**ICT 203 Assignment 2 Final Report**

*Murdoch University*

EDITED BY

**Keith Neo Kai Si**

**35107628**

*A Comprehensive Evaluation of AI techniques for Handwritten Digit Recognition*

Table of Contents

[**Introduction** 2](#_Toc172387963)

[**Background** 4](#_Toc172387964)

[**AI Techniques** 6](#_Toc172387965)

[**Evaluation Method** 8](#_Toc172387966)

[**Results** 10](#_Toc172387967)

[**Discussion and Conclusion** 13](#_Toc172387968)

[**Acknowledgements** 16](#_Toc172387969)

[**User Guide** 18](#_Toc172387970)

[**References** 19](#_Toc172387971)

## **Introduction**

The primary objective of this project is to investigate and implement classification algorithms leveraging PyTorch, with a specific focus on recognising handwritten digits. This task holds significant importance as it emulates applications akin to the US Postal Service’s mail sorting system, which utilises similar technology. Handwritten digit recognition has many real-world applications, including Optical Character Recognition (OCR), automated form processing, and digitised archival systems.

**Problem Domain:**

The central problem revolves around accurately classifying images of handwritten digits (0-9) into their respective classes using machine learning techniques. The challenge lies in developing robust classifiers that can generalise effectively to unseen data and achieve high accuracy despite variations in writing styles and image quality.

**Dataset Overview:**

Sample images of handwritten digits (0-9) are shown below to help you visually understand the dataset used for this assignment. Each image is grayscale and has dimensions of 28x28 pixels.

A collage of numbers

Description automatically generated

*Fig. 1 – dataset overview* (mnist, 2024)

**Objectives:**

The main objectives of this assignment are:

1. **Implementation of Classification Algorithms:** Design and compare at least two classifiers:

* **Naïve Bayes Classifier:** A probabilistic model known for its simplicity and efficiency in classification tasks.
* **Alternative Classifier:** A neural network-based model (CNN) chosen for its ability to capture intricate patterns and relationships in image data.

1. **Practical Applications:** Gain hands-on experience in:

* Understanding and preprocessing the dataset of handwritten digits.
* Training, validating, and testing classifiers using PyTorch.
* Evaluating and comparing the performance of different classifiers based on metrics like accuracy and computational efficiency.

**Significance:**

By successfully implementing these classifiers, the project aims to enhance understanding of:

* Classification techniques and their practical application in image recognition
* The impact of model complexity and architecture on performance metrics.
* The role of machine learning in automated data analysis and decision-making processes.

## **Background**

In exploring handwritten digit recognition, I have delved into various AI techniques, each offering unique insights and perspectives on this challenging problem. By investigating and gaining hands-on experience with these different approaches, I have developed a more comprehensive understanding of the field and the trade-offs associated with each methodology.

1. **Naïve Bayes Classifier:**

I began my journey by investigating the use of Naïve Bayes classifiers, which provided a solid foundation for understanding the application of probability theory for digit classification. These models proved effective for specific text-based tasks, where the feature independence assumption holds reasonably well. However, in the context of image data, I found that Naïve Bayes struggled to capture the complex spatial relationships inherent in handwritten digits. This limitation prompted me to explore more sophisticated techniques to handle better the nuanced patterns presented in visual data.

1. **Convolutional Neural Network (CNN):**

Exploring CNN has been a transformative experience in my journey. These deep learning models have revolutionised the field of image recognition by automatically learning hierarchical features directly from pixel data. Their ability to effectively discern intricate patterns and textures has made them highly suitable for handwritten digit classification, where local characteristics are crucial for accurate identification. The data-driven nature of CNN and its remarkable performance have sparked my fascination with the intersection of neural networks and computer vision.

1. **Decision Trees:**

While the powerful capabilities of deep learning models like CNN have been compelling, I have also explored alternative approaches, such as decision trees. These interpretable models offered insights into developing AI systems that can handle nonlinear relationships between features and labels. However, their performance in image classification tasks highlights the importance of feature engineering and the challenges in scaling them to high-dimensional data, such as images, without extensive preprocessing. This experience has underscored the need to balance model interpretability and performance, mainly when dealing with complex visual data.

1. **Natural Language Processing (NLP):**

Although not directly applicable to image classification, my investigation of Natural Language Processing (NLP) techniques, such as recurrent neural network (RNN) and transformer models, underscored the significance of context and sequential information in AI systems. These insights have influenced my understanding of practical model design principles, even in handwritten digit recognition. While the specific architectures and algorithms may differ, the fundamental concepts of capturing contextual dependencies and leveraging sequential data have provided valuable perspectives I can apply to my ongoing work.

**Justification for CNN as the Alternative Classifier:**

My decision to adopt CNN as the alternative classifier stems from a profound appreciation for their:

* Ability to autonomously extract meaningful features from digit images.
* Capacity to capture intricate spatial dependencies critical for accurate classification.
* Potential to surpass traditional models in performance and adaptability to diverse handwriting styles.

## **AI Techniques**

In developing a robust handwritten digit recognition system, I comprehensively evaluated various AI techniques and their strengths and weaknesses in handling image classification tasks. Based on this evaluation, I selected Convolutional Neural Network (CNN) as the alternate classifier for the assignment. This decision was guided by the unique capabilities of CNN that make them well-suited for handwritten digit recognition.

**Justification for Selecting CNN:**

**Feature Extraction:**

CNN excels at automatically extracting relevant features from raw pixel data, capturing the spatial hierarchies and local patterns essential for image recognition tasks. This capability surpasses traditional methods like Naïve Bayes and linear regression, which require extensive manual feature engineering and struggle with high-dimensional data.

**Handling non-linearity:**

Compared to decision trees and random forest classifiers, which model complex interactions among features, CNN can inherently capture non-linear relationships through their multiple convolutional layers and non-linear activation functions. This allows for a more nuanced understanding of the image data, leading to higher accuracy in handwritten digit recognition.

**Scalability:**

CNN has demonstrated superior scalability compared to linear regression and Naïve Bayes models. My experience has shown that while the latter models struggled with large datasets, CNN maintained robust performance across different scales, making them a more suitable choice for your handwritten digit recognition system.

**Established Effectiveness in Image Recognition:**

Extensive research, such as the studies by (Yann LeCun, 1998) on the LeNet architecture and the advancements by (Alex Krizhevsky, 2017), has firmly established CNN as the state-of-the-art in computer vision tasks, including image recognition. This prior research strongly supports the effectiveness of CNN in this assignment.

**Parameter for Optimising CNN:**

**Learning Rate:**

The learning rate is a crucial parameter that controls the step size in the optimisation process. Fine-tuning the learning rate is essential for balancing convergence speed and stability. Through experimentation, I identified a learning rate 0.001 as optimal for my model, ensuring steady convergence without overshooting the minima.

**Batch Size:**

The batch size determines the number of samples processed before updating the model parameters. It influences the stability and speed of training. I experimented with batch sizes of 16, 32, and 64 and found that a batch size of 64 provided the best trade-off between memory efficiency and gradient estimation accuracy.

**Epochs:**

The number of epochs defines how often the training dataset passes through the model. It impacts the model’s ability to generalise and avoid overfitting. I evaluated the model performance over 8, 16, 32, and 64 epochs and settled on 32 epochs to achieve optimal training without overfitting.

**Optimiser:**

The choice of optimiser affects the training dynamics and convergence. I selected the Adam (Adaptive Moment Estimation) Optimiser because it adapts learning rates and accelerates convergence, as it efficiently handles sparse gradients and provides a stable and fast training process.

**Early Stopping:**

Early stopping is a technique that monitors the validation loss to prevent overfitting by halting training when performance stagnates. I employed early stopping with a patience of 5epochs and a minimum delta of 0.01, ensuring that training ceased upon convergence.

## **Evaluation Method**

A detailed comparative study was conducted to thoroughly assess and compare the performance of the Naïve Bayes and Convolutional Neural Network (CNN) classifiers. This section outlines the key metrics and observations used to measure the performance of both models.

1. **Accuracy:**

**Description:** Accuracy represents the proportion of correctly classified instances out of the total instances. It provides a general indication of the model’s overall performance.

**Formula:**

**Observation:** High accuracy suggests the model makes many correct predictions but does not account for potential class imbalances.

1. **F1 Score:**

**Description:** The F1 score is the harmonic mean of precision and recall. It offers a balanced measure between precision and recall, which is particularly useful for imbalanced datasets.

**Formula:**

**Observation:** A high F1 score indicates that the model balances precision and recall well.

1. **Confusion Matrix:**

**Description:** A confusion matrix is a tabular representation used to describe the performance of a classification mode. It shows the actual versus predicted classifications, enabling a detailed analysis of the model’s errors.

**Components:** True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN)

**Observation:** The confusion matrix provides insights into the model’s performance on individual classes, helping identify areas where the model misclassifies instances.

1. **Precision:**

**Description:** Precision measures the proportion of true positive predictions out of all positive predictions made by the model. It indicates the accuracy of the model’s positive predictions.

**Formula:**

**Observation:** High precision indicates that the model has a low false positive rate.

1. **Recall:**

**Description:** Recall measures the proportion of true positive predictions out of all actual positive instances. It indicates the model’s ability to capture all relevant instances.

**Formula:**

**Observation:** High recall indicates the model has a low false negative rate.

1. **ROC Curve and AUC:**

**Description:** The Receiver Operating Characteristic (ROC) curve plots the true positive rate against the false positive rate at various threshold settings. The Area Under the Curve (AUC) measures the model's overall performance.

**Observation:** An ROC curve closer to the top-left corner and a high AUC value suggest better model performance

1. **Class-wise Metrics:**

**Description:** Evaluating precision, recall, and F1 scores for each class provides a detailed view of the model’s performance in individual classes.

**Observation:** This analysis helps identify the classes where the model performs well and those that require further improvement.

1. **Precision-Recall Curve:**

**Description:** The precision-recall curve plots precision against recall at various threshold settings. It is particularly useful for evaluating models on imbalanced datasets.

**Observation:** A precision-recall curve closer to the top-right corner indicates better performance.

## **Results**

This section summarises the performance metrics of the Naïve Bayes and CNN models on the classification task. The metrics considered include accuracy, F1 score, precision, recall, and ROC curves.

|  |  |
| --- | --- |
| **Accuracy Comparison**  The overall accuracy of both models is compared in the bar chart below. | **F1 Score Comparison**  The F1 scores of the two models are compared in the bar chart below. |
| **A red white and blue flag  Description automatically generated**  *Fig. 2 Accuracy comparison between Naïve Bayes and CNN models.* | **A flag of italy with red green and white stripes  Description automatically generated**  *Fig. 3 F1 Score comparison between Naïve Bayes and CNN models.* |
| **Precision and Recall Comparison**  Both models' precision and recall scores are compared side by side in the bar charts below. | |
| **A red and blue rectangles  Description automatically generated**  *Fig. 4 Precision and Recall comparison between Naïve Bayes and CNN models.* | |

|  |  |
| --- | --- |
| **Detailed Metrics by Class**  This section provides a detailed breakdown of precision, recall, and F1 score for each class in the dataset. | |
| **A graph with blue and orange lines  Description automatically generated**  *Fig. 5 Class-wise Precision, Recall, and F1 Score for NB model.* | **A graph of a bar chart  Description automatically generated with medium confidence**  *Fig. 6 Class-wise Precision, Recall, and F1 Score for CNN model.* |
| **Confusion Matrix**  The confusion matrices for both models provide a detailed view of the classification results. | |
| **A purple and green chart with yellow and green squares  Description automatically generated**  *Fig. 7 Confusion Matrix for the Naïve Bayes model.* | **A chart with numbers and labels  Description automatically generated**  *Fig. 8 Confusion Matrix for the CNN model.* |
| **ROC Curves**  ROC curves illustrates the trade-off between true positive rates and false positive rates for each class. | |
| **A graph of a person with a line  Description automatically generated with medium confidence**  *Fig. 9 ROC Curve for Naïve Bayes model.* | **A graph of a curve  Description automatically generated**  *Fig. 10 ROC Curve for CNN model.* |

|  |  |
| --- | --- |
| **Precision-Recall Curves**  Precision-recall curves provide another view of the trade-off between precision and recall. | |
| **A graph of a line  Description automatically generated with medium confidence**  *Fig. 11 Precision-Recall Curve for the Naïve Bayes model.* | **A graph of a graph showing different types of curves  Description automatically generated with medium confidence**  *Fig. 12 Precision-Recall Curve for the CNN model.* |

## **Discussion and Conclusion**

**Interpretation of Results**

The findings from my experiments offer significant insights into the performance of the Naïve Bayes and CNN models concerning the classification task.

**Accuracy an F1 Score:**

The CNN model consistently surpasses the Naïve Bayes model in both accuracy and F1 score. The CNN achieved superior accuracy and F1 Score, indicating its effectiveness in correctly classifying the test set. Additionally, the precision and recall scores for the CNN model were higher, suggesting enhanced capability to identify true positives while minimising false positives and negatives.

**Confusion Matrix:**

The confusion matrices for both models demonstrated that the CNN model had fewer misclassifications than the Naïve Bayes model. This indicates that the CNN model better understands the classification boundaries among different classes.

**ROC and Precision-Recall Curves:**

The ROC curves for the CNN model exhibited higher area under the curve (AUC) values for each class, reflecting improved performance in differentiating between classes. Furthermore, the precision-recall curves indicated that the CNN model maintained elevated precision and recall across various threshold values, which is vital for precision and recall.

**Was the Outcome Expected?**

Yes, the outcome was anticipated. Considering the classification task's complexity and the data's characteristics, it is generally expected that deep learning models like CNNs will outperform traditional machine learning models such as Naïve Bayes, mainly when dealing with high-dimensional data like images or intricate features.

**Key Learnings**

* **Effectiveness of CNN:** CNN demonstrated superior performance to Naïve Bayes due to their capability to learn hierarchical features and capture complex patterns within the data. This reinforces the notion that CNN is particularly well-suited for tasks that involve intricate feature spaces.
* **Model Limitations:** While Naïve Bayes is computationally efficient and straightforward, it struggled with the dataset’s complexity. This outcome aligns with the model’s feature independence assumption, which often does not hold in real-world scenarios.
* **Evaluation Metrics:** Employing multiple evaluation metrics (accuracy, F1 score, precision, recall, ROC curves) offered a comprehensive assessment of model performance. It underscores the strengths of CNN in addressing multi-class classification challenges.

**What Went Well**

**Model Training and Evaluation:**

Both models were effectively trained and evaluated. The code for generating evaluation metrics and visualisations functioned as intended, yielding clear and informative representations of model performance.

**Clear Visualisation:**

The visualisations for accuracy, F1 score, precision-recall, and ROC curves successfully illustrated the performance differences between the models.

**What Went Poorly**

**Naïve Bayes Performance:**

The Naïve Bayes model underperformed relative to expectations, likely due to its simplistic design and feature independence assumption, which may not have been applied to this dataset.

**Complexity of Tuning:**

Tuning deep learning models like CNN demands substantial computational resources and experimentation, making the process time-consuming.

**Recommendations for Improvement**

**Data Augmentation:**

In future experiments, augmenting the training data could enhance the performance of both models, particularly the CNN, by increasing the variety of training examples.

**Hyperparameter Optimisation:**

Further tuning of hyperparameters, especially for the CNN model, may yield improved performance. Techniques such as random search could be utilised for this purpose.

**Model Ensembling:**

Leveraging the strengths of multiple models through ensembling techniques could boost overall performance and robustness.

**Advanced techniques:**

Investigating advanced CNN architectures or employing transfer learning with pre-trained models may produce better results.

**Conclusion**

The results indicate that the CNN model significantly outperforms the Naïve Bayes model across all key metrics. This conclusion is supported by the consistently high accuracy, F1 score, precision, and recall achieved by the CNN model and its superior performance reflected in the ROC and precision-recall curves. The CNN's capability to learn complex features and patterns from the data is crucial to its enhanced performance.

In summary, while the Naïve Bayes model is more straightforward and faster, the CNN model offers a more accurate and robust solution for this classification task. These findings highlight the effectiveness of deep learning techniques in addressing complex classification problems and suggest that investing in CNN or similar advanced models is beneficial for achieving superior performance.

## **Acknowledgements**

**CNN Model Codes**

**Original Code Snippet:**

****

*Adapted from* (Deepak, 2023)

**Modified code Snippet:**

****

**Original Code Snippet:**

****

*Adapted from (Soni, 2021)*

**Modified Code Snippet:**

****

## **User Guide**

This guide provides instructions for using the models from the classifiers developed in this project. It also describes replicating the steps to build and test the models.

**System Requirements**

* **Operating System:** This assignment has been developed and tested on Manjaro Linux. Compatibility with operating systems such as Windows or macOS is possible but may require adjustments.
* **Python Version:** 3.12.4
* **Visual Studio Code Version:** 1.91.1
* **Dependencies:** Refer to **requirements.txt** for the list of required Python packages.

**Setup the Environment**

1. **Create A Python Virtual Environment:**

Open your terminal and navigate to the **ICT203-Assignment-2-Keith-Neo-35107628** folder. Create a virtual environment by running:

**“python -m venv assignment2”**

1. **Activate the Virtual Environment:**

On Linux and macOS:

**“source assignment2/bin/activate”**

On Windows:

“**.\assignment2\Scripts\activate”**

1. **Install Required Packages:**

With the virtual environment activated, install the dependencies by running:

**“pip install -r requirements.txt”**

**Running the Program**

The program is run using the command line interface. Below are examples of how to execute different commands:

* **Train Naïve Bayes Classifier:**

**“python code/main.py -c nb -d digitdata -m train”**

This command trains the Naïve Bayes classifier on the digit dataset.

* **Train Alternative Model (CNN):**

**“python code/main.py -c alt -d digitdata -m train”**

This command trains the alternative model. Users will be prompted to choose between predefined best parameters or a dynamic search for the best parameters. **Note that the dynamic option may take longer, depending on system capabilities.**

A screen shot of a computer

Description automatically generated

*Fig. 13 User prompt training type.*

* **Compare Models:**

**“python code/main.py --compare -d digitdata”**

This command compares the performance of the Naïve Bayes and CNN models and generates graphs comparing the models and graphs for each model.

A screen shot of a computer code

Description automatically generated

*Fig. 14 All graphs.*

**Important Notes**

**Running on Windows:** The program has known issues when running on Windows. Specifically, Windows systems require **numpy<2.0.0,** while Linux and macOS can use **numpy 2.0.0**. Ensure that the **numpy** version is appropriately installed for your operating system.

## **References**

Alex Krizhevsky, I. S. (2017). ImageNet Classification with Deep Concolutional Neural Networks. *Communications of the ACM*, 84-90.

Deepak, D. (1 March, 2023). *LeNet-5 Implementation on MNIST in PyTorch*. Retrieved from Medium: https://medium.com/@deepeshdeepakdd2/lenet-5-implementation-on-mnist-in-pytorch-c6f2ee306e37

mnist. (1 June, 2024). *Datasets*. (Tensoflow) Retrieved 19 July, 2024, from TensorFlow: https://www.tensorflow.org/datasets/catalog/mnist

Soni, P. (9 January, 2021). *CNN Model with PyTorch For Image Classification*. Retrieved from Medium: https://medium.com/thecyphy/train-cnn-model-with-pytorch-21dafb918f48

Yann LeCun, L. B. (1998). GradientBased Learning Applied to Document. *Proceedings of the IEEE*, 2278-2324.