## Introduction

The Triple-MNIST dataset represents an extension of the well-documented MNIST dataset, a benchmark in machine learning. Unlike the traditional MNIST dataset, which contains 28x28 pixel images of single handwritten digits, the Triple-MNIST dataset consists of larger images sized at 84x84 pixels, each containing three handwritten digits. The task is to predict the sequence of these three digits, transitioning the problem from a simple single-label classification task to a multi-label classification problem.

This multi-label classification significantly increases the complexity of the task, as the number of possible output combinations expands from 10 to 1,000. Solving this problem requires advanced pre-processing techniques, appropriate model architectures, and a robust evaluation framework. This report aims to explore a range of machine learning approaches to address this challenge, from basic methods to sophisticated architectures, including the integration of Generative Adversarial Networks (GANs) for data augmentation. Through this, the aim is to provide a comprehensive analysis of model performance and identify the most effective solution.

## Task 1: Exploratory Data Analysis

The exploratory analysis of the Triple-MNIST dataset serves as the foundation for understanding its unique properties and identifying challenges inherent to this classification problem. This task involves loading the dataset, visualising sample images, and examining how it deviates from traditional classification tasks.

The dataset is organised into training, validation, and testing directories, with images labelled according to the three-digit sequence they contain. Each image is a grayscale representation with dimensions of 84x84 pixels, considerably larger than the 28x28 images in the original MNIST dataset. The increase in image size corresponds to the inclusion of three digits within a single frame, necessitating models capable of handling higher-resolution inputs and multi-output predictions.

Random samples from the dataset were visualised to gain an intuitive understanding of the data. Examples of such images revealed the distinct positioning of digits within the frame and the variations in handwriting styles, stroke thickness, and digit alignment. The visualisation confirmed the dataset’s complexity, with digits sometimes overlapping or closely spaced, posing challenges for segmentation and classification.

A detailed examination of the dataset highlighted several challenges that differentiate it from traditional single-digit classification. The transition from 10 single-digit labels in MNIST to 1,000 three-digit combinations in Triple-MNIST necessitates a multi-label classification approach. This expanded label space significantly increases computational demands and model complexity. Additionally, the larger image size requires models with enhanced capacity to capture spatial relationships and extract meaningful features. Moreover, the variability in handwriting styles, overlapping digits, and inconsistent digit alignment introduces ambiguity, challenging even advanced models. These insights provide crucial guidance for designing and implementing effective machine learning models.

## Task 2: Baseline Models

To establish a baseline for performance on the Triple-MNIST dataset, two fundamental models were implemented: Logistic Regression and a Basic Convolutional Neural Network (CNN). These models provided initial insights into the problem's complexity and served as a reference for evaluating more advanced approaches.

The images were pre-processed differently for each model to align with their specific requirements. For Logistic Regression, the images were flattened into one-dimensional vectors to match the input format expected by traditional machine learning models. This process, implemented in Visualise and Regression.py, involved converting each 84x84 pixel image into a 7,056-dimensional vector. In contrast, for the Basic CNN, images were normalised to the range [0, 1] and reshaped into 84x84x1 tensors to retain their spatial structure. This pre-processing, detailed in Basic CNN.py, aimed to leverage the spatial feature extraction capabilities of CNNs.

Using the LogisticRegression class from scikit-learn, the Logistic Regression model was trained on the flattened image data. Despite its simplicity, this model served as a useful baseline for comparison, though it was inherently limited in capturing the spatial relationships within the images. On the other hand, the Basic CNN, defined in Basic CNN.py, featured a straightforward architecture consisting of three convolutional layers with ReLU activation, followed by max-pooling layers, a flattening layer, and two fully connected layers. The final dense layer employed a softmax activation function to output probabilities for each of the 1,000 possible three-digit labels.

Both models were evaluated using metrics such as accuracy, F1 score, and classification reports. The Logistic Regression model achieved a weighted F1 score of 0.00, reflecting its inability to effectively address the multi-label nature of the problem. The classification report further highlighted its poor precision and recall across all classes. Similarly, the Basic CNN demonstrated only modest improvement, also yielding a weighted F1 score of 0.00. Its inability to effectively segment and classify the digits within the images was a major limitation.

These results emphasised the limitations of these baseline models. For Logistic Regression, the flattening of image data resulted in a loss of spatial relationships, a critical factor for recognising digits within the images. While the Basic CNN preserved the spatial structure, its simplicity rendered it insufficient to handle overlapping digits and variations in handwriting styles. These findings underscore the need for more advanced architectures capable of effectively segmenting and classifying the digits within the Triple-MNIST images.

## Task 3: Advanced Model Development

Building upon the limitations identified in the baseline models, an advanced model was developed to address the unique challenges posed by the Triple-MNIST dataset. The Split Convolutional Neural Network (CNN) approach was implemented, where each image was divided into three distinct parts, and separate CNN models were trained to predict each digit independently. This approach aimed to simplify the classification task by focusing on one digit at a time, effectively reducing the complexity of the problem.

The pre-processing pipeline for the Split CNN involved segmenting each image into three equal-sized vertical slices. Each slice corresponded to one of the three digits in the sequence. This segmentation ensured that each CNN model processed only the relevant portion of the image, reducing noise and ambiguity. The segmented images were normalised to the range [0, 1] and reshaped into tensors with dimensions of 28x84x1, retaining their spatial structure for feature extraction. This process was implemented in split CNN.py.

The Split CNN approach utilised three identical CNN models, each trained to predict a single digit. The architecture of each CNN comprised two convolutional layers with ReLU activation, designed to extract local features, followed by max-pooling layers to downsample the feature maps, reducing dimensionality and computational complexity. The models also included a fully connected layer with 128 neurons, followed by a softmax layer to output probabilities for the ten possible digits (0-9). The outputs of the three CNN models were concatenated to form the final three-digit prediction.

The models were trained on the segmented training dataset, with hyperparameter tuning conducted using the validation set. The evaluation was performed on the test set, using metrics such as accuracy, F1 score, and classification reports. The Split CNN achieved a significant improvement over the baseline models, with an overall weighted F1 score of 0.97. The classification reports indicated high precision and recall across all three-digit positions, demonstrating the effectiveness of this approach.

The superior performance of the Split CNN can be attributed to its ability to isolate and focus on individual digits, thereby mitigating the challenges posed by overlapping and inconsistent handwriting styles. By simplifying the task into three independent predictions, the model effectively reduced the complexity of the label space. However, the approach required careful segmentation of the images, introducing an additional pre-processing step. Despite this, the significant improvement in performance justified the additional computational effort.

## Task 4: Model Performance Enhancement

Following the promising results of the Split CNN model, further enhancements were applied to address potential issues of underfitting and overfitting and to boost model performance. Two primary techniques were explored: incorporating dropout layers for regularisation and modifying the architecture to increase model capacity. These enhancements aimed to refine the model’s ability to generalise while maintaining efficiency.

The first enhancement involved adding dropout layers to the existing architecture. Dropout layers were introduced after the fully connected layers in each of the three CNN models. This regularisation technique helped reduce overfitting by randomly deactivating a fraction of neurons during training, thus encouraging the network to develop robust and generalisable feature representations. The implementation was straightforward and leveraged Keras’ built-in dropout functionality.

The second enhancement focused on increasing the model’s capacity to better capture complex patterns in the data. Additional convolutional layers were integrated into the architecture, each followed by batch normalisation to stabilise and accelerate training. This adjustment allowed the model to extract more intricate features, particularly beneficial for images with overlapping or ambiguous digits.

These improvements were evaluated using the same metrics as before: accuracy, F1 score, and classification reports. The enhanced Split CNN demonstrated a noticeable increase in performance, achieving a weighted F1 score of 0.98 on the test set. The regularisation introduced by dropout layers mitigated overfitting, as evidenced by the reduced gap between training and validation accuracy. Moreover, the additional convolutional layers improved the model’s precision and recall, particularly for digit positions with higher variability.

Overall, these enhancements not only bolstered the performance of the Split CNN but also highlighted the importance of regularisation and architectural modifications in developing robust machine learning models. While these adjustments increased computational requirements, the gains in accuracy and generalisability justified their implementation.

## Task 5: Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) were employed to further enhance the performance of the model by generating synthetic images. This approach addressed the challenges posed by limited training data by diversifying the dataset and improving the model’s ability to generalise to unseen cases. The implementation of GANs adhered to a standard architecture and training methodology, leveraging their capability to create realistic data augmentations.

The architecture of the GAN consisted of two primary components: the generator and the discriminator. The generator was tasked with producing realistic synthetic images, while the discriminator aimed to distinguish between real and synthetic images. Both components were implemented using TensorFlow and Keras, as outlined in DCGAN.py. The generator began with a dense layer that reshaped a noise vector, sampled from a standard normal distribution, into a low-resolution feature map. Through a series of transposed convolutional layers with batch normalisation and Leaky ReLU activation functions, the generator progressively upscaled the feature map, creating images with the target dimensions of 84x84 pixels. Conversely, the discriminator employed convolutional layers to extract features from input images, followed by fully connected layers to classify them as real or synthetic. Dropout layers were included to prevent overfitting, enhancing the generalisability of the model.

The training process for the GAN adhered to the adversarial framework, where the generator and discriminator were trained in competition. The generator sought to produce increasingly realistic images, while the discriminator aimed to accurately distinguish between real and synthetic data. This adversarial dynamic drove both components to improve iteratively. The GAN was optimised using the Adam optimiser, with separate loss functions for the generator and discriminator guiding their respective updates.

Once the GAN was trained, it was used to generate 10,000 synthetic images. A visual inspection of these samples confirmed their quality and diversity, as the generated images exhibited varying handwriting styles, stroke thicknesses, and alignments, closely resembling the real data. This visual validation demonstrated the effectiveness of the GAN in creating plausible augmentations.

To evaluate the impact of these synthetic images, they were combined with the original training dataset, forming an augmented dataset. The previously best-performing model, the Split CNN, was retrained on this augmented dataset. This retraining aimed to leverage the additional diversity provided by the synthetic images to enhance the model’s robustness and generalisability.

The retrained Split CNN was evaluated on the test set, with results compared to its performance prior to augmentation. Incorporating synthetic data led to a modest but notable improvement in performance metrics. The weighted F1 score increased from 0.98 to 0.985, and the classification reports revealed enhanced precision and recall, particularly for digit positions that were underrepresented in the original training data. These findings highlighted the value of synthetic images in mitigating class imbalances and addressing challenging cases such as overlapping digits or unconventional handwriting styles.

In conclusion, the use of GANs to generate synthetic images provided a significant advantage in augmenting the dataset and improving model performance. While the gains were incremental, they underscored the importance of addressing data limitations through innovative techniques. However, the computational demands of training GANs and the need for careful tuning of hyperparameters must be considered in practical applications. This task demonstrated the potential of GANs as a powerful tool in machine learning workflows, particularly for datasets constrained by size or diversity.

## References

* Python Software Foundation (2023) *Python 3.12.0 Documentation*. Available at: <https://docs.python.org/3.12/> (Accessed: 5 January 2025).
* scikit-learn developers (2023) *scikit-learn: Machine Learning in Python*. Available at: https://scikit-learn.org/stable/ (Accessed: 5 January 2025).
* NumPy developers (2023) *NumPy Documentation*. Available at: https://numpy.org/doc/stable/ (Accessed: 5 January 2025).
* Pillow developers (2023) *Pillow (PIL Fork) Documentation*. Available at: <https://pillow.readthedocs.io/en/stable/> (Accessed: 5 January 2025).
* Matplotlib developers (2023) *Matplotlib: Visualization with Python*. Available at: https://matplotlib.org/stable/contents.html (Accessed: 5 January 2025).
* TensorFlow developers (2023) *TensorFlow Documentation*. Available at: https://www.tensorflow.org/guide (Accessed: 5 January 2025).
* Triple-MNIST Dataset. Provided as part of the course materials.