

Vision Transformer Project Report

WEEK 0

Python - Learnt about **Loops** which are used to iterate over dataset multiple times to help the model learn in, **Functions** allows to wrap a block of code and give it a name so we can use it repeatedly without rewriting it, **Lists** are Ordered sequences. **Dictionaries** are Key-Value pairs. **Classes** are the blueprints for objects.

Matplotlib - It provides a way to turn numbers into insights through charts, plots, and figures for better analyzation.

Numpy - Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays

Assignment -1

Section 1: Python Basics (Classes & Logic)

- **Problem 1 (Classes):** Built a DataSample class to store features and labels. Implemented **Min-Max Normalization** (scaling data between 0 and 1) and a scaling method.
- **Problem 2 (Functions & Sorting):** Wrote a function to sort a list of strings based on how many **unique characters** they have.

Section 2: NumPy (Vectorization & Math)

- **Problem 1 (Array Manipulation):** Covered creating 10x10 matrices, using **boolean masks** to replace values, slicing submatrices.
- **Problem 2 (Normalization Logic):** Implemented a manual version of the **Softmax** process.

Section 3: Pandas & Matplotlib (Data & Batching)

- **Data Analysis:** Made student performance dataset, performed sorting, and visualized the data.

WEEK 1

Activation & Loss Functions- Resources cover the theoretical basis for choosing mathematical functions that introduce non-linearity and measure model error.

Regression & Gradient Descent- The materials explain Linear and Logistic Regression along with Gradient Descent as the fundamental algorithm for optimizing model weights.

Closed-Form Solution- A specific guide explains the direct algebraic method for solving linear regression without iterative optimization.

K-Means Clustering: The algorithmic steps and visual representations of this popular unsupervised learning method is used for data grouping.

Neural Network Foundations:

Evaluation Metrics: The text provides a summary of the statistical tools used to assess the accuracy and reliability of machine learning models.

Regularization Techniques: L1, L2, and Dropout methods are presented as essential strategies to prevent models from overfitting to training data.

Optimization Strategies: Stochastic Gradient Descent (SGD) and advanced optimizers like Momentum, RMSProp, and Adam are used for efficient model training.

Assignment-2

- I built a 2-layer neural network from the ground up using only NumPy to classify handwritten digits from the MNIST dataset.
- To keep the training balanced and efficient, I curated the dataset so that each digit (0–9) had exactly 1,000 sample images.
- I initialized the model by giving it "small" random weights to prevent the learning from stalling and set all biases to zero.
- I implemented ReLU and Tanh activation functions to help the network learn complex patterns, ending with a Softmax layer to turn raw numbers into probabilities.
- I coded the forward propagation to make predictions and used categorical cross-entropy to calculate exactly how much the model missed the mark.
- Through backpropagation, I calculated the gradients and used Gradient Descent to refine the weights after every iteration.
- Final Results: I wrapped everything into a single training function, monitored the falling cost curve, and finally verified the model's accuracy on unseen test data.

Assignment- 3

- I explored the dimensionality of gradients in matrix calculus and practiced calculating key performance metrics **Accuracy, Precision, Recall, and F1 score** using a real-world medical diagnosis scenario.
- I defined the role of a **Confusion Matrix** in visualizing model errors and discussed the critical balance between **Overfitting** (learning noise) and **Underfitting** (failing to learn patterns).
- I investigated the causes of **Vanishing and Exploding Gradients**, detailing how these issues can stall or destabilize training and the specific techniques used to prevent them.
- I broke down the differences between standard **Gradient Descent**, **Stochastic Gradient Descent (SGD)**, and **Mini-Batch Gradient Descent**. Finally, I detailed how modern optimizers like **Momentum**, **RMSprop**, and **Adam** use moving averages to navigate the loss landscape more efficiently.

WEEK - 2

Convolutional Neural Networks (CNNs): The resources provide an in-depth exploration of CNN architectures, which are the fundamental building blocks for modern computer vision tasks. We studied the specialized architecture of **Convolutional Neural Networks (CNNs)**, learning how they serve as the backbone for modern computer vision, the mechanics of how these networks extract spatial features from images. **Batch Normalization:** The materials explain how normalizing the inputs of each layer during training stabilizes the learning process and significantly speeds up model convergence.

Assignment 4

- I started by implementing a classic Convolutional Neural Network (CNN) using PyTorch, focusing on how layers like Conv2d and MaxPool2d extract spatial features from images.
- I built the core "engine" of a Vision Transformer (ViT), which uses MultiheadAttention and LayerNorm to allow the model to focus on different parts of an image simultaneously.
- I implemented a Multi-Layer Perceptron (MLP) using the GELU activation function and dropout layers, which serves as the processing unit for the transformer's encoded information.
- I created a system to break down full images into smaller, digestible "patches", which is the critical step that allows a Transformer to "read" an image similar to how it reads words in a sentence.
- I analyzed the theoretical shift from CNNs, which are hard-coded to look at local patterns, to ViTs, which have a "global" perspective and learn relationships across the entire image from scratch.

WEEK 3

Recurrent Neural Networks (RNNs) are a specialized type of neural network designed to process sequential data or time-series information by "remembering" previous inputs. Unlike traditional neural networks that assume all inputs are independent, RNNs use a hidden state that acts as a memory. This allows information from earlier parts of a sequence to influence the processing of later parts.

Assignment 5

- I implemented a custom VanillaRNN class from scratch using PyTorch to understand the fundamental mechanics of recurrent layers.
- I developed the recursive forward pass logic where the current hidden state is calculated using both the new input and the previous state. I manually set up the weight matrices for inputs and hidden states, applying the tanh activation function to maintain the "memory" of the sequence.
- To put the theory into practice, I built a SimpleRNNRegressor to perform a sequence prediction task, specifically training it to forecast a sine wave.
- The algorithm has started to generate plausible dinosaur names towards the end of the training. At first, it was generating random characters, but towards the end I could see dinosaur names with cool endings.

WEEK 4

LSTMs are designed to combat the vanishing gradient problem by using a complex "cell state" that acts as a long-term memory track. **GRUs** are a streamlined version of LSTMs that achieve similar results with a simpler architecture. Traditional RNNs only look at the past (information from earlier in the sequence). Bi-RNNs look at both the **past and the future**.