PROJECT REPORT

Deep Learning

Alexis TREMELLAT
Mathieu PERRIOT
Francky RASATAHARISOA

January 2025

Etablissement de formation :
Université Gustave Eiffel, 12 avenue Blaise Pascal, 77420 Champs-sur-Marne

Summary

1. Introduction & Problem Understanding

The task at hand involves developing a robust deep learning model capable of accurately identifying digits from a vast dataset of tens of thousands of handwritten images. This problem is a classic example of image classification, a fundamental task in the field of computer vision. The primary objective is to create a model that can generalize well to unseen data, ensuring high accuracy in recognizing and classifying handwritten digits.

Problem Definition: The goal is to build a deep learning model that inputs images of handwritten digits and outputs the corresponding digit label (0-9). The dataset typically consists of grayscale images, each representing a single digit. This problem is crucial for various real-world applications, such as postal mail sorting, bank check processing, and digitizing handwritten forms.

Objectives:

- Develop proficiency in designing and implementing deep learning architectures, such as Convolutional Neural Networks (CNNs), Multilayer Perceptrons (MLPs), or other architectures.
- Apply advanced training techniques and optimization algorithms to improve model performance.
- Critically evaluate different model architectures and training strategies.
- Effectively communicate the engineering process, insights, and results.

Challenges:

- **Model Architecture:**
    - **Choosing the Right Deep Learning Architecture:** Selecting an appropriate neural network architecture is crucial. CNNs are typically the go-to choice for image classification tasks due to their ability to capture spatial hierarchies in images. However, determining the optimal depth, number of layers, and filter sizes requires careful experimentation.
    - **Balancing Complexity and Performance:** A model that is too complex might achieve high accuracy on the training set but fail to generalize to new data, leading to overfitting. Conversely, a model that is too simple might underfit, failing to capture the necessary features of the images.
-

- **Data Preprocessing:**
    - **Normalization and Augmentation:** Preprocessing steps such as normalizing pixel values and augmenting the dataset through transformations (e.g., rotations, translations) are essential to improve model performance and robustness.
    - **Handling Imbalanced Data:** Ensuring that the dataset is balanced across all digit classes is critical. An imbalance could lead to biased predictions towards more frequent classes.
- **Training Challenges:**
    - **Choosing the Right Loss Function and Optimizer:** The choice of loss function and optimization algorithm can significantly impact the model's learning process. Cross-entropy loss is commonly used for classification tasks, and optimizers like Adam or SGD with momentum are popular choices.
    - **Hyperparameter Tuning:** Finding the right set of hyperparameters (learning rate, batch size, number of epochs) requires extensive experimentation and fine-tuning.
- **Evaluation Metrics:**
    - **Accuracy vs. Other Metrics:** While accuracy is a primary metric, it's also important to consider precision, recall, and F1-score, especially if there is any class imbalance.
    - **Cross-Validation:** Implementing cross-validation can provide a more reliable assessment of model performance and ensure that the model generalizes well to different subsets of data.

By addressing these challenges methodically, the objective is to develop a high-performing, generalizable deep learning model for handwritten digit recognition that can be applied effectively in practical scenarios.

2. Model Architecture Design

Convolutional Neural Networks (CNN)

Convolutional Neural Networks are designed specifically for image recognition tasks. They are composed of convolutional layers that perform feature extraction, pooling layers for dimensionality reduction, and fully connected layers for classification. CNNs are highly effective for handling image data as they automatically learn hierarchical features from raw pixels, making them translation-invariant and robust when dealing with large datasets. However, CNNs can be computationally expensive and require large datasets to avoid overfitting.

Multilayer Perceptron (MLP)

Multilayer Perceptrons are primarily used for tabular data classification. They consist of fully connected layers, where input features pass through one or more hidden layers before reaching the output layer. MLPs are versatile and can model complex relationships in structured, tabular data, making them adaptable for a wide range of classification and regression tasks. The main drawback is that MLPs are prone to overfitting, especially on smaller datasets, and often require regularization techniques to prevent this, particularly for complex models.

Recurrent Neural Networks (RNN)

Recurrent Neural Networks are designed to handle sequential data, where each output is dependent on previous outputs. They are well-suited for tasks such as time series analysis and text processing, as they can capture temporal dependencies. RNNs are highly effective for sequential data, making them ideal for tasks like speech recognition and time-series forecasting. However, they are computationally intensive, especially with long sequences, and are susceptible to vanishing or exploding gradients, which can hinder learning in deep networks.

3. Training Optimization Strategies

In our effort to build an effective Convolutional Neural Network (CNN) for the Digit Recognizer competition, we employed several key optimization strategies to enhance model performance and generalization. Below are the details of the training algorithms, optimization techniques, and regularization methods used, along with the rationale behind their selection.

Gradient-Based Optimizers

We utilized the Adam optimizer, a popular choice in deep learning for its efficiency and ability to handle sparse gradients. Adam combines the advantages of two other extensions of stochastic gradient descent (SGD): Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). The default learning rate of 0.001 was used, which works well with Adam since it adjusts the learning rate for each parameter dynamically, aiding in faster convergence.

Regularization Methods

To prevent overfitting, we incorporated the method Data Augmentation, used to artificially increase the training dataset size by applying random transformations like rotations, shifts, and zooms. This exposes the model to a wider variety of data points, reducing overfitting.

Batch Normalization

Batch normalization layers were included after each convolutional layer to stabilize the learning process by normalizing the inputs of each layer. This allows the model to converge faster and perform better.

These optimization strategies slightly improved model accuracy and generalization. In conclusion, the integration of these training and optimization strategies was important in enhancing the model's performance and ensuring that it generalizes well to the test set, fulfilling the competition's objectives.

4. Model Evaluation & Validation

Validation Framework

To evaluate the performance of our Convolutional Neural Network (CNN), we split the original training dataset into a training set and a validation set. This approach allowed us to monitor the model's performance during training and adjust hyperparameters to minimize overfitting.

Training and Validation Split: The dataset was split into 80% for training and 20% for validation. This split ensures that the model is trained on a substantial portion of the data while retaining enough samples to evaluate its performance reliably.

Performance Metrics

We primarily used **accuracy** as the performance metric to assess the model's ability to correctly predict the hand-drawn digits. Accuracy is defined as the ratio of correctly predicted instances to the total number of instances.

Validation Accuracy: The model's accuracy on the validation set was monitored after each epoch to track performance. This metric helps in determining the model's generalization capability to unseen data.

Impact of Validation Framework

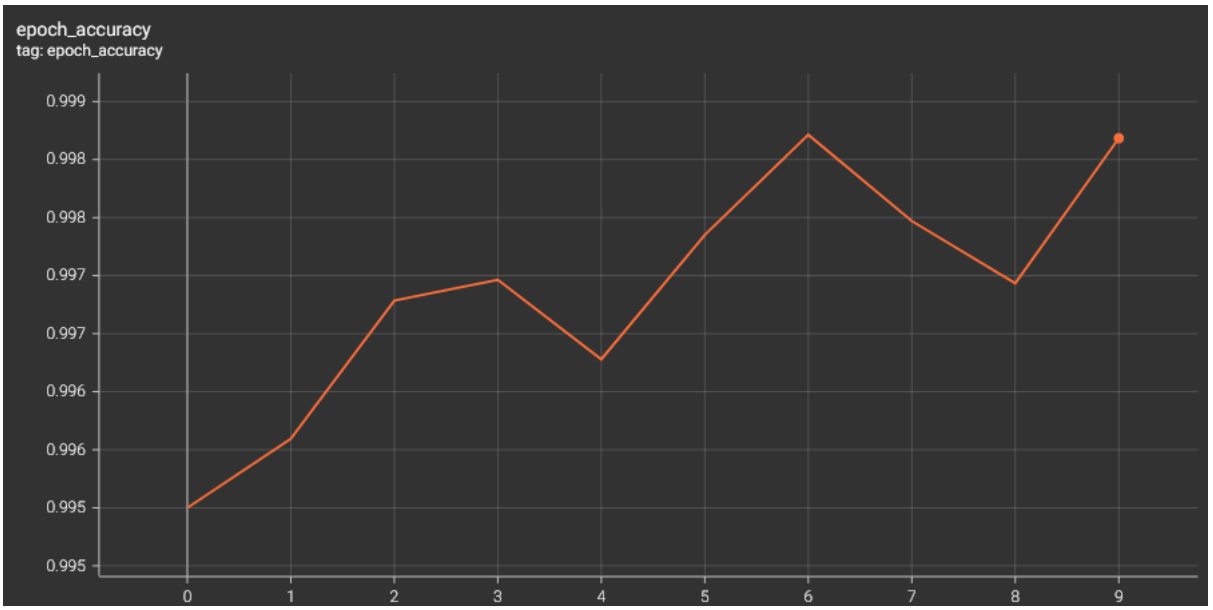
The use of a separate validation set allowed for real-time monitoring of the model's performance, helping to identify potential overfitting. Adjustments such as learning rate scheduling and dropout were informed by trends in validation accuracy, leading to a more robust final model.

5. Results & Analysis

Performance Outcomes

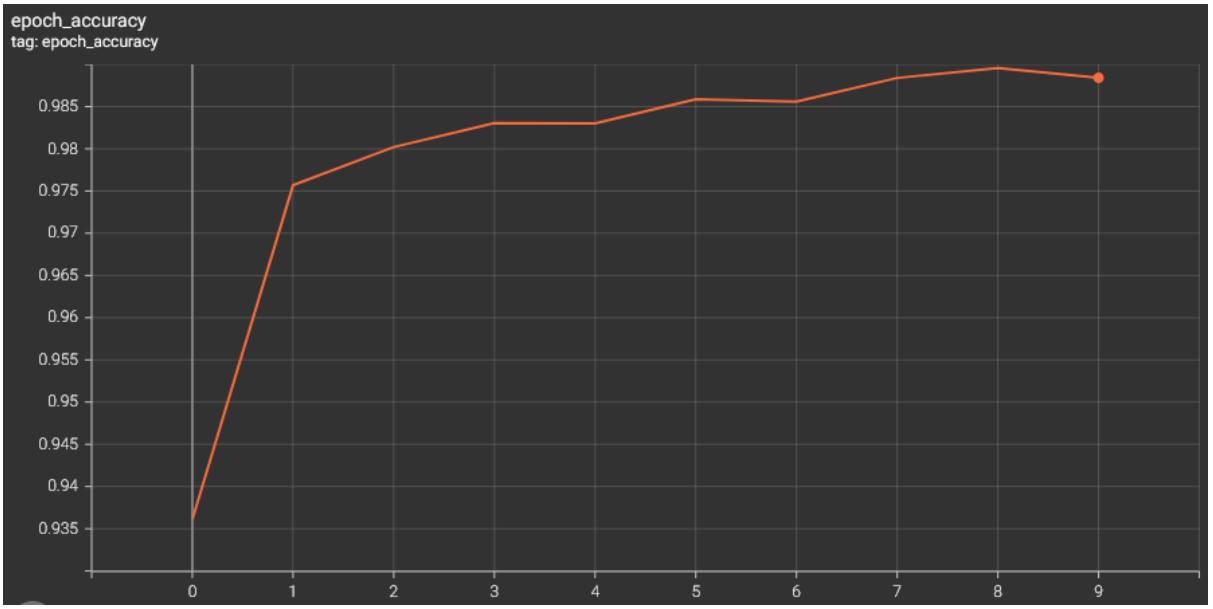
Model 1: Simple CNN

- **Training and Validation Accuracy:**
  - Final training accuracy: 99.12%
  - Final validation accuracy: 98.54%
- **Kaggle Submission Score:** 0.98585
- **Kaggle Ranking:** 709th out of 1437 participants



Model 2: Improved CNN

- **Training and Validation Accuracy:**
  - Final training accuracy: 98.84%
  - Final validation accuracy: 98.90%
- **Kaggle Submission Score:** 0.98614
- **Kaggle Ranking:** 693rd out of 1437 participants



Analysis of Results

Positive Elements:

- **Model 2's Architecture:** The improved CNN demonstrated better performance in both validation accuracy and Kaggle ranking. This can be attributed to the additional layers, regularization techniques, and data augmentation, which helped the model generalize better to unseen data.
- **Consistency in Accuracy:** Both models achieved high accuracy on the validation set, But the evolution was completely different. the complex model's accuracy evolved quite linearly, which was not the case for the simple model, which has a much chaotical evolution.
- **Fine-Tuning:** Adjustments in the learning rate and the use of a learning rate schedule likely contributed to the improved performance of Model 2 over Model 1.

Areas for Improvement:

- **Initial Overfitting in Model 2:** Despite its overall better performance, Model 2 showed signs of overfitting in the early epochs, as seen in the significant drop in validation accuracy from Epoch 1 to Epoch 2. This suggests that the model initially struggled to generalize but improved after the learning rate adjustments.
- **Kaggle Ranking:** While Model 2 outperformed Model 1 slightly on Kaggle, both models ranked in the middle of the leaderboard. Further improvements in data augmentation, hyperparameter tuning, and potentially increasing the model's capacity might be necessary to move into the top tiers.

- **Validation Loss Fluctuations:** The validation loss for Model 2 showed some fluctuations (e.g., between Epochs 7 and 10), indicating room for further stability. Techniques like more aggressive learning rate decay or early stopping could help in achieving more consistent performance.

In summary, the improved CNN model achieved slightly better performance and generalization, as evidenced by the higher validation accuracy and Kaggle score. However, there remains potential for further optimization, particularly in reducing overfitting and stabilizing validation performance.

6. Conclusion & Lessons Learned

The CNN models demonstrated the effectiveness of this architecture for handwritten digit recognition. The basic model achieved strong performance with high accuracy, while the enhanced model, incorporating techniques like data augmentation and regularization, further improved results. This project highlighted the importance of data augmentation in improving generalization and preventing overfitting, as well as the role of regularization in making the model more robust.

Despite the improvements, we faced challenges, especially with overfitting in the early stages of training for the enhanced model. While the enhanced model outperformed the basic model, there were fluctuations in validation loss, suggesting further adjustments could have stabilized performance. Additionally, while the model performed well on Kaggle, there is still room for improvement to compete at the top levels, and exploring deeper architectures or transfer learning could further enhance results.

In conclusion, this project reinforced the value of an iterative approach—testing different architectures, fine-tuning hyperparameters, and incorporating regularization techniques. For future projects, early stopping and better resource management will be key to handling larger datasets and further improving model performance.