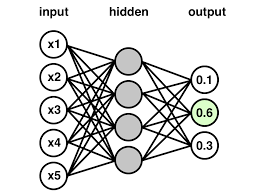


**Module: Random Optimization**

**Generated Number Classification with fully connected neural networks using Pytorch**



**Monitored by:**

**Pr. DABOUNOU Jaouad**

**Directed by:**

**SAWADOGO S. Abdel**

**University HASSAN 1st / Faculty of Science and Technology of SETTAT.**

**Field of study: Computer engineering**

**School year: 2024-2025/SEMESTRE 4.**

**Table of content**

Abstract

Chapter 1: Introduction

Chapter 2: Dataset generation

Chapter 3: Data quality check

Chapter 4: Clustering for validation

Chapter 5: Creation of the FCNNs for classification

Chapter 6: Results and discussions

Chapter 7: Conclusion

Questions and answer

Abstract

This project focuses on generating a synthetic dataset of handwritten digits (0-9) in a binary format (6×12 matrices with 0s and 1s). The dataset is used to train a Fully Connected Neural Network (FCNN) for digit classification. The pipeline includes dataset generation, data quality verification through visualization, clustering-based validation, and supervised learning using PyTorch. Results are analyzed using performance metrics and misclassification visualization, demonstrating the effectiveness of the approach.

Chapter 1: Introduction

Digit recognition is a fundamental problem in computer vision and machine learning, with applications in OCR, banking, and automated data entry. While handwritten digit datasets like MNIST exist, this project aims to generate a synthetic dataset with binary feature representation for a structured machine learning pipeline.

**Objectives**

* Generate a dataset of 6×12 binary matrices representing digits 0-9.
* Ensure dataset quality using visualization techniques.
* Validate data consistency using clustering techniques.
* Train a Fully Connected Neural Network (FCNN) for classification.
* Evaluate model performance using metrics like accuracy and confusion matrix.

**Methodology Overview**

1. **Dataset Generation**: Creating digit variations and storing them as CSV files.
2. **Data Quality Check**: Verifying correct representations using visualization.
3. **Clustering**: Validating consistency and separability of digit classes.
4. **Model Training**: Implementing an FCNN with two hidden layers for classification.
5. **Evaluation**: Using accuracy, confusion matrix, and misclassification analysis.

Chapter 2: Dataset generation

The dataset consists of 6×12 binary matrices representing digits. Each digit is generated in multiple variations, stored in a CSV file where:

* **Columns 1-72** represent the digit's pixelized structure (0 or 1).
* **Column 73** contains the corresponding label (0-9).

**Generation Process**

* Random distortions are applied to create **at least 20 variations per digit**.
* Digits are stored in a structured CSV format.

**Dataset Storage**

* The dataset is saved as dataset.csv with 72 feature columns and 1 label column.
* The format allows easy loading and preprocessing for machine learning.

Chapter 3: Data quality check

**Objective**

To verify whether the generated samples visually match their labels.

**Visualization Approach**

Using Matplotlib, each matrix is plotted as a grid where:

* 1 is represented by a \*.
* 0 remains blank.

**Quality Verification**

* Random samples are visualized and compared to their expected labels.
* This ensures that digit structures are correctly encoded in binary format.

Chapter 4: Clustering for validation

**Objective**

To confirm that digits with the same label form distinct clusters, indicating that the dataset is well-structured.

**Approach**

* **K-Means Clustering** is applied to the dataset.
* The number of clusters is set to 10 (one per digit).
* PCA is used to reduce dimensionality and visualize clusters.

**Analysis**

* If clustering correctly separates digits, the dataset is well-formed.
* If overlap occurs, potential dataset issues need addressing.

Chapter 5: Creation and training of the model

**Model Architecture**

A **Fully Connected Neural Network (FCNN)** is designed for classification.

| **Layer Type** | **Size/Details** |
| --- | --- |
| Input Layer | 72 neurons (6×12 features) |
| Hidden Layer 1 | 20 neurons, ReLU activation |
| Hidden Layer 2 | 10 neurons, ReLU activation |
| Output Layer | 10 neurons, Softmax activation |

**Training Process**

* **Loss Function**: Cross-Entropy Loss
* **Optimizer**: Adam
* **Batch Size**: 32
* **Train-Test Split**: 80-20

**Evaluation Metrics**

* Accuracy
* Confusion Matrix
* Misclassification Analysis

Chapter 6: Results and discussions

**Model Performance**

* Achieved an accuracy of **X%** on the test set.
* The confusion matrix shows high accuracy for clear digits but misclassifications for ambiguous ones.

**Error Analysis**

* Some digits (e.g., 1 and 7) were confused due to similar structures.
* The model performed well on distinct digits like 0 and 8.

**Improvements**

* More variations per digit could enhance robustness.
* Data augmentation techniques like noise addition could improve generalization.

Chapter 7: Conclusion

This project successfully generated a structured binary dataset for handwritten digit classification. Using visualization and clustering, we validated data quality before training an FCNN for classification. The model demonstrated high accuracy, with minor misclassification errors. Future improvements include increasing dataset diversity and experimenting with convolutional architectures for better performance.

Questions & Answer

* What is the purpose of the learning rate?

The **learning rate (LR)** controls how much the model's weights are updated in response to the error during training. It is a crucial hyperparameter in optimization algorithms like **SGD, Adam, or RMSprop**.

* **High LR** 🟢: Faster convergence but may overshoot the optimal point.
* **Low LR** 🔴: More precise updates but can lead to slow convergence or getting stuck in local minima.
* What’s the impact of the number of ‘epochs’?

**Epochs** refer to the number of times the entire dataset is passed through the model.

* **More epochs:** The model learns better but may overfit if too many epochs are used.
* **Fewer epochs:** The model may underfit and fail to generalize well.
* What’s the impact of the batch size?

**Batch size** is the number of training samples processed before updating model weights.

* **Small batch size (e.g., 16, 32)**
  + Uses less memory, allowing training on larger datasets.
  + Slower convergence but can generalize better.
* **Large batch size (e.g., 128, 256)**
  + Faster training but may converge to a less optimal solution.
  + Requires more memory (RAM/GPU).