1 Introduction

The act of transcribing is to understand and copy text from one medium to another. As such, transcription is relevant to Computer Vision. It is as an archetypic activity of Computer Vision and as a research program itself, as seen in the many handwriting datasets. In this research project, we explore transcribing handwritten mathematical expressions to LaTeX.

We chose the research topic of transcribing handwritten mathematics into LaTeX because firstly, typing mathematics into LaTeX can be a tedious and time-consuming activity amenable to improvement. As a personal anecdote, the author has spent much valuable time typing written problem sets into LaTeX for online submission, only to debug errors afterwards. An application to automate this process is of significant value. Secondly, LaTeX is the universally used throughout industry and academia for writing research papers, even this paper is in LaTeX! Finally, as expanded on in section 1.1, there is significant exposure and previous research into the topic of transcribing mathematics.

1.1 Related work

There have been many different computer vision pipelines to solve the typeset image to LaTeX transcription task. The problem has been an active field of research thanks to OpenAI, who recommended the problem in their yearly "Requests for Research"(1).The topic is also closely related to the field of image captioning, so we can also survey the architectures and pipeline there.

There pipelines for transcribing mathematics can be broadly categorized into two groups. The first group uses end-to-end deep learning pipelines. Research papers featuring such pipelines combine state of the art machine learning models, such as Sequence-to-Sequence (Seq2Seq) recurrent neural networks (RNN), convolutional neural networks (CNN), such that input is the image and the output is the desired label (2). The model is trained with backpropagation to learn salient features in the data to produce the correct output. In our case we input an image of the handwritten mathematics and expect the LaTeX as output. Some pipelines opt to build on Seq2Seq and use an attention mechanism to focus on relevant part of the image when predicting the next token (2).

Notable research includes: "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", which uses the attention mechanism for image caption generation, "What You Get Is What You See: A Visual Markup Decompiler" which applies a Seq2Seq model for transcribing images (3).

The benefit of such models is that they use Seq2Seq, which is suitable to the problem of generating variable length data such as LaTeX strings, however the downside of end-to-end models is that they require a large amount of data and compute resources.

The second group of pipelines is not end-to-end, it focuses on traditional computer-vision techniques, as seen in Gowda (4). Gowda first segments the image into symbols, via connected components, then computes feature descriptors (Histogram of Gradients) over the image. A classifier is used to predict the segmented symbols over the mathematical equation. Finally, the LaTeX formula is derived using a look-up table and heuristics about the position of symbol relative to the image.

Unlike the models from the two groups above, our models combines features from both groups in interest of saving time and computation. Rather than pass in an image directly to and end-to-end model, we compute a sequence of features over the formula image using classifiers as a form of dimensionality reduction, which is then used as training input for a Seq2Seq recurrent model. This is expanded on in Section 2.

2 Methodology

We started evaluating Handwritten Mathematical Expression on the CROHME dataset (5). The dataset is overall composed of 8902 training formulas and 1502 test formulas, in an inkML (pen stroke) format over the 2011, 2012, 2013 years. The inkML formulas were transformed into jpeg images using **CROHME_extractor/extract_formulas.py** script, written by myself as a derivative of Thomas Lech's CROHME_extractor (6). Some sample formulas are shown in Figure 1.

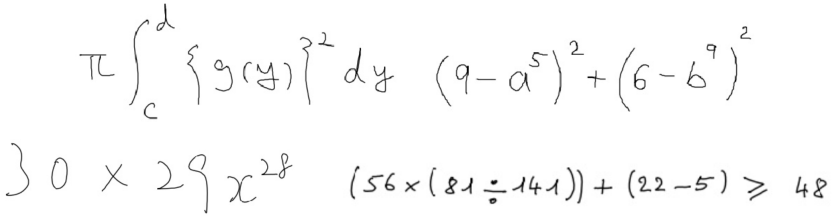

Figure 1: Sample formulae from the CROHME dataset

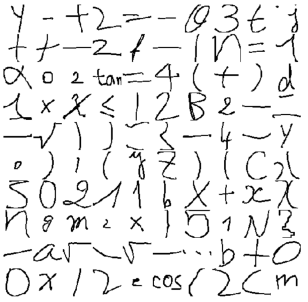

Figure 2: 100 random symbols from the CROHME dataset

2.1 Segmentation of math symbols

Next, all mathematical symbols in the CROHME dataset are segmented into 101 classes using **CROHME_extractor/extract_symbols.py** modified by my partner as derivative of Thomas Lech's CROHME_extractor (6). There are 114,027 symbols in the training dataset and 15,818 symbols in the test dataset.Because the inkML format provided in the CROHME dataset groups pen strokes by the symbol the strokes constitute, no computer vision techniques were used in symbol segmentation (5). The output symbol image has dimensions $(60, 60)$, a size amenable to further image processing and feature computation for algorithms as CNN and HOG, but not low enough that strokes look binarized and blocky.

The 101 classes each represent a mathematical operand such as one of alphabet, digits or Greek symbols, sequence of characters representing a mathematical function, such as "\sin", "\lim", "\int" etc. It is important to note that some frequently used operators with scope, such as "\sqrt" are included, but others such as "\frac" are not. Figure 2 shows a 100 random symbols.

In addition, each symbol's centeroid is computed, normalized by the width and height of the formula image size, then stored to later pass into our Seq2Seq model.

2.2 First Classifier - Convolutional Neural Network

Two Convolutional Neural Networks were jointly written by my partner and I, both were trained on the dataset of math symbols extracted in Section 2.1, using script **src/cnn.py**. The CNNs were trained with a train-validation test split of 15% and batch size of 64, and Adam optimizer was used for training process. The first, a shallow CNN ran for 20 epochs on an NVIDIA K80 GPU. The second, much deeper CNN ran for 20 epochs. The architecture of both CNNs is shown in Figure 3. Despite the large imbalances in the symbols distribution data as discussed in Section 3, the CNN was trained without taking into account the large imbalance in the data Figure 7.

2.3 Second Classifier - Support Vector Machine on HOG features

As an alternative to the CNN classifier, my partner was responsible for writing a Support Vector Machine (SVM) on trained on the Histogram of Gradients (HOG) model for each mathematical symbol in the training set. The script responsible for this is **src/hog_svm_rnn.py** with the **USE_CNN** flag set to **False**.

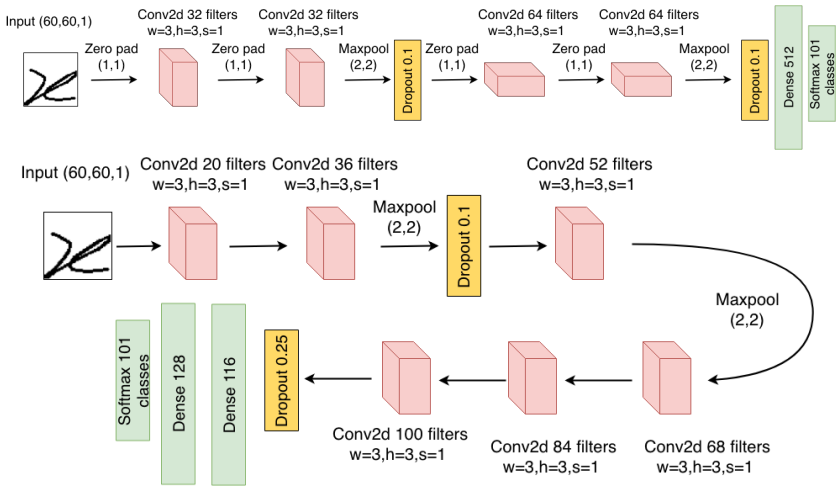

Figure 3: CNN classifier architectures, the shallow CNN is on top.

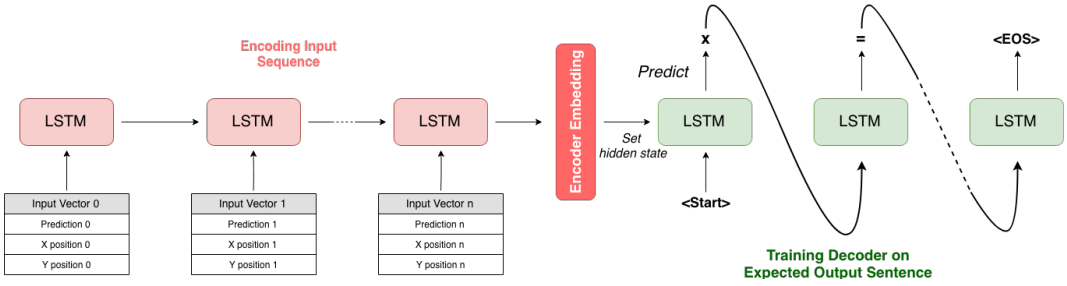

Figure 4: Seq 2 Seq model architecture

Due to restrictions in the machine learning library scikit-learn, a SVM with a linear kernel was trained with stochastic gradient descent and a hinge loss function (7).

2.4 Sequence to Sequence LSTM

Finally, one of either classification models in Section 2.2 and 2.3 were interfaced with a Seq2Seq language model, written by myself in **src/hog_svm_rnn.py**, see the **USE_CNN** Flag. The architecture of the Seq2Seq model is shown in Figure 4. A detailed description of the interfacing follows.

As seen on the left side of figure 4, for every formula in the dataset, symbols from the formula are segmented as in 2.1, and fed to the classifier for prediction. A sequence of triplets of the classifier's predictions, alongside the relative x and relative y position of the symbol's centroid in the formula image, are the input vectors.

The Seq2Seq architecture uses an encoder to process the source sequence, thereby creating encoding of the symbols and their positions in the formula (8). The encoder's output is next passed into the decoder as it's initial hidden state (8). The decoder is trained on characters over the ground truth LaTeX formula. Starting from a start of sentence character, at each time-step the decoder is to predict the character at the next time-step over the complete ground truth.

The model does inference on test images as follows: the images are first segmented and the triplet sequence is fed into the encoder to generate the encoder state. The decoder is conditioned on the encoding state and sampled from using either greedy or beam search sampling, also implemented by myself.

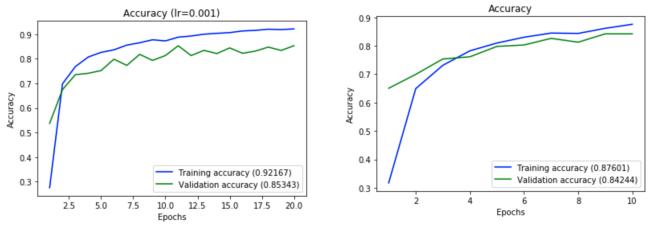

Figure 5: Training and validation curves for deep CNN (right) and shallow CNN of figure 3

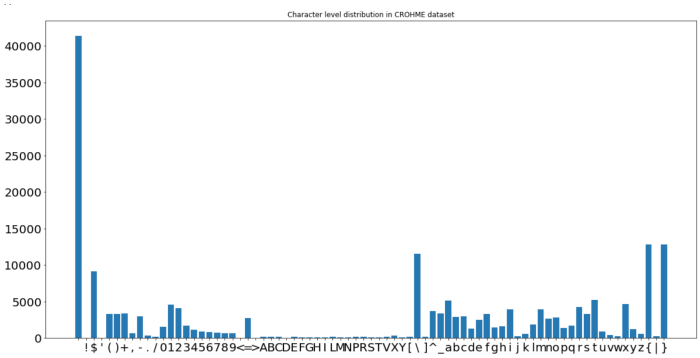

Figure 6: Distribution of *LaTeX* characters in CROHME.

3 Results and Findings

3.1 CROHME Dataset Distribution

Over the course of training models on the CROHME datasets, both my partner and I noticed strange attributes, such as non-converging losses, affecting the quality of models. After some investigation we believe this is due to two flaws in the CROHME dataset itself.

Firstly, there is an imbalance of characters in the CROHME data distribution (Figure 6, 6). Secondly, as seen in Fig. 2, symbols can be highly ambiguous, of the same shape but of multiple meanings without context. Examples of ambiguous pairs include "\times" and "x", "+" and "t", "o", "0" and ".".

3.2 Convolutional Neural Network

Both CNN models performed amicably, but not excellently after training on input symbols, as below.

CNN model	Test accuracy	Test log loss
Shallow	84.53%	0.5453
Deep	88.21%	0.7512

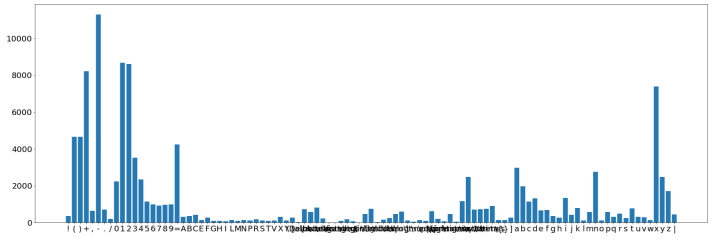

Figure 7: Distribution of classes in CROHME

Of the two CNN models, the deep model achieved a better accuracy than the shallow model, but had a larger logarithmic loss on the validation and showed evidence of overfitting, such as a high training accuracy in the 90%, see Figure **??**acccnn_accce the shallow CNN model was chosen, as accuracy can be a misleading metric. Despite numerous attempts, it was difficult to breach 90% without overfitting. We also noticed oscillations in the training curves. The author believes this is due to imbalance in the input distribution as discussed in 3.1.

3.3 HOG + SVM

My partner Keith was in charge of writing and training the HOG SVM classifier. The HOG SVM classifier achieves a 70.16% accuracy over symbols dataset. My partner ensured that the SVM accounted for imbalance in the class dataset by modifying the loss function to penalize misclassification of less frequent symbols. This lead to lower recall for the most the frequent symbols and higher recall for the less frequent symbols.

3.4 Sequence to Sequence

In our final Seq2Seq result we noticed that SVM classifier produced better results than CNN classifier, this is because the SVM took into accounts the bias in the data distribution during training phase, leading to better precision than the CNN. The CNN did not. The per character level accuracy of SVM was 48%. For CNN it was 41%. Both results are significantly better than random prediction. We show examples generated by Seq2Seq on SVM on the Test set.

```
Ground Truth: f=\frac{\sqrt{5}}{RC}
Greedy: \frac{1}{x}+\frac{x}{x}
Beam Search: \frac{1}{\cos\sin(x)}

Ground Truth: \sinx
Greedy: \sinx
Beam Search: \phi\inY

Ground Truth: 9.0
Greedy: 9.9
Beam Search: 7.1

Ground Truth: 2=\sqrt{2+2}
Greedy: x=12x=2
Beam Search: \frac{\sqrt5}{\frac{8}{n}}{n}}{}\mbox{pB}_{\mbox{R}d}\mbox

Ground Truth: \frac{1}{1-c}
Greedy: \frac{1}{x^{2k}}
Beam Search: \frac{1}{\frac{n}{2}}n\lam

Ground Truth: 1-(-1)^d
Greedy: 1-(-1)^d
Beam Search: \frac{1!}{(x)^{2}}3

Ground Truth: {x^{2}}+3m=3i
Greedy: x^{2}}+3i=\sqrt{70}
Beam Search: \frac{5}{\frac{5}{\frac{5}{5}\theta}{5}\theta\right)
```

We noticed that our Seq2Seq model is unable to handle lengthy input sequences, hoever it can learn shorter sequences.We think this is due to using a character level seq2seq model, rather than a word level seq2seq model, a. Character level models have to models have to deal with long input sequences than a word level models. Additionally, we believe higher accuracy for the classifier used would improve model results.

In addition using beam search does the reverse of what is expected, there is repetition in the same output. This is also due to the model learning incorrect weights.

Finally, we noticed for input formulas containing scoped operators such as "\frac" or "\sqrt", the triplet of (predictions, relative x, relative y) will not be enough to infer the scope of such operators.

3.5 Challenges faced and Future Work

Through the course of the project, I specialized in building Seq2Seq model used in two different pipelines. The first pipeline was the image HOG-SVM-Seq2Seq, the second was the CNN-Seq2Seq pipeline. My second responsibility was to increase accuracy, and decrease log loss for the aforementioned models, this lead to writing beam search over Seq2Seq predictions. Overall I thought getting Seq2Seq working was more difficult than predicted, in addition to being time consuming to train.

Joint efforts in the project with my partner included training CNNs to perform mathematical symbol segmentation. Here, we found it challenging to accommodate a biased dataset without causing model training to diverge. We tried weighting the loss function similar to SVM, however this was not fruitful.

Given future time and effort towards the problem, I would like to implement an end-to-end model towards the task as in (2)(3), and implement better metrics such as BLEU and METEOR to evaluate sentence level.

4 Conclusion

In conclusion, two classifiers, an SVM trained on HOG features, and a CNN were trained to produce predictions over of mathematical symbols in the CROHME dataset. During training, images of formula were segmented into images of mathematical operands and operators. A character based Seq2Seq model was trained on the sequence of triplets to predict the $\LaTeX$ formula: math symbol as predicted by classifier, the relative x position in formula, the relative y position in formula . It was found that our model, despite generalizing over the simpler equations, lacked generalization as the length of the corresponding $\LaTeX$ equation increased; this can be attributed to imbalance of formulas and symbols in the dataset and the inability of our model to deal with scoped operators. Overall the author believes significant improvement can be made towards mathematical formula transcription by using end-to-end models.

References

[1] "Requests for research: Im2latex.." `https://openai.com/requests-for-research/#im2latex`. Accessed: 2018-12-03.

[2] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *CoRR*, vol. abs/1502.03044, 2015.

[3] Y. Deng, A. Kanervisto, and A. M. Rush, "What you get is what you see: A visual markup decompiler," *CoRR*, vol. abs/1609.04938, 2016.

[4] "Keerthi gowda: Im2latex.." `https://keerthigowda.github.io/Im2LaTex/`. Accessed: 2018-12-03.

[5] "Competition on recognition of online handwritten mathematical expressions.." `www.isical.ac.in/~crohme/result.html`. Accessed: 2018-12-03.

[6] "Github: Thomaslech/crohme_extractor." `https://github.com/ThomasLech/CROHME_extractor`, 2018.

[7] "Scikit-learn sgdclassifier." `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html`. Accessed: 2018-12-03.

[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.