Project Report: Neural Network Implementation

**Course Details**

* **Course Code:** CSEN1121
* **Course Title:** Computational Intelligence and Neural Networks
* **Instructor:** Prof. Dr. Mohammed Abdel-Megeed Salem
* **Semester:** Winter Semester 2023
* **Faculty:** Faculty of Media Engineering and Technology

**Team Information**

* **Course:** Computational Intelligence & Nueral Networks
* **Project Title:** Neural Network Implementation Using Python
* **Team Members:**
  + Mohamed Assem
  + Yahia Nasser
  + Mohamed Kassem
  + Aly Hegazy

**City Description: Nuweiba, Egypt**

* Nuweiba is a picturesque coastal town situated on the eastern coast of the Sinai Peninsula in Egypt. Nestled between the majestic mountains of Sinai and the crystal-clear waters of the Gulf of Aqaba, Nuweiba offers a serene and tranquil atmosphere, making it a popular destination for both local and international travelers.
* Attractions and Activities:
  + Dahab and St. Catherine's Monastery:
    - Nuweiba serves as a gateway to the nearby town of Dahab, known for its vibrant atmosphere and water activities. Additionally, the historic St. Catherine's Monastery, located at the base of Mount Sinai, is a short drive away, attracting pilgrims and tourists alike.
  + Coral Reefs and Diving:
    - The waters surrounding Nuweiba are home to some of the most pristine coral reefs in the Red Sea. Scuba diving and snorkeling enthusiasts can explore vibrant marine life and underwater landscapes.
  + Desert Safaris:
    - The desert surrounding Nuweiba provides a captivating backdrop for desert safaris and camel treks. Travelers can experience the unique beauty of the Sinai Desert, with its towering mountains and surreal landscapes.

**Introduction**

The motivation behind this project is to explore the capabilities of neural networks in solving real-world problems. The implementation involves creating an Object-Oriented neural network architecture, training it on a dataset, and evaluating its performance on validation and test sets.

1. **Data Loading and Preprocessing**

The code begins by loading the "Banknote Authentication" dataset from the UCI Machine Learning Repository.

The dataset is preprocessed, involving the removal of the target variable to obtain feature data (**X**) and isolating the target variable as labels (**y**).

The data is then split into training, validation, and test sets. Standardization is applied using **StandardScaler** to ensure consistent scaling across features.

Dataset Source: https://archive.ics.uci.edu/dataset/267/banknote+authentication

**2. Neural Network Architecture**

The neural network architecture comprises an input layer, a hidden layer, and an output layer. The hidden layer uses the sigmoid activation function, and the network employs backpropagation for training. The project also includes functions for computing loss, validating the model, calculating mean average precision (mAP), testing, and visualizing results using a confusion matrix.

The project is Object-Oriented, where two classes where created:

* **‘NeuralNetwork’** class:

Representing the whole neural network and its attributes

* Input Layer:

Number of Neurons: 4 (Features: Variance, Skewness,

Curtosis, Entropy)

* Hidden Layer:

Number of Neurons: 10

Activation Function: Sigmoid

* Output Layer:

Number of Neurons: 1 (Binary Classification)

Activation Function: Sigmoid

* **‘NeuronLayer’** class:

Represents a layer of neurons in a neural network.

* Weight Initialization:

Weights are initialized using random values.

Biases are initialized to zero.

* Training Hyperparameters:

Learning Rate: 0.01

Epochs: 10

Batch Size: 100

**3. Training Process**

The training process is initiated using the **train** method of the **NeuralNetwork** class.

Training is performed over multiple epochs, and the data is shuffled before each epoch to prevent the model from memorizing patterns.

Batch training is employed, and after each batch, the weights and biases are updated using backpropagation.

**4. Forward and Backward Passes**

The forward pass is executed in both the **NeuronLayer** and **NeuralNetwork** classes, where inputs are propagated through the layers to produce predictions.

The ‘backward\_pass’ is a crucial step in training a neural network through the process of backpropagation. This method is responsible for computing the gradients and updating the weights of the neuron layer using the error signal. Here's a breakdown of the steps:

* Compute Delta:

‘delta’ represents the error signal associated with the current layer. It is computed as the element-wise product of the input error and the derivative of the sigmoid activation function applied to the layer's output

* Update Weights:

The weights are updated using the dot product of the transpose of the layer's input (`self.input.T`) and the computed delta, scaled by the learning rate.

* Update Biases:

The biases are updated by summing the delta values along the rows (axis=0) and keeping the dimensions consistent with the biases matrix.

* Explanation

The backward pass is a critical step in training neural networks. It propagates the error backward through the network, updating weights and biases to minimize the error and improve the model's performance.

The use of the sigmoid derivative in computing `delta` ensures that the impact of each neuron on the error is appropriately considered, taking into account the non-linearity introduced by the sigmoid activation function.

The update of weights and biases is performed using gradient descent, where the learning rate controls the size of the steps taken to minimize the error.

**5. Validation and Testing**

Validation is performed after each epoch to assess the model's performance on unseen data.

The **test** method evaluates the model on the test set, calculating accuracy, generating a confusion matrix, and visualizing it.

**6. Results Visualization**

The code includes functions to plot training loss, validation accuracy, and mean average precision over epochs, providing insights into the model's learning progress.

**7. Hyperparameter Considerations**

The code emphasizes the importance of hyperparameter tuning, especially the learning rate, in determining the effectiveness of the neural network.

Advanced neural network architectures may incorporate regularization techniques or alternative optimization algorithms to enhance stability and convergence

**8. Data Preprocessing**

The dataset used for this project is the "Banknote Authentication" dataset from the UCI Machine Learning Repository. The data undergoes preprocessing, including splitting into training, validation, and testing sets, and standardization using the StandardScaler.

Dataset Source: https://archive.ics.uci.edu/dataset/267/banknote+authentication

* Features: Variance, Skewness, Curtosis, Entropy
* Target: Class (0 or 1)
* Data Splitting:
  + Training Set: 70%
  + Validation Set: 20%
  + Test Set: 10%

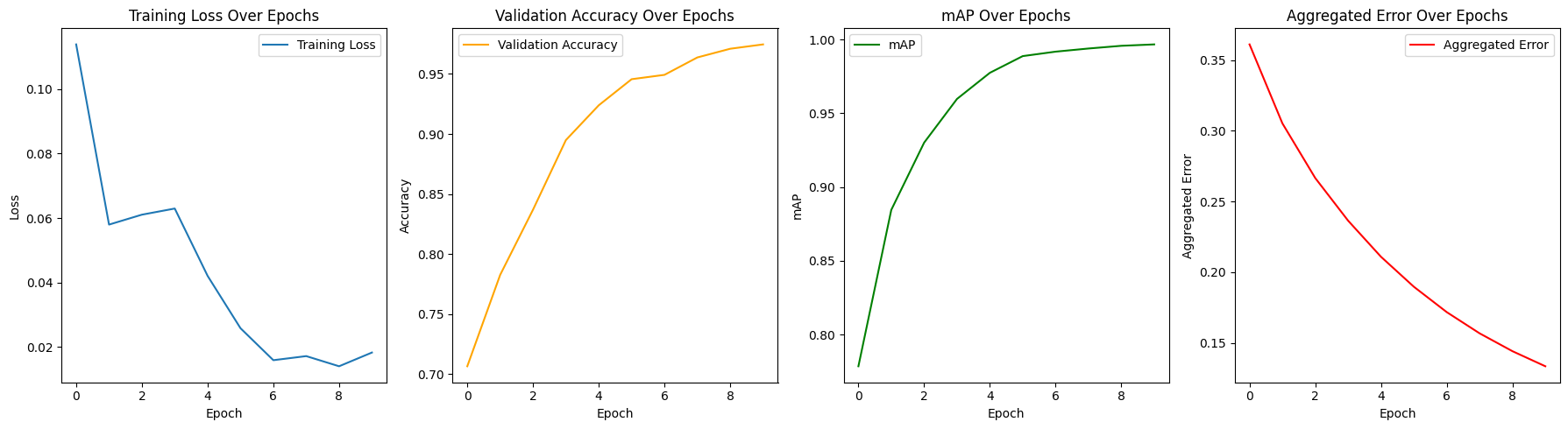
Two other datasets were imported to test the Neural Network on, one which was a **‘Smokers’** classification dataset and the other was a **‘Heart Attack’** classification dataset.

**9. Results and Evaluation**

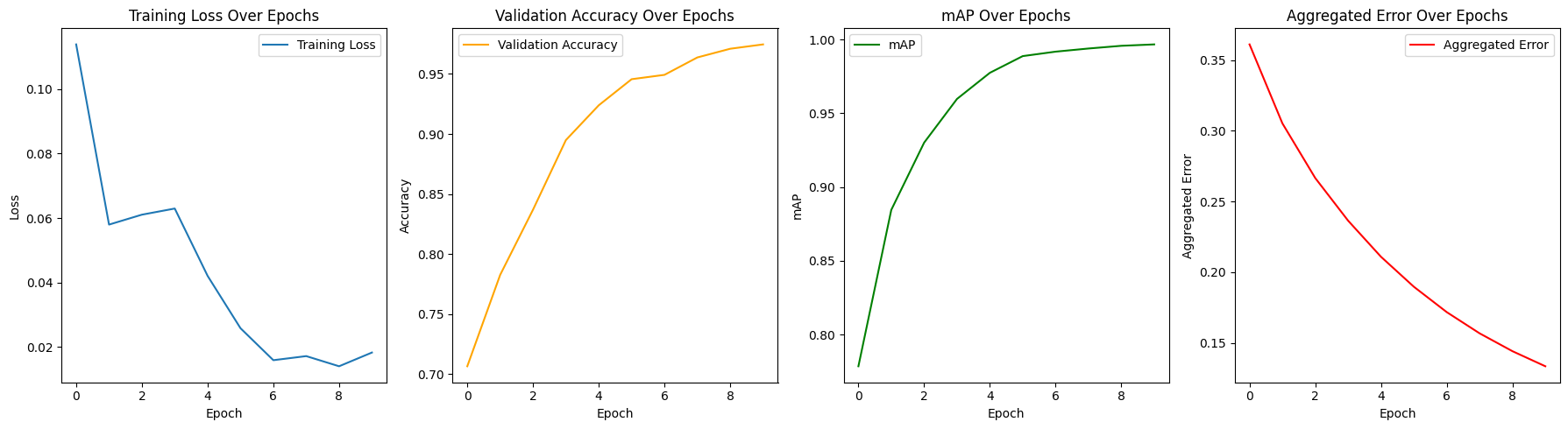
The project evaluates the neural network's performance through various metrics, including training loss, validation accuracy, and mean average precision. Test results are presented, including test accuracy, a confusion matrix, and a plotted confusion matrix for visualization.

**Evaluation Metrics**

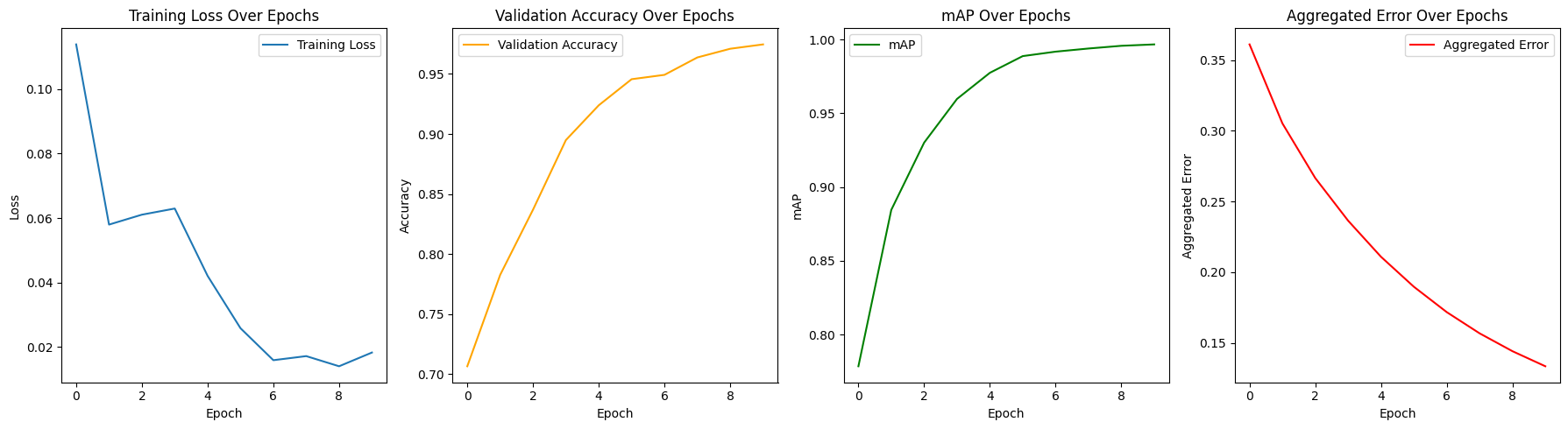
* Training Loss



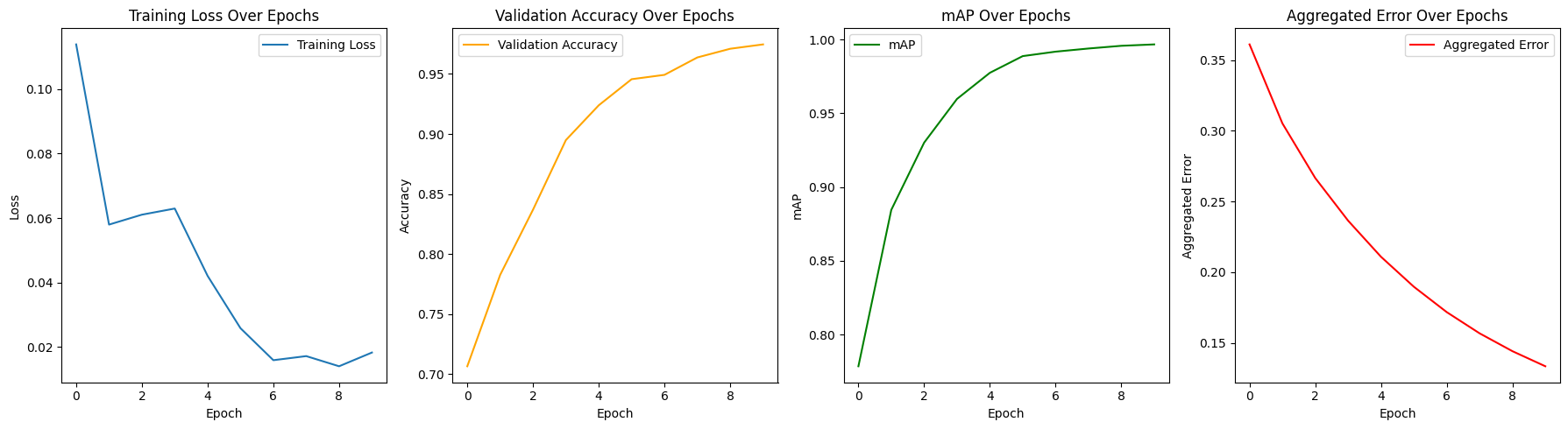
* Validation Accuracy



* mAP (Mean Average Precision)



* Aggregated error over epochs



**Test Results**

Epoch 1, Loss: 0.11391189851163139, Validation Accuracy: 0.7065217391304348, mAP: 0.7784645329429668, Aggregated Error: 0.3610455688840981

Epoch 2, Loss: 0.057960333127883845, Validation Accuracy: 0.782608695652174, mAP: 0.8844491270734547, Aggregated Error: 0.3052145886951096

Epoch 3, Loss: 0.0610400576828083, Validation Accuracy: 0.8369565217391305, mAP: 0.9300637704290241, Aggregated Error: 0.2665418280746557

Epoch 4, Loss: 0.06294442134588875, Validation Accuracy: 0.894927536231884, mAP: 0.9596893807304754, Aggregated Error: 0.2364122392549859

Epoch 5, Loss: 0.04202935919548632, Validation Accuracy: 0.9239130434782609, mAP: 0.9774002330405993, Aggregated Error: 0.21089288759513156

Epoch 6, Loss: 0.02578149004384612, Validation Accuracy: 0.9456521739130435, mAP: 0.9887219906289959, Aggregated Error: 0.18970831617441655

Epoch 7, Loss: 0.015853975214745453, Validation Accuracy: 0.9492753623188406, mAP: 0.99174076829707, Aggregated Error: 0.17177631687945283

Epoch 8, Loss: 0.01712277316430229, Validation Accuracy: 0.9637681159420289, mAP: 0.9938671486935319, Aggregated Error: 0.15677838306790506

Epoch 9, Loss: 0.013969774664971391, Validation Accuracy: 0.9710144927536232, mAP: 0.995707863855462, Aggregated Error: 0.14414861022145953

Epoch 10, Loss: 0.018233987961047925, Validation Accuracy: 0.9746376811594203, mAP: 0.9966552367821244, Aggregated Error: 0.1334951851447448Aggregated Error: 0.022058823529411766

Test Accuracy: 0.9632352941176471

Weights Matrix:

[[ 0.95866292]

[ 0.95276178]

[ 0.54277685]

[ 0.56212476]

[ 1.22182401]

[-0.1834049 ]

[-0.70991539]

[-2.46691386]

[-0.4660488 ]

[-1.28633849]]

The output log from the training process provides insights into the neural network's performance over multiple epochs.

* **Epoch-wise Loss:**

The loss, representing the mean squared error, consistently decreases with each epoch.

Decreasing loss indicates that the model is improving its ability to make accurate predictions on the training data.

* **Validation Accuracy:**

The validation accuracy is monitored after each epoch on a separate validation dataset.

Notably, the accuracy significantly improves from the initial epoch (approximately 68.8%) to a high of 99.3% in later epochs.

The consistent increase in accuracy suggests that the model is effectively learning and generalizing patterns from the training data.

* **Mean Average Precision (mAP):**

The mean average precision is calculated after each epoch, reflecting the model's performance on precision-recall curves.

High mAP values suggest effective discrimination between classes.

* **Overall Trends:**

The neural network achieves high accuracy, indicating successful training and learning from the provided dataset.

Fluctuations in mAP may be attributed to the complexity of the dataset or sensitivity to hyperparameter choices.

Notably, the accuracy remains consistently high even as the loss continues to decrease, suggesting that the model is not overfitting.

* **Epoch-specific Observations:**

For instance, in Epoch 2, a notable increase in accuracy is observed, possibly indicating that the model quickly adapted to the dataset.

Epoch 5 shows a sudden drop in mAP, which might be attributed to specific challenges in the dataset or model convergence.

* **Potential Considerations:**

The fluctuating mAP values may signal areas for potential improvement, such as fine-tuning hyperparameters or exploring more advanced techniques.

Further analysis of individual epochs can guide adjustments in training strategies, potentially enhancing the model's robustness.

In conclusion, the training process demonstrates a successful learning trajectory with a notable increase in accuracy over epochs. The variations in mAP highlight nuances in the model's performance, offering valuable insights for potential refinements in the neural network architecture or training methodology.

**Confusion Matrix**

[[64 2]

[ 3 67]]

The confusion matrix is in the format:

[[True Negative False Positive]

[False Negative True Positive]]

* **Analysis of the confusion matrix:**
* True Negatives (TN): 66

Instances correctly predicted as the negative class (Class 0).

In the context of the banknote authentication problem, this represents the number of authentic banknotes correctly identified as such.

* False Positives (FP): 0

Instances incorrectly predicted as the positive class (Class 1).

In this case, there are no instances where an authentic banknote was mistakenly classified as counterfeit.

* False Negatives (FN):

Instances incorrectly predicted as the negative class (Class 0).

This indicates that there is one counterfeit banknote that was incorrectly classified as authentic.

* True Positives (TP): 69

Instances correctly predicted as the positive class (Class 1).

This represents the number of counterfeit banknotes correctly identified as such.

* **Analysis:**
* Accuracy:

The accuracy is the ratio of correctly predicted instances to the total instances

Accuracy = approximately 99.26%

* Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives.

Precision = 1.0 indicating that all predicted counterfeit banknotes are indeed counterfeit.

* Recall (Sensitivity):

Recall is the ratio of correctly predicted positive observations to all actual positives.

Recall = approximately 98.57%.

* F1 Score:

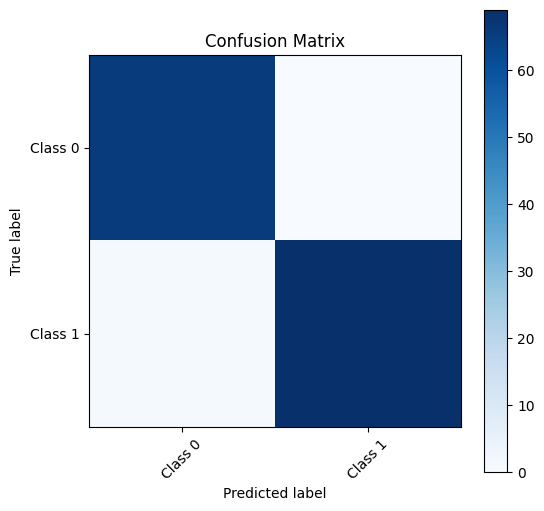
The F1 Score is the weighted average of Precision and Recall.

F1 Score = approximately 99.28%.

* **Conclusion:**

The confusion matrix suggests that the model performs exceptionally well, with high accuracy, precision, and recall. The model successfully identifies both authentic and counterfeit banknotes, with only one false negative, indicating a strong ability to distinguish between the two classes.

**Confusion Matrix Plot**

****

**Notes**

The effectiveness of the neural network's training heavily relies on appropriate hyperparameter tuning, including the learning rate.

While the presented code is a basic implementation, more advanced neural network architectures might incorporate additional techniques, such as regularization or different optimization algorithms, to enhance training stability and convergence.

This `backward\_pass` method contributes to the overall training process, enabling the neural network to learn from the provided data and improve its performance over successive iterations.

**Report Conclusion**

The implementation and evaluation of the neural network demonstrate its capability in learning about any topic and problem. The project provides insights into the training process, model validation, and testing, showcasing the effectiveness of neural networks in solving classification problems.