability compared to the baseline, which struggled with high loss variance.

Week 5: Sequence Modeling & Attention Mechanisms

Objective: To bridge the gap between sequential data processing (RNNs) and the parallel processing of Transformers (Attention).

Part A: Recurrent Neural Networks (RNNs)

- **RNN Cell Implementation:** Coded the fundamental RNN cell from scratch.

- **Text Generation:** Applied the model to a character-level language generation task (generating Dinosaur names). Implemented a sampling function to generate novel sequences from the learned probability distribution

- **Gradient Clipping:** Implemented gradient clipping to prevent the "exploding gradient" problem common in RNN training.