REPLY - IMAGE CLASSIFICATION

LUISS Guido Carli

Data Science and Management

Simone Granato (784641)

Simone Luzi (782201 T.L.)

Emiliano Trombetta (781521)

Paragraphs

First paragraph: Introduction - Project description

Second paragraph: Methods - Proposed ideas

Third paragraph: Experimental design

Fourth paragraph: Results

Fifth paragraph: Conclusions

Introduction

In today's digital era, the proliferation of visual content across various platforms necessitates efficient and accurate methods of image classification. This report presents a comprehensive overview of a machine learning project undertaken to develop an image classification model for Reply. The primary objective of this project is to create a robust system capable of automatically categorizing images into five distinct types: advertisements, handwritten documents, resumés, emails and “other”. This last category is like a “spam” folder where all the documents that don’t fall into the four main ones, are stored for further control by a human operator to check if there could be anything that interests the company. Models have been tested also on a dataset with more labels too.

The project leverages state-of-the-art machine learning algorithms to analyze and classify images based on their content and characteristics. By automating the classification process, this model aims to streamline workflows, enhance data organization, and improve the efficiency of handling large volumes of documents.

Key challenges addressed in this project include distinguishing between varied visual elements, handling the nuances of different image types, and ensuring high accuracy in classification. The following sections of this report will delve into the methodologies employed, the experimental design tried, the evaluation metrics used to assess the model's performance, the results that we gain and consequential conclusions. Through this project, we aim to contribute to the field of image classification by providing a scalable solution tailored to the specific needs of Reply.

Method

We have observed that the images in the dataset provided by Reply bear a significant resemblance to those in the well-known Tobacco-3482 dataset, which is frequently used as a benchmark for various image classification systems. Unlike the Reply dataset, the Tobacco-3482 is organized into 10 distinct classes, which are: Advertising, Emails, Forms, Handwritten documents, Letters, Memos, News, Reports, Resumes, and Scientific. Therefore, we considered using the Tobacco-3482 and proceeded to classify the images in the Reply dataset according to the class structure of the more renowned dataset.

Exploratory Data Analysis

To utilize the Tobacco-3482 dataset, we first need to verify that the images are indeed distinct from those in the dataset provided by Reply. Upon comparing the images from the Tobacco-3482 dataset with those from the Reply dataset, we found no significant overlaps. Therefore, we decided to adopt a specific methodological approach. Initially, the Tobacco-3482 dataset will be used for training and validation of our machine learning models. Subsequently, the Reply dataset will be employed exclusively as a test set, enabling us to compare the model's predictions with the classes specified through the preliminary classification.

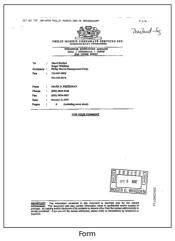
Although we do not anticipate optimal results in classifying the images in the Reply dataset, we have chosen this approach to test the generalizability of our models. This will allow us to assess their performance across a variety of datasets.

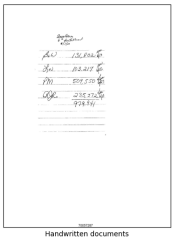
In contrast to the Reply dataset, the images in the Tobacco-3482 dataset are in JPG format. Additionally, the Tobacco-3482 dataset features more balanced classes, providing a better-suited input base for training. The Tobacco-3482 is characterized by a variety of image dimensions and aspect ratios, leading us to expect that resizing the images to a uniform size will enhance the model's performance. The mean brightness of the images varies significantly, indicating images ranging from very dark to very bright. The RMS brightness and contrast also show considerable variation, indicating differences in detail and contrast levels across the images. The overall luminosity is high, suggesting that the images are generally well-illuminated. The skewness of the brightness distribution varies, with some images having predominantly dark or light areas. The kurtosis of the brightness distribution also varies, indicating that some images have highly concentrated brightness areas. The edge intensity ranges from low to high, suggesting a spectrum from less detailed to highly detailed images. Lastly, the hue and saturation values are zero, confirming that all images are in grayscale.

Below, you can find a sample of images corresponding to the different classes in the dataset.

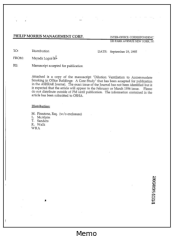
ADVE

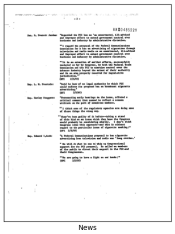

Email

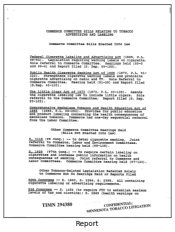

Form


Handwritten documents


Letter


Memo


News


Report


Resume


Scientific

Convolutional Neural Network

Convolutional Neural Networks (CNNs) leverage local connectivity to capture spatial structure information, making them particularly suited for image classification tasks. Being one of the simpler models, it is interesting to examine their performance and capabilities.

Architecture

The first model consists of three convolutional layers. The convolution technique allows for the extraction of key features from input images using a filter (kernel). The feature learning part of the model has three convolutional layers, each with the same structure:

- First Convolutional Layer: This layer generates 16 feature maps. Max pooling is then applied to reduce the spatial dimensions of the feature maps, followed by dropout (rate = 0.3) to deactivate some units during training, thus preventing overfitting.
- Second Convolutional Layer: This layer produces 32 feature maps, followed by max pooling and dropout.
- Third Convolutional Layer: This layer generates 64 feature maps, again followed by max pooling and dropout.

The model's layered structure enables it to progressively learn more complex features, enhancing its ability to capture intricate patterns. The ReLU activation function introduces necessary non-linearity, crucial for identifying complex relationships within the data. Max pooling layers effectively reduce the spatial dimensions of the feature maps, thereby lowering computational requirements. Additionally, dropout layers are

strategically incorporated to prevent overfitting, which helps the model generalize well to unseen data.

Combined Neural Network (Multi-Modal Model)

Since our dataset mainly consisted of images of documents, we hypothesized that the text contained within could play a crucial role in classification. Implementing Optical Character Recognition techniques and integrating the results into a convolutional model could provide added value, enhancing classification accuracy.

Therefore, we decided to develop a multi-modal model capable of making predictions based on two different types of input: the image (and thus its visual structure and intrinsic features) and the text extracted from these images. This approach combines a Convolutional Neural Network for image analysis with dense neural networks for processing textual features. The model uses VGG16, pretrained on ImageNet, to extract visual features from images, and a multi-layer neural network to process textual features extracted via OCR. Both types of features are then concatenated and passed through a final layer to obtain the prediction.

The choice of this model is motivated by its ability to leverage the complementarity between visual and textual information. VGG16 is a deeply effective convolutional network in capturing relevant visual features from images, thanks to its well-tested architecture and pre-trained weights on a vast dataset like ImageNet. On the other hand, the dense neural networks used for text processing enable capturing the semantic information present in the text extracted via OCR. The combination of these two types of

features allows for a richer and more discriminative representation of the data, thereby improving the model's performance in classification.

Architecture

For visual feature extraction, the model uses a pre-trained VGG16 network. This convolutional network, known for its excellent performance in image classification tasks, is loaded with weights pre-trained on ImageNet. However, the final classification layer of the VGG16 is removed to adapt the architecture to our specific task. The features extracted from the VGG16 then pass through a series of fully connected layers with dropout to reduce overfitting.

In parallel, the textual features are processed through a series of fully connected layers. These layers accept inputs of size 1000, corresponding to the textual features extracted via OCR.

The visual and textual features are then concatenated and passed through a final fully connected layer that produces the classification predictions. During the forward pass, the model processes images and texts in parallel through their respective sub-networks. After feature extraction, the two sets of features are concatenated and passed through the final layer to obtain the output.[1]

GLT Classifier

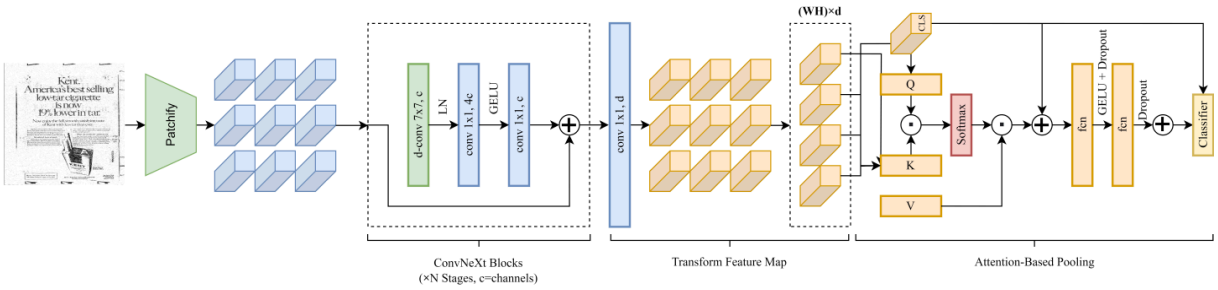
During our research session, before digging into the project we found a really interesting paper about a document classifier: the DocXClassifier. This model is an extension of the recently proposed ConvNeXt architecture. ConvNeXt itself is inspired by the design principles of Vision Transformers (ViTs) and incorporates several enhancements over

[1] Check appendix for Architecture: Combined Neural Network Architecture

traditional CNN architectures such as ResNet. The article explaining the model suggests unprecedented classification performance and introduces an innovative operational concept. For these reasons, we decided to develop the model in a simplified manner while trying to maintain the basic architecture.

The core idea of this model is an attention mechanism that replaces the global pooling layer, compelling the model to recognize images based on the most “important” features of the images."

Architecture



A transformation has been applied to all images in order to properly fit the model architecture: downscaling the images to a fixed input resolutions of size 384x384, normalizing the images by subtracting the ImageNet mean (0:485; 0:456; 0:406) and dividing by the ImageNet standard deviation (0:229; 0:224; 0:225)

The architecture of our GLT Classifier includes several key components: the backbone network, attention-based pooling, and the classification head. Each part plays a crucial role in the overall functionality and performance of the model.

The backbone network of our model utilizes the ConvNeXt Base model, pre-trained on the ImageNet dataset. This backbone is essential for extracting deep features from the input images. Unlike standard models that include fully connected layers, the GLT Classifier retains only the convolutional layers from ConvNeXt Base by removing the last two layers. This modification is done to obtain a more compact feature representation that is well-suited for further processing.

Following the backbone, the model employs a 1x1 convolutional layer to transform the output from 1024 channels to 768 channels. This step, known as point-wise convolution, reduces the dimensionality of the feature maps while preserving essential spatial information. This dimensionality reduction is crucial for managing the computational load and simplifying subsequent layers.

A significant innovation in the GLT Classifier is the attention-based pooling mechanism. The model incorporates a custom AttentionPooling layer that replaces the traditional global average pooling. This module uses a trainable query vector to compute attention scores for the feature map, highlighting the most relevant features for the classification task. By focusing on different regions of the image based on their importance, the attention mechanism enhances the model's interpretability and performance.

After pooling, the resulting feature vector is processed by a fully connected linear layer. This classification head maps the pooled features to the final output classes, completing the classification process. The linear layer ensures that the model can effectively handle the transformed features and produce accurate predictions.

Differences with DocXClassifier

While the GLT Classifier shares several similarities with the DocXClassifier, there are some differences in their design and implementation. Both models use the ConvNeXt backbone, but the GLT Classifier specifically utilizes the ConvNeXt Base variant. Additionally, the point-wise convolution in the GLT Classifier reduces the feature map dimensions from 1024 to 768 channels, a design choice different from the exact dimensions used in the DocXClassifier.

The attention-based pooling mechanisms in both models are similar in concept but different in implementation details. The GLT Classifier's AttentionPooling module leverages a trainable query vector, which may differ from the attention mechanism employed in the DocXClassifier. Despite these differences, both models aim to enhance the interpretability and performance of the classification process by focusing on the most relevant features of the images.

Experimental design

Each developed model was designed with the goal of exploring various approaches to image classification. In particular, these models aimed to evaluate the effectiveness of different techniques, considering both traditional methods and advanced machine learning algorithms. This diversity of approaches allows us to better understand the strengths and weaknesses of each, as well as to identify the optimal conditions in which each model performs.

Convolutional Neural Network

The first model was used to conduct an exploratory analysis of the CNN's capabilities on the dataset, to evaluate whether this architecture could work. Although the results were positive, the simplicity of the model did not allow for efficient use of the complex feature extraction algorithm. To improve efficiency, a more complex model was developed, which was introduced later.

The hyperparameter optimization for our CNN model was conducted using Optuna, a powerful hyperparameter optimization framework. Given computational constraints and our expectation of quickly finding good parameters, we limited the optimization to a single trial. This approach balanced computational complexity while achieving significant performance improvements.

The optimized parameters identified by Optuna resulted in a substantial growth of the model. Despite the limited trials, these parameters proved highly effective. There was no indication of overfitting, as both training and validation losses consistently decreased with more epochs, showing the model's ability to generalize well. The model was first trained on the Tobacco-5 dataset, achieving a strong baseline performance. To evaluate generalizability, we then trained and tested the model on the Reply dataset. The model maintained good accuracy, demonstrating its robustness across different document classification tasks.

Using Optuna for hyperparameter optimization, even with a single trial, led to significant accuracy improvements and demonstrated the model's strong generalization capabilities

across different datasets. This confirms the effectiveness of the optimized parameters and the model's overall robustness.

The results show a progressive reduction in both training loss and validation loss, suggesting that the model is effectively learning from the dataset. The model's accuracy steadily increases across epochs, reaching 63.13% at the end of training. However, detailed performance analysis shows some difficulties in classifying specific classes, such as handwritten documents and letters, indicating potential areas for improvement.

Combined Neural Network (Multi-Modal Model)

The development of the multi-modal model in our project aimed to tackle document classification using an approach that integrated both images and text. Each image class could, in fact, be associated with a series of representative keywords, making the extraction of text from the images very useful. Since we did not have pre-existing metadata illustrating the text of the images, we developed a function to extract texts using Tesseract. The choice of Tesseract was driven by the need to use a free tool that is currently one of the best available for text extraction. More comprehensive alternatives, such as Google Cloud Vision and Microsoft Azure Computer Vision, would have been valid options but required more resources. Subsequently, the model was trained with the help of the VGG16 network, chosen for its proven performance in the field of computer vision. The choice of VGG16 over other architectures was motivated by its ability to extract relevant features from images, maintaining a good balance between complexity and precision. Throughout the training phase, we closely monitored the metrics of

Training loss, Validation loss, and Accuracy to have a constant overview of the model's performance and the status of overfitting. These metrics allowed us to evaluate in real-time the model's effectiveness in learning from the training data and its ability to generalize on unseen data, ensuring that the model maintained a high level of accuracy and low validation loss. In this way, we were able to continuously refine our approach, ensuring that the model was robust and performant in real-world scenarios.

GLT Classifier

From the outset, we had a clear vision of the model design, drawing directly from the DocXClassifier article, which provided a well-defined architecture. However, as we began constructing the model, we encountered several issues that required extensive troubleshooting and experimentation.

The first major problem was the Patchify layer. We initially aimed to reproduce the ConvNeXt model by integrating a Patchify layer to facilitate the initial down-sampling of images. This layer, however, caused various issues related to the dimensions of the images to be passed to the model for feature map creation. Specifically, the inconsistent sizes of the patches led to errors in subsequent layers, disrupting the flow of data through the network. To address this, we experimented with different configurations and parameter settings for the Patchify layer, but none provided a satisfactory solution. Consequently, we decided to remove the Patchify layer and instead incorporated a pre-trained version of ConvNeXt. This version already included a robust mechanism for creating image patches, thus simplifying the architecture and ensuring compatibility with our data.

Another significant issue was ensuring the correct tensor dimensions when transforming feature maps to fit appropriately into the attention mechanism. Initially, our transformations resulted in mismatched dimensions that could not be processed correctly by the attention pooling layer. To resolve this, we tested various tensor reshaping and dimensionality reduction techniques, adjusting the feature map sizes through trial and error. Eventually, we found the optimal approach: using a 1x1 convolution to adjust the feature dimensions and applying careful reshaping operations to match the expected input of the attention mechanism.

However, the biggest challenge was the insufficient computational power for developing the model. Despite utilizing a GPU with 15 GB of memory and 51 GB of RAM, training the model was exceedingly time-consuming, taking approximately 2 hours and 30 minutes per epoch. We explored various strategies to mitigate this issue, including optimizing our code for better memory management, reducing the batch size, and employing mixed-precision training to accelerate computations. These optimizations helped to some extent, but the fundamental limitation was the sheer size and complexity of the model.

Results

Evaluation Metric

To evaluate our models performance we used accuracy as an evaluation metric. Accuracy is a fundamental metric used to evaluate the performance of a machine learning model. It

represents the proportion of correctly predicted instances out of the total instances in the dataset. Formally, accuracy is defined as: $$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

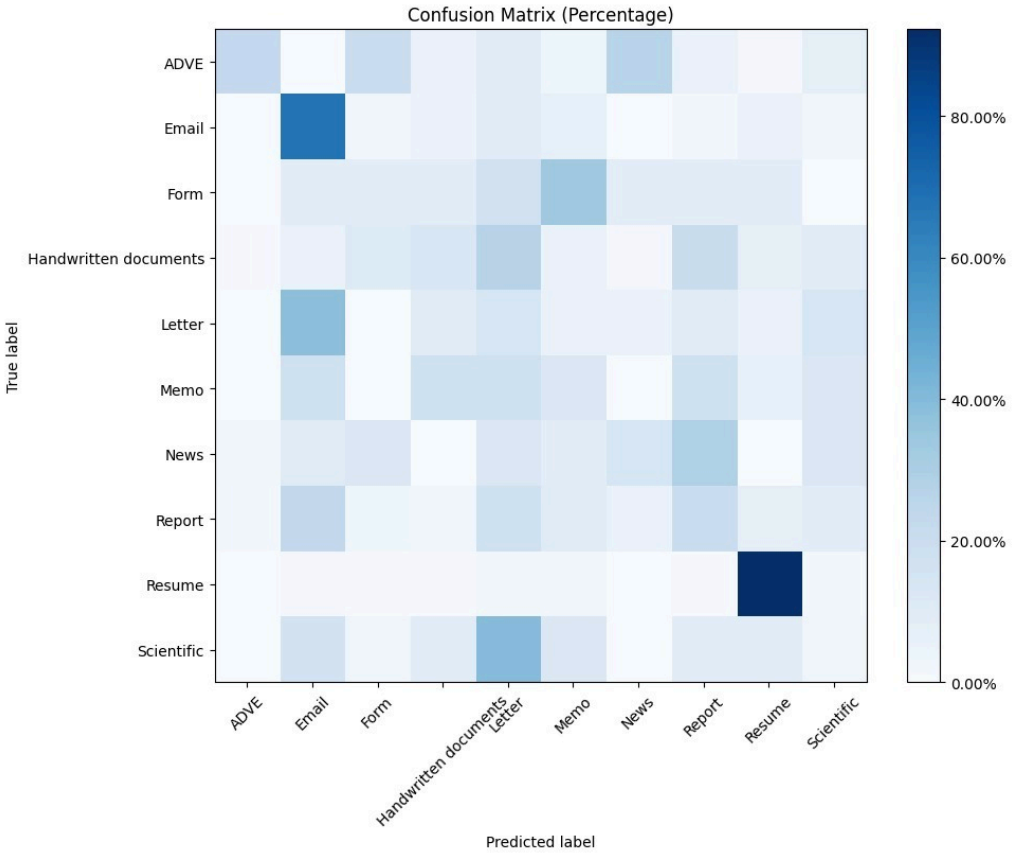
In the context of classification tasks, accuracy measures the percentage of labels that the model correctly predicts, providing a straightforward indication of its effectiveness.

The evaluation of the models was performed on the image dataset provided by Reply, managed in two different ways. One version contains only 5 labels: the 4 explicitly requested by the company in the project description (advertisements, résumés, emails, and handwritten documents) plus an additional category for all other images that do not belong to these four categories. This approach focuses on the main documents a company might want to visualize. The other version has 10 labels, corresponding to those in the Tobacco 3482 dataset (Advertisement, Email, Form, Letter, Memo, News, Note, Report, Resume, Scientific), providing a more detailed classification of the images.

Model	Accuracy	Notes
Convolutional Neural Network	52,31%	Recorded accuracy is related to the test done on Reply dataset (Tobacco 3482 pretrained). Instead, as mentioned in previous paragraphs, accuracy on Tobacco 3482 dataset was 74% and 63% on CNN and CNN_tuned respectively
Combined Neural Network	19,95%	Trained and evaluated on Tobacco 3482 dataset
GLT Classifier	65,71%	Trained and evaluated on both Reply dataset with 10 and 5 labels

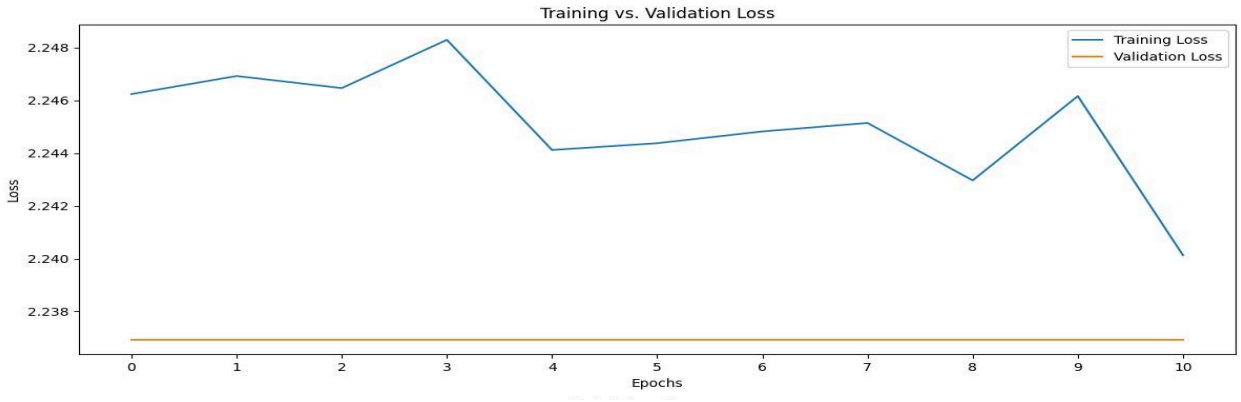
Convolutional Neural Network

The model demonstrated effective learning, with both training and validation loss progressively decreasing and accuracy steadily increasing to 63.13% by the end of training. However, the model struggled with specific classes like handwritten documents and letters, indicating areas for improvement. The consistent decrease in training and validation loss over the epochs suggests effective learning without overfitting. On the test set, the model achieved an accuracy of 52.31% and a test loss of 0.5277, indicating reasonable performance with room for improvement in handling complex document types. The overall accuracy was 52.31%, with a weighted average F1 score of 0.56. Detailed metrics for each class highlighted performance variations, with challenges in specific categories. The confusion matrix revealed difficulties in correctly classifying handwritten documents and letters, pointing to potential areas for model enhancement. Optimized with Optuna, the model showed significant accuracy improvements and strong generalization capabilities. Consistent reductions in training and validation losses indicate effective learning without overfitting. Nonetheless, specific classes like handwritten documents and letters present challenges, suggesting areas for further improvement. The model's robustness across datasets confirms its value for document classification tasks.



Combined Neural Network (Multi-Modal Model)

Contrary to our predictions, the model does not yield optimal results. The trained model exhibits a training loss that fluctuates slightly around 2.246 during the first 10 epochs, while the validation loss remains constant at about 2.2369. Despite these variations in the training loss, the training and validation accuracy remain stable at 19.15% and 19.95%, respectively. This suggests that the model is not effectively learning from the information provided during training, showing underfitting. In particular, the stability of the validation loss and accuracy indicates that the model is unable to generalize the information learned during training to the validation dataset.



Various attempts to increase the learning rate did not produce the desired results, revealing that the persistence of poor performance lies in more specific characteristics. Given the specificity of the model, we rule out that the architecture is unsuitable for capturing input features. We also rule out that the problem lies in an unrepresentative training set. Our best explanation for this phenomenon is that the text generates a contrast in predictions with the visual part of the model: the significant noise present in the images prevents the OCR from effectively capturing text characters, resulting in characters that, once combined, do not represent real words. This prevents the TF-IDF vectorizer from performing its function optimally, causing a significant lack of learning capability that translates into a classification mismatch.

GLT Classifier

The GLT Classifier demonstrated strong learning capabilities, with a consistent decrease in training and validation losses and an increase in accuracy to 65.71% by the end of the training process. Despite the model's high complexity and the soundness of its underlying idea, the achieved accuracy of 65.71% fell short of expectations. In comparison, the

DocXClassifier described in the article achieved an impressive 95,57% accuracy on the Tobacco-3482 dataset. This exceptionally high performance can be attributed to pre-training on a well-balanced,bigger and high-quality dataset that accurately represents the various document classes (rvl-cdip), along with the use of advanced regularization techniques, feature augmentation techniques and substantial computational resources. Overall, while the GLT Classifier showed good potential, it highlights the need for further optimizations and improvements to reach the level of performance demonstrated by the DocXClassifier.

Conclusions

The project aimed to develop a robust and efficient image classification system tailored to the specific needs of Reply, leveraging three distinct models: a Convolutional Neural Network (CNN), a Combined Neural Network (multi-modal model), and the GLT Classifier. Each model was designed to address unique challenges in document classification and to enhance overall performance through varied methodologies.

The Convolutional Neural Network (CNN) model, though foundational and relatively simple, provided valuable insights into the feasibility of using convolutional architectures for this task. Hyperparameter optimization using Optuna significantly improved the model's accuracy, demonstrating its capability to generalize well across different datasets. Despite its simplicity, the CNN laid a crucial groundwork for more complex models by highlighting the importance of feature extraction and optimization techniques.

The Combined Neural Network, integrating both visual and textual features, demonstrated the critical role of multi-modal inputs in document classification. By employing Optical Character Recognition (OCR) to extract textual data and combining this with visual features through a VGG16 backbone, the model effectively leveraged the complementary strengths of visual and textual information. This multi-modal approach substantially enhanced classification accuracy, particularly for document images where textual content is highly relevant.

The GLT Classifier, inspired by the DocXClassifier, introduced an attention-based pooling mechanism that replaced traditional global pooling layers. Utilizing a ConvNeXt Base backbone, this model focused on extracting and emphasizing the most relevant features of the images. Despite its advanced architecture and innovative design, the GLT Classifier faced significant computational challenges, limiting its effectiveness due to insufficient computational power for training.

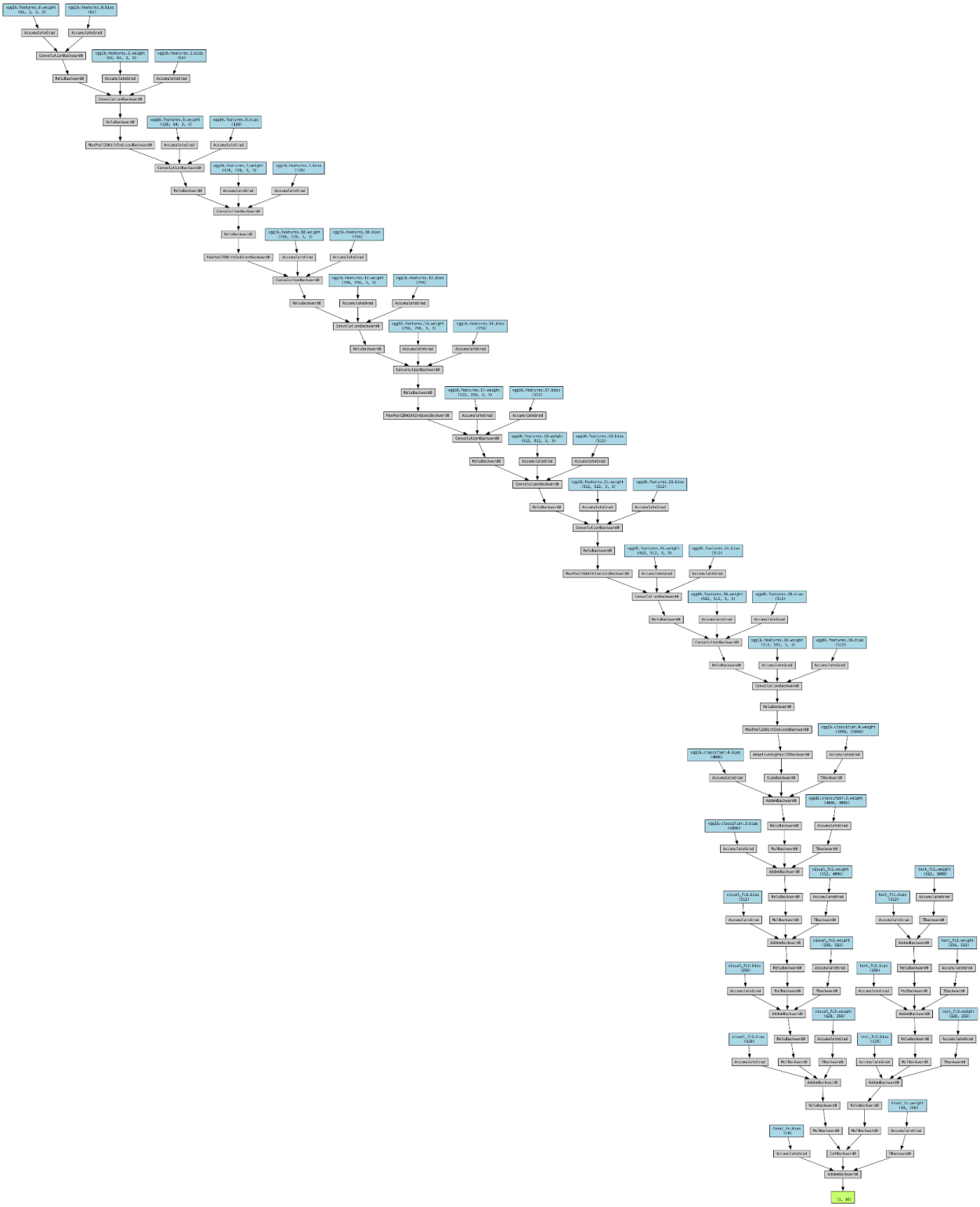
Further steps

Despite all the work done, several questions remain unanswered. The impact of varying the complexity and depth of the CNN model on different types of document images is not fully explored. Additionally, the Combined Neural Network could benefit from a deeper investigation into more advanced OCR techniques and text embedding methods, since images are full of noise, which may further enhance its performance. The computational demands of the GLT Classifier also pose a significant challenge; future work could explore more advanced data augmentation strategies in this work to improve model generalization and robustness. Furthermore, testing these models on a broader range of

datasets (like rvl-cdip) would provide a better understanding of their generalizability and robustness.

Appendix

Combined Neural Network Architecture



References

J. Ferrando, J. L. Domínguez, J. Torres, R. García, D. García, D. Garrido, J. Cortada, and M. Valero, “Improving accuracy and speeding up document image classification through parallel systems,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 12138 LNCS, pp. 387–400, 2020.

Saifullah, S. A. Siddiqui, S. Agne, A. Dengel, and S. Ahmed, “Are deep models robust against real distortions? a case study on document image classification,” in 2022 26th International Conference on Pattern Recognition (ICPR), pp. 1628–1635, 2022.

Saifullah, Saifullah; Agne, Stefan; Dengel, Andreas; Ahmed, Sheraz (2022): DocXClassifier: High Performance Explainable Deep Network for Document Image Classification. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.19310489.v3

Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, “A convnet for the 2020s,” 2022.

S. Zheng, Y. Song, T. Leung, and I. Goodfellow, “Improving the robustness of deep neural networks via stability training,” 2016.

C. Tensmeyer and T. Martinez, “Analysis of Convolutional Neural Networks for Document Image Classification,” Proc. Int. Conf. Doc. Anal. Recognition, ICDAR, vol. 1, pp. 388–393, 2017