Final Project Report

Arlen Dumas & Tuyetlinh Nguyen

Dr. Marco Alvarez
CSC 561: Deep learning and Neural Networks
Department of Computer Science
University of Rhode Island
May 11, 2023

1 Introduction

A subset of handwriting recognition (HRW), mathematical expression recognition (MER) focuses on the automatic detection and transcription of handwritten mathematical expressions into their corresponding digital form. While the earliest forms of HWR appeared in the 1980s, the use of recurrent neural networks and feed-forward neural networks began in 2009 exploded the scope of research. MER in particular benefited from the introduction of these deep networks as it allowed for not only the "basic" mathematical expressions to be recognized, but it also opened up space for the more complex and unique characters to be introduced.

MER has always been a subject of interest for researchers, but it began receiving significantly more attention after the inception of CROHME (Competition on Recognition of Online Handwritten Mathematical Expression) in 2011. Both online recognition (which is focused on deriving information from symbols written electronically) and offline recognition (which is based upon pen-and-paper handwriting that is then scanned in) have peaked the interest of computer scientists and mathematicians across the world.

The interest in MER comes from the fact that though mathematics is a universal language, the way that we write our symbols varies greatly from person to person. This not only provides a rich environment for deep learning, but it also opens up an opportunity to generalize variations to a single universally understood ground truth.

For our research purposes we wanted to find an intersection of our fields of study that we both found interesting, as we both know from experience that interest in a topic from all parties gives the best chance at exceptional results. The idea for MER specifically began with a complaint regarding how LaTeX expressions aren't always intuitive and can be arduous to learn, especially when you're looking for a specific formatting or want your symbols to appear a specific way. The idea of automatically recognizing handwritten mathematical expressions and preventing the additional strain of learning yet another markup language came quickly became an idea that evolved into this project.

2 Related works

Interest in MER has waned in recent years, but has remained as a point of interest for competitions and small projects, though the literature available is robust enough to support multiple surveys. Recognition of handwritten formulae is a difficult structural pattern recognition task as there are significant fewer enforced standards for how mathematical formulas

as written when compared to plain English, with overlapping symbols and large alphabet sizes being a primary complaint to researchers[1]. Additionally it appears that though mathematical equations themselves are frequently used by researchers many of them are already proficient at using LaTeX when rendering equations, which renders the topic of MER to be somewhat redundant and more of a "pet subject" for those interested.

Because of the complications that come with offline recognition (image quality, smudges, lighting changes), most modern researchers seem to prefer online recognition[2][3]. This is likely not only because online images are cleaner and don't require the same level of prepossessing but also because it is much easier to generate datasets from online images captured through digital means than it is to scan in a handwritten image.

Inital methods for MER focused on stroke segmentation and normalization in thew form of slant collection, point interpolation, and resampling [4][5], but more modern methods are utilizing methods based on the Bidirectional Long Short-Term Memory (BLSTM) neural networks with Connectionist Temporal Classification (CTC) first introduced in 2006 [6]. While these methods are often used for tasks such as classifying speech, it's important to note that the structure (or lack thereof) gives context to what a mathematical expression means. The main feature of BLSTM models is that they can use contextual information over a long period, considering both the next part of the input sequence and the previous one. CTC refers to the scoring functions, and the use of CTC implies the introduction of an extra ' Blank' symbol in an alphabet. This seems to be a powerful way to implement symbol recognition as it can be used to identify when one stroke ends and the next begins [7]. Additionally, CNNS are often used for symbol recogition in this feild as well and seem to be a popular choice for small individual projects due to the easier implementation when compared to BLSTM methods [8].

Understanding the spatial relationship between symbols is a significantly more complicated task, as determining the correct relationship between two variables is vital to the overall understanding of the mathematical expression. Approaches based on geometrical features [9] and shape features [10] both appear commonly and have similar accuracy. Grammar-based approaches have been existed for nearly fifty years and appear to have had a recent renewal in interest with 2D Stochastic Context-Free Grammars (SCFG) are the most studied in the ME recognition domain [11]

The most recent advancements in MER have been utilizing attention as a secondary processing step in order to detect multi-line formulas and separate them in order to improve model accuracy [12], though it appears that these models still lack a fundamental misunderstanding

in the role that structure plays in a mathematical expression.

While refering to these papers was interesting to us and gave us some good insight on what was feasible in this domain, what was more interesting was the slew of projects that have been done by varying individuals over the years. As we explain later, we ran into quite a few issues when trying to implement our own MER recognition system but we did find multiple individuals who had implemented their own version of MER with the end goal of transferring the results into LaTeX format [13][14][15][16]. Though none of these projects are recent, it gave us a good look into what people were interested in.

3 Changing the project

3.1 InkML to LaTeX

When we began this project, we elected to use the CROHME 2023 dataset available through the TC11 datasets repository. Comprised of Ink Markup Language (InkML) images, the dataset was already split into online, offline, and bimodal recognition tasks and contained a wide array of non-standard symbols. The possibility of decreased processing time along with the varied data and existing and papers utilizing the dataset [16][17] made the CROHME dataset very attractive to use.

In practice, however, we learned that the CROHME dataset and the tools supporting the use of this dataset are outdated and to a large degree simply do not work anymoew. While issues such as the dataset being too large to push to github were able to be dealt with, the bigger issues were that the CROHMELib and lgEval tools required for the data to be usable simply don't work anymore. Githubs that attempt to get around this issue exist [18] but also face issues with core libraries being updated and no are no longer usable. Due to the lack of interest in MER for the last five years, there are no tools that allow InkML images to be used in the way there were intended to. In addition, there are almost no posts within the last five years relating to these issues on stackoverflow or similar sites. The handful of questions thaty do exist often do not have answers or are answered by directing users to [18], which we have determined do not work. Because the dataset we had chosen was such a pivotal part of our poject, we needed to step back and reevaluate what could be done in the time that we had remaining.

3.2 images to LaTex

After over a week of struggling to get the CROHME dataset working, we pivoted and looked back at the papers we had chosen for our literature review. We selected a few papers to look into more deeply and found that "What You Get Is What You See: A Visual Markup Decomplier" [citation] was closely related to what we were attemping but that it came with the added benefit of a piblically available github related to the project [citation] that contained the png images that were used in the project.

When we attempted to write a dataloader using this data, however, we ran into a multitude of issues. We're not entirely sure what the original authors of this paper did to preprocess their images, but our dataloader took over a day to get working and even when we were able to load in our images, we quickly ran into issues with our models. We spent a large amount of time fine-tuning our models so that we were able to uitlize the images provided, but after multiple days of trying we were unable to find any model that was able to take in these images for processing. Our best guess is that the images were corrupted some way either upon upload to the github or through the data loading process and had dimensions incompatible with any processing methods that we knew of.

3.3 Benchmarking MNIST

While we initially set out to complete a significantly more interesting and complex project, issues with the only available datasets that we could find prevented us from completing that part of the project, though we did gain some important knowledge about what we were actually trying to understand through this process. More than producing a given file or even recognizing mathematical expressions in general, wanted to rationalize how different models would impact a given recognition task. In our proposal we spoke about using a CNN/RNN and comparing that to a model that was pretrained using transfer learning and through what we had learned in class we realized that visual transformers were also a valid model for recognition tasks.

Because we did not have the time to select a new dataset after spending more three quarters of the time we had for this project doing literature review and fighting with the datasets we had chosen, we had to pick something we knew was reliable enough for us to benchmark anything. While it's a very small dataset without much variation, MNIST was a logical choice. While in an ideal world there would be a similar data set with more variation, it was more important for us to be able to show that this different models do make a difference in visual recognition tasks.

4 Methods

4.1 Models

Due to the aforementioned issues with our dataset and implementation coupled with the time restraint, we elected to experiment with the MNIST dataset using a CNN and a pre-trained ViT model. Our primary goal was to compare two different approaches for an image recognition task; training from scratch and transfer learning.

5 Experiments

5.1 Working with MNIST

Experiments using CNN-based models preformed well, reach a 90% to 95% accuracy quickly. This is likely due in part to the limited amount of labels provided by the MNIST dataset. There is a consideration that the CNN was overtrained and too deep, as we know that good performance on a small dataset can be due to the models memorizing the dataset. While the results don't explicitly indicate this, it's hard to not imagine how the results would differ on a more diverse dataset.

Our final CNN architecture is as follows:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_7 (Conv2D)           (None, 26, 26, 32)        320

 batch_normalization_7 (Batc  (None, 26, 26, 32)       128
 hNormalization)

 conv2d_8 (Conv2D)           (None, 24, 24, 32)        9248

 batch_normalization_8 (Batc  (None, 24, 24, 32)       128
 hNormalization)

 conv2d_9 (Conv2D)           (None, 12, 12, 32)        25632

 batch_normalization_9 (Batc  (None, 12, 12, 32)       128
 hNormalization)

 dropout_3 (Dropout)         (None, 12, 12, 32)        0

 conv2d_10 (Conv2D)          (None, 10, 10, 64)        18496

 batch_normalization_10 (Bat  (None, 10, 10, 64)       256
 chNormalization)

 conv2d_11 (Conv2D)          (None, 8, 8, 64)          36928

 batch_normalization_11 (Bat  (None, 8, 8, 64)         256
 chNormalization)

 conv2d_12 (Conv2D)          (None, 4, 4, 64)          102464

 batch_normalization_12 (Bat  (None, 4, 4, 64)         256
 chNormalization)

 dropout_4 (Dropout)         (None, 4, 4, 64)          0

 conv2d_13 (Conv2D)          (None, 1, 1, 128)         131200

 batch_normalization_13 (Bat  (None, 1, 1, 128)        512
 chNormalization)

 flatten_1 (Flatten)         (None, 128)               0

 dropout_5 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 327,242
Trainable params: 326,410
Non-trainable params: 832
_________________________________________________________________
```

5.2 Using a ViT through Transfer learning

When looking for pre-trained models to work with and beginning to evaluate various models on MNIST, we faced issues with input image and and output compatibility. Eventually, we were able to use the following model from Huggingface:

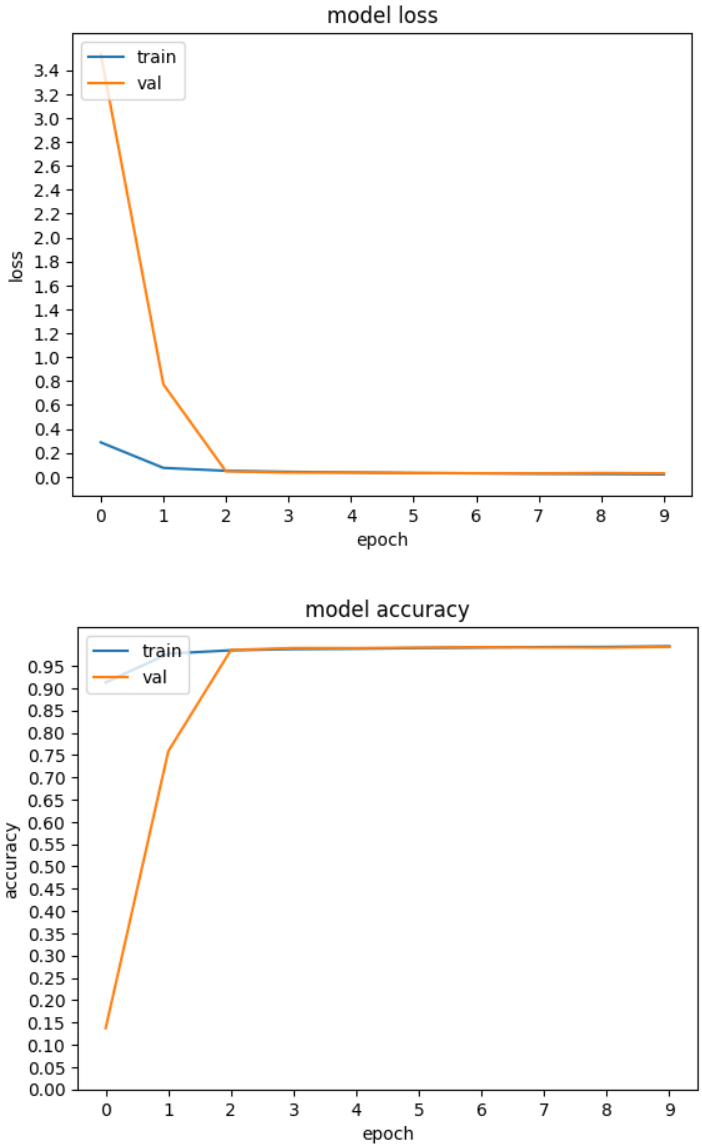
https://huggingface.co/farleyknight-org-username/vit-base-mnist

While it's hard to say what exactly this ViT was trained on, it's probable that at least some kind of numeric data was fed through at some point in the form of images or handwriting samples, which would make this transformer robust to the MNIST dataset in particular.

6 Conclusion

6.1 Findings

The following graphs depict metrics, namely loss and accuracy, for our final and best CNN model:



When evaluated on the test dataset of MNIST, our CNN model had 99.24% accuracy and a loss of 0.0246, as compared to the pre-trained ViT mentioned earlier, which has an accuracy of 98.29%. Given current literature, it is expected that the ViT perform at least as well as a CNN, which do not correlate with our results. This may be due to a few circumstances.

For example, the ViT required input of 224x224x3. Therefore, in order to compare our CNN

model to this ViT using the same dataset, we had to reshape the image accordingly. The increased channel dimension was handled by duplicating the grayscale MNIST value, which should not have affected the performance. However, we have reason to believe the rescaling of MNIST by a factor of 8 could have interfered with performance. Moreover, we did not train the ViT, as we evaluated the given model with the reshaped images.

However, Huggingface documentation says this model is a version of the google/vit-base-patch16-224-in21k model that has been finetuned on MNIST. According to documentation, the ViT model achieved an evaluation loss of 0.0236 and evalutation accuracy of 99.49%, which does support our initial expectation of better performance from the ViT model over the CNN.

https://huggingface.co/google/vit-base-patch16-224-in21k

6.2 Future considerations

In a perfect world there would be a dataset available for any task. Given that most of our issues came from a lack of good quality data, it would be vitally important for future iterations of this work to have data that is reliable, correctly labelled, and is of sufficent complexity. We would even go as far as to say that the best quality data would likely come from "in-house" sources at the URI computer science and Mathematics department. While this method would require a large amount of planning and prepossessing of the data, this project has reaffirmed to us that nothing can be done without good-quality data.

While it was too late for us to implement, in the last few days of our project we did find that CNN/RNN encoder-decoders may be useful to complete this task. There seems to be some question about how to best tokenize data when using encoder-decoders which does create some questions about how to preprocess data, but they seem to be as powerful as they are complex.

If we were to have more time to focus on the models instead of the data portion of this project, we would have liked to focus on benchmarking more models and playing around with non-standard activation functions, very short and very long epochs, and different splits for the data. Ideally, however, we would have liked to know about the significant issues with data in this project with enough time to gather and prepossess our own data.

All that being said, MER and converting images to LaTeX is an understudied field and while it is unfortunate that we weren't able to complete this project as we had initially intended there is a great opportunity to experiment with multiple methodologies and architectures to

complete this task.

References

[1] D. Zhelezniakov, V. Zaytsev and O. Radyvonenko, "Online Handwritten Mathematical Expression Recognition and Applications: A Survey," in IEEE Access, vol. 9, pp. 38352-38373, 2021, doi: 10.1109/ACCESS.2021.3063413.

[2] Deng, Yuntian , Kanervisto, Anssi , Rush, Alexander. (2016). What You Get Is What You See: A Visual Markup Decompiler.

[3] Deng, Yuntian, et al. "Image-to-markup generation with coarse-to-fine attention." International Conference on Machine Learning. PMLR, 2017.

[4] B. Q. Huang, Y. B. Zhang and M. T. Kechadi, "Preprocessing techniques for online handwriting recognition", Proc. 7th Int. Conf. Intell. Syst. Design Appl. (ISDA), pp. 793-800, Oct. 2007.

[5] Koschinski, H.-J. Winkler and M. Lang, "Segmentation and recognition of symbols within handwritten mathematical expressions", Proc. Int. Conf. Acoust. Speech Signal Process., pp. 2439-2442, 1995.

[6] M. Liwicki, A. Graves, S. Fernàndez, H. Bunke and J. Schmidhuber, "A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks", Proc. Int. Conf. Document Anal. Recognit., pp. 367-371, 2007.

[7] T. Zhang, "New architectures for handwritten mathematical expressions recognition", 2017.

[8] https://arxiv.org/pdf/1609.04938.pdfThis

[9] A. D. Le and M. Nakagawa, "A system for recognizing online handwritten mathematical expressions by using improved structural analysis", Int. J. Document Anal. Recognit., vol. 19, no. 4, pp. 305-319, Dec. 2016.

[12] L. Hu and R. Zanibbi, "MST-based visual parsing of online handwritten mathematical expressions", Proc. 15th Int. Conf. Frontiers Handwriting Recognit. (ICFHR), pp. 337-342, Oct. 2016.

[11] F. Alvaro, J.-A. Sánchez and J.-M. Benedi, "Recognition of printed mathematical expressions using two-dimensional stochastic context-free grammars", Proc. Int. Conf. Document Anal. Recognit., pp. 1225-1229, Sep. 2011.

[12] Zhou M, Cai M, Li G, Li M. An End-to-End Formula Recognition Method Integrated Attention Mechanism. Mathematics. 2023; 11(1):177. https://doi.org/10.3390/math11010177

[13] https://github.com/Chelagat/latex-project

[14]https://modelzoo.co/model/im2latex

[15]https://github.com/harishB97/Im2Latex-TensorFlow-2

[15] https://guillaumegenthial.github.io/image-to-latex.html

[16] https://cs229.stanford.edu/proj2013/JimenezNguyenMathFormulasfinalpaper.pdf

[17] http://cs229.stanford.edu/proj2017/final-reports/5241761.pdf

[18] https://github.com/ThomasLech/CROHME$_e$*xtractor*

[19] http://cs229.stanford.edu/proj2017/final-reports/5241761.pdf