Applied Deep Learning - Final Report

Marko Gugleta, 12016483

January 21, 2025

1 What is the problem that you tried to solve?

The primary problem addressed in this project is optical character recognition (OCR) for receipts. The goal was to not only recognize and extract text from scanned or photographed receipts but also to classify the extracted information into predefined labels such as date, total amount, merchant name, and address. This enables more structured data processing for financial tracking and business intelligence.

Many industries and consumers rely on receipt processing to manage expenses. In fields like accounting, financial auditing, and reimbursement systems, accurate and automated extraction of receipt data can lead to significant time savings. Automating this process reduces human errors and eliminates repetitive manual data entry. With the increasing digitalization of expense management, automating OCR for receipts contributes to better integration with financial tools and cloud-based management systems.

2 Why is it a problem?

Extracting data from receipts is challenging due to several reasons:

- **Format Variety**: Receipts vary widely in format, font size, and layout depending on the issuer. Different merchants use different templates, which complicates template-based extraction systems.
- **Quality Issues**: Low-quality images, faded text, and background noise introduce additional complexity and require robust preprocessing.
- **Unstructured Data**: Receipts contain information that is often not organized in tabular form. Items may be listed irregularly, and the information hierarchy can vary significantly.
- **Manual Limitations**: Manual data entry is error-prone, time-consuming, and not scalable for large volumes of receipts.

Automating this task provides significant benefits in terms of efficiency, accuracy, and cost reduction for both personal finance management and enterprise-level bookkeeping. It also facilitates better analytics and reporting, enabling users to gain insights into spending patterns and cost control.

3 What is your solution?

Implemented solution is a fine-tuned implementation of the LayoutLM model, a transformer-based architecture tailored for document understanding. LayoutLM leverages textual content, spatial layout, and visual cues to interpret document structures effectively. Unlike traditional OCR systems, it integrates visual and text-based features to build a more context-aware understanding of document layout.

Steps involved:

- **Data Preprocessing**: Preparing receipt images by transforming them to desired input for LayoutLM was an important step. Since input is in textual format images needed to be processed in multiple steps in order to assure correct input for the fine-tuning of the model.
- **Model Fine-Tuning**: Adapting LayoutLM with our custom-labeled receipt dataset to optimize for receipt-specific text extraction and classification. Fine-tuning involved training the model with labeled data that included bounding boxes for specific fields such as dates, totals, and vendor names. We used transfer learning to leverage pre-existing weights from general document models.
- **Evaluation**: Measuring performance using F1-score, precision, recall, and loss metrics to validate the effectiveness of the model.

4 Why is it a solution? (And in particular, why is or isn't deep learning a solution?)

Deep learning is particularly well-suited for this problem because:

- **Traditional Limitations**: Traditional OCR methods do not account for the spatial relationships between text elements, leading to poor accuracy on complex layouts.
- **Layout Awareness**: LayoutLM models text in combination with its positional and visual features, enabling it to understand hierarchical structures inherent to receipts.
- **Transfer Learning Benefits**: The flexibility of transfer learning allows for efficient customization with domain-specific data, reducing the need for extensive data from scratch.

Deep learning, however, requires significant computational resources and a large annotated dataset, which can pose challenges in practical deployment. It also demands proper hyperparameter tuning and significant storage for the model weights. Additionally, fine-tuning complex models like LayoutLM can be time-intensive.

5 The main take-aways and insights you gained from your project

- **Data Quality Matters**: Well-annotated and diverse training data significantly improves model performance. Consistency in annotation reduces noise in learning.
- **Model Complexity vs. Performance**: While LayoutLM offers superior accuracy, it also demands greater computational resources than simpler OCR approaches. Selecting the right balance between model complexity and real-time processing is key.
- **Metrics as a Comprehensive Indicator**: F1-score combines precision and recall, making it a better performance indicator than using accuracy alone, especially for imbalanced datasets.

6 If you would do the same project again, what - if anything - would you do differently?

- **Expand the Dataset**: Incorporate more receipts from varied sources to improve model generalization across different formats. Including multilingual receipts would also enhance usability in global applications.
- **Optimize Hyperparameters**: Conduct a more comprehensive search for optimal hyperparameters to maximize accuracy and minimize loss. Techniques such as Bayesian optimization or grid search could be employed for better results.
- **Explore Post-Processing**: Implement rule-based corrections to refine predictions and address errors in classification. For instance, using regex patterns to validate and correct dates or amounts.
- **Pipeline Integration**: Integrate the model into a complete processing pipeline with automated input handling, error logging, and user interfaces for reviewing and correcting results.
- **Deploying to Cloud or Mobile Platforms**: Consider the practical aspects of deploying the solution to cloud services or mobile apps for widespread use.

7 How much time did you spend on your project?

The project required approximately 75 hours, distributed as follows:

- First assignment: 5h
- Reformulating the approach: 5h
- Getting a new dataset and preparing the data: 10h
- Training the model: 30h
- Evaluation: 10h
- Building demo app: 10h
- Making the presentation: 2h
- Writing the report: 3h