Vision Transformer (ViT)

Mid-Evaluation Report

Name: KANHAIYA KUMAR
Roll No.: 240515

---

1. Introduction

The objective of this project is to study and practically implement **Vision Transformers (ViTs)** for image classification tasks. The project begins with revisiting core machine learning and deep learning fundamentals and gradually transitions to modern attention-based architectures used in computer vision.

Initially, image classification is explored using traditional neural networks and Convolutional Neural Networks (CNNs). Subsequently, transformer-based ideas such as self-attention are adapted for vision by dividing images into fixed-size patches and processing them as token sequences. The CIFAR-10 dataset is used throughout the project to compare the performance of CNNs and Vision Transformer models.

---

2. Week 0

2.1 Python Foundations

Before starting machine learning concepts, a revision of Python programming fundamentals was conducted. This included an overview of Python syntax, control structures, functions, object-oriented programming, and data structures. Libraries such as **NumPy**, **Pandas**, and **Matplotlib** were introduced for numerical computation, data handling, and visualization.

Key topics covered included array operations, indexing and broadcasting, dataframe manipulation, and plotting techniques, all of which form the backbone of later ML implementations.

2.2 Assignment 1: Python, NumPy, Pandas, and Matplotlib

The first assignment focused on strengthening programming fundamentals. Tasks included building a custom data-handling class, performing array manipulation using NumPy, working with Pandas dataframes for data splitting and analysis, and creating visualizations using Matplotlib. This assignment ensured familiarity with essential tools required for subsequent machine learning experiments.

---

3. Week 1

3.1 Activation Functions and Loss Functions

Activation functions introduce non-linearity into neural networks, enabling them to learn complex patterns. Commonly used activation functions studied include Sigmoid, Tanh, ReLU, and Softmax. Each function serves a specific purpose depending on the task and network architecture.

Loss functions measure the discrepancy between predicted outputs and ground truth values. Regression losses such as MAE and MSE, and classification losses such as Binary Cross Entropy and Categorical Cross Entropy were studied and compared.

3.2 Regression Models and Gradient Descent

Linear regression models the relationship between inputs and outputs using a straight-line approximation. Logistic regression extends this idea to classification by passing linear outputs through a sigmoid function. Gradient Descent was introduced as an optimization technique to iteratively minimize loss by updating model parameters in the direction of the negative gradient.

3.3 K-Means Clustering

K-Means is an unsupervised learning algorithm used for clustering data into K groups based on similarity. The iterative process of centroid initialization, distance computation, cluster assignment, and centroid update was studied in detail.

3.4 Theoretical Concepts

3.4.1 Evaluation Metrics

Model performance evaluation metrics such as Accuracy, Precision, Recall, and F1-score were studied, along with their mathematical formulations and practical significance.

3.4.2 Regularization Techniques

To address overfitting, regularization methods such as L1 and L2 regularization were explored. Dropout layers were also studied as a technique to improve generalization by randomly disabling neurons during training.

3.4.3 Gradient Descent Variants

Batch, Stochastic, and Mini-Batch Gradient Descent methods were compared based on convergence speed, stability, and computational efficiency.

3.5 Assignment 2: Neural Network from Scratch

A two-layer neural network was implemented from scratch using NumPy for digit classification on the MNIST dataset. The network employed ReLU and Softmax activations, cross-entropy loss, and gradient descent optimization. The model achieved high training and test accuracy, validating the correctness of the implementation.

---

4. Week 2

4.1 Convolutional Neural Networks

CNNs were introduced to address the limitations of fully connected networks for image data. Concepts such as convolution, feature maps, receptive fields, padding, stride, and pooling were studied. Core CNN components, including Conv2D, MaxPooling, Flatten, Dense layers, and ReLU activation, were implemented.

4.2 Batch Normalization

Batch Normalization was studied as a technique to stabilize training by normalizing layer activations. Its benefits include faster convergence, higher learning rates, and improved generalization.

4.3 Assignment 3: Machine Learning Theory

This assignment focused on written explanations of theoretical concepts, including evaluation metrics, overfitting vs underfitting, vanishing and exploding gradients, dropout, gradient descent variants, and optimizers such as Momentum, RMSprop, and Adam.

---

5. Week 3

5.1 Recurrent Neural Networks

RNNs were introduced for modeling sequential data where temporal dependencies exist. Concepts such as autoregressive models, hidden states, tokenization, and language modeling were discussed.

Backpropagation Through Time (BPTT) was studied as the training mechanism for RNNs, along with gradient clipping to handle exploding gradients.

5.2 Assignment 4: Batch Normalization in CNNs

A baseline CNN and a Batch-Normalized CNN were implemented and trained on the CIFAR-10 dataset. The Batch Normalization model demonstrated faster convergence and higher test accuracy, highlighting the effectiveness of normalization techniques.

---

6. Week 4

6.1 Long Short-Term Memory (LSTM)

LSTMs were studied as an improvement over vanilla RNNs to handle long-term dependencies. The roles of forget, input, and output gates in controlling information flow were analyzed.

6.2 Gated Recurrent Units (GRU)

GRUs were introduced as a simplified alternative to LSTMs with fewer gates and lower computational complexity while maintaining comparable performance.

6.3 Bidirectional RNNs

Bidirectional RNNs process sequences in both forward and backward directions, enabling richer contextual understanding. Their advantages and limitations were discussed.

6.4 Encoder–Decoder Architecture

Sequence-to-sequence models using encoder–decoder architectures were studied for tasks involving variable-length input and output sequences, such as machine translation.

6.5 Assignment 5: Character-Level RNN

A character-level RNN was implemented to generate text sequences(dinosaurs). Gradient clipping and sampling techniques were applied to stabilize training and generate meaningful

outputs. Over training iterations, the model successfully learned character-level patterns and produced realistic text samples.
