

### Project Report

University of Hull

MSc. Artificial Intelligence

**Machine Learning & Deep Learning (771948\_A23\_T3A)**

L4097-Computer Science

Henny Purwadi

202370140

Nareshwar Raju Vaneshwar

202410472

Supervisor: Prof. Adil Khan, Khadija Fraz

Date of submission: 10th August 2024

Contents

[Project Report ii](#_Toc173012812)

[List of Figure iii](#_Toc173012814)

[List of Tables iii](#_Toc173012814)

[List of Abbreviation](#_Toc173012814) iv

[Task 1 – Classification Problem (Single label Classification)……………………………………](#_Toc173012813) 1

[Introduction 1](#_Toc173012814)

[Dataset Overview and Pre-processing 1](#_Toc173012815)

[Data Loading and Initial Exploration 1](#_Toc173012816)

[Pre-processing Steps 1](#_Toc173012817)

[Handling Missing Data 1](#_Toc173012818)

[Challenges 1](#_Toc173012819)

[Methodology and Techniques 2](#_Toc173012820)

[Results and Discussion 2](#_Toc173012821)

[Visualizations 2](#_Toc173012822)

[Model Comparison and Selection 5](#_Toc173012823)

[Hyperparameter Tuning on Random Forest classifier 5](#_Toc173012824)

[Conclusion 5](#_Toc173012837)

[Task 2 – Triple MNIST (Multi labels dataset) 6](#_Toc173012825)

[Dataset Overview and Pre-processing 6](#_Toc173012826)

[Data Loading and Initial Exploration 6](#_Toc173012827)

[Pre-processing Steps 6](#_Toc173012828)

[Challenges 7](#_Toc173012829)

[Methodology and Techniques 7](#_Toc173012830)

[Model Comparison and Selection 7](#_Toc173012831)

[Hyperparameter Tuning 7](#_Toc173012832)

[Results and Discussion: 7](#_Toc173012833)

[Save the Tuned Model and Load to Predict on Test Image 11](#_Toc173012834)

[Visualization 12](#_Toc173012834)

[Conclusion 19](#_Toc173012837)

[Ethical, Legal and Social Considerations in AI 19](#_Toc173012838)

[References 20](#_Toc173012825)

# List of Figures

Figure 1. Feature Importance on Random Forest 3

Figure 2. Limited Depth of Decision Tree 4

Figure 3. Partial Dependence Plots 4

Figure 4. ROC Curve using Random Forests 5

Figure 5. Example of Triple MNIST Dataset 6

Figure 6. Sigmoid Activation and Categorical Cross Entropy as the Loss Function 7

Figure 7. Sigmoid Activation and Binary Cross Entropy as the Loss Function 8

Figure 8. SoftMax Activation and Binary Cross Entropy as the Loss Function 9

Figure 9. Keras Tuner Best Parameter on SoftMax Activation & Binary Cross Entropy 10

Figure 10. Save the Tuned Model and Load to Predict on test image 11

Figure 11. Confusion Matrix for Digit 0 12

Figure 12. Confusion Matrix for Digit 1 and Digit 2 13

Figure 13. Visual Representation of The Fitters/Kernels and Feature Maps 15

Figure 14. Basic Saliency Map 16

Figure 15. Grad Cam 16

Figure 16. Deep Dream Visualization 17

Figure 17. Guided Back Propagation Saliency Map 17

# List of Tables

Table 1. Model Comparison and Selection 5

# List of Abbreviations

AI Artificial intelligence.

ANN Artificial Neural Network.

CNN Convolutional Neural Network.

OCR Optical Character Recognition.

ML Machine learning.

KNN K-Nearest Neighbours.

ROC Receiver Operating Characteristic.

SVM Support Vector Machine.

Task 1 – Classification Problem (Single label Classification)

Introduction

In this assignment, we demonstrate several machine learning techniques to solve classification problems. By processing the datasets (both numerical and categorical features), we explore the processes of data pre-processing, model building, and evaluation.

This assignment showed the importance of data preparation, model selection, and fine-tuning to achieve the most optimal performance in machine learning.

Dataset Overview and Pre-processing

Dataset1 consisted of 925 rows and includes the features var1, var2, var3, var4, var5, var6, and var7, with the target column as the label.

Among these features, var3 and var6 are categorical, while var7 contains datetime values. The other features such as var1, var2, var4, and var5, are numerical.

Data Loading and Initial Exploration

The dataset was loaded, and initial exploration was conducted to understand its structure and identify any data quality issues.

The first step involved checking for missing values, and this revealed a significant amount of missing data in the var4 column. Missing data in var4 about 600 missing values (about 65% missing).

Pre-processing Steps

For the categorical features, we converted var3 and var6 into numerical values using label encoding.

Regarding the datetime feature, var7, we converted it to a datetime format and split it into var7\_date and var7\_time.

Handling Missing Data

Since var4 had around 65% missing values, we explored how to impute missing values in a dataset using a KNN imputer.

The KNN (K-Nearest Neighbour) imputer is a method that replaces missing values based on the mean or median values of the nearest neighbours.

Since KNN works with distance computations, first we scale the dataset.

Challenges

We faced several challenges during the pre-processing phase of the dataset.

One major issue was handling a significant amount of missing data in var4, which required careful consideration of imputation techniques or potential removal. Ensuring the correct encoding of categorical features, such as var3 and var6, was important to maintain the integrity of the data.

Properly managing and splitting datetime values in var7 also posed a challenge, as it was needed to accurately separate and format the date and time components for further analysis.

Methodology and Techniques

Model Selection: Applied Logistic Regression, Decision Trees, Random Forests, and Artificial Neural Networks (ANN) to the classification problem.

Hyperparameter Tuning: Use grid Search and cross-validation to fine-tune model parameters.

Evaluation Metrics: Assessed models using accuracy, precision, recall, and F1-score.

Results and Discussion

Random Forests model with or without normalization outperformed other models and achieved the highest accuracy, around 96.2%.

Important features for the Random Forest model are var1, var5, and var 2. We plotted feature importance on Random Forests, Limited Depth Decision Trees, and Partial Dependence Plots to compare the features.

Visualizations

Feature importance

**Figure 1. Feature Importance on Random Forests**

A graph with a bar graph

Description automatically generated with medium confidence

**Source: Own representation.**

Limited Depth Decision Tree

**Figure 2. Limited Depth of Decision Tree**

A diagram of a diagram

Description automatically generated

**Source: Own representation.**

#### Partial Dependence Plots

**Figure 3. Partial Dependence Plots**

A graph of a number of blue lines

Description automatically generated with medium confidence

**Source: Own representation.**

Since var4 is not a significant feature, imputing the mean or removing the column does not affect the accuracy of the model.

The Confusion Matrix and heatmap for the final selected model shows that the model has high and reliable performance.

#### ROC Curve using Random Forests

**Figure 4. Partial Dependence Plots**

**A graph of a line

Description automatically generated with medium confidence**

**Source: Own representation.**

Model Comparison and Selection

**Table 1. Model Comparison and Selection**

|  |  |
| --- | --- |
| Classifier | Accuracy |
| Random Forests | 0.9621621621621622 |
| Artificial Neural Networks | 0.9567567706108093 |
| Logistic Regression | 0.9513513513513514 |
| Support Vector Machines | 0.9424460431654677 |
| Decision Trees | 0.9405405405405406 |

**Source: Own representation.**

Hyperparameter Tuning on Random Forest classifier

Best Parameters:

{

'bootstrap': [False],

    'max\_depth': [None],

    'max\_features': ['sqrt'],

    'min\_samples\_leaf': [1],

    'min\_samples\_split': [10],

    'n\_estimators': [200]

}

Conclusion

The Random Forest classifier achieved the highest accuracy on the training dataset compared to other models. Initially, its train accuracy was 0.962, which increased to 0.978 after hyperparameter tuning. The test set accuracy was 0.950

Effective data pre-processing is important for model performance. Ensemble methods like Random Forest can provide significant improvements. Fine-tuning and cross-validation are necessary steps to optimize model performance.

In future, we can explore more advanced techniques for handling missing data and feature engineering. Experiment with other methods and deeper neural networks and investigate the impact of feature selection and dimensionality reduction techniques.

Task 2 – Triple MNIST (Multi labels dataset)

Dataset Overview and Pre-processing

Task 2 assignment is focusing on the objectives and importance of multi-label classification on multiple digits from images, where each image contains several digits.

The dataset for this project consists of images each labelled with three digits.

**Figure 5. Example of Triple MNIST Dataset**

A black and white image of circles

Description automatically generated

**Source: Triple MNIST Dataset.**

Data Loading and Initial Exploration

To load and explore the dataset, we define the load\_images\_from\_folder function, which reads images from subdirectories, resizes them, converts them to grayscale, and assigns labels based on subdirectory names. Directories for training, validation, and test datasets are specified to organize the data.

We print the contents of these directories to verify their structure. We load the datasets into arrays of images and corresponding labels. This process prepares the dataset for further processing and model training.

Pre-processing Steps

Pre-processing steps were taken to prepare the data for modelling.

This includes reshaping and normalizing pixel values, one-hot encoding of labels to convert labels to categorical, creation of a custom data generator for efficient data handling, setting up learning rate scheduler to adjust the learning rate during training, and configuring call-backs for early stopping to prevent overfitting by stopping the training when the validation loss stops improving.

Challenges

The challenges during pre-processing involved reshaping the images to fit the model's input requirements, normalizing pixel values to the range [0, 1], and encoding the labels for a multi-label setup using one-hot encoding. We faced several challenges when we needed to choose which activation function and which loss function, we need to use.

Methodology and Techniques

CNN models were developed, comparing categorical cross entropy with Binary cross entropy as loss functions and Sigmoid versus SoftMax as activation functions. Techniques like Early Stopping, Drop out were used to improve training performance and avoid overfitting.

Model Comparison and Selection

Binary cross entropy as the loss function outperformed categorical cross entropy, showing lower loss and higher accuracy for multi-label classification, demonstrating its suitability for this task.

While recommendations suggest Sigmoid for multi-label and SoftMax for multi-class, our experiment shows the use of SoftMax in the output layer achieved the highest accuracy.

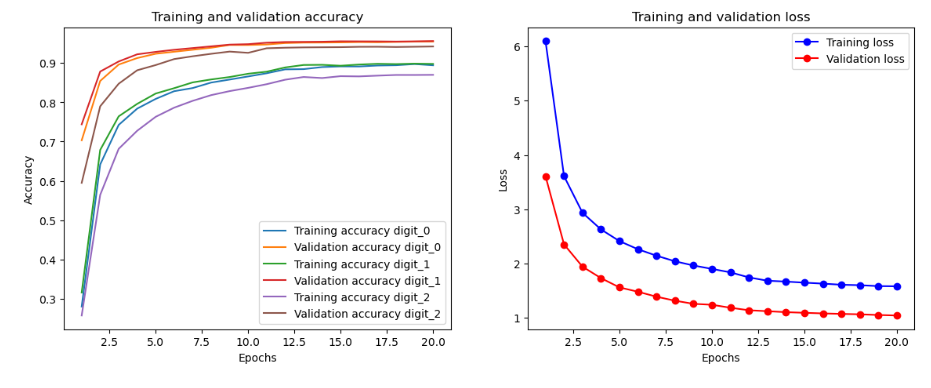
Hyperparameter Tuning

Hyperparameter tuning is performed using Keras-Tuner to find the best parameters, adjust the model parameters like learning rate and increase K-fold cross-validation to more accurately evaluate model performance across different subsets of data, and implement batch normalization in the model to normalize the inputs for each mini-batch, reducing internal covariate shift and stabilizing training.

Results and Discussion:

Initial models using **Sigmoid activation and categorical cross entropy** as the loss function achieved high accuracy, indicating no underfitting or overfitting.

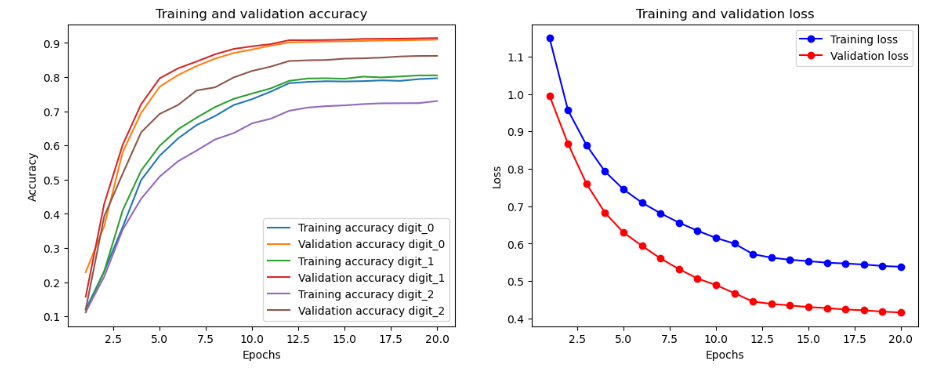
**Figure 6. Sigmoid activation and Categorical Cross Entropy as the Loss Function**



**Source: Own representation.**

Switching to **Sigmoid activation and Binary cross entropy** as the loss function reduced losses (around 0.3), but lowered accuracy.

**Figure 7. Sigmoid Activation and Binary Cross Entropy as the Loss Function**



**Source: Own representation.**

The model using **SoftMax activation and Binary Cross Entropy** as the loss function showed better performance with higher accuracy and lower loss.

Using **Keras-Tuner**, we found the best parameters and implemented into the model:

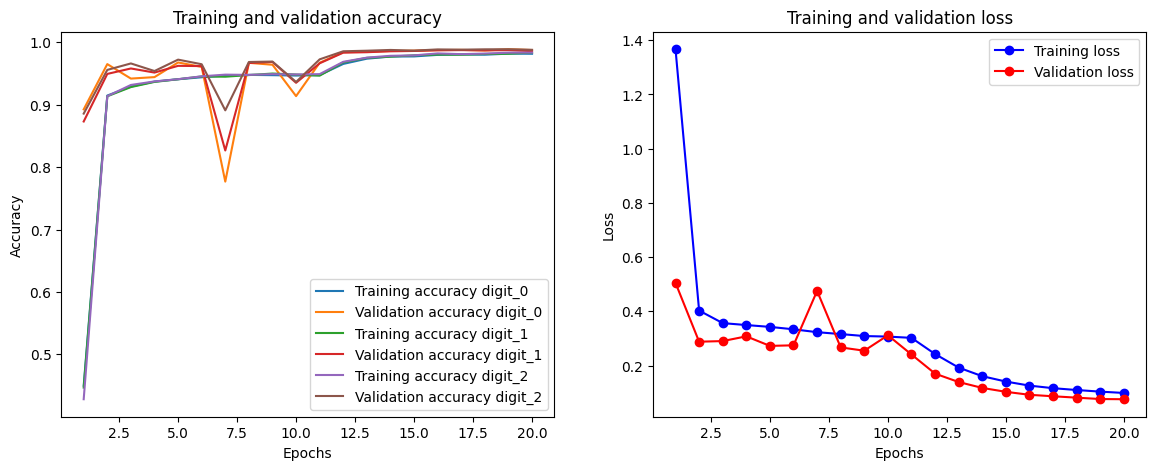
val\_loss: 0.07059618085622787

Best val\_loss So Far: 0.06173119693994522

Total elapsed time: 05h 47m 46s

{'conv\_1\_filters': 96, 'conv\_2\_filters': 128, 'conv\_3\_filters': 128, 'dense\_units': 256}

**Figure 8. SoftMax Activation and Binary Cross Entropy as the Loss Function**



**Source: Own representation.**

After hyperparameter tuning, the model demonstrated improvement and better performance.

digit\_0\_accuracy: 0.9817 - digit\_1\_accuracy: 0.9839 - digit\_2\_accuracy: 0.9847 - loss: 0.0987 - val\_digit\_0\_accuracy: 0.9876 - val\_digit\_1\_accuracy: 0.9858 - val\_digit\_2\_accuracy: 0.9879 - val\_loss: 0.0758

In multi-class classification, the final layer of the model employs a SoftMax function to predict the class, and the training utilizes categorical cross-entropy as the loss function. Conversely, in multi-label classification, the last layer uses a sigmoid function to predict labels, with binary cross-entropy serving as the loss function. (franky, 2018).

But from our experiment, final testing on unseen data after Hyperparameter Tuning on SoftMax activation function and Binary Cross Entropy as the loss function achieved the model's highest accuracy (around 98%) and lowest loss (around 0.07).

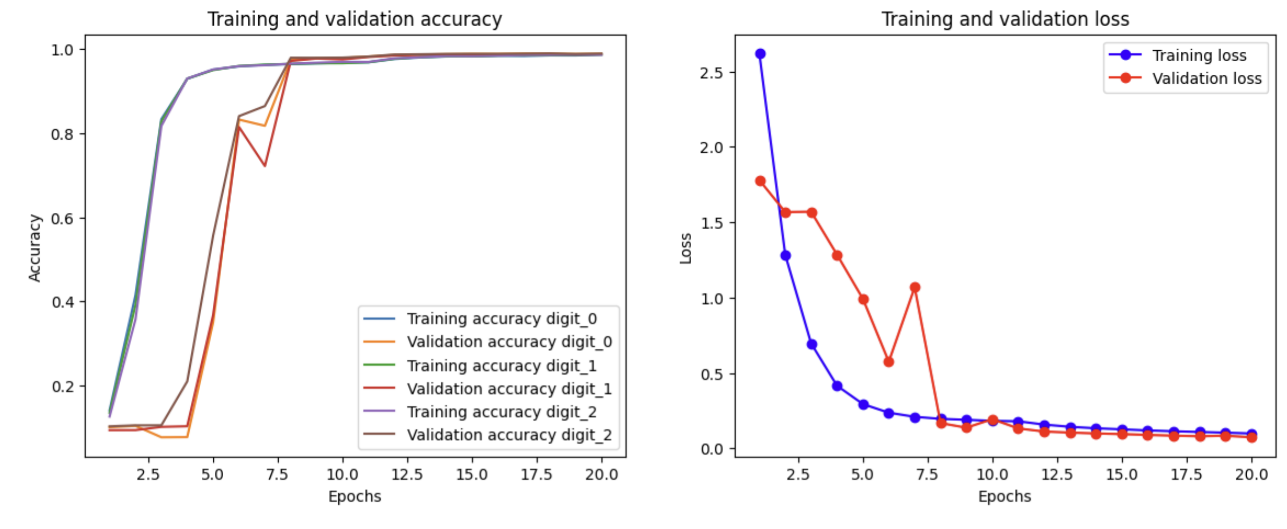
Test loss: 0.07689374685287476

0.986299991607666

0.9864500164985657

0.9870499968528748

**Figure 9. Keras Tuner Best Parameter on SoftMax Activation and Binary Cross Entropy**



**Source: Own representation.**

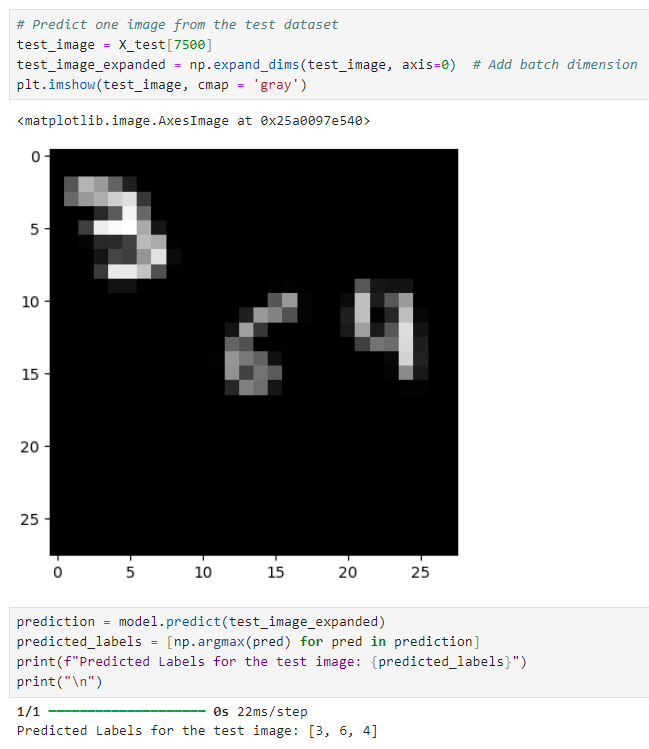
Save the tuned model and load to predict on test image

To know whether the trained model capable of making predictions on test images, after tuning and training the model, we save it to a file using the.keras format.

To make predictions, we load the saved model and use it to make predictions on individual images from the test set.

The model is designed to make accurate predictions on the test images. By loading the saved model and using it to predict the labels of a test image, we demonstrate its capability to correctly identify the labels.

**Figure 10. Save the tuned model and load to predict on test image**



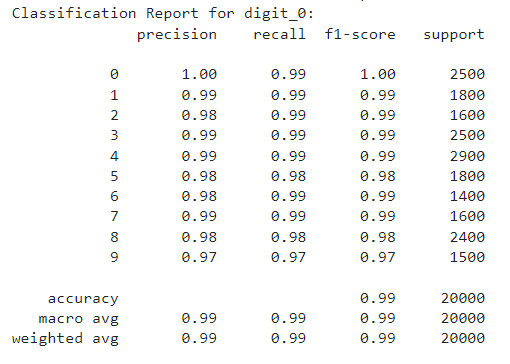
**Source: Own representation.**

Visualization

We visualize the results using classification report, confusion matrix heatmaps, feature maps, Pydot, saliency maps, Grad-CAM, Deep Dream Visualization, and Guided Back Propagation Saliency maps.

Confusion Matrix

**Figure 11. Classification Report for digit\_0**

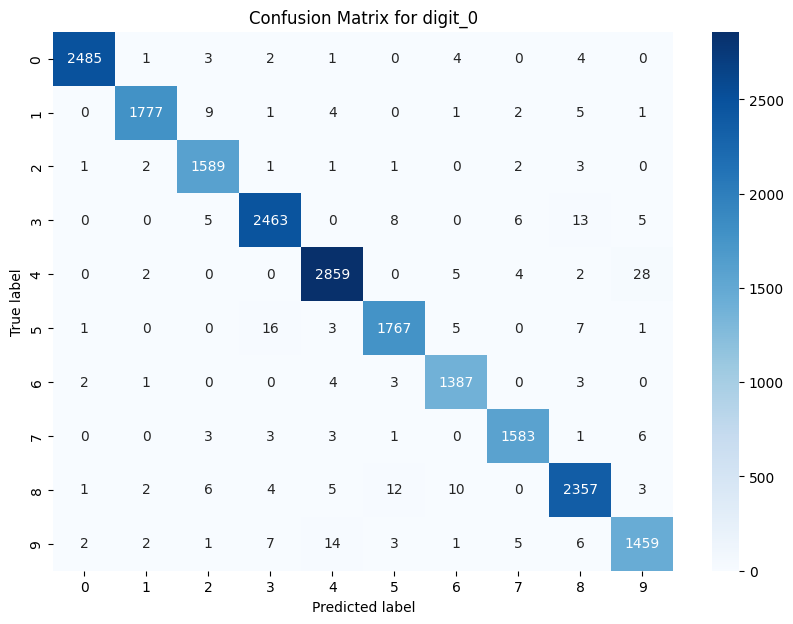


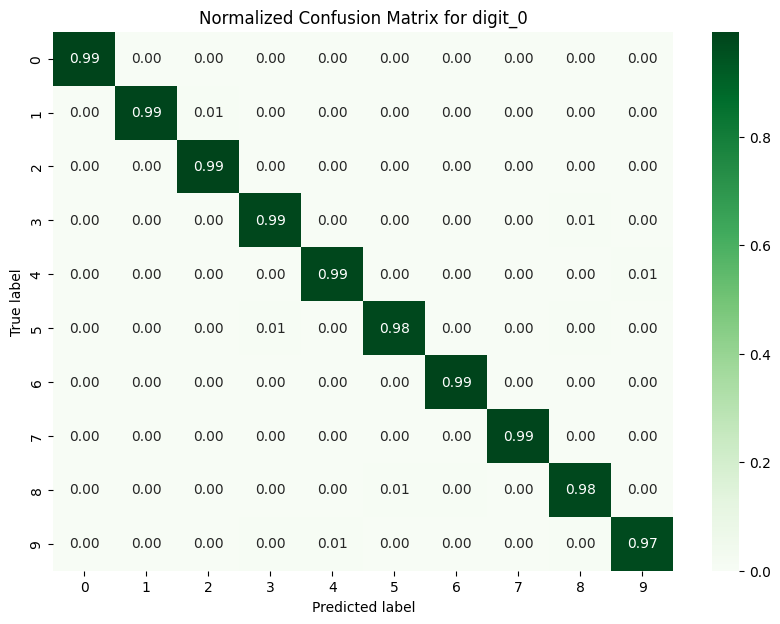
**Source: Own representation.**

The model shows excellent performance with high precision, recall, and F1-scores across all classes. The accuracy is 99%, indicating that the model correctly predicts most instances.

Confusion Matrix

**Figure 12. Confusion Matrix and Normalized Confusion Matrix for digit\_0**

**Source: Own representation.**



**Source: Own representation.**

The diagonal elements represent the correctly classified ones, with the highest values along the diagonal showing correct predictions. Off-diagonal values show where the model is making errors.

Visual Representation of The Filters/ Kernels and Feature map

Filter/Kernel is small matrices that slide over the input image to detect specific features such as edges, textures, or patterns. Each filter is applied across the entire input image, and the result is a feature map.

The visual representation of the filters helps to understand what each filter is focusing on in the input images. The filters have dimensions (3, 3, 1, 96), where 3 and 3 are the height and width of each filter. 1 is the number of input channels for grayscale image; there is only one channel. 96 is the number of filters in this layer.

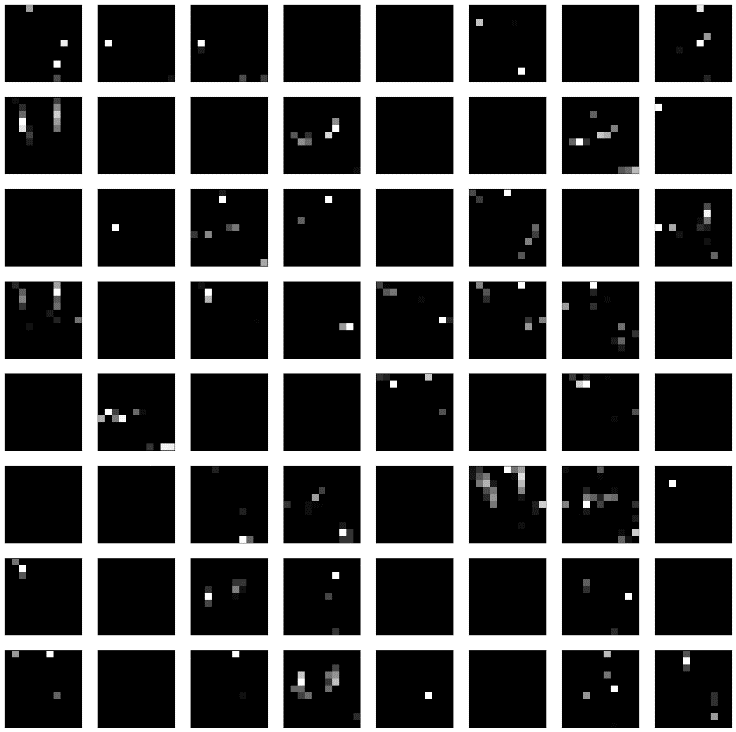
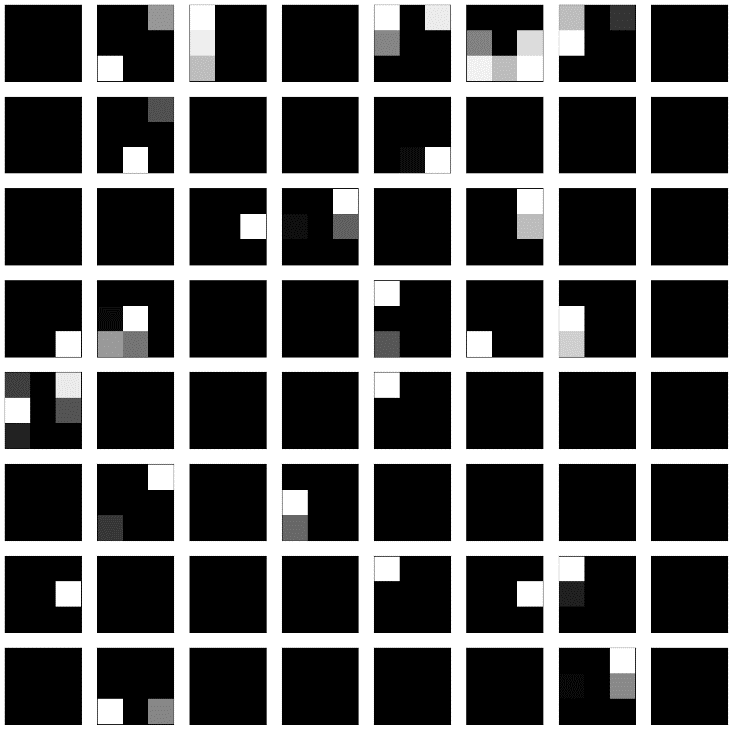
The filters have purpose of detect specific features within the input images. During training, the weights of these filters are adjusted to minimize the loss function, thereby learning the features that are most relevant for the task. Visualizing the filters can give insights into what the model is learning.

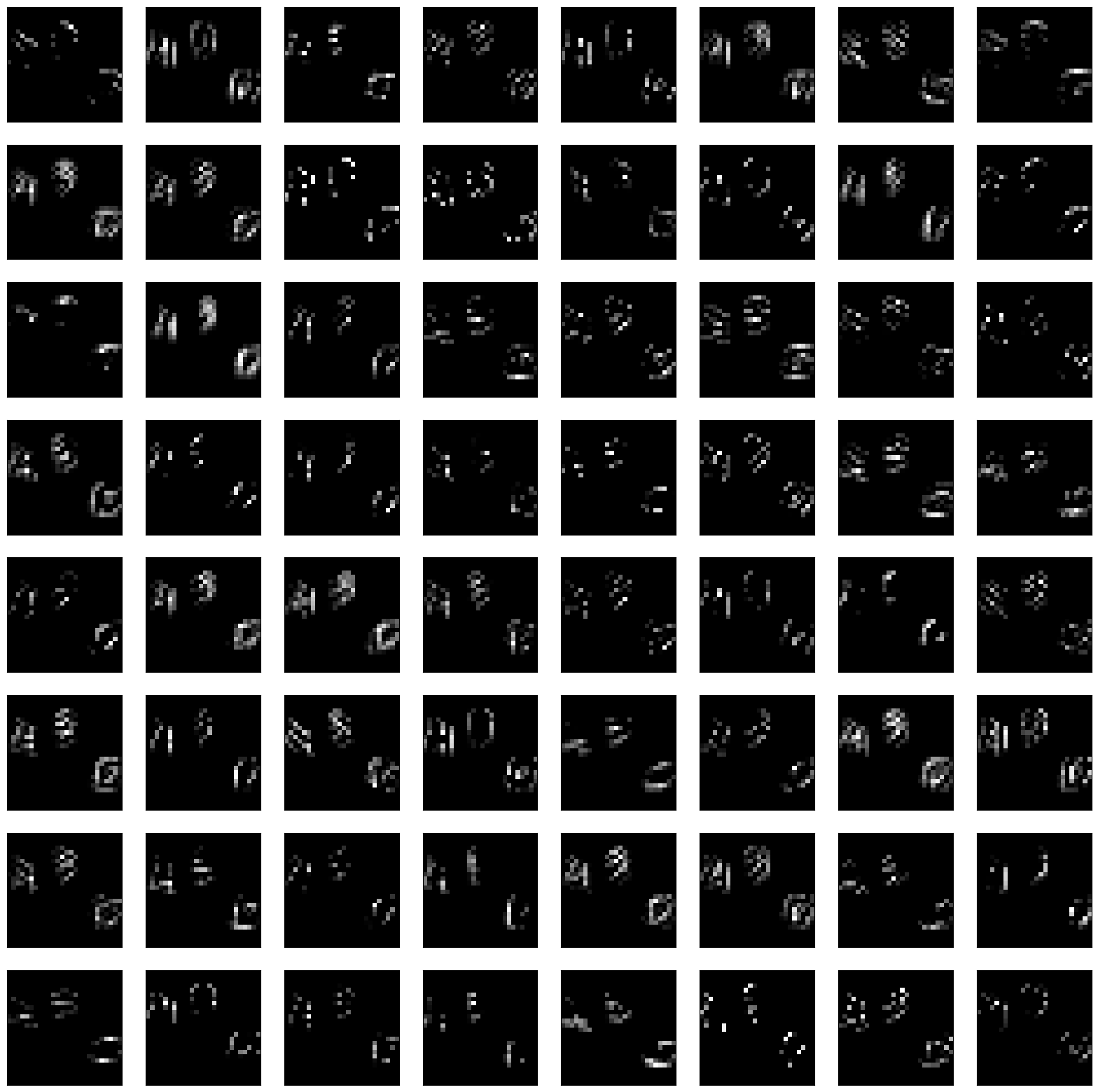
In this example, we chose a random handwritten image (number 430) from the training dataset and predicted the feature outputs.

A number on a black background

Description automatically generated

**Figure 13. Visual Representation of The Filters/ Kernels and Feature map**





**Source: Own representation.**

The patterns seen in the feature show what each filter has learned. For example, some may detect edges, other textures, and some specific shapes. Early layers earn simple features, then deeper layers learn more complex features and patterns.

**Basic Saliency Map**

Basic Saliency maps highlight the most important areas of the input image at pixel level, showing which parts of the input image have the greatest impact on the model’s output prediction, regardless to make it better or worse.

**Figure 14. Basic Saliency Map**

A red and yellow pixelated image

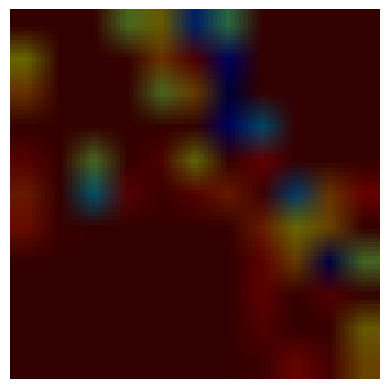
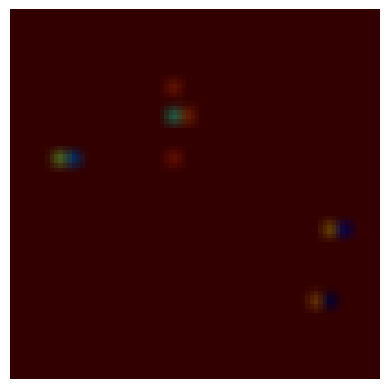
Description automatically generated

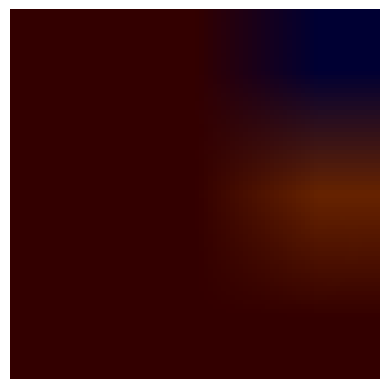
**Source: Own representation.**

**Grad-CAM**

Similar purpose with the Saliency Map but smoother, Grad-CAM highlights the most important areas of the input image at the region level, showing which parts of the input image contribute most to the output prediction.

**Figure 15. Grad Cam**





**Source: Own representation.**

**Deep Dream Visualization**

DeepDream visualization takes the input image(s), and enhances the patterns and curves what a network would typically see, and then create a dream-like visuals to showcase the learned features.

**Figure 16. Deep Dream Visualization**

A group of colorful letters

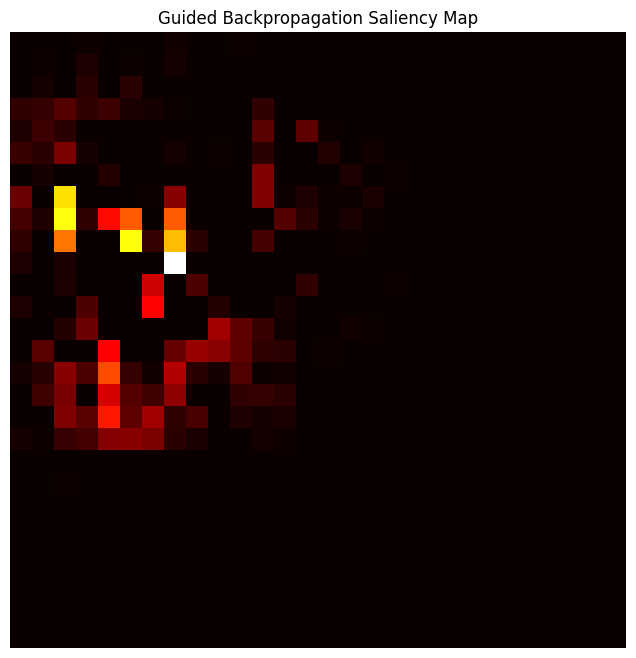
Description automatically generated

**Source: Own representation.**

**Guided Back Propagation Saliency Map**

Unlike basic saliency map which allows positive and negative gradients if they make greatest contribution to the output prediction, guided back propagation saliency map method refined the basic saliency map by only allowing positive gradients to backpropagate through the network. Therefore, it refines the basic saliency map to highlight more relevant features.

**Figure 17. Guided Back Propagation Saliency Map**



**Source: Own representation.**

Conclusion

The project successfully demonstrated the application of CNNs to multi-label image-based digit classification, with SoftMax activation and binary cross entropy as the loss function showing the highest performance over Sigmoid activation and categorical cross entropy as the loss function. They are improving more after hyperparameter tuning. This is showing the importance of selecting appropriate loss functions and optimizing model architectures.

Future work could explore OCR (Optical Character Recognition) Systems. The dataset can be utilized by researchers and developers to train and evaluate OCR systems for the recognition of handwritten multi-digit numbers.

Furthermore, researchers could more complex architectures, additional regularization techniques, and further hyperparameter tuning to enhance model performance and robustness, potentially experimenting with different convolutional layer configurations to improve focus on relevant parts of the image.

Ethical, Legal, and Social Considerations in AI

When developing AI models for multi-label image classification, such as **handwritten** triple MNIST digits, it's important to address ethical, legal, and social issues.

Data protection laws like GDPR or CCPA must be adhered to, ensuring that any personal data is handled with strict privacy measures.

The ownership of the AI model, its training data, and its outputs need to carefully consider ensuring responsible and beneficial AI deployment.

# References

franky (2018) 'Multi-Label Classification and Class Activation Map on Fashion-MNIST', Towards Data Science, 2 July. Available at: https://towardsdatascience.com/multi-label-classification-and-class-activation-map-on-fashion-mnist-1454f09f5925 (Accessed: 20 June 2024).