Mid-Term Project Report
Vision Transformer

Darsh Jain
EEA Winter Project

Contents

1 Foundations of Programming and Data Processing

1.1 Essential Python Programming Concepts

Python serves as the primary programming language for implementing machine learning and deep learning models. Core programming constructs such as variables, data types, conditional statements, iterative loops, and user-defined functions enable the development of modular, readable, and reusable code. Adhering to clean coding practices is essential for efficient debugging, experimentation, and scaling of machine learning workflows.

1.2 Numerical Computation with NumPy

NumPy provides high-performance numerical computation through efficient multi-dimensional array structures. Operations such as array creation, indexing, slicing, reshaping, and broadcasting allow vectorized computation, significantly improving execution speed. Fundamental mathematical operations including mean, variance, summation, and element-wise transformations are extensively used in data preprocessing and model computations.

1.3 Managing Structured Data using Pandas

Pandas offers powerful tools for handling structured datasets through Series and DataFrame objects. Essential operations include loading datasets, selecting and filtering rows and columns, managing missing values, and computing descriptive statistics. These steps play a crucial role in preparing real-world datasets before applying machine learning algorithms.

1.4 Data Visualization Techniques

Data visualization facilitates better understanding of data distributions and model behavior. Common visualization techniques include line plots, scatter plots, and bar charts. These graphical representations assist in exploratory data analysis and help identify patterns, trends, and anomalies within datasets.

Assignment 1: Python Programming and Numerical Computing (Colab Submission)

As part of Week 0, I completed Assignment 1 using Google Colab to apply core Python and numerical computing concepts in a practical setting. The assignment focused on strengthening programming fundamentals essential for machine learning workflows. Key tasks included implementing a DataSample class to understand data encapsulation, designing custom Python functions for logical operations, and performing NumPy-based array manipulations such as slicing, diagonal extraction, and vectorized computations relevant to neural network operations. Basic Pandas data handling and Matplotlib visualization were also used to analyze and present results.

Results

The assignment was completed successfully, with all components producing correct outputs. It reinforced proficiency in Python programming, numerical computation, and data handling, forming a strong foundation for subsequent machine learning and Vision Transformer–related work.

2 Core Concepts of Machine Learning

2.1 Machine Learning as a Mapping Function

Machine learning models can be interpreted as mathematical functions that map input features to corresponding outputs. While linear models provide a simple baseline, they lack the representational capacity required for complex, high-dimensional data. Non-linear models are therefore essential for capturing intricate patterns, particularly in domains such as image analysis.

2.2 Role of Activation Functions

Activation functions introduce non-linearity into neural networks, enabling them to learn complex mappings. Commonly used activation functions include sigmoid, hyperbolic tangent (tanh), Rectified Linear Unit (ReLU), linear activation, and softmax. The selection of an activation function directly impacts gradient propagation, convergence behavior, and overall model performance.

2.3 Error Measurement using Loss Functions

Loss functions measure the discrepancy between predicted outputs and ground truth labels. Mean Squared Error (MSE) and Mean Absolute Error (MAE) are typically employed for regression tasks, while Binary Cross-Entropy and Categorical Cross-Entropy are widely used for classification problems. Choosing an appropriate loss function is critical for effective model training.

2.4 Optimization using Gradient Descent

Gradient Descent is an iterative optimization technique used to minimize the loss function. Model parameters are updated by computing gradients of the loss with respect to the parameters. The learning rate determines the step size of updates and plays a vital role in balancing convergence speed and training stability.

2.5 Linear and Logistic Regression Models

Linear Regression models continuous outputs as weighted linear combinations of input features. Logistic Regression extends linear models to classification tasks by estimating

output probabilities using the sigmoid function. These regression techniques form the foundation of supervised learning.

Assignment 2: Mathematical Foundations and Core ML Algorithms (Colab Submission)

As part of Week 1, I completed Assignment 2 to apply mathematical concepts and core machine learning algorithms through hands-on implementation in Google Colab. The assignment focused on translating theoretical understanding into practical model behavior. Key components included implementing linear and logistic regression, analyzing loss functions for regression and classification, and applying gradient descent–based optimization to minimize errors. The assignment also involved experimenting with evaluation metrics to assess model performance and understanding the behavior of learning algorithms.

Results

The assignment was completed successfully, with correct implementation of regression models and optimization procedures. It strengthened conceptual clarity in loss minimization, optimization dynamics, and model evaluation, providing a solid mathematical foundation for deeper neural network and Vision Transformer–related learning.

3 Image Learning with Convolutional Neural Networks

3.1 Drawbacks of Fully Connected Neural Networks

Fully connected neural networks are inefficient for image-based tasks due to their high number of parameters and inability to exploit spatial structure. This limitation motivates the use of convolutional architectures for visual learning.

3.2 Fundamentals of the Convolution Operation

The convolution operation applies learnable filters to local regions of an input image to extract spatial features. The resulting feature maps represent the activation strength of filters across the image, capturing local patterns such as edges and textures.

3.3 CNN Design Principles and Architecture

Convolutional Neural Networks leverage locality and parameter sharing to efficiently learn hierarchical feature representations. Key concepts include local receptive fields, translation invariance, and progressive abstraction of features across layers.

3.4 Key Hyperparameters in CNNs

Important hyperparameters in CNNs include kernel size, stride, and padding. These parameters control the spatial resolution of feature maps and influence the effective receptive field of convolutional layers.

3.5 Dimensionality Reduction using Pooling

Pooling layers such as max pooling and average pooling reduce the spatial dimensions of feature maps while preserving salient information. Pooling improves computational efficiency and increases robustness to minor spatial variations.

Assignment 3: Convolution Operations and CNN Fundamentals (Colab Submission)

As part of Week 2, I completed Assignment 3 to gain hands-on experience with convolutional operations and core CNN concepts using Google Colab. The assignment emphasized understanding how spatial features are captured from image-like data. The tasks involved implementing convolution and cross-correlation operations, analyzing the effect of kernel size, stride, and padding, and observing how these parameters influence output feature maps. This practical work helped bridge theoretical concepts with actual computational behavior in CNNs.

Results

The assignment was completed successfully, with correct implementation and visualization of convolutional operations. It strengthened understanding of spatial feature extraction and CNN mechanics, providing a strong foundation for comparing convolution-based models with transformer-based vision architectures.

4 Sequence Learning and Recurrent Neural Models

4.1 Characteristics of Sequential Data

Sequential data exhibits temporal dependencies between observations. Unlike independent data samples, sequence data requires models capable of capturing order-dependent and contextual relationships.

4.2 Auto-Regressive Sequence Models

Auto-regressive models predict future values based on previous observations. Latent auto-regressive models use hidden states to summarize historical information over time.

4.3 Text Tokenization and Vocabulary Design

Tokenization converts raw text into discrete units known as tokens. These tokens are mapped to numerical indices using a vocabulary, with special symbols assigned to unknown or rare tokens.

4.4 Probabilistic Language Models

Language models estimate the probability of a sequence of tokens. Perplexity is commonly used as an evaluation metric to measure how effectively a model predicts sequences.

4.5 Recurrent Neural Network Architecture

Recurrent Neural Networks process sequential data by maintaining hidden states across time steps. Training is performed using Backpropagation Through Time (BPTT).

4.6 Training Difficulties in RNNs

RNNs often suffer from vanishing and exploding gradient problems. Techniques such as gradient clipping and truncated BPTT are employed to stabilize training.

Assignment 4: Sequence Modeling with RNNs (Colab Submission)

As part of Week 3, I completed Assignment 4 to implement and analyze sequence modeling concepts using recurrent neural networks in Google Colab. The assignment focused on understanding temporal dependencies and training dynamics in sequential data. Key tasks included implementing RNN-based models, applying one-hot encoding for sequential inputs, analyzing language modeling behavior, and studying training challenges such as vanishing and exploding gradients, addressed using gradient clipping.

Results

The assignment was completed successfully, demonstrating correct sequence modeling behavior and stable training with gradient clipping. It strengthened understanding of temporal learning mechanisms, directly contributing to readiness for attention-based and transformer-style architectures.

5 Advanced Recurrent Neural Architectures

5.1 Long Short-Term Memory Networks (LSTM)

LSTM networks introduce memory cells and gating mechanisms that enable learning long-term dependencies. The forget, input, and output gates regulate information flow within the network.

5.2 Gated Recurrent Units (GRU)

GRUs simplify the LSTM architecture by reducing the number of gates while retaining the ability to model long-term dependencies.

5.3 Enhanced and Deep Recurrent Models

Advanced architectures include stacked RNNs and bidirectional RNNs, which enhance representational capacity and contextual understanding.

5.4 Encoder–Decoder Based Sequence Modeling

Encoder–decoder models map input sequences to output sequences of variable lengths. These architectures form the conceptual basis for transformer-based models, including Vision Transformers.

Assignment 5: Advanced RNNs and Encoder–Decoder Modeling (Colab Submission)

As part of Week 4, I completed Assignment 5 to gain hands-on experience with advanced recurrent architectures and sequence-to-sequence modeling in Google Colab. The assignment focused on understanding how gated mechanisms and encoder–decoder structures improve long-range dependency modeling. Key tasks included implementing and analyzing LSTM and GRU-based models, experimenting with deep and bidirectional RNNs, and studying the working of encoder–decoder architectures for sequence transformation tasks.

Results

The assignment was completed successfully, demonstrating stable training and improved handling of long-term dependencies. It strengthened conceptual and practical understanding of architectures that directly precede and motivate transformer-based models, including Vision Transformers.