Handwriting recognition with neural network

January 15, 2026

Introduction

Machine learning plays a central role in pattern recognition tasks, particularly in computer vision. Among these tasks, *handwritten digit recognition* is a well-known benchmark problem used to evaluate and compare different modeling approaches. In this project, we focus on the *supervised classification of handwritten digits* using the **MNIST dataset**, which consists of grayscale images representing digits from 0 to 9.

Each image in the MNIST dataset is described by a matrix of pixel intensities, resulting in a high-dimensional input space and complex relationships between the input data and the target classes. To address this problem, we employ *artificial neural networks*, and more specifically the *Multi-Layer Perceptron (MLP)*, which is well suited for modeling non-linear relationships.

The main objective of this project is to **train and compare three types of models**: a *linear model*, a *shallow MLP*, and a *deep MLP*. This comparison aims to analyze the impact of network depth on model performance. The models are evaluated using several criteria, including the learned *weights*, the evolution of the *training and validation loss*, and the *classification accuracy*.

A detailed graphical analysis is conducted to provide insights into the advantages and limitations of deeper architectures compared to simpler models in the context of MNIST digit classification.

1. Theoretical and Mathematical Background

1.1 Linear Model for Classification

The linear model represents the simplest form of a neural network and serves as a baseline for comparison. In the context of classification, each image is represented as a vector

$$\mathbf{x} \in \mathbb{R}^d,$$

The linear model computes the output as

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b},$$

where $\mathbf{W} \in \mathbb{R}^{K \times d}$ is the weight matrix, $\mathbf{b} \in \mathbb{R}^K$ is the bias vector, and $K = 10$ is the number of classes.

To obtain class probabilities, the Softmax function is applied:

$$\hat{y}_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}.$$

This model is equivalent to a multiclass logistic regression and is limited to learning linear decision boundaries in the feature space.

1.2 Multi-Layer Perceptron (MLP)

1.2.1 General Architecture

A Multi-Layer Perceptron (MLP) is a fully connected neural network composed of an input layer, one or more hidden layers, and an output layer. Each layer performs an affine transformation followed by a non-linear activation function.

For layer l, the forward propagation is given by:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)},$$
$$\mathbf{a}^{(l)} = f\left(\mathbf{z}^{(l)}\right),$$

where $\mathbf{a}^{(l-1)}$ denotes the activations from the previous layer and $f(\cdot)$ is the activation function.

Shallow MLP

A shallow MLP consists of a single hidden layer. Its formulation is:

$$\mathbf{a}^{(1)} = f\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right),$$
$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)}.$$

The output logits are then transformed using the Softmax function to obtain class probabilities. Compared to the linear model, the shallow MLP can learn non-linear decision boundaries due to the presence of the hidden layer and non-linear activation function.

Deep MLP

A deep MLP is composed of multiple hidden layers stacked together:

$$\mathbf{a}^{(l)} = f\left(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right), \quad l = 1, \ldots, L.$$

Increasing the depth allows the network to learn hierarchical representations of the data. Early layers tend to capture simple patterns such as edges or strokes, while deeper layers represent more abstract structures corresponding to digit shapes.

However, deeper architectures may lead to challenges such as overfitting, optimization difficulties, and increased computational cost.

1.3 Activation Functions

Activation functions introduce non-linearity, which is essential for learning complex mappings.

ReLU

$$f(x) = \max(0, x).$$

ReLU is widely used due to its simplicity and effectiveness in deep networks.

Softmax: The Softmax function is used in the output layer for multiclass classification:

$$\hat{y}_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}.$$

Hyperbolic tangent (**Tanh**) is also used to introduce non-linearity.

1.4 Loss Function

For MNIST classification, the categorical cross-entropy loss is employed:

$$\mathcal{L} = -\sum_{k=1}^{K} y_k \log(\hat{y}_k),$$

where y_k is the true label encoded as a one-hot vector and $\hat{y}_k$ is the predicted probability.

1.4 Training and Backpropagation

The learning process consists of minimizing the loss function over the training dataset. The model parameters

$$\theta = \{\mathbf{W}, \mathbf{b}\}$$

are updated using an optimizer like $AdamW$:

$$g_t = \nabla_\theta \mathcal{L}(\theta_{t-1}) \quad \text{(gradient of the loss with respect to the parameters)}$$
$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad \text{(first moment estimate, i.e., momentum)}$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \quad \text{(second moment estimate, i.e., gradient variance)}$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad \text{(bias-corrected estimates)}$$
$$\theta_t = \theta_{t-1} - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \theta_{t-1} \right) \quad \text{(parameter update with weight decay)}$$

Parameters explanation:

- θ: model parameters (weights and biases)
- g_t: gradient of the loss at iteration t
- m_t: first moment estimate (moving average of gradients)
- v_t: second moment estimate (moving average of squared gradients)
- $\hat{m}_t, \hat{v}_t$: bias-corrected first and second moment estimates
- β_1, β_2: exponential decay rates for the moment estimates (commonly 0.9 and 0.999)
- η: learning rate
- ϵ: small constant to avoid division by zero (e.g., 10^{-8})
- λ: weight decay coefficient (L2 regularization)

Gradients are efficiently computed using the backpropagation algorithm, which applies the chain rule from the output layer back to the input layer.

1.5 Relation to the Project Objectives

This theoretical framework enables a comparative study between a linear model, a shallow MLP, and a deep MLP. By analyzing the learned weights, validation loss, and accuracy, the effect of network depth on learning dynamics and generalization performance can be systematically evaluated.

1.6 Dataset Description

This project uses the **MNIST dataset**, a standard benchmark for handwritten digit classification. The dataset consists of **70,000 grayscale images** of handwritten digits ranging from **0 to 9**, with **60,000 images for training** and **10,000 images for testing**.

Each image has a resolution of 28×28 pixels and is represented as a vector in $\mathbb{R}^{784}$. Pixel intensities originally range from 0 to 255 and are **normalized to the range** $[0, 1]$ to improve numerical stability during training.

The labels correspond to the digit classes and are encoded using **one-hot vectors** for classification with a softmax output layer. The training set is further split into **training and validation subsets** in order to monitor generalization performance and detect overfitting.

The MNIST dataset is well suited for this project as it enables a direct comparison between a **linear model**, a **shallow MLP**, and a **deep MLP**, allowing the analysis of the effect of network depth on learning behavior and classification performance.

1.7 Model Architectures

In order to analyze the impact of network depth on learning behavior and performance, three different architectures are considered in this project: a **linear model**, a **shallow Multi-Layer Perceptron (MLP)**, and a **deep MLP**. All models take the same input representation and produce class probabilities over the ten MNIST digits.

1.7.1 Linear Model

The linear model serves as a baseline for comparison. Each input image is represented as a vector

$$\mathbf{x} \in \mathbb{R}^{784},$$

which is directly mapped to the output layer through a single affine transformation:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b},$$

where $\mathbf{W} \in \mathbb{R}^{10\times784}$ and $\mathbf{b} \in \mathbb{R}^{10}$.

A softmax activation function is applied to obtain class probabilities. This model does not include any hidden layers and therefore learns only linear decision boundaries in the input space.

1.7.2 Shallow MLP

The shallow MLP extends the linear model by introducing a single hidden layer, enabling the network to model non-linear relationships.

The architecture is defined as:

$$\mathbf{a}^{(1)} = f\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right),$$

$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)},$$

where $f(\cdot)$ denotes a non-linear activation function, such as ReLU.

The output layer uses a softmax activation to produce class probabilities. Compared to the linear model, the shallow MLP has a higher representational capacity due to the presence of the hidden layer.

1.7.3 Deep MLP

The deep MLP consists of multiple hidden layers stacked sequentially, allowing the network to learn hierarchical feature representations.

For a network with L hidden layers, the forward propagation is defined as:

$$\mathbf{a}^{(l)} = f\left(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right), \quad l = 1, \ldots, L.$$

The final layer applies a softmax activation function to produce the output probabilities. Increasing the depth of the network enhances its expressive power but also introduces challenges such as overfitting and increased computational cost.

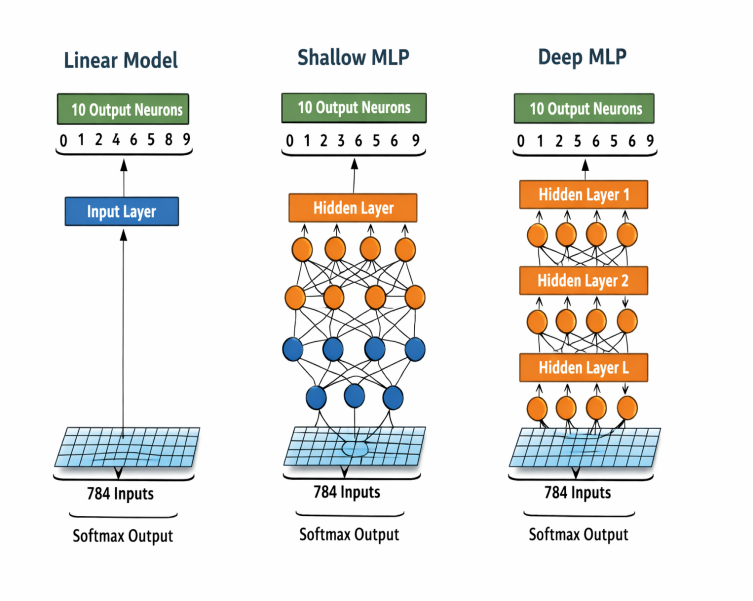

Figure 1: Architectures of Linear, ShallowMLP and DeepMLP models

1.7.4 Comparison Objective

All three architectures are trained under comparable conditions in order to fairly assess the effect of depth. The comparison focuses on:

- the evolution of training and validation loss,
- classification accuracy,
- the structure and magnitude of the learned weights.

This experimental setup enables a systematic study of how network depth influences optimization behavior and generalization performance on the MNIST dataset.

1.8 Training Procedure

This section describes the training protocol used for all models in order to ensure a fair and consistent comparison between the three models.

1.8.1 Data Splitting

The MNIST training set is divided into two subsets:

- a **training set**, used to update the model parameters,
- a **validation set**, used to monitor generalization performance and detect overfitting.

All models are trained using the **categorical cross-entropy loss**, which is well suited for multi-class classification problems.

1.8.2 Optimization

Model parameters are optimized using gradient-based methods. In this project, the considered adaptive optimizer is **AdamW**.

1.8.3 Training Configuration

All models are trained for a fixed number of epochs using mini-batch learning. The same training configuration is applied to all architectures in order to isolate the effect of network depth.

During training, the following quantities are monitored:

- training loss,
- validation loss,
- classification accuracy.

These metrics are recorded at each epoch and later used for graphical analysis.

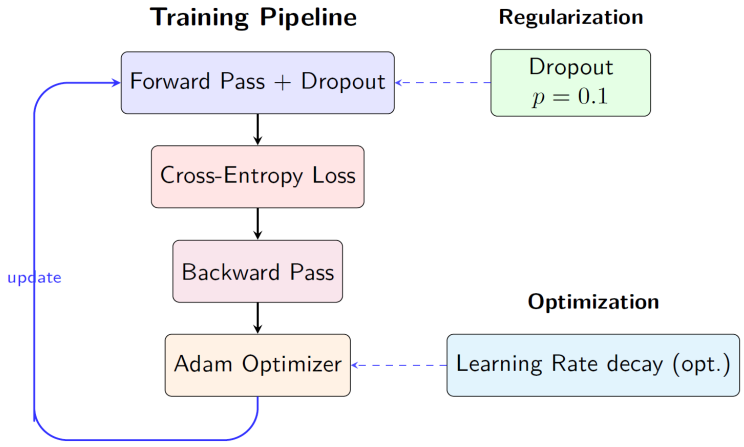

Figure 2: Training procedure

1.8.4 Model Selection

The validation set is used to assess the generalization capability of the models and to identify potential overfitting. The final comparison between architectures is based on the validation performance as well as on the analysis of the learned weights.

1.9 Experimental Results and Analysis

This section presents and analyzes the experimental results obtained for the linear model, the shallow MLP, and the deep MLP on the MNIST dataset.

1.9.3 Model Comparison: *test loss vs. epochs for each model*

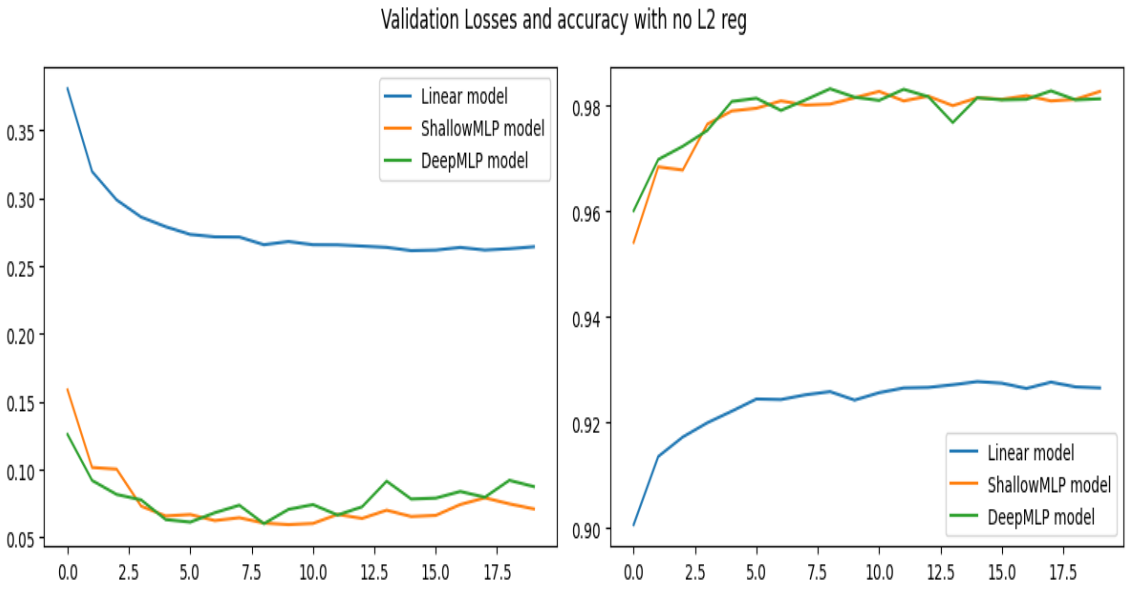

Figure 3: test loss accuracy vs. epochs for each model

The linear model exhibits a relatively high validation loss throughout training. Although the loss decreases during the first epochs, it quickly reaches a plateau around a higher value compared to the MLP-based models. This behavior highlights the limited representational capacity of linear models when applied to complex tasks such as handwritten digit classification.

In contrast, both the shallow MLP and the deep MLP achieve significantly lower validation losses. The introduction of hidden layers allows the models to learn non-linear decision boundaries, leading to a substantial improvement in performance. The shallow MLP shows a rapid decrease in loss during the early epochs and converges to a low and stable value.

The deep MLP reaches a comparable, and in some epochs slightly lower, validation loss than the shallow MLP. However, its loss curve exhibits slightly higher variability, which may indicate increased sensitivity to optimization and a higher risk of overfitting due to the increased depth, especially in the absence of L2 regularization.

Summary: These results show that:

- Adding hidden layers dramatically improves performance compared to a linear model
- A single hidden layer is sufficient to achieve strong performance on MNIST.
- Increasing network depth provides only marginal gains in accuracy, while potentially increasing training instability when no regularization is applied.

Overall, while deeper architectures offer higher representational capacity, their benefits on MNIST are limited without appropriate regularization, and a shallow MLP already provides an excellent trade-off between performance and model complexity.

1.9.4 Effect of L2 Regularization: *Analysis of Learned Weights*

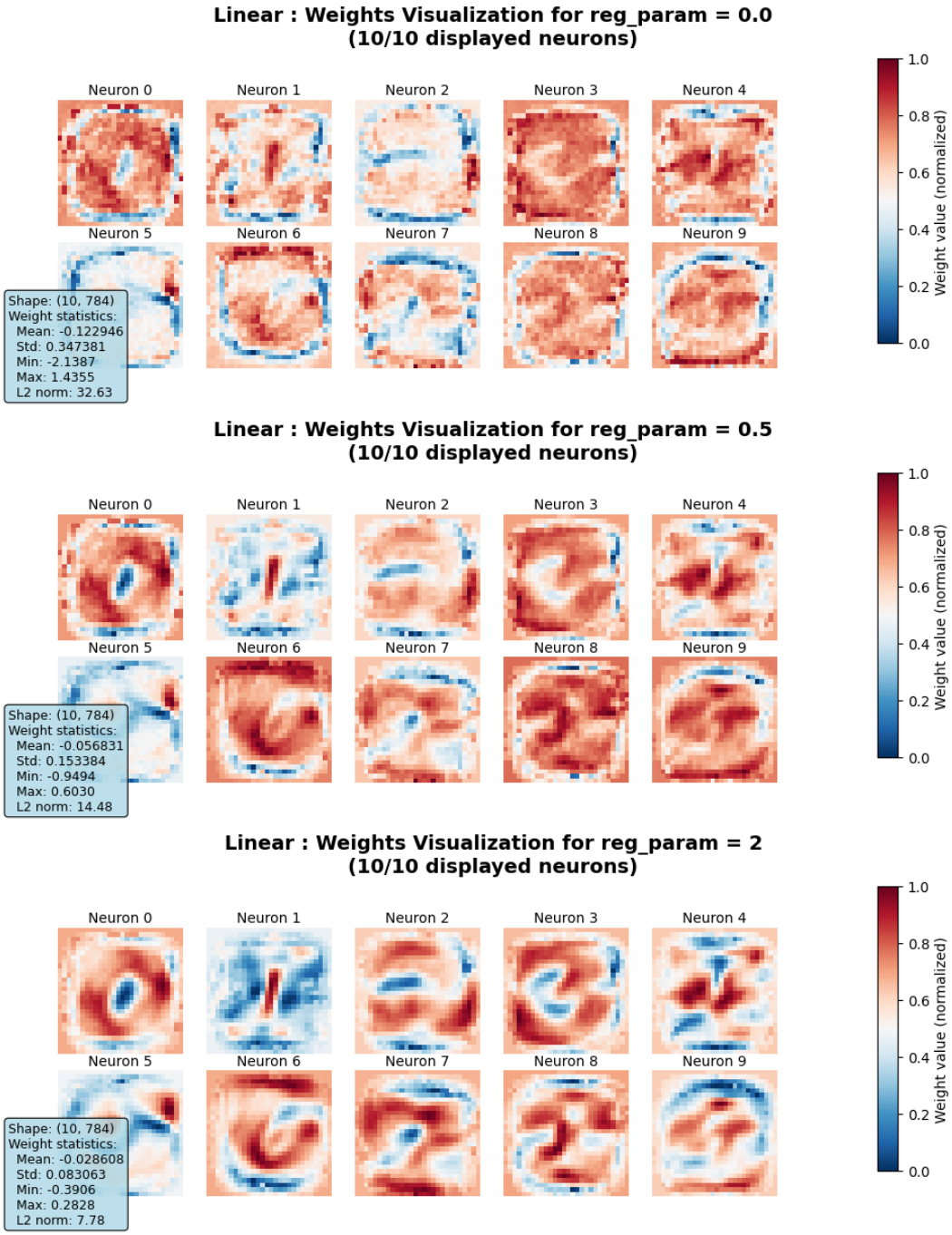

Figure 4: Plot of the 3 linear model's weights

No Regularization ($\lambda = 0.0$)**:** when no L2 regularization is applied, the learned weight maps exhibit high-amplitude and noisy patterns. The weights strongly emphasize specific pixel regions, resulting in sharp contrasts and irregular structures. This behavior indicates that the model heavily adapts to the training data, potentially capturing noise and spurious correlations. The large L2 norm of the weight matrix reflects the reliance on large weight magnitudes, which increases the risk of overfitting.

Low Regularization ($\lambda = 0.5$)**:** with a low regularization strength, the learned weights become smoother and more structured. The dominant digit shapes remain clearly identi-

fiable, while extreme weight values are attenuated. This level of regularization provides a balance between data fitting and model simplicity, encouraging a more robust distribution of importance across pixels and improving generalization.

High Regularization ($\lambda = 2.0$)**:** for higher regularization values, the weight maps appear significantly smoother and less contrasted. The digit patterns become more diffuse due to the strong penalty on large weights, as reflected by the reduced L2 norm. While this improves stability and reduces overfitting, excessive regularization may suppress discriminative features, potentially leading to underfitting.

Summarize: these visualizations highlight the role of L2 regularization as an effective complexity control mechanism. Increasing the regularization parameter reduces weight magnitudes and produces smoother weight patterns. Moderate regularization yields a favorable trade-off between interpretability, robustness, and classification performance, whereas overly strong regularization may limit the expressive capacity of the model.

1.9.5 Evaluation and Confusion Matrix: *best-performing MLP and confusion matrix.*

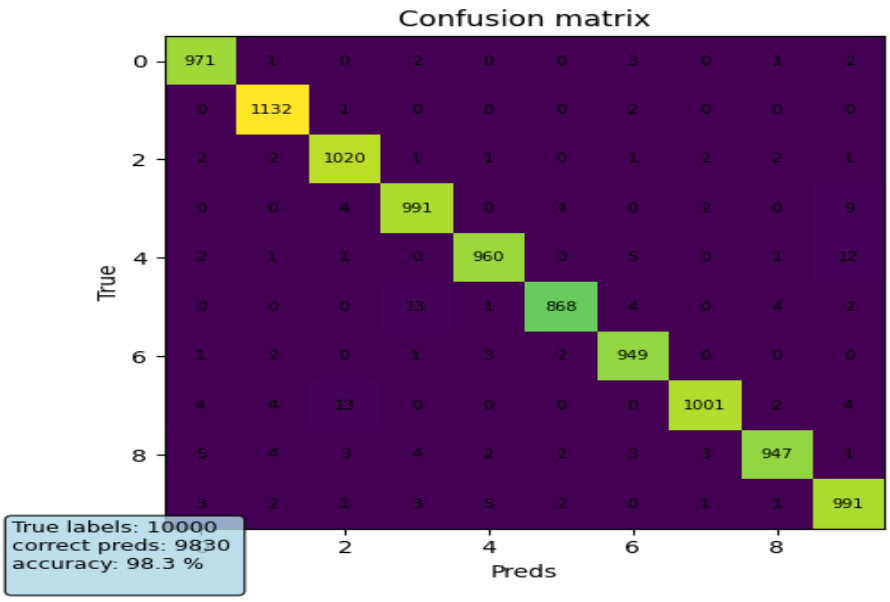

Figure 5: Confusion Matrix (ShallowMLP)

0.0.1 Evaluation and Confusion Matrix

The strong diagonal dominance indicates that the model correctly classifies the majority of samples across all digit classes.

Digits such as 0, 1, and 7 are particularly well recognized, with very few misclassifications, suggesting that their visual representations are distinctive and consistently captured by the model.

Most errors occur between visually similar digits. In particular, confusions are observed between digits 3 and 5, as well as between 4 and 9, which can be explained by similarities in handwritten shapes. The digit 8 is sometimes misclassified as 0, 3, 5, or

9, likely due to its complex structure. Additional confusions between digits 2 and 7 or 2 and 8 are also observed, which are common in MNIST classification tasks.

Overall, the confusion matrix indicates that misclassifications are primarily due to inherent ambiguities in handwritten digits rather than poor model generalization.

1.9.6 Regularization and Overfitting: *the chosen MLP with different L2 strengths.*

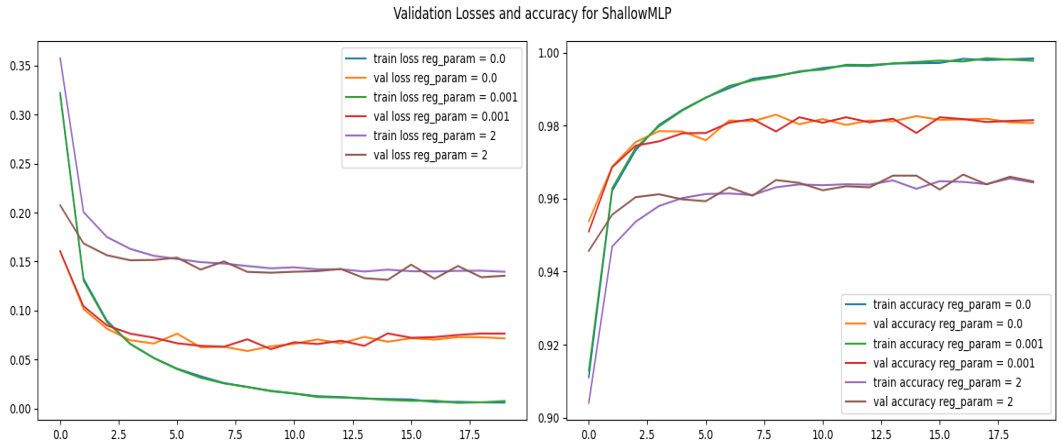

Figure 6: Regularization and Overfitting (ShallowMLP)

No Regularization ($\lambda = 0.0$) Without regularization, the training loss decreases rapidly and reaches very low values, while the validation loss stops improving and slightly increases after several epochs. The growing gap between training and validation curves indicates overfitting. This behavior is further confirmed by the training accuracy continuing to increase while the validation accuracy plateaus.

Low Regularization ($\lambda = 0.001$) With a small L2 regularization value, the training and validation losses remain close throughout training, and the validation accuracy reaches its highest and most stable values. This setting effectively reduces overfitting while preserving strong predictive performance.

High Regularization ($\lambda = 2$) For higher regularization strengths, the training loss decreases more slowly and remains relatively high. The validation accuracy stabilizes at a lower level, indicating that the model is overly constrained and suffers from underfitting.

Optimal Regularization Overall, a low L2 regularization value ($\lambda \approx 0.001$) provides the best trade-off between model complexity and generalization performance for the shallow MLP.

1.9.7 Model's per class accuracy

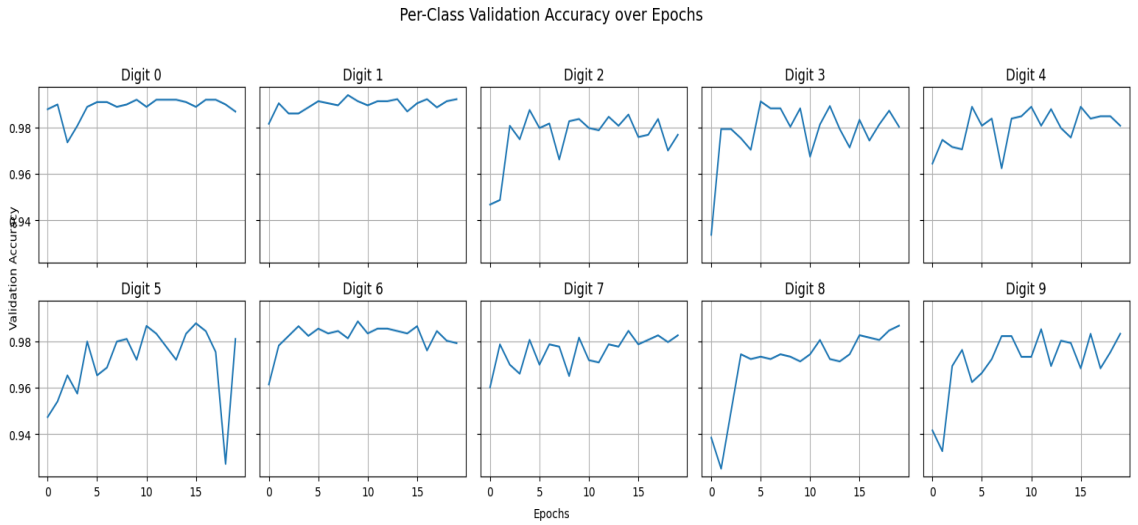

Figure 7: Model's per class accuracy (Shallow with $\lambda = 0.001$)

Digits with simple, distinctive shapes and low ambiguity (0,1,6 and 7) converge faster, while visually complex and similar digits require more training to reach stable performance.

Conclusion

In this project, we studied the behavior of neural network models of increasing complexity on the MNIST handwritten digit classification task. A linear model, a shallow Multi-Layer Perceptron (MLP), and a deep MLP were trained and compared in order to analyze the impact of network depth, regularization, and optimization on learning dynamics and generalization performance.

The experimental results show that introducing hidden layers significantly improves performance compared to a linear model. While the linear classifier is limited to learning linear decision boundaries, both the shallow and deep MLPs are able to capture non-linear patterns in the data, leading to substantial gains in accuracy. However, the performance gap between shallow and deep architectures remains relatively small on MNIST, suggesting diminishing returns when increasing depth for this dataset.

The analysis of training and validation curves highlighted the importance of regularization. Without regularization, models tend to overfit, whereas an appropriate level of L2 regularization effectively reduces the generalization gap and stabilizes training. Excessive regularization, on the other hand, can lead to underfitting by overly constraining the model capacity.

Visualizations of learned weights further illustrated the role of regularization as a complexity control mechanism, with increasing regularization producing smoother and more stable weight patterns. In addition, the per-class accuracy analysis revealed that digits with simple and distinctive shapes converge more quickly during training, while visually complex or ambiguous digits require more training iterations to achieve stable performance.

Finally, the confusion matrix analysis confirmed that most classification errors arise from intrinsic ambiguities in handwritten digits rather than systematic model failures.

Overall, this project demonstrates the trade-offs between model complexity, regularization, and generalization, and highlights that carefully tuned shallow architectures can already achieve strong performance on MNIST.