CS230

CS230 - Spring 2020 Final Report

Ricky Grau
Department of Computer Science
Stanford University
jrgrau@stanford.edu

Taylor Song
Department of Mechanical Engineering
Stanford University
jhtsong@stanford.edu

Daniel Xia
Institute of Mathematical and Computational Engineering
Stanford University
jx643@stanford.edu

1 Introduction

There has been numerous effort in recent technology to connect the analog and digital experience in reading and writing. For example, the pen scanner is one of the pieces of modern technology that is able to assist text recognition over printed text with Optical Character Recognition(OCR) technology and convert to encoded text. Such technology allows for easier text editing, text-to-speech applications, translation, preservation of data, and so on (5). However, most of the products in the market fall short of scanning mathematical equations but resort to saving them as an image (1).

Hence we try to tackle the non-standard OCR task with real-world rendered expressions with LaTeX labels through a deep learning model. Our attempts in incorporating the deep learning architecture include applying the VGG model, and further developing to combine CNN with RNN to generate latex from images.

2 Related Work

OCR for mathematical expressions has been actively sought amongst researchers. Mathematical expressions with sub/superscript, operator symbols of different sizes, and nested fractions have made mathematical expression recognition a non-standard OCR task (2).

Prior approaches include an integrated OCR system called INFTY, which conducts layout analysis, character recognition, structure analysis, and error correction (9). Other approaches include an attention-based neural encoder-decoder model (3).

There also has been numerous effort in recognizing online and off-line handwriting of alphabets and numbers (8). The CROHME competitions, first held in 2011, were intended to facilitate study of math recognition amongst researchers (6).

One interesting paper we encountered attempted to classify over 300 mathematical symbols in handwritten text (7). Their model used a Simple Linear Iteratve Clustering (SLIC) method to group connected pixels and isolate individual characters in the handwritten text. The researchers then used a pre-trained SqueezeNet cnn architecture for classifying these identified regions. The SqueezeNet architecture consists of a combination of 1x1 and 3x3 convolutional filters, alternating between squeezing and expanding. This has the effect of both reducing the total number of parameters in the model and to add an extra learning step between these so called "fire modules."

,

3 Dataset and Features

For our dataset, we are using the the typed latex expressions collected by Deng et al. for their paper on translating images to latex(4). This dataset consists of roughly 100,000 images of typed mathematical expressions with their corresponding latex representation. We first attempted to use a dataset of handwritten mathematical formulas from kaggle(10), but after some initial experimentation we found that interpreting different handwritten versions of mathematical characters and then translating to latex was too difficult a task at the moment.

Each image in our dataset is an A4 sized image with a typed mathematical expression towards the top of the page, and comes with a corresponding latex expression. We preprocess the images by cropping to just the region containing the mathematical formula using a program provided by Deng et al. (4). The size of this cropped image can vary, but is generally much wider than it is tall. The length of the latex expression also spans a wide range, with the maximum length of a latex expression being 941 characters. The dataset provided was already split into a train, dev and test set. The training set contained 83870 examples, while the testing and validation sets each contained 10355 examples.

We found that using the entire dataset was prohibitively expensive computationally and limited our ability to iterate and develop a model, so we decided to filter the dataset and only use examples where the latex expression is under 80 tokens in length. This reduced our training set size to 66900 examples for our training set, and 8258 examples for our test set.

We split the latex expressions word based, and used the model to predict the following word in a sequence. Using this model, our vocabulary contains 3583 words, including lowercase letters, digits, and some special characters (including "{", "(", and "_" among others).

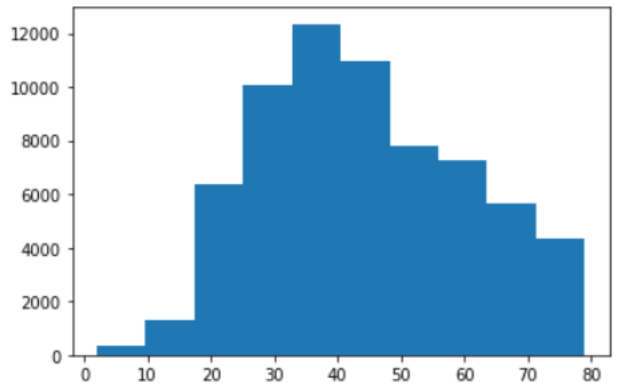

Figure 1: Distribution of words in filtered training dataset

4 Methods

4.1 Overall Task

Our overall task is to try and recreate the LaTeX output from an image of a typed mathematical formula. Since OCR model fails to represent spatial relationship between symbols and is limited to recognizing separate symbols, they cannot be expressed and labeled in LaTeX format. Hence we chose to focus only on utilizing the deep learning model. However, there are many complications that arise when trying to take an input image and generate a LaTeX expression with variable length. Following models and approach were found during literature study (4), and we decided to use a CNN to reduce the input images to feature vectors, then pass those vectors to an LSTM encoder-decoder to predict the characters making up the latex expression for the mathematical formula in the image.

4.2 Model and Approach

The current OCR models are not suited for the task since latex expression contains special symbols which are not included in the OCR dictionaries and also there are some latex expression that are corresponded to not only the symbols but also things like the relative position between symbols. For example, in Figure2,the expression \frac does not only correspond to a line but also the relative position of the following symbols. However we want to use some of the concept of OCR. One approach of the OCR model is Convolutional Recurrent Neural Network(CRNN). We want to use a similar structure to build the model, so the model might be similar to photo captioning model which will combine CNN and RNN models.

$$ds_{11}^2 = dx^+dx^- + l_p^9\frac{p_-}{r^7}\delta(x^-)dx^-dx^- + dx_1^2 + \cdots + dx_9^2$$

Figure 2: Example Latex Expression

Our model consists of two main components. The first component is a CNN model which is meant to extract features from the images of our mathematical formulas. The output of this CNN is then passed to the second component of our model, which consists of a RNN using Long Short Term Memory units. This part of the model takes the output from our CNN and attempts to convert it to a sequence of latex tokens extracted from our training data set.

4.2.1 Baseline Model/initial approach

Our initial approach is to build a character level based prediction model. We break down the latex expressions into single characters and try to generate sequence of characters from the latex image. Because the model's complexity depends on the maximum length of the output sequence, we filter the training data and only focus on the expressions that have less than 50 characters. For the convolution part of the model, we have utilize the pre-trained VGG model to extract the photo features. We loaded and resized the photo to (224,224) and passed it through the VGG model. VGG model is a CNN model that was developed for large-scale visual recognition. We load the model and remove the last layer of the model which is used for classifying image. We are not interested in classifying the image but we are interested in the internal representation of the image, which are the internal features. The parameters and weights of the VGG model were pretrained and loaded from the internet. We used this model to convert the images into vector of size (4096,) containing such features of the image.

The second part of the model is a sequence model and a feature extractor model followed by a decoder model. We used ”?” as the starting symbol and ”@” as the ending symbol for latex expressions since these two symbols are not commonly used in latex expression. The model we develop will generate one character each time and the previously generated sequence will be used as input to generate the next character. The starting sequence will be ”?” to start the generation and if ”@” is generated or the sequence reaches its maximum length, the generation will stop. All the char are encoded into integers for representation. The final loss function used is categorical cross-entropy.

4.2.2 Current Approach

Through discussion, we think it might be better to use a word-level model since there is only a limited dictionary of words in latex and by using a word level tokenization, we can have more suitable training examples. The filter that train on samples with less than 50 characters limits the size of training set. And in addition, we have decided to build our own CNN layers since the input size of the VGG model does not suit our image and resizing the image to (224,224) makes the image not possible to be recognised by human. The VGG model was trained on images that are not relevant to the current task, so we have decided to train our own CNN layers.

CNN Component The table 1 shows the specific hyper-parameters for our CNN model. The inputs are padded and resized into the size of (500,100,1) and are input to the layers. The output of the CNN layers will be the features extracted from the image and is later input into the second part of the model. We tried to structure the CNN component of our model to closely mimic the architecture used by Deng et al. (4) In some cases, we reduced the number of convolutional filters and increased the stride in order to reduce the total number of parameters in our model and accelerate training. By shrinking the output of the convolutional portion, we were able to significantly reduce the number of trainable parameters in the following dense layer. The table below describes the layers used in our final model.

COV	POOL
c:256,f:(3,3) ,s:(1,3),bn	Po:(2,2),s:(2,2)
c:256,f:(3,3) ,s:(1,3),bn	Po:(1,2),s:(1,2)
c:128,f:(3,3) ,s:(1,1)	Po:(2,1),s:(2,1)
c:128,f:(3,3) ,s:(1,1),bn	-
c:64,f:(3,3) ,s:(1,1),bn	Po:(2,2),s:(2,2)
c:32,f:(3,3) ,s:(2,2)	Po:(2,2),s:(2,2)

Table 1: CNN Layers. "c" is convolution layer followed by the number of layers, "Po" is max-pooling layer followed by the filter size,"f" is the size of the convolution block, "s" is the stride size, "bn" is batch normalization, the order of the filter is from bottom to top

RNN Component We have not change much for the second part of the model compared to the baseline model. It is still a sequence model followed by a decoder model. This time we have decided to use a word level tokenizer and the latex expressions are converted into arrays of integers. The start and the end of the expression are padded with start_token and end_token to symbolize the start and end of the expression. The general structure of the second part of the model remains unchanged. The final loss function is still remained as categorical cross-entropy.

5 Experiments, Results, Discussion

We evaluated our predictions using the Bilingual Understudy Evaluation (BLEU) score. This metric is appealing because it is easy and efficient to compute on predicted sentences and is easy to understand. The metric is calculated by counting the total number of matching n-grams between the predicted and the actual sentence, regardless of position. If the predicted sentence contains all n-grams present in the reference sentence, then it will have a higher BLEU score.

The metric also punishes predicted sentences that have more characters than the reference text. One could imagine a case where a predicted translation contains all of the n-grams as the reference translation, but is inaccurate because it contains many superfluous characters. Thus, to acheive a perfect translation score of 1, a predicted sentence must contain all n-grams present in the reference, with no extra characters.

In our first evaluation of our model, we used a very small training set (50 example images) with a large latex maximum length (250 characters). With this model, our bleu score was disappointingly (but not unexpectedly) 0. Further, it took a long time (30 minutes) to calculate a prediction for just a single image. We then decided to filter the dataset and use only images whose corresponding latex expression was under 50 characters. This sped up both the training of the model and the time required to make predictions, allowing us to make predictions on 100 test images. On this test dataset, we achieved an average BLEU score of 0.143, with a maximum score of 0.485 and a minimum score of essentially 0.

These results were in fact worse than their low scores would suggest. After some further inspection of the predicted latex expressions, we found that the model was repeatedly predicting the same character (namely "̈). Since this is a fairly common character in most mathematical latex expressions, our BLEU score was much higher than it should have been.

After adjusting our preprocessing methods and changing the architecture of our model, our BLEU scores went down significantly. Our average BLEU score dropped down to 0.046, with a maximum BLEU score of 0.077. These scores are still far from where we want them to be, and although they are lower than our scores when using the VGG model, qualitatively the output more closely resembles a real latex expression than the results of our VGG model. This is likely because of the changes we made in preprocessing our latex, and choosing to split on latex tokens as opposed to characters.

In addition, the BLEU scores achieved by the VGG network were calculated on shorter sequences (maximum of 50 characters), while our new architecture was tested on sequences with a maximum of 80 tokens, where each token consists of some number of characters. This also explains some of the decrease in BLEU score. Despite matching more characters accurately with our new model, the extended length of the reference latex ensures that the ratio of matching characters to sequence length is smaller, resulting in a lower BLEU score.

6 Conclusion / Future Work

Our model does not come close to achieving the results demonstrated by Deng et al. (4). We tried to structure our model similarly to theirs, but one key difference is that we did not incorporate attention in the RNN component of our architecture. By utilizing attention, we could ensure that each predicted latex token is focusing it's "attention" on the relevant part of the image, which could improve the accuracy of our output predictions.

Nevertheless, it is unlikely that the addition of attention would improve our model from close to 0% accuracy to almost 100% accuracy. We believe there may still be some issues with our preprocessing of the data that could be preventing our model from making accurate predictions. A deeper analysis of the inputs to our model could shed light on changes to our preprocessing steps that would increase model performance.

7 Contributions

Each member of the team contributed equally to the final project. We "pair programmed" for the entire project, and always made decisions as a group.

References

[1] Mark BMark. Best pen scanners and digital highlighters of 2018, Sep 2019. URL: `https://www.mbreviews.com/best-pen-scanner-digital-highlighter/`.

[2] Kam-fai Chan and Dit-Yan Yeung. Mathematical expression recognition: A survey. *International Journal on Document Analysis and Recognition*, 3, 02 2001. `doi:10.1007/PL00013549`.

[3] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. Image-to-markup generation with coarse-to-fine attention, 2016. `arXiv:1609.04938`.

[4] Yuntian Deng, Anssi Kanervisto, and Alexander Rush. What you get is what you see: A visual markup decompiler. 09 2016.

[5] emerging technologies team. Pen scanner evaluation summary. URL: `https://library.stanford.edu/projects/emerging-technologies-team/technology-assessments/pen-scanner-evaluation-summary`.

[6] Harold Mouchère, Christian Viard-Gaudin, Richard Zanibbi, Utpal Garain, Dae Kim, and Jin Kim. Icdar 2013 crohme: Third international competition on recognition of online handwritten mathematical expressions. 08 2013. `doi:10.1109/ICDAR.2013.288`.

[7] Tavakolian N. Fitzpatrick D. Fernando C. Suen C. Y. Nazemi, A. Offline handwritten mathematical symbol recognition utilising deep learning. 10 2019.

[8] R. Plamondon and S. N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[9] Masakazu Suzuki, Fumikazu Tamari, Ryoji Fukuda, Seiichi Uchida, and Toshihiro Kanahori. Infty: an integrated ocr system for mathematical documents. pages 95–104, 01 2003. `doi:10.1145/958220.958239`.

[10] Kaggle. Rachel Tatman. Handwritten mathematical expressions. URL: `https://www.kaggle.com/rtatman/handwritten-mathematical-expressions`.