SAPIENZA
UNIVERSITÀ DI ROMA

Handwritten Mathematical Expression Recognition and Translation in LaTeX

Professor:
Fabio Galasso

Students:
Federico Alvetreti (1846936),
Aurora Bassani (1852791),
Erica Luciani (1868647),
Laura Mignella (1920520)

Academic Year 2023/2024

Contents

1 Abstract **2**

2 Introduction **2**

3 Related work **2**

4 Proposed method **2**

5 Dataset and Benchmark **3**

6 Experimental results **3**

7 Conclusions and Future work **4**

8 Appendix **5**

8.1 Tasks Division 5

8.2 Plots 5

References **9**

1 Abstract

This project aims to generate LaTeX sequences from given mathematical expressions images.

Our proposed approach employs a cascade model consisting of two stages: initially utilizing YOLOv8 for the detection of symbols within images, followed by an LSTM network for sequence-to-sequence translation into LaTeX representation.

After hyperparameters tuning we obtained 97% mAP50 on the YOLO and a 66% accuracy on the LSTM translation.

The cascade model achieved an overall accuracy of 26%.

2 Introduction

In response to the imperative to seamlessly translate handwritten mathematical expressions into LaTeX format, this report meticulously details a machine learning initiative that employs fine-tuning and a cascade model. The core objective is to expound upon the methodologies implemented in training algorithms, specifically designed to discern and transform the inherent chaos within handwritten mathematical notations into the structured elegance demanded by LaTeX equations.

You can find the whole project here .

3 Related work

In Handwritten Mathematical Expression Recognition (HMER), approaches historically fell into two categories: grammar-based and encoder-decoder models. Grammar-based methods relied on predefined rules (e.g., context-free or relational grammars), lacking adaptability to large datasets. Conversely, encoder-decoder models, such as LSTMs or Transformers, drew inspiration from the successes in machine translation and speech recognition, gaining significance in this field.

4 Proposed method

We propose a cascade model consisting of two phases:

- a detection phase, obtained by fine-tuning YOLOv8, that would recognize each mathematical symbol in the image and its position;
- a translation phase, obtained by training from scratch a Long Short-Term Memory model, that would take as input the symbols and positions found by YOLO and translate them into the actual latex phrase.

5 Dataset and Benchmark

We leveraged the **Aida Calculus Math Handwriting Recognition Dataset** from Kaggle, comprising $100,000$ handwritten mathematical expression images and their corresponding targets. Post data cleaning, which involved the exclusion of images containing symbols with excessively low frequency 1, our refined dataset comprised $37,548$ images, encompassing 51 distinct mathematical tokens 2.

Among these, $2,000$ images were reserved for testing the cascade model, while the remaining $35,548$ were allocated for training both the LSTM and YOLO models. All images were used for LSTM training, whereas a subset of $5,000$ images was employed for YOLO model training.

We evaluated the YOLO detection results using Precision, Recall, mAP50, and mAP50-95 metrics. For assessing the LSTM and cascade model performance, we employed overall accuracy (the count of correctly translated sequences over total sequences) and a variant of accuracy introducing an error tolerance termed "patience." This tolerance factor allows a flexible evaluation, accommodating minor deviations in sequence translation. Making this "patience" parameter range between 0 and 30 we ended up obtaining a curve, which is referred as "accuracy curve" in the plots.

We developed the whole project using Google Colab.

6 Experimental results

In the following section, we will analyze the results of both the model and the overall results of the cascade model.

Before directly training YOLO on our dataset we performed an hyperparameters tuning using its built-in function ".tune" 3. This exploration led to the following choise of hyperparameters:

Table 1: Best choice of hyperparameters (Yolo)

Hyperparameter	**Values**
Learning Rate (lr0)	0.00744
Momentum	0.87391
Weight Decay	0.00045
Warmup Epochs	3.73217
Warmup Momentum	0.82597
Box	10.69948
Cls	0.40047
Dfl	1.60313

Using these hyperparameters we trained the model for 50 epochs and achieved a mAP50 of 97% and a mAP50-95 of 89% 4 5.

A grid search was also performed for the LSTM model. We ended up using this combination of hyperparameters:

Table 2: Best choice of hyperparameters (LSTM)

Hyperparameter	**Values**
Embedding dim	256
Hidden units	1024
Learning Rate	0.001
Dropout	0.2

achieving. After a further training of 40 epochs (50 in total) 6, it achieved an overall accuracy on the test of 66%.

Using the $2,000$ samples test (never seen from both the models) we evaluated the performance of our cascade model, obtaining an overall accuracy on the cascade model of 26% 8.

In order to evaluate the importance of each part of our total model we performed an ablation study.

The LSTM receiving the right input (not predicted by Yolo, without errors) behaved as expected achieving 63% of accuracy, while the Yolo ability to predict the right input sequence from images obtained 33,5% of accuracy 7.

We also evaluated the LSTM on increasingly wronger input by the YOLO model. At first we passed as input to the LSTM just correct YOLO predictions, then we added YOLO predictions with just one error and so on 9.

7 Conclusions and Future work

In conclusion, our outcomes are reasonably satisfactory given the resources available to us. The ablation study underscores the potential for enhancing the overall model performance by refining the capabilities of Yolo. Subsequent efforts could focus on improving the fine-tuning process of Yolo by leveraging more powerful computational resources. Furthermore, these resources could be employed to conduct a more comprehensive grid search on the LSTM.

8 Appendix

8.1 Tasks Division

At first, Federico and Erica took care of the cleaning of the dataset and wrote the initial draft of the Yolo part, while Laura and Aurora took care of the EDA and of the draft of the LSTM. Afterwards, we all worked together to refine both parts of the project, put them together and do the overall test.

8.2 Plots

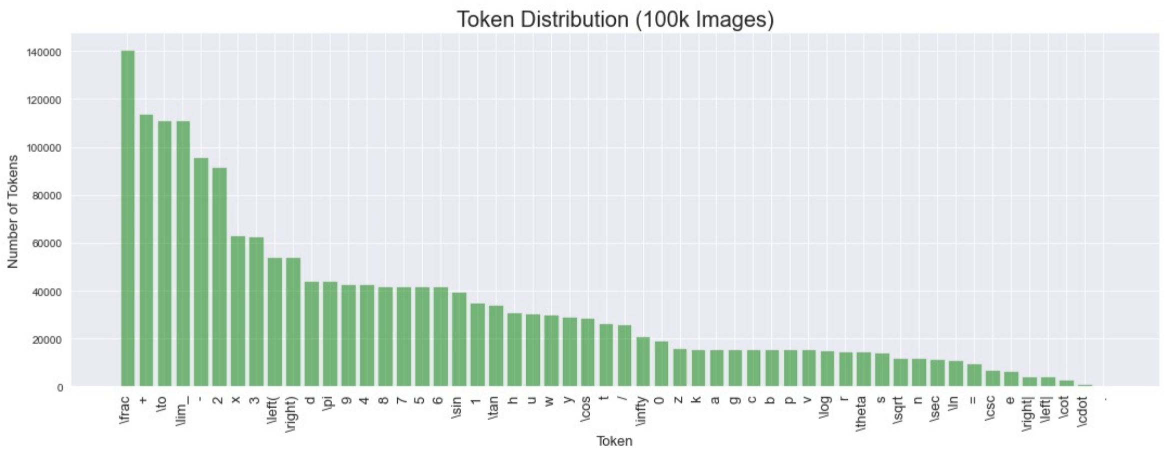


Figure 1: Initial data tokens distribution.

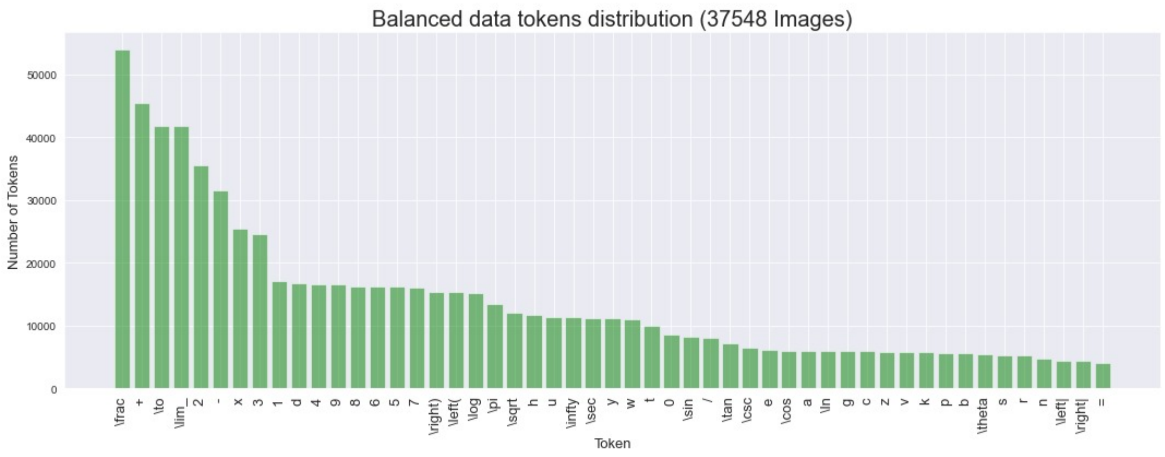


Figure 2: Final data tokens distribution.

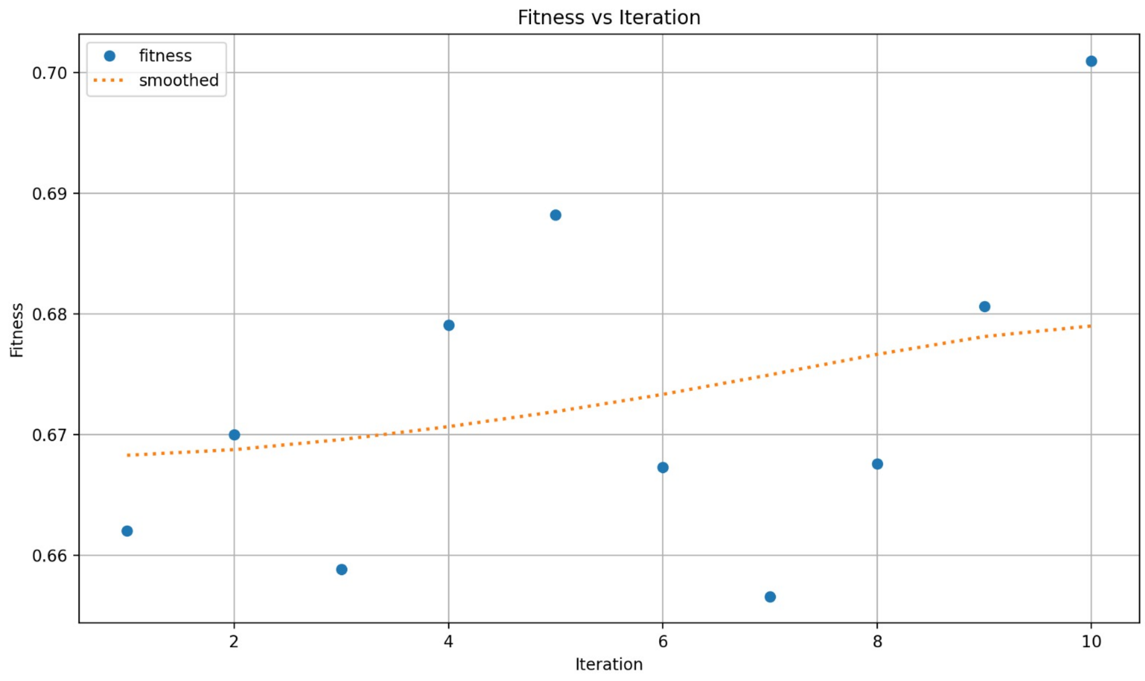


Figure 3: Yolo hyperparameters tuning.

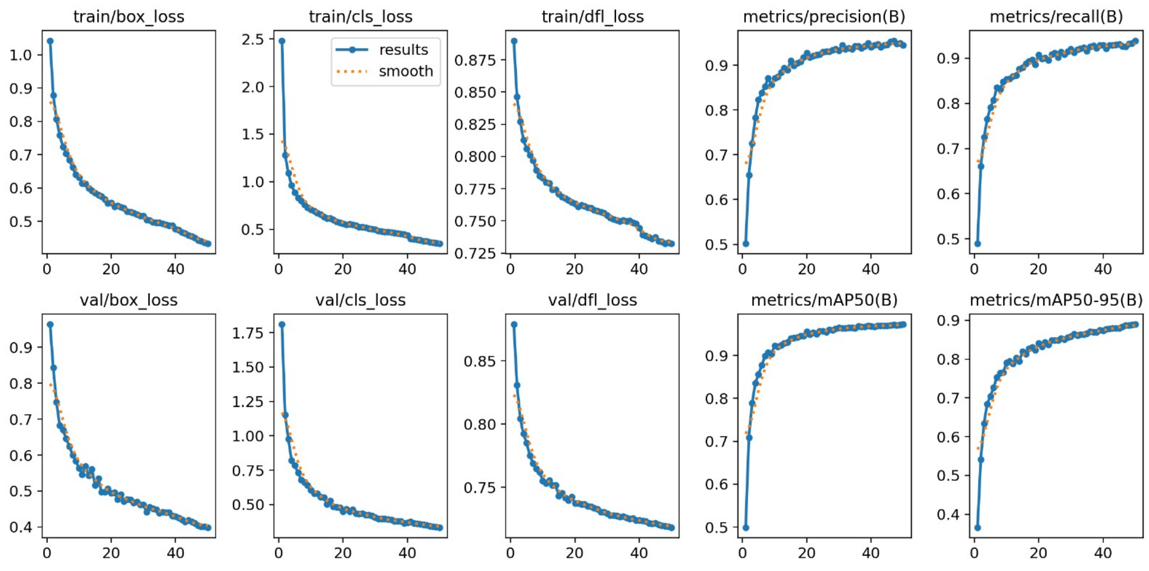


Figure 4: Yolo training results.

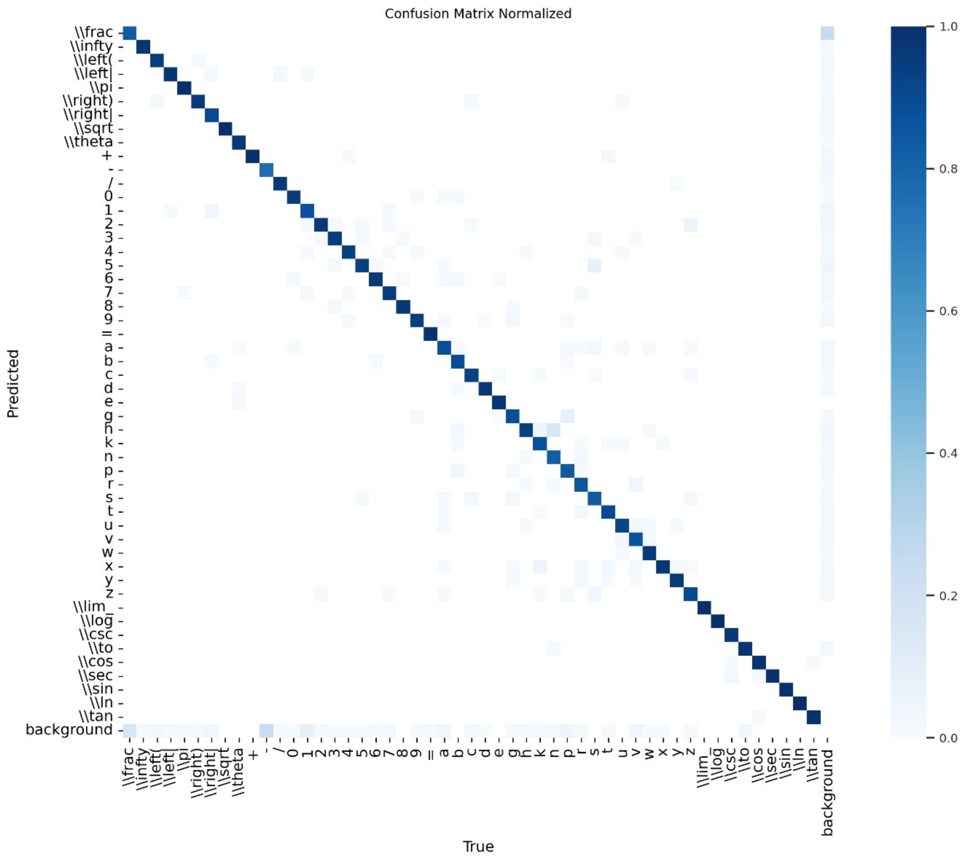
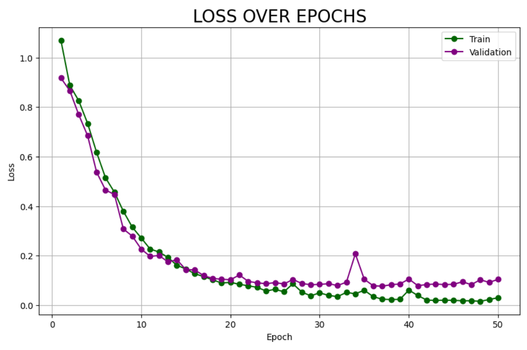
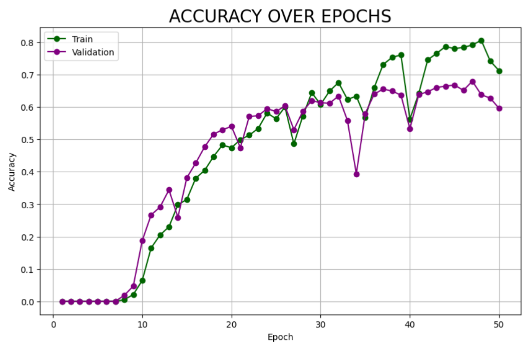


Figure 5: Yolo confusion matrix.

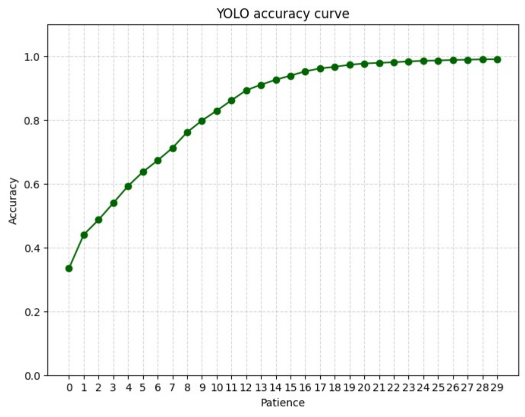


(a) Train and validation loss over epochs

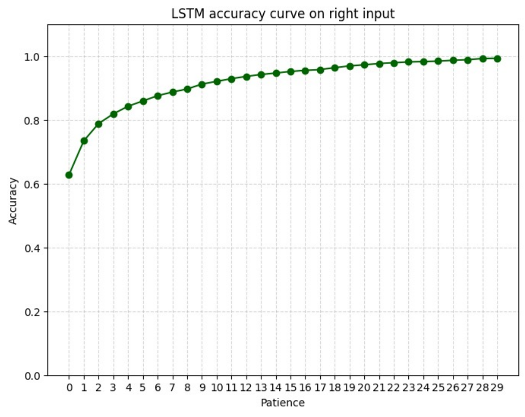


(b) Train and validation accuracy over epochs

Figure 6: Train and validation loss and accuracy for the LSTM model.



(a) Yolo accuracy curve



(b) LSTM accuracy on right input

Figure 7

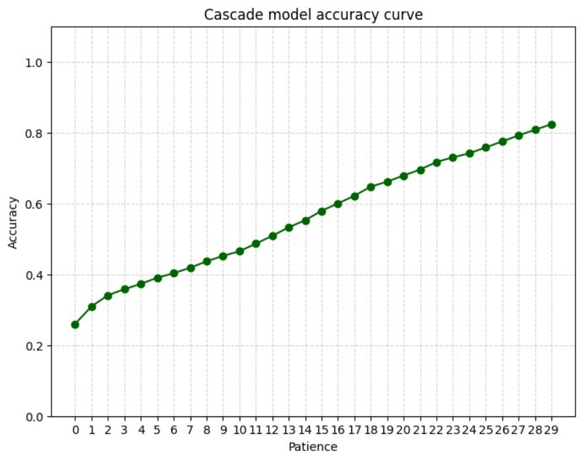
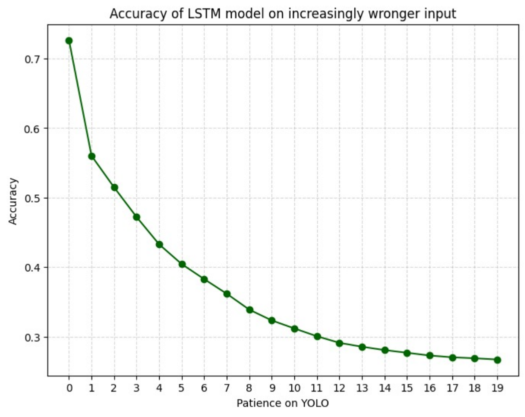
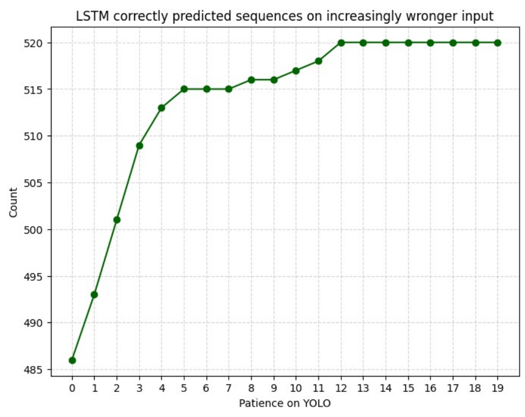


Figure 8: Cascade model accuracy curve



(a) accuracy of LSTM on increasingly wronger input



(b) number of LSTM correct prediction on increasingly wronger input

Figure 9: YOLO influence on LSTM

References

[1] https://www.kaggle.com/datasets/aidapearson/ocr-data/data.

[2] https://github.com/ultralytics/ultralytics.

[3] https://docs.ultralytics.com/models/yolov8/.

[4] https://arxiv.org/pdf/2308.05820.pdf.

[5] https://arxiv.org/pdf/1609.04938.pdf.

[6] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8978113.

[7] https://chat.openai.com/.