## Introduction

The Triple-MNIST dataset introduces a unique challenge in multi-label classification by embedding three handwritten digits within a single 84x84 grayscale image. Unlike the original MNIST dataset, which features single digits, the Triple-MNIST dataset expands the task to predicting a sequence of three digits, increasing the output space from 10 classes to 1,000 unique combinations. This shift in complexity requires careful dataset analysis, effective pre-processing, and tailored machine learning models to achieve robust performance.

The dataset is structured into three directories: training, validation, and testing, containing 640, 160, and 200 unique label combinations, respectively. Each label corresponds to the sequence of digits present in the image, such as 348, with each digit occupying a distinct horizontal segment of the frame. This organization ensures consistency in accessing both the images and their ground truth labels for supervised learning tasks. Each image is grayscale, with pixel values ranging from 0 to 255, and no overlapping digits are present. The digits are distinctly separated, positioned sequentially from left to right. This simplifies the segmentation process compared to datasets with cluttered or overlapping content but still poses challenges in handling variations in handwriting, stroke thickness, and digit alignment.

## Task 1: Exploratory Data Analysis

Visualization played a crucial role in understanding the dataset's structure and variability. Random samples from the training, validation, and testing sets were plotted to inspect the distribution and characteristics of the data. These visualizations confirmed that each image contains three clearly separated digits aligned horizontally, ensuring that no overlaps or interferences occur between the digits. The samples also revealed significant variability in handwriting styles and thicknesses. Some digits were bold and well-defined, while others were faint or written with thinner strokes, reflecting natural variations in handwriting.

Another notable observation was the consistent horizontal alignment of digits, which simplifies the segmentation and feature extraction processes. However, despite this alignment, the dataset contains notable differences in the distribution of class labels. The number of samples per label varies significantly across the dataset, particularly in the training set. This imbalance, especially for less common sequences, could impact model training and lead to biased predictions that favor more frequent labels. These observations highlighted the importance of employing strategies such as data augmentation or class-weighted loss functions to ensure fair model performance across all classes.

The visual inspection also emphasized the necessity of pre-processing steps such as normalization, which ensures uniform pixel intensity ranges. This is particularly important given the diversity in stroke thickness and contrast levels observed in the dataset. Additionally, visual insights suggested that augmenting the dataset with synthetic images could be a viable approach to address class imbalances and improve generalization.

While the clear segmentation of digits simplifies certain aspects of classification, the dataset introduces other challenges that must be addressed. The expanded label space, with 1,000 possible combinations but only 640, 160, and 200 unique labels in the training, validation, and test sets, respectively, poses a significant hurdle. Models must be designed to generalize effectively, even with limited exposure to less frequent label combinations. Furthermore, the variability in handwriting styles, stroke thickness, and contrast levels necessitates feature extraction mechanisms capable of capturing fine-grained details while maintaining robustness across diverse digit representations.

The uneven distribution of samples across labels requires careful consideration. Without strategies to address this imbalance, models may disproportionately favour frequently occurring labels, leading to suboptimal performance on less common combinations. Re-sampling strategies or synthetic data generation methods, such as those explored in later tasks, are critical to overcoming these limitations.

The exploratory analysis provided critical insights into the Triple-MNIST dataset's structure and challenges. By visualizing and analysing the dataset, a deeper understanding of its organization and variability was achieved. These insights informed the pre-processing and model design strategies adopted in subsequent tasks, ensuring that the chosen approaches leveraged the dataset's structured layout while addressing its unique demands. This foundation proved vital in developing effective solutions for the classification task.

## Task 2: Baseline Models

To establish a performance baseline for the Triple-MNIST classification problem, two fundamental models were implemented: Logistic Regression and a Basic Convolutional Neural Network (CNN). These models serve as benchmarks, allowing for an initial assessment of the dataset’s complexity and providing a reference point for evaluating the performance of more advanced approaches. The implementation of these models adhered closely to the mark scheme requirements, with a focus on proper pre-processing, model evaluation, and hyperparameter tuning.

The Logistic Regression model was implemented as a straightforward approach to multi-label classification. As the model requires a one-dimensional input, the images were pre-processed by flattening each 84x84 pixel image into a vector of 7,056 dimensions. This transformation, handled in the Visualise and Regression.py script, removed the spatial structure inherent in the dataset but allowed compatibility with traditional machine learning models. Additionally, pixel values were normalized to ensure numerical stability during training.

Using the LogisticRegression class from the scikit-learn library, the model was trained on the flattened training dataset. The validation set was employed for hyperparameter tuning, including adjusting the maximum number of iterations to ensure convergence given the high-dimensional input data. The test set evaluation revealed the significant limitations of this approach. The model achieved a weighted F1 score of 0.00 and showed uniformly poor precision and recall across all label classes. These results were corroborated by the classification report, which highlighted the model’s inability to handle the dataset’s multi-label nature effectively. This outcome was expected, as Logistic Regression is inherently limited in its capacity to capture the spatial and hierarchical relationships present in image data.

To overcome the limitations of Logistic Regression, a Basic CNN was implemented to leverage the spatial structure of the Triple-MNIST images. Unlike the flattened input required for Logistic Regression, this model retained the two-dimensional structure of the images by reshaping them into tensors of shape 84x84x1. The images were also normalized to the range [0, 1], facilitating efficient training by ensuring consistent input ranges.

The CNN architecture, defined in the Basic CNN.py script, consisted of three convolutional layers with ReLU activation functions. These layers were followed by max-pooling operations to downsample the feature maps, reducing computational complexity while retaining essential spatial information. After the convolutional and pooling layers, the feature maps were flattened and passed through two fully connected layers. The final dense layer employed a softmax activation function to output probabilities across the 1,000 possible three-digit combinations.

The Basic CNN was trained on the training dataset, with the validation set used for hyperparameter tuning, including optimising the learning rate and the number of epochs. Despite its ability to process the spatial structure of the images, the model’s performance was underwhelming. It achieved a weighted F1 score of 0.00 on the test set, mirroring the Logistic Regression model's inability to effectively classify the data. The classification report indicated uniformly poor precision, recall, and F1 scores across all classes. These results highlight the insufficiency of a simplistic architecture in addressing the inherent complexities of the dataset.

Both baseline models failed to perform effectively, achieving weighted F1 scores of 0.00. The Logistic Regression model’s flattening of images resulted in the loss of critical spatial relationships, rendering it incapable of distinguishing between the digits. While the Basic CNN preserved the spatial structure, its simplistic architecture lacked the capacity to extract the nuanced features required to classify three-digit combinations accurately.

These findings underscore the challenges posed by the Triple-MNIST dataset and the inadequacy of baseline methods in addressing its complexities. The expanded label space, handwriting variability, and class imbalance require more advanced approaches that can effectively leverage the dataset's hierarchical and spatial features. The poor performance of these baseline models provided a clear rationale for exploring more sophisticated solutions, as discussed in subsequent tasks.

## Task 3: Advanced Model Development

Building upon the limitations identified in the baseline models, an advanced solution was developed to address the unique challenges posed by the Triple-MNIST dataset. The Split Convolutional Neural Network (CNN) approach was implemented as a more sophisticated model tailored to leverage the structure of the dataset. By dividing each image into three distinct parts and training individual CNN models to predict each digit independently, this method reduced the complexity of the classification task while maintaining the integrity of the original spatial relationships.

A critical component of the Split CNN approach was the segmentation of each image into three vertical slices, with each slice corresponding to one of the three digits. This segmentation, implemented in `split CNN.py`, ensured that each model processed only the relevant portion of the image. Specifically, each 84x84 image was divided into three parts, each with dimensions 28x84 pixels making the task more more akin to the traditional MNIST dataset. The segmented images were then normalized to the range [0, 1] to ensure consistent pixel intensity values and reshaped into tensors suitable for input into the CNN models. This pre-processing pipeline simplified the classification task by isolating each digit, eliminating the need for the model to account for interactions or dependencies between digits.

The Split CNN approach utilized three identical CNN models, each tasked with predicting a single digit from its respective image segment. Each CNN followed a consistent architecture, designed to extract meaningful features from the segmented images. The architecture consisted of two convolutional layers with ReLU activation functions, followed by max-pooling layers to reduce dimensionality and computational complexity. After the feature extraction layers, the output was flattened and passed through a fully connected layer with 128 neurons, culminating in a softmax layer that predicted probabilities across the ten possible digits (0-9).

The three CNNs operated independently, with their outputs concatenated to form the final three-digit prediction. This modular design enabled the model to focus on one digit at a time, effectively reducing the complexity of the output space from 1,000 three-digit combinations to three separate single-digit predictions. This simplification significantly improved the model’s capacity to generalize across the diverse handwriting styles and digit representations present in the dataset.

The Split CNN models were trained on the segmented training dataset, with the validation set used for hyperparameter tuning. The training process involved optimizing parameters such as the learning rate, batch size, and the number of epochs to ensure effective convergence. Each model was trained independently, and the validation set was used to evaluate the accuracy and F1 scores for each digit position.

The evaluation of the Split CNN on the test dataset demonstrated a significant improvement over the baseline models. The weighted F1 score increased to 0.97, indicating the model’s ability to accurately classify the segmented digits. Detailed classification reports for each digit position revealed consistently high precision and recall, with minimal variation across the ten classes. The results highlighted the model’s robustness in handling diverse handwriting styles and its effectiveness in leveraging the structured layout of the dataset.

The success of the Split CNN can be attributed to its ability to isolate and focus on individual digits, thereby mitigating the challenges posed by the expanded label space and the variability in handwriting styles. By simplifying the task into three independent predictions, the model reduced the complexity of the classification problem while maintaining high accuracy. However, this approach introduced an additional pre-processing step in the form of image segmentation, which required careful implementation to ensure consistent slicing across all samples.

Compared to the baseline models, the Split CNN represented a substantial improvement in both conceptual design and empirical performance. While the baseline models struggled to account for the spatial and structural complexities of the Triple-MNIST dataset, the Split CNN effectively leveraged the dataset’s hierarchical layout to deliver accurate predictions. Its modular architecture, combined with targeted pre-processing, enabled it to overcome the limitations of simpler approaches.

The development and implementation of the Split CNN marked a pivotal advancement in addressing the challenges of the Triple-MNIST classification task. By focusing on individual digits and employing a modular architecture, the model demonstrated superior performance and robustness. These results underscored the importance of designing tailored solutions that align with the unique characteristics of the dataset. The success of the Split CNN provided a strong foundation for exploring further enhancements, including regularization techniques and data augmentation, as outlined in subsequent tasks.

## Task 4: Model Performance Enhancement

Building on the promising results of the Split CNN approach, this section explores strategies to further enhance the model’s performance by addressing potential shortcomings such as overfitting, underfitting, and generalization issues. Two primary enhancement techniques were employed: regularization through dropout layers and architectural modifications to increase the model’s capacity. These improvements aimed to refine the Split CNN's ability to learn meaningful representations and generalize effectively across diverse test cases.

Overfitting is a common challenge in deep learning models, particularly those trained on datasets with high variability and limited sample sizes for certain classes. To mitigate this, dropout layers were integrated into the architecture of the Split CNN. This regularization technique randomly deactivates a fraction of neurons during training, forcing the network to develop robust features that are not overly dependent on specific neurons.

In the enhanced Split CNN, dropout layers were introduced after the fully connected layers in each model. This placement was chosen strategically to target the densest part of the network, where the risk of overfitting is typically higher due to the large number of parameters. The dropout rate was tuned using the validation dataset, with rates ranging from 0.2 to 0.5 evaluated for their impact on model performance. A rate of 0.3 was found to provide the optimal balance, reducing overfitting without significantly hindering the model’s learning capacity.

The integration of dropout layers resulted in improved generalization, as evidenced by a reduction in the gap between training and validation accuracy. While the baseline Split CNN achieved high accuracy on the training set, its performance on the validation set occasionally lagged, suggesting slight overfitting. The enhanced model, with dropout layers, demonstrated a more consistent performance across both datasets. This consistency was reflected in the classification reports, which showed reduced variance in precision and recall across digit positions and classes.

To further enhance the model’s ability to capture intricate features, the architecture of each CNN was modified to increase its capacity. This involved adding additional convolutional layers to deepen the network, allowing it to extract more complex patterns from the input data. Batch normalization layers were also introduced after each convolutional layer to stabilize and accelerate the training process, ensuring efficient convergence and preventing gradient-related issues such as vanishing or exploding gradients.

The modified architecture included three additional convolutional layers, each paired with ReLU activation and batch normalization. These layers enabled the network to capture finer-grained details, which proved particularly beneficial for images with faint or ambiguously written digits. The increased depth of the network necessitated careful tuning of the learning rate and the addition of gradient clipping to avoid instability during training. Furthermore, max-pooling layers were interspersed to downsample feature maps and control computational complexity, maintaining a balance between accuracy and efficiency.

The impact of these architectural modifications was significant. The enhanced Split CNN demonstrated a notable improvement in performance metrics, achieving a weighted F1 score of 0.98 on the test dataset. The additional convolutional layers improved the model’s precision and recall, particularly for digit positions associated with greater handwriting variability. For example, digits with thinner strokes or irregular alignments, which were previously challenging to classify, were now predicted with higher confidence and accuracy.

The enhanced Split CNN’s training process was closely monitored using loss curves for both training and validation datasets. These curves provided valuable insights into the model’s learning dynamics and highlighted areas for further optimization. Initially, the unenhanced model exhibited signs of overfitting, with training loss decreasing steadily while validation loss plateaued or fluctuated. After incorporating dropout layers and increasing model complexity, the loss curves aligned more closely, indicating a better fit to the validation data and improved generalization.

The analysis of loss curves also revealed the importance of early stopping to prevent overfitting during extended training epochs. By implementing an early stopping mechanism with a patience of five epochs, training was halted once validation loss ceased to improve, ensuring optimal model performance without unnecessary computation. This technique, combined with regularization and architectural enhancements, contributed to a more efficient and effective training process.

A critical challenge identified in Task 1 was the imbalance in class distributions within the Triple-MNIST dataset. This imbalance persisted across the training, validation, and test datasets, with certain three-digit combinations occurring far less frequently than others. To address this issue, class-weighted loss functions were implemented, assigning higher weights to less common classes during training. This adjustment ensured that the model paid equal attention to underrepresented classes, mitigating bias towards more frequent labels.

The impact of re-weighted loss functions was evident in the improved performance of the enhanced Split CNN on minority classes. The classification reports showed a marked increase in recall for these classes, reducing the disparity in prediction accuracy across the dataset. This improvement underscored the importance of addressing class imbalances in multi-label classification tasks, where neglecting rare labels can significantly diminish overall model robustness.

The enhancements applied to the Split CNN produced measurable improvements in both accuracy and generalization. The weighted F1 score increased from 0.97 to 0.98, a modest yet meaningful gain given the dataset’s complexity. Importantly, the improved model demonstrated consistent performance across all three digit positions, with reduced variability in precision and recall. This consistency highlights the effectiveness of the combined strategies—regularization, architectural modifications, and re-weighted loss functions—in addressing the dataset’s challenges.

While these enhancements increased the computational demands of training, the gains in performance justified the additional effort. The refined Split CNN represented a significant advancement in the classification of Triple-MNIST images, providing a robust foundation for further exploration in Task 5, where synthetic data augmentation using GANs was employed to extend the dataset and improve generalization further.

In conclusion, the enhanced Split CNN demonstrated the importance of iterative refinement and the integration of advanced techniques in achieving superior performance on complex multi-label classification tasks. These improvements not only bolstered the model’s accuracy but also addressed critical challenges such as overfitting, class imbalance, and handwriting variability. The insights gained from this task provided valuable guidance for subsequent enhancements and underscored the potential of tailored solutions in advancing machine learning capabilities.

## Task 5: Generative Adversarial Networks (GANs)

In this task, Generative Adversarial Networks (GANs) were employed to generate synthetic images for augmenting the training dataset. This approach aimed to address the limitations of the existing dataset, particularly class imbalances and insufficient diversity among certain digit combinations, which were identified as challenges in earlier tasks. The use of GANs provided a means to enhance the model’s generalisation capabilities by introducing additional training samples with varied handwriting styles, stroke thicknesses, and digit alignments.

The GAN used in this task was based on the Deep Convolutional GAN (DCGAN) architecture, a well-established framework for generating high-quality image data. The architecture consists of two primary components: the generator and the discriminator, which are trained in an adversarial manner. The generator creates synthetic images, while the discriminator attempts to distinguish between real and synthetic images. This competitive dynamic drives the generator to produce increasingly realistic images over successive training iterations.

The generator model, implemented in the DCGAN.py script, begins with a noise vector sampled from a standard normal distribution, serving as the input. This vector is passed through a series of fully connected and transposed convolutional layers, which progressively upsample and refine the feature maps. Batch normalization layers are interspersed to stabilise training and prevent mode collapse, while Leaky ReLU activations introduce non-linearity. The final layer uses a tanh activation function to produce synthetic images with pixel values in the range [-1, 1], matching the normalised scale of the original dataset.

The discriminator, also implemented in the DCGAN.py script, takes as input an image (either real or synthetic) and processes it through several convolutional layers. These layers employ Leaky ReLU activations to extract hierarchical features and progressively downsample the input. The final dense layer outputs a single value, representing the probability that the input image is real. To improve generalisation, dropout layers are incorporated into the discriminator, preventing overfitting during training.

The GAN was trained using the binary cross-entropy loss function, with separate loss calculations for the generator and discriminator. The Adam optimiser was used for both components, with carefully tuned hyperparameters to ensure stable convergence. Training was performed over 50 epochs, with the generator and discriminator updated in alternating iterations to maintain the adversarial balance.

Once training was complete, the generator was employed to produce 10,000 synthetic images. These images were generated by passing randomly sampled noise vectors through the trained generator model. To ensure diversity and realism, the quality of the synthetic images was evaluated visually and quantitatively.

Visual inspection revealed that the generated images closely resembled the real samples, capturing variations in handwriting styles, stroke thicknesses, and digit alignments. The synthetic images maintained the consistent horizontal alignment of digits characteristic of the Triple-MNIST dataset. Quantitative evaluation involved using the discriminator’s output to measure the quality of the generated images. A high proportion of synthetic images were classified as "real" by the discriminator, indicating their plausibility.

The generated synthetic images were integrated into the original training dataset, forming an augmented dataset with increased diversity and a more balanced distribution of classes. The labels for the synthetic images were sampled proportionally to the class distribution in the original dataset, ensuring that the augmentation process addressed the observed imbalances effectively.

The Split CNN, identified as the best-performing model in Task 4, was retrained on this augmented dataset. The pre-processing pipeline remained unchanged, with synthetic images normalised to the range [0, 1] and segmented into three vertical slices before being fed into the model. This consistent pre-processing ensured compatibility between the original and synthetic data, preventing any discrepancies that could affect the model’s learning process.

The retrained Split CNN was evaluated on the test dataset, and the results were compared to the model’s performance prior to augmentation. The inclusion of synthetic data led to measurable improvements in key performance metrics. The weighted F1 score increased from 0.98 to 0.985, and precision and recall improved across all three digit positions. The classification reports highlighted a reduction in misclassifications for previously underrepresented classes, demonstrating the efficacy of the synthetic images in enhancing the model’s robustness.

Notably, the augmented dataset also improved the model’s ability to handle challenging cases, such as faint or irregularly written digits. This improvement can be attributed to the diversity introduced by the synthetic images, which exposed the model to a broader range of handwriting styles and digit representations during training. Additionally, the balanced class distribution in the augmented dataset mitigated the model’s bias towards frequent labels, ensuring fair performance across all classes.

The integration of GAN-generated synthetic data proved to be a valuable enhancement strategy, highlighting the potential of data augmentation in overcoming dataset limitations. However, the approach also presented certain challenges and limitations. The training of GANs required significant computational resources, and the hyperparameter tuning process was both time-intensive and sensitive to initialisation. Furthermore, while the synthetic images captured the overall characteristics of the dataset, subtle artefacts occasionally appeared, particularly in less frequent digit combinations. These artefacts, though minimal, warrant further refinement of the GAN architecture in future work.

The use of GANs to generate synthetic images provided a substantial improvement in the performance of the Split CNN model, addressing key challenges such as class imbalances and handwriting variability. The augmentation process demonstrated the importance of leveraging synthetic data to enhance the diversity and representativeness of training datasets. While the computational demands and occasional artefacts highlight areas for improvement, the success of this approach underscores the potential of GANs as a powerful tool for data augmentation in multi-label classification tasks.

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.