Image to Latex Translation with Machine Learning

Team: Albert Wang, Irwin Deng, Kevin Lamb. **Project Mentor TA:** Mahesh

1) Abstract

For this project, we study the problem of optical character recognition (OCR) by diving into image to LaTeX translation. This is important for demystifying LaTeX as a barrier of entry from making a professional document and can serve students, resume-writers, and LaTeX users as a whole. Our primary target contribution is to collect a new training dataset for this problem. Second, we would like to evaluate the machine learning models we created by training and testing them on the dataset of images we generated. We wrote a program that produced images of handwritten mathematical formulae that could be translated to LaTeX commands. We generated 100 thousand images in addition to their corresponding correct LaTeX translations. We then wrote, trained, and tested 3 machine learning models on the dataset, the first being our best attempt to replicate the Texas A&M model, and the other two being models we created through what we learned this semester. We measure accuracy in two ways, and perhaps most interestingly, we find that one of our modifications improved the accuracy over the original model.

2) Introduction

We propose to create a program to recognize mathematical formulae from handwritten images and translate it to LaTeX commands. The inputs to our system are Image-LaTeX pairs (x,y) where $x \in R^{HxW}$ is a grayscale .PNG image of a mathematical formula with height H and width W, and $y = [y_1, y_2, \dots, y_t]$ is a sequence of *t* LaTeX commands and characters. The trained model should then be able to take some image x which corresponds to a LaTeX sequence y and produce the LaTeX sequence y (or at least a very good approximation). The training and testing data will be produced using a program that generates valid LaTeX and a corresponding image of the handwritten math equation. We will be using a modified version of the BLEU score to evaluate our model.

If we are successful in making an accurate model, it could serve as an invaluable resource for STEM students, professionals, and anyone who uses LaTeX to type up complex mathematical formulae. We've noticed how often Detextify is used among the CIS community here at Penn. However, a major limitation of Detexify is that it only recognizes single symbols. We hope that this project can help any future students, resume-writers, or anyone who writes in LaTeX. The main impact of our project is demystifying LaTeX because it is generally considered a barrier of entry for many people who want to make a professional document.

3) Background

1. Zelun Wang, Jyh-Charn Liu, "Translating Math Formula Images to LaTeX Sequences Using Deep Neural Networks with Sequence-level Training", Texas A&M 2019. See paper and code implementations here. The paper proposes a neural network model for

translating images of mathematical formulas into LaTeX. Much of this paper discusses techniques for vectorizing the source image to be better understood by the machine learning algorithm. We could use these techniques as a starting point for testing other ML algorithms, or improve upon the image preprocessing algorithms implemented in this paper.

2. Yuntian Deng, Anssi Kanervisto, Alexander Rush, “What You Get Is What You See: A Visual Markup Decompiler”, Harvard 2016. This paper is cited frequently in the aforementioned Texas A&M paper, and proposes a very similar model to the Texas A&M paper. The code for this paper, as well as pre-trained models from their training data, are readily available online. This paper not only focuses on building a model for decompiling an image of LaTeX output into the underlying LaTeX, but more generally on decompiling an image produced by any defined markup language. They also build a model to produce HTML from an image of a small webpage. The more general nature of their model allows for more customization in our own testing, and also opens the door for more future research by looking at other markup languages like HTML in a similar light (could we build a model that maps a hand-drawn mock-webpage to approximate HTML to produce that webpage?).

4) Summary of Our Contributions

1. **Contribution(s) in Data:** We created an algorithm that generates semantically logical mathematical expressions in a pseudo-random manner, then samples symbols from a handwritten math symbol dataset on Kaggle to create a .PNG image of the equation. We used our algorithm to create over 100,000 unique image-LaTeX pairs that we then used to train and test various ML models. This is a unique contribution into the research space as most papers in the past use images of typed math equations rather than handwritten ones, and our algorithm can easily be expanded to produce more data and cover more latex commands.
2. **Contribution(s) in Algorithm:** We will build upon the work of the Texas A&M paper by testing different types of ML algorithms. The work in the paper mostly used a CNN and LSTM. Our goal was to find if substituting in newer models, such as a Capsule Neural Network and Hidden Markov Model, could result in better performance

5) Detailed Description of Contributions

5) Data Contribution:

5.1) Methods: Getting a large dataset of handwritten math formulae organically is not feasible, and would cause issues with testing evaluation. For a given datapoint, we would need an image of the handwritten formula as well as the latex formatting that exactly fits the intended formula. Without access to a pre-existing dataset meeting these conditions, it doesn’t seem feasible for a group of 3 students to produce a sufficiently large dataset without paying people to produce some handwritten math formulae and corresponding latex. A sound compromise would be to use a dataset of handwritten math symbols and systematically convert LaTeX formulae into pseudo-handwritten images using these symbols. We can simulate human handwriting by

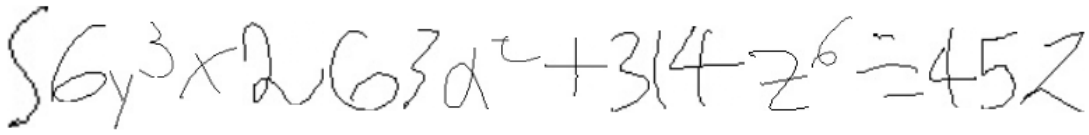
randomly selecting one of the handwritten symbols k_i each time a symbol k appears in a formula.

Originally, we intended to create a randomized font using the handwritten math symbol dataset we found, then type up all of the LaTeX strings in the im2LaTeX100k dataset and render them using our new font. This turned out to be infeasible, as any font-making software available for a reasonable price did not allow fonts to include some of the symbols we needed (this includes many characters like +, -, =, …), and the handwritten math symbol dataset we had did not include many of the more advanced characters found in im2LaTeX100k.

Instead, we created a semi-random method of generating math formulae that included only symbols we had a sufficient number of handwritten samples for, and produced the corresponding "handwritten" image. In some ways, producing images of math formulae in this way gives us a dataset with more variance than a handwritten image would, as the variance in a given symbol drawn twice by the same person likely is smaller than the difference between some of the symbols in the Kaggle dataset, which were produced by multiple different people. Because of this, we suspect that any model that performs well on our produced dataset would perform even better on an actual handwritten dataset.

5.2) Results: We have successfully created an algorithm in Python that parses LaTeX code and generates a grayscale image output of dimensions 1024 x 128 by randomly picking symbols from the Kaggle dataset. Since all of the handwritten symbols are 90x90 pixels, simply placing the 90x90 squares creates excessive whitespace. To avoid these issues, our algorithm takes the input image, crops it, and then inserts it into the generated image. The algorithm slightly varies the positioning of the symbols to avoid placing symbols perfectly in line with each other. These perturbations ensure that any algorithm that is trained on the data is robust to minor amounts of noise. Below is a sample image that was generated from the LaTeX code

$$\int 6y\text{^}3 \times 263\alpha\text{^}2 + 314z\text{^}6 \div 452$$



As can be seen from the sample image output, our program is able to generate images that look like they could be handwritten. Such data is labeled and ML-ready. Since we produced all of these images with machine learning in mind, we didn't really need to do any preprocessing before training our models; the preprocessing is essentially handled in the data generation.

Our full dataset of 100,000 equations and expressions can be found here: https://drive.google.com/file/d/1PKxqr4lHdwPoQ52suzszlRj_PDP2Lkep/view?usp=sharing

Each image in the dataset is named by its corresponding LaTeX code, with all backslashes being replaced with ampersands to avoid file naming issues in Windows. (For example, the image above would be $\&int\ 6y\text{^}3\ \&times\ 263\&alpha\text{^}2 + 314z\text{^}6\ \&div\ 452.png$)

6) Algorithm Contribution:

6.1) Methods: The Texas A&M paper uses a CNN model. While CNN models typically do well with problems involving computer vision, there are some drawbacks to CNNs. For example, CNNs may not be very robust to noise and non-translational transformations [More et. al.]. That paper analyzes a model called “capsule networks” that is shown to work better on MNIST data. There is also literature that suggests that combining a CNN architecture with different models such as SVM can improve classification accuracy of Arabic handwriting [Ali et. al.].

Similarly, the Texas A&M paper uses a combination of two models: The visual information is first encoded into a more linear, one dimensional representation of the image. The encoding is then translated into LaTeX commands through a decoder that sequentially processes the encoded information. This two part approach was shown to produce very good results because it utilized the strengths of each model. We tested whether performance can be improved by adjusting either the encoding model or the decoding model. We will thus experiment with the following three models:

Model 1: CNN + LSTM (Baseline)

We attempted to replicate the Texas A&M model, which we will use as a baseline. This model consists of a CNN as the encoder and a LSTM as the decoder.

Model 2: Capsule Neural Network + LSTM

Capsule Neural networks are created by adding “capsules” to convolutional neural networks. These “capsules” can be trained to only activate under certain conditions, making individual neurons more independent. Capsule neural networks have been shown to offer improvements over CNNs in tasks such as computer vision (Vijayakumar).

The first model we propose uses a capsule neural network as an encoder, and uses the same LSTM as a decoder.

Model 3: CNN + HMM

Hidden Markov Models are probabilistic graphical models that predict the current state based on previous observations. Since language, including LaTeX, is generally sequential from left to right, Hidden Markov Models work well for translation tasks. They have been shown to outperform LSTMs in translation tasks (Chang).

The second model we propose uses the same CNN encoder as the baseline model, and uses a Hidden Markov Model as a decoder.

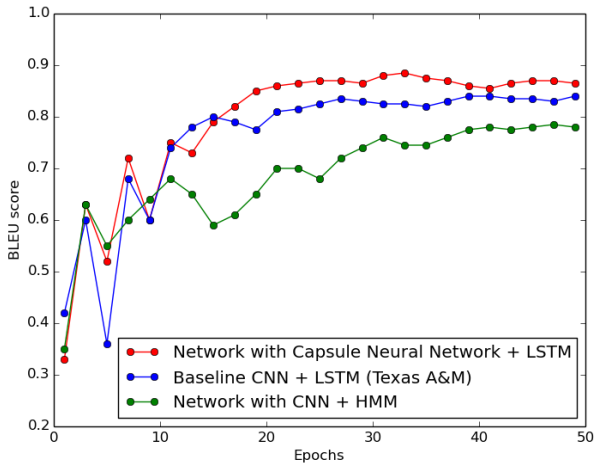
We plan to use an evaluation metric based on the BLEU score (which is used to evaluate language translations) to evaluate the performance of our network. This is the metric used in the Texas A&M paper and was shown to outperform other training metrics. As a more concrete evaluation metric, we will also use “accuracy”, which we define as the Hamming distance between the actual LaTeX command and the predicted command. Since LaTeX is in most cases indifferent about spaces, we ignored spaces when calculating the accuracy.

6.1.3) Training

We split the dataset into approximately 80% train, and 20% test. We augmented our training dataset by applying transforms, such as minor translations, rotations, and dilations. We used a modified BLEU score for the training loss. We used Adam as our optimizer, and after testing a few different hyperparameter settings, we found the most success with a learning rate of 0.001 and a weight decay of 0.0001. We found that our models typically stopped improving after around 50 epochs, which is how long we trained them for.

6.2) Results:

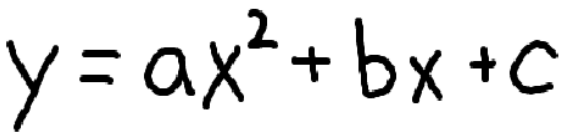
Below is a plot of the BLEU score during the training of the three models:



The BLEU scores and accuracy of the models on the test dataset are as follows:

Model	BLEU score	Accuracy
Baseline CNN + LSTM (Texas A&M)	0.80	0.77
Capsule Neural Network + LSTM	0.83	0.81
CNN + HMM	0.74	0.72

We then drew the following sample image and inputted it into all three trained models:



Although none of our models had not been trained with our handwriting, they were all able to correctly predict the LaTeX sequence $y = ax\text{^}2 + bx + c$

7) Compute/Other Resources Used

All computational work for this project was done locally, without the use of resources like AWS. While we were given access to AWS credits, we had some difficulty figuring out how to get it to work properly. Since we had relatively decent hardware and were able to generate a reasonable number of data points and train our models locally, this was a fine solution for the purposes of this project.

8) Conclusions

The outcomes of this project were two-fold. We generated a dataset of over 100 thousand images containing handwritten math formulae that could be translated to LaTeX. Additionally, we created two new machine learning models and compared them to a baseline model. These models were all trained with images from the dataset we generated. In our testing, we evaluated our models through two accuracy methodologies, and we were impressed by the performance of all three models, especially our model consisting of a capsule neural network that performed better than the original Texas A&M model. This semester-long project was a great introduction to machine learning for all of us, and we genuinely enjoyed going through the process of creating our own data. Going through and training, testing, and evaluating our model with our accuracy methodologies, and then forming conclusions based on the results was also very rewarding. The whole process was a valuable learning experience for everyone, and we do believe that our project produced something that others may find valuable because we're contributing to the demystification of LaTeX.

In hindsight, the project has evolved greatly from our initial conceptual idea. At first, we were invested in creating a user interface that would allow users to submit their own images and have their LaTeX translated. However, based on the feedback given to us, we decided against this direction, instead focusing on the machine learning and research aspect of the project. Looking back, this change allowed us to focus more on what we enjoyed learning about. Perhaps someone looking to extend our work could build an interface, and our dataset could be improved through user-submitted images. One could also use more symbols to generate equations. One major roadblock we encountered was debugging our models in general. Given that machine learning is complex and many of the models we used are fairly new, we struggled significantly to understand the research and actual programming. Another challenge we encountered during this project that seems to be fundamental to any research dealing with handwriting is generating usable data. While we were able to come up with a way to algorithmically produce pseudo-handwritten images, the data was not actually written by a human. The process of collecting large handwritten datasets and their LaTeX markups remains something that requires a lot of time and money to do. I would be curious to see how results would have varied if we instead used 100 thousand actual handwritten math equations. Similar to what we mentioned in our previous report, our project does not necessitate significant resources. As a result, there are no ethical considerations. For a broader social impact, we only hope that our work can impact people who are trying to learn LaTeX in a positive way.

Other Prior Work / References cited in the text:

1. Ali, Mallaiah, “Intelligent handwritten recognition using hybrid CNN architectures based-SVM classifier with dropout”, https://www.sciencedirect.com/science/article/pii/S1319157821000148
2. Chang, Yunpeng, et al. “Improving Language Translation Using the Hidden Markov Model.” Computers, Materials & Continua, vol. 67, no. 3, 2021, pp. 3921–3931., https://doi.org/10.32604/cmc.2021.012304.
3. More, Shirodkar, Joshi, Thakur, “Overcoming the Drawbacks of Convolutional Neural Network Using Capsule Network”, https://www.iosrjournals.org/iosr-jce/papers/Vol21-issue2/Series-3/B2102030611.pdf
4. T, Vijayakumar. “Comparative Study of Capsule Neural Network in Various Applications.” Journal of Artificial Intelligence and Capsule Networks, vol. 01, no. 01, 2019, pp. 19–27., https://doi.org/10.36548/jaicn.2019.1.003.