Mid-Term Project Report

Vision Transformer

Kamana Gupta

EEA Winter Project

Contents

1 Programming and Data Handling Foundations

1.1 Python Programming Fundamentals

Python is the primary programming language used for implementing machine learning and deep learning algorithms. Fundamental constructs such as variables, data types, conditional statements, loops, and functions enable modular and reusable code development. Clean programming practices are essential for debugging, experimentation, and scalability of machine learning pipelines.

1.2 Numerical Computing using NumPy

NumPy provides efficient numerical computation through multi-dimensional arrays. Array creation, indexing, slicing, and broadcasting enable vectorized operations that significantly improve performance. Mathematical operations such as mean, variance, summation, and element-wise transformations form the basis of data preprocessing and model computation.

1.3 Structured Data Handling using Pandas

Pandas enables handling of structured datasets using Series and DataFrames. Key operations include reading datasets, selecting rows and columns, filtering data, handling missing values, and computing descriptive statistics. These steps are crucial for preparing real-world data before applying learning algorithms.

1.4 Data Visualization using Matplotlib

Visualization techniques help in understanding data distributions and model behavior. Common plots include line plots, scatter plots, and bar charts. Visualization supports exploratory data analysis and aids in interpreting trends and anomalies in datasets.

Assignment 1: Data Handling and Batching

This assignment focused on structured data handling and preprocessing using Python libraries. The dataset was divided into multiple batches to enable efficient processing. Basic inspection of each batch was performed using tabular previews, verifying correct data segmentation and feature integrity before further analysis.

Results

```
print("Number of batches:", len(Batches))
print("\nFirst few rows of Batches[0]:")
print(Batches[0].head())
```

```
Number of batches: 17

First few rows of Batches[0]:
     gender race/ethnicity parental level of education         lunch  \
219    male        group B            some high school      standard
278  female        group C            some high school  free/reduced
218    male        group B                 high school  free/reduced
920    male        group D                 high school  free/reduced
130    male        group D             master's degree      standard

    test preparation course  math score  reading score  writing score
219               completed          61             56             56
278                    none          65             86             80
218                    none          66             77             70
920                    none          69             70             67
130                    none          89             84             82
```

Output showing the number of data batches and a preview of the first batch, confirming correct data partitioning.

2 Machine Learning Fundamentals

2.1 Machine Learning as Function Approximation

Machine learning models can be viewed as functions that map input features to outputs. Linear models provide a baseline but lack expressive power for complex data. Non-linear models are required to capture intricate patterns in high-dimensional datasets such as images.

2.2 Activation Functions

Activation functions introduce non-linearity into neural networks. Common activation functions include sigmoid, hyperbolic tangent (tanh), Rectified Linear Unit (ReLU), linear activation, and softmax. The choice of activation function affects gradient flow, convergence behavior, and overall model performance.

2.3 Loss Functions

Loss functions quantify the error between predicted and actual outputs. Mean Squared Error (MSE) and Mean Absolute Error (MAE) are commonly used for regression tasks, while Binary Cross-Entropy and Categorical Cross-Entropy are used for classification problems. Selecting an appropriate loss function is essential for effective learning.

2.4 Gradient Descent Optimization

Gradient Descent is an iterative optimization algorithm used to minimize loss functions. Model parameters are updated using gradients of the loss function with respect to the parameters. The learning rate controls convergence speed and training stability.

2.5 Regression Models

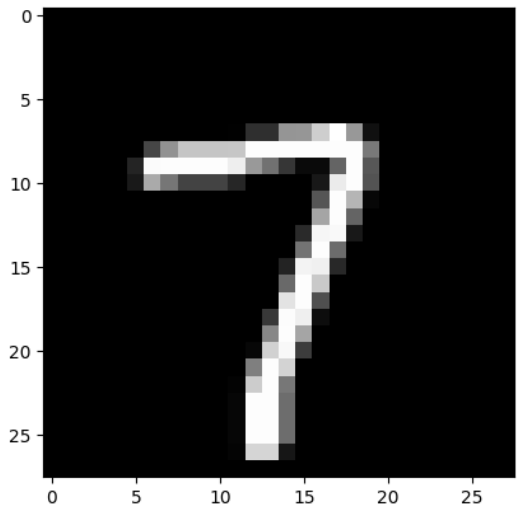
Linear Regression models continuous outputs as linear combinations of input features. Logistic Regression extends linear models to classification by predicting probabilities using the sigmoid function. These models form the foundation of supervised learning.

Assignment 2: Neural Network Fundamentals

This assignment focused on understanding the mathematical foundations of neural networks, including activation functions, loss functions, and gradient descent optimization. The objective was to analyze how model parameters are updated during training and how learning behavior depends on optimization dynamics.

Results

```
import random
idx = np.random.randint(0, X_test_acc.shape[1])
plt.imshow(X_test[idx], cmap='gray')
plt.show()
x_single = X_test_acc[:, idx].reshape(-1, 1)
cache = forward_prop(x_single, Parameters)
a_pred = np.argmax(cache['a2'], axis=0)
print("Our model says it is:", a_pred[0])
print("Actual label is:", Y_test[idx])
```




```
Our model says it is: 7
Actual label is: 7
```

Prediction result for a randomly selected test image. The model correctly classifies the handwritten digit by matching the predicted label with the ground truth.

3 Convolutional Neural Networks

3.1 Limitations of Fully Connected Networks

Fully connected neural networks are inefficient for image data due to high parameter count and lack of spatial inductive bias. This motivates the use of convolution-based architectures for visual learning tasks.

3.2 Convolution Operation

Convolution applies learnable kernels to local regions of an input image to extract spatial features. Feature maps represent the activation of kernels across the image, capturing local patterns such as edges and textures.

3.3 CNN Architecture and Key Concepts

Convolutional Neural Networks exploit locality and parameter sharing to efficiently learn hierarchical representations from images. Important concepts include local receptive fields, translation invariance, and hierarchical feature extraction.

3.4 CNN Hyperparameters

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

3.5 Pooling Layers

Pooling layers such as max pooling and average pooling reduce spatial dimensions while retaining important features. Pooling improves computational efficiency and robustness to small spatial variations.

Assignment 3: Convolutional Neural Networks

This assignment focused on implementing and understanding convolutional neural networks for image-based data. The objective was to analyze how convolution operations extract spatial features and how CNN architectures improve classification performance compared to fully connected networks.

4 Sequence Modeling and Recurrent Neural Networks

4.1 Sequential Data

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

4.2 Auto-Regressive Models

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

4.3 Tokenization and Vocabulary Construction

Tokenization converts text into discrete units, which are mapped to numerical indices using a vocabulary. Unknown or rare tokens are handled using special symbols.

4.4 Language Models

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

4.5 Recurrent Neural Networks

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

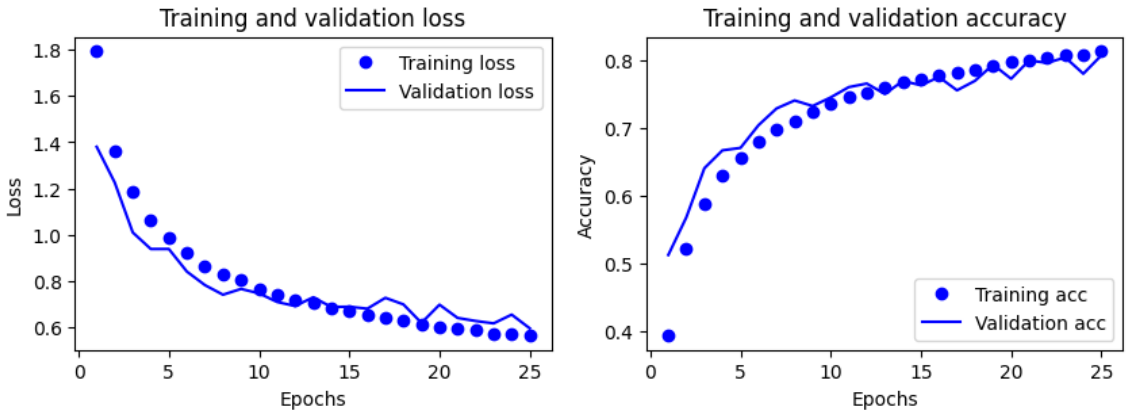
4.6 Training Challenges in RNNs

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

Assignment 4: Sequence Modeling with Recurrent Neural Networks

This assignment focused on modeling sequential data using recurrent neural networks. The objective was to understand how hidden states capture temporal dependencies and how training dynamics affect sequence-based predictions.

Results



Training and validation loss (left) and accuracy (right) across epochs for the recurrent neural network model. The plots show stable convergence with decreasing loss and improving accuracy, indicating effective sequence learning.

5 Modern Recurrent Architectures

5.1 Long Short-Term Memory Networks

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

5.2 Gated Recurrent Units

GRUs simplify LSTMs by reducing the number of gates while maintaining the ability to model long-term dependencies.

5.3 Advanced Recurrent Architectures

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

5.4 Encoder–Decoder Architecture

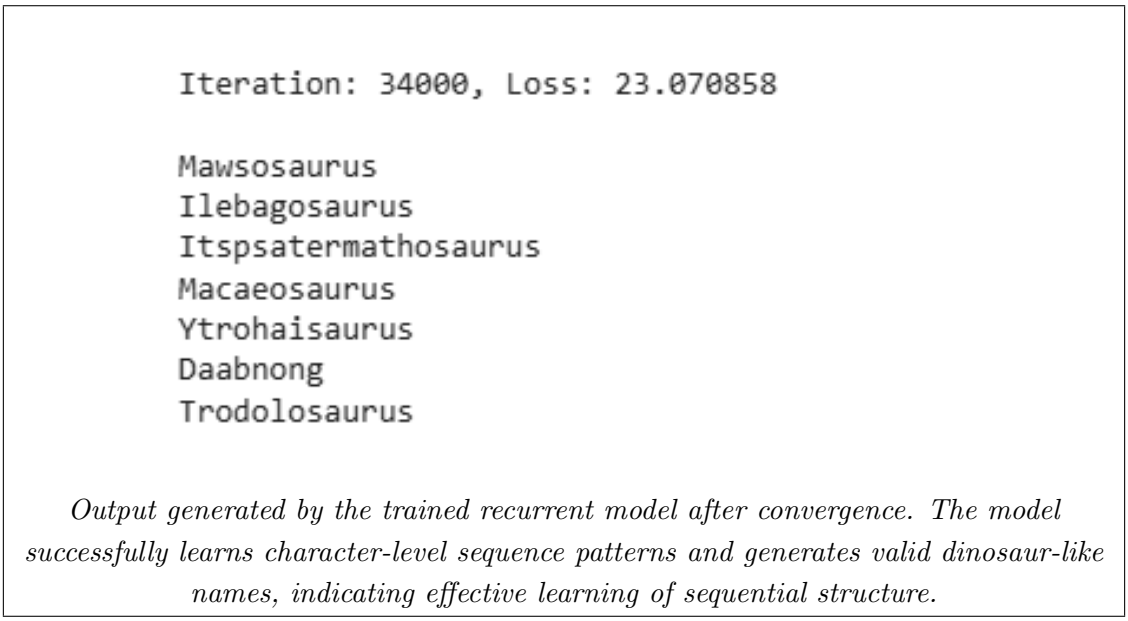
Encoder–decoder models map input sequences to output sequences of variable lengths. These architectures form the conceptual foundation for transformer-based models used in Vision Transformers.

Assignment 5: Modern Recurrent Architectures (LSTM / GRU)

This assignment focused on understanding modern recurrent neural network architectures such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units

(GRUs). The objective was to analyze how gating mechanisms help preserve long-term dependencies and improve training stability compared to vanilla RNNs.

Results



Output generated by the trained recurrent model after convergence. The model successfully learns character-level sequence patterns and generates valid dinosaur-like names, indicating effective learning of sequential structure.