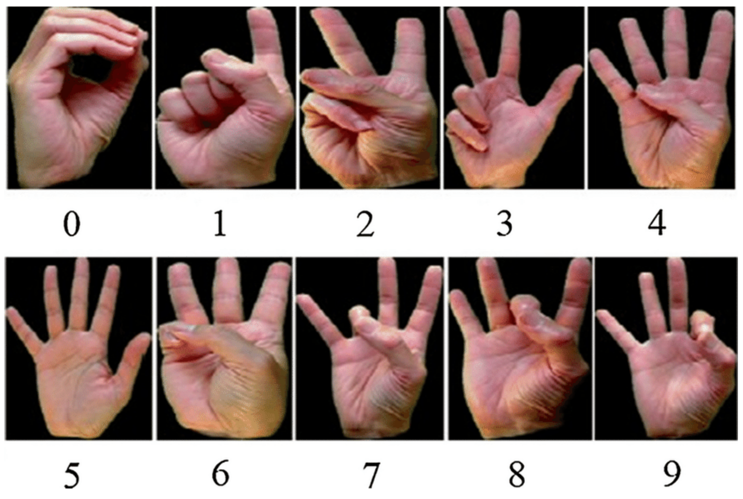
# Basic Of Deep Learning - Mid Semester Project Part 2

Amos Zohar (ID. 311402812), Gal Havshush (ID. 207389909))

Submitted as mid-semester project report for Basic Of Deep Learning course, Colman, 2024



## 1 Introduction

The classification of hand-sign digits presents a significant and important challenge within the domain of deep learning, with profound implications for accessibility technologies aimed at enhancing communication for individuals with hearing impairments. This project addresses this challenge by designing and implementing a deep learning framework capable of accurately discerning hand-sign digits. Leveraging the strengths of deep learning models, which excel at recognizing patterns and extracting features from complex datasets, the project utilizes multi-layered architectures to process data hierarchically, making them ideal for image classification. By applying these methods to a hand-sign digit dataset, the project converts raw pixel data into accurate and meaningful classifications, showcasing the adaptability and efficacy of neural network architectures in resolving real-world tasks.

### 1.1 Data

The dataset employed in this study is the Sign Language Digits dataset, comprising 5,000 grayscale images uniformly distributed across 10 categories. Each image measures 28x28 pixels and represents hand signs corresponding to digits 0 through 9. The pixel intensity values range from 0 to 255, encoding the visual patterns of each gesture in a manner that supports effective feature extraction for classification tasks.

This dataset offers an optimal foundation for exploring both binary classification, focusing on specific pairs of digits, and multiclass classification, encompassing all 10 digits. Its balanced class distribution ensures robust evaluation metrics, while its structured design facilitates streamlined preprocessing and effective deployment of fully connected deep learning models. The dataset's compactness and visual richness underscore its suitability for the methodological and experimental objectives of this project.

### 1.2 Problem

The primary aim of this project is to leverage deep learning to address two related classification tasks: binary classification, which serves as an introductory step by distinguishing between two specific digits, and multiclass classification, which builds upon this foundation to identify all 10 digits in the dataset. The binary task serves as an introductory framework to test and fine-tune essential techniques, while the multiclass problem requires a more advanced architectural design and a comprehensive training strategy to handle the increased complexity effectively, Both tasks were tackled using fully connected neural networks implemented with the Keras framework. To solve the multiclass problem, a base dense network was first used to establish an initial performance benchmark. The model was then refined through architectural modifications and hyperparameter tuning to improve its ability to classify all 10 digits accurately.

The optimized model, developed through these iterative enhancements, achieved superior accuracy and demonstrated robust performance in distinguishing all 10 hand-sign digits. This methodology highlighted the importance of experimentation and intelligent neural network design in complex classification tasks.

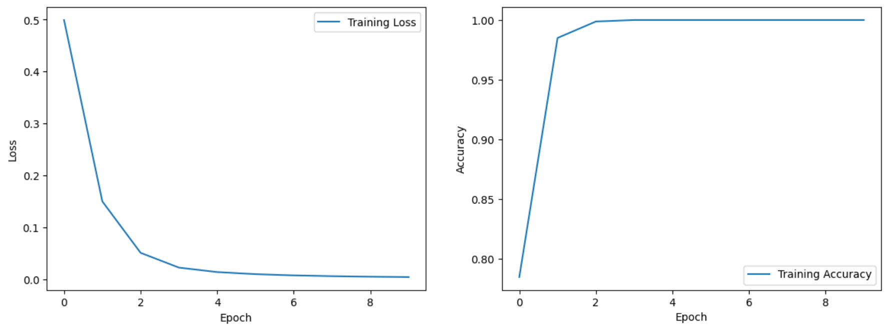
## 2 Solution

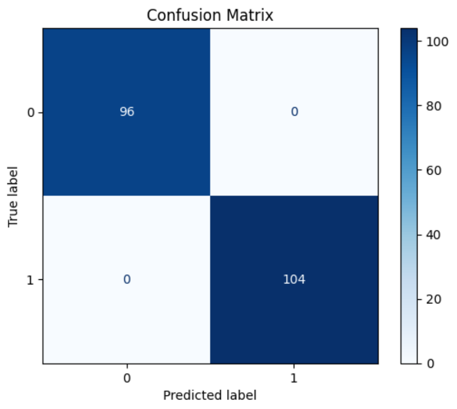
### 2.1 General approach

A simple neural network was initially designed for binary classification to distinguish between two digits, offering foundational insights for developing the multiclass model. For multiclass classification, a fully connected neural network was progressively refined through experimentation involving architectural adjustments and hyperparameter tuning. This approach enabled the model to achieve robust accuracy and effectively classify all 10 digits.

## 2.2 Design

The binary classification model featured an input layer of 784 neurons (corresponding to the size of the image 28x28), a hidden layer with 10 neurons and sigmoid activation, and two output neurons with softmax activation utilizing one-hot encoding. While a single output neuron with sigmoid activation is standard for binary classification, the more complex output layer was used because it provided a consistent framework for transitioning to the multiclass model architecture, which required handling multiple categories efficiently. The binary classification model achieved 100% training and test accuracy, with consistently decreasing loss and perfect performance visualized through a confusion matrix.





The multiclass classification model employed a more complex architecture to handle all 10 digits. It began with a base fully connected neural network featuring multiple dense layers. ReLU activation was used in the hidden layers as it avoids the vanishing gradient problem associated with sigmoid and allows for faster convergence during training, while the output layer utilized a softmax activation function to address the multiclass nature of the task. The model was trained using the Adam optimizer, which combines the benefits of momentum and adaptive learning rates to efficiently handle sparse gradients and ensure faster convergence, the categorical cross-entropy loss function was used to measure classification performance. Managing overfitting and finding the optimal architecture and hyperparameters, such as learning rates, batch sizes, and the number of epochs, were achieved through experimentation and iterative refinement, leading to superior accuracy and consistent performance across all 10 classes.

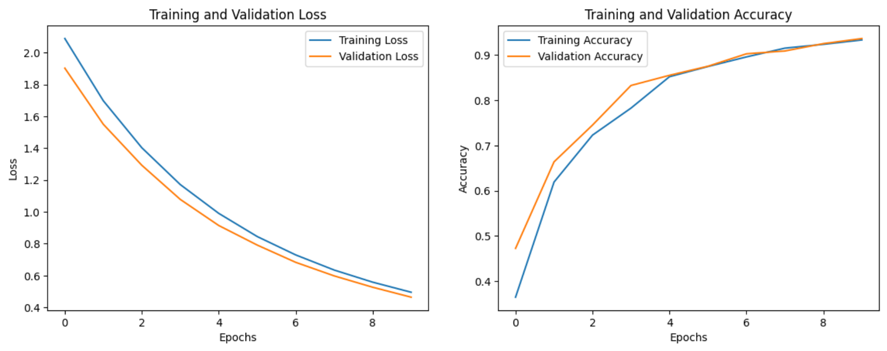
The following sections will detail the base model's initial setup, the series of experiments conducted to refine the architecture and hyperparameters, and the characteristics of the final optimal model that achieved the highest performance.

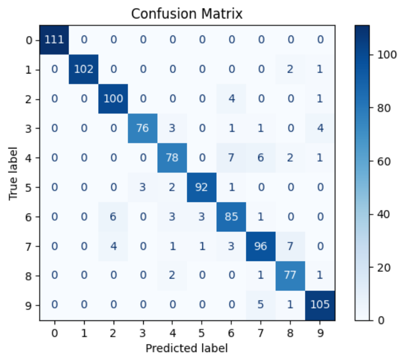
### 2.3 Base Model

The base model was structured as follows:

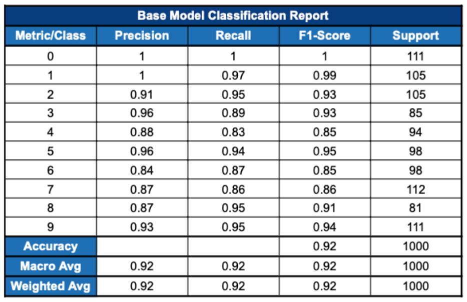
- **Input Layer:** 784 neurons (28x28 pixels flattened)
- **Hidden Layer:** 32 neurons with sigmoid activation
- **Output Layer:** 10 neurons with softmax activation

This simple architecture served as the initial benchmark for multiclass classification. The model was trained using the Adam optimizer with a learning rate of 0.001, categorical cross-entropy loss function. After 10 epochs with a batch size of 32, the model achieved a test accuracy of 92.2%. Analysis of the confusion matrix revealed areas for improvement in overall classification accuracy.





|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 111 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 102 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 |
| 2 | 0 | 0 | 100 | 0 | 0 | 0 | 4 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 76 | 3 | 0 | 1 | 1 | 0 | 4 |
| 4 | 0 | 0 | 0 | 0 | 78 | 0 | 7 | 6 | 2 | 1 |
| 5 | 0 | 0 | 0 | 3 | 2 | 92 | 1 | 0 | 0 | 0 |
| 6 | 0 | 0 | 6 | 0 | 3 | 3 | 85 | 1 | 0 | 0 |
| 7 | 0 | 0 | 4 | 0 | 1 | 1 | 3 | 96 | 7 | 0 |
| 8 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 1 | 77 | 1 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 1 | 105 |



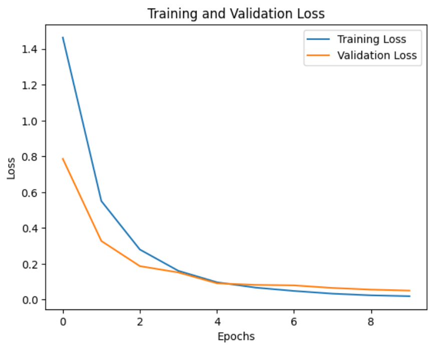
| Base Model Classification Report | | | | |
|---|---|---|---|---|
| Metric/Class | Precision | Recall | F1-Score | Support |
| 0 | 1 | 1 | 1 | 111 |
| 1 | 1 | 0.97 | 0.99 | 105 |
| 2 | 0.91 | 0.95 | 0.93 | 105 |
| 3 | 0.96 | 0.89 | 0.93 | 85 |
| 4 | 0.88 | 0.83 | 0.85 | 94 |
| 5 | 0.96 | 0.94 | 0.95 | 98 |
| 6 | 0.84 | 0.87 | 0.85 | 98 |
| 7 | 0.87 | 0.86 | 0.86 | 112 |
| 8 | 0.87 | 0.95 | 0.91 | 81 |
| 9 | 0.93 | 0.95 | 0.94 | 111 |
| Accuracy |  |  | 0.92 | 1000 |
| Macro Avg | 0.92 | 0.92 | 0.92 | 1000 |
| Weighted Avg | 0.92 | 0.92 | 0.92 | 1000 |

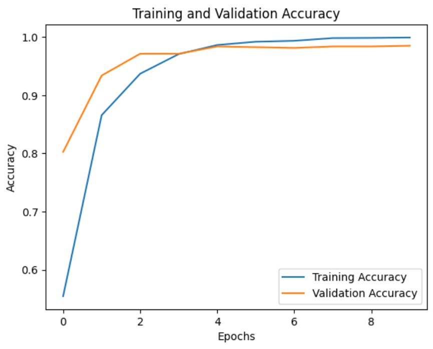
## 2.4 First Experiment

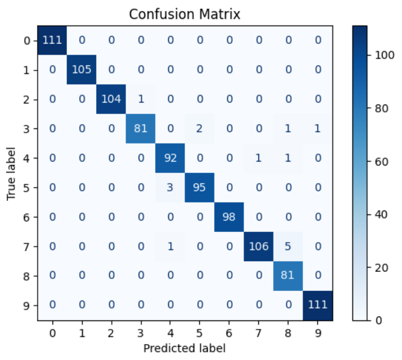
In the first experiment, the architecture was modified by replacing the sigmoid activation function with ReLU and increasing its complexity. ReLU activation was chosen because it mitigates the vanishing gradient problem associated with sigmoid, enabling faster and more efficient convergence during training. This updated architecture became a more advanced structure capable of extracting richer features. The updated architecture included:

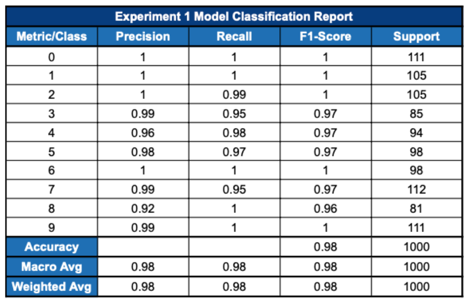
- **Input Layer:** 784 neurons
- **First Hidden Layer:** 128 neurons with ReLU activation
- **Second Hidden Layer:** 32 neurons with ReLU activation
- **Output Layer:** 10 neurons with softmax activation

The goal of this modification was to increase the model's ability to learn more complex patterns. The model was trained for 10 epochs with a batch size of 32, resulting in a test accuracy of 98.4%. The confusion matrix showed a significant reduction in errors, particularly among similar digits, demonstrating the effectiveness of this enhanced architecture.









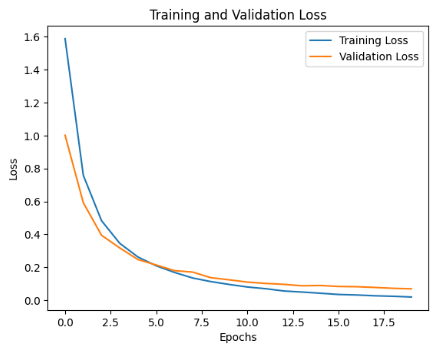
| Experiment 1 Model Classification Report | | | | |
|---|---|---|---|---|
| **Metric/Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| 0 | 1 | 1 | 1 | 111 |
| 1 | 1 | 1 | 1 | 105 |
| 2 | 1 | 0.99 | 1 | 105 |
| 3 | 0.99 | 0.95 | 0.97 | 85 |
| 4 | 0.96 | 0.98 | 0.97 | 94 |
| 5 | 0.98 | 0.97 | 0.97 | 98 |
| 6 | 1 | 1 | 1 | 98 |
| 7 | 0.99 | 0.95 | 0.97 | 112 |
| 8 | 0.92 | 1 | 0.96 | 81 |
| 9 | 0.99 | 1 | 1 | 111 |
| **Accuracy** | | | 0.98 | 1000 |
| **Macro Avg** | 0.98 | 0.98 | 0.98 | 1000 |
| **Weighted Avg** | 0.98 | 0.98 | 0.98 | 1000 |

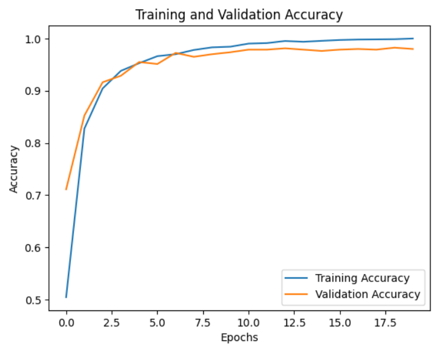
## 2.5 Second Experiment

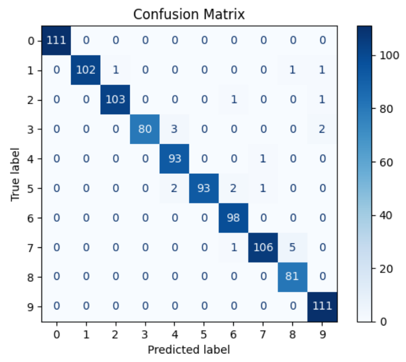
In the second experiment, the focus was on tuning hyperparameters while retaining the architecture of the base model. Adjustments included:

- **Batch Size:** Reduced to 16
- **Number of Epochs:** Increased to 20

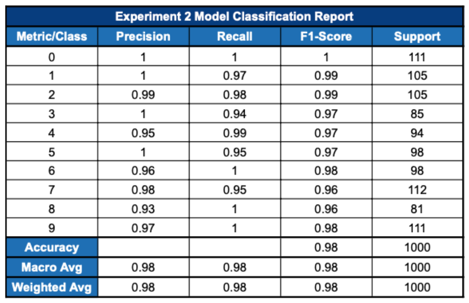
These changes allowed for more frequent weight updates and longer training time, promoting better generalization. The test accuracy reached 97.8%, with noticeable improvements in resolving challenging digit pairs. This experiment demonstrated that even with a simpler architecture, hyperparameter tuning can lead to substantial performance gains.







| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 111 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 102 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 103 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 80 | 3 | 0 | 0 | 0 | 0 | 2 |
| 4 | 0 | 0 | 0 | 0 | 93 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 2 | 93 | 2 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 98 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 106 | 5 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 81 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 111 |



**Experiment 2 Model Classification Report**

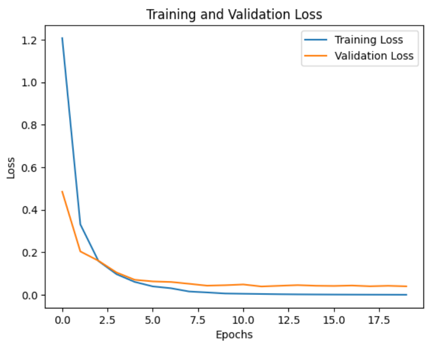
| Metric/Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 111 |
| 1 | 1 | 0.97 | 0.99 | 105 |
| 2 | 0.99 | 0.98 | 0.99 | 105 |
| 3 | 1 | 0.94 | 0.97 | 85 |
| 4 | 0.95 | 0.99 | 0.97 | 94 |
| 5 | 1 | 0.95 | 0.97 | 98 |
| 6 | 0.96 | 1 | 0.98 | 98 |
| 7 | 0.98 | 0.95 | 0.96 | 112 |
| 8 | 0.93 | 1 | 0.96 | 81 |
| 9 | 0.97 | 1 | 0.98 | 111 |
| Accuracy | | | 0.98 | 1000 |
| Macro Avg | 0.98 | 0.98 | 0.98 | 1000 |
| Weighted Avg | 0.98 | 0.98 | 0.98 | 1000 |

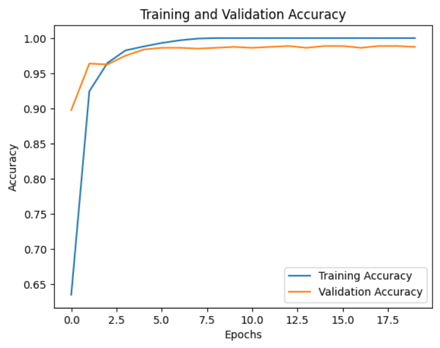
## 2.6 Best model Results and Metrics

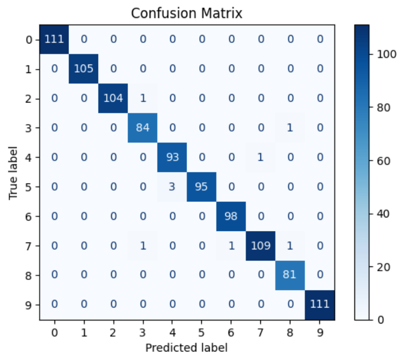
The optimal model integrated architectural enhancements from the first experiment and hyperparameter tuning from the second. Its final structure was:

- **Input Layer:** 784 neurons
- **First Hidden Layer:** 128 neurons with ReLU activation
- **Second Hidden Layer:** 32 neurons with ReLU activation
- **Output Layer:** 10 neurons with softmax activation

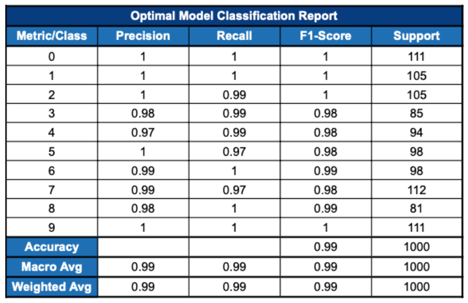
The model was trained for 20 epochs with a batch size of 16. It achieved the highest test accuracy of 99.1%. Both loss and accuracy curves indicated stable convergence, and the confusion matrix analysis revealed minimal errors. This optimal configuration effectively balanced model complexity and generalization, showcasing the importance of combining architectural design and hyperparameter optimization.







| True label \ Predicted label | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 111 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 105 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 104 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 84 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 0 | 93 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 3 | 95 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 98 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 109 | 1 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 81 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 111 |



| Optimal Model Classification Report | | | | |
|---|---|---|---|---|
| Metric/Class | Precision | Recall | F1-Score | Support |
| 0 | 1 | 1 | 1 | 111 |
| 1 | 1 | 1 | 1 | 105 |
| 2 | 1 | 0.99 | 1 | 105 |
| 3 | 0.98 | 0.99 | 0.98 | 85 |
| 4 | 0.97 | 0.99 | 0.98 | 94 |
| 5 | 1 | 0.97 | 0.98 | 98 |
| 6 | 0.99 | 1 | 0.99 | 98 |
| 7 | 0.99 | 0.97 | 0.98 | 112 |
| 8 | 0.98 | 1 | 0.99 | 81 |
| 9 | 1 | 1 | 1 | 111 |
| Accuracy | | | 0.99 | 1000 |
| Macro Avg | 0.99 | 0.99 | 0.99 | 1000 |
| Weighted Avg | 0.99 | 0.99 | 0.99 | 1000 |

## 3 Discussion

The experiments underscored the pivotal role of iterative refinement in deep learning, highlighting how methodical adjustments to architectural design and hyperparameter configurations can drive substantial improvements in model performance. By enhancing the complexity of the network's structure, the model was able to capture more intricate patterns and relationships within the data, thereby achieving heightened levels of accuracy and reliability. Furthermore, hyperparameter tuning fine-tuned the training process, ensuring a balance between convergence speed and generalization capacity. The final model exemplifies the synergy between these approaches in tackling complex classification tasks effectively. Moving forward, incorporating a convolutional neural network (CNN) will enable the model to effectively process spatial hierarchies and patterns inherent in image data, offering superior accuracy and robustness for real-world applications.

## 4 Code

The complete implementation can be accessed via the Colab notebook:
**mid2_311402812_207389909_basics_2024**

## References

- **Keras API Documentation**
- **Keras to_categorical Utility**
- **Keras Dense Layer**
- **Adam Optimizer in Keras**
- **Scikit-learn Accuracy Score**
- **Scikit-learn Confusion Matrix**
- **Scikit-learn Classification Report**