Translating Image to LaTeX with Adaptive Attention

Jiancong Gao
Georgia Institute of Technology
North Avenue, Atlanta, GA 30332
jgao320@gatech.edu

Qihang Zhang
Georgia Institute of Technology
North Avenue, Atlanta, GA 30332
qihang.zhang@gatech.edu

Abstract

*LaTeX has become the most popular typesetting system in both scientific and industrial fields with its ability to generate beautiful mathematical equations. However, once an equation is rendered, the output cannot be modified without access to the underlying code, and it is very time consuming and error-prone for someone to re-type lengthy equations by just looking at the image. To improve working efficiency, our group believes that it is worth studying to translate mathematical equations directly from image to LaTeX code. As a sub-problem under image captioning, we decided to solve the problem with an encoder-decoder model framework [12, 6], started with the model structure proposed by [5], and implemented adaptive attention mechanism [7] as our new approach. Thus, our model consists of a Convolutional Neural Network (CNN) encoder, a Long Short Term Memory (LSTM) decoder with adaptive attention which knows for each time step whether it is worth looking at the image. Our model significantly outperformed the existing models [6, 5] with a BLEU score of **88%** and an image edit distance of **78%**.*

1. Introduction/Background/Motivation

1.1. Introduction

Nowadays, LaTeX has been the most popular tools for researchers to compose scientific research papers. However, it has always been a cumbersome task for not only starters but also experienced users to type in mathematical formulas by hand. The difficulties come from both long expressions that require writers to type in correctly and complex syntax of math formula expression. In this project, we aim to create an automated process which translates images of formulas into LaTeX code for the user by constructing an encoder-decoder framework with adaptive attention implementation [7]. It has already been proven by [6, 5] that a model with attention based decoder gives decent performance in image-LaTeX translation tasks. However, intuitively speaking, it is not necessary for the model to look for visual signals while translating some specific characters. Researchers have already studied this problem and in 2017, [7] introduced adaptive attention mechanism which allows the model to decide whether to look for image information at each time step. In this project, we aim to build a new image-LaTeX translation model with adaptive attention based decoder. In the next subsection, we are going to give a brief introduction on existing works and the data set we used in this project.

1.2. Background

Automatic recognition of mathematical expressions is one of the key components in converting documents in scientific and engineering disciplines into electronic form. As mentioned in [4], this problem typically consists of two major stages, symbol recognition and structural analysis. The mathematics-recognition problem was first defined in [3] follower by a survey of existing work according to the two major sub-parts of the problem mentioned above. In terms of symbol recognition, although many existing symbol recognition techniques have shown decent performance in various applications, many of them can only work with isolated symbols. In a mathematical expression, there usually exist multiple symbols, e.g., sub and superscript notations, special symbols, and nested fractions. In order to apply symbol recognition techniques, we must first segment each symbol from the expression properly [4]. For structural analysis, a list symbols with associated attributes, such as the location, size, and identity, are needed. Then on can build a hierarchical structure (a parse tree or a relation tree) for the objects to perform the analysis [4]. To combine the two sub-parts together and perform mathematics-recognition, [4] and [2] have proposed with syntactic-based approaches, while [9] has managed to build systems by exploiting character segmentation and grammar reconstruction. As mentioned in [6], a prime example of the approach in [9] is the INFTY system introduced by [11], which can convert printed mathematical expressions to LaTeX and other markup formats.

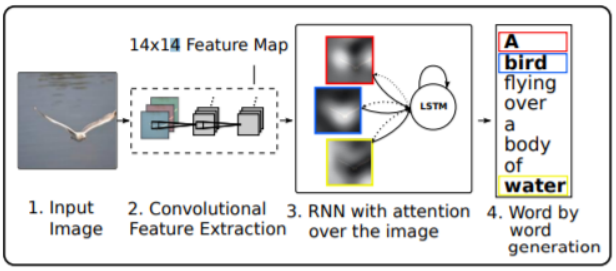
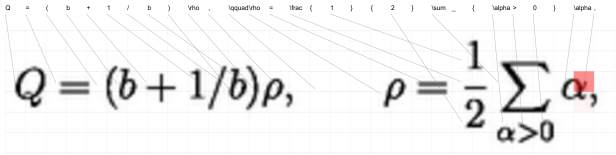


Figure 1. Learning process of the encoder-decoder model from [12]



$$Q = (b + 1/b)\rho, \qquad \rho = \frac{1}{2}\sum_{\alpha>0}\alpha,$$

Figure 2. Example output given by [6] with the generated LaTeX code on the top of the image

To consider this problem from a macro perspective, mathematics-recognition is a combined problem of both Computer Vision (CV) and Natural Language Processing (NLP), as it requires joint processing of both image and text data. In the past few years, both domains have draw more and more attentions with the huge progress made in deep learning. In NLP, sequence-to-sequence models based on recurrent neural networks (RNNs) have outperformed other models in machine translation [10]. By adding attention [1], the combined model has been served as a new standard in Machine Translation systems.

Combining sequence-to-sequence with Image Captioning techniques, [12] encoded the image in a fixed size vector with a Convolutional Neural Network (CNN), and then decoded the vector step by step. At each step, a new caption word is generated and then fed as the input to the next step. Furthermore, an attention mechanism was also included to enable the decoder to look and attend at the encoded image at each time step, thus compute a representation of this image with respect to the current state of the decoder. An example of the process is shown in Figure 1.

In a recent work by [6], the authors took this same approach and successfully generated LaTeX code from images of formulas. Same soft attention mechanism was applied as in [8], and [6] achieved 77% of exact match score on the test dataset. An example output of their model is shown in Figure 2.

Although [6] has generated pretty satisfying results, there is still room to improve the performance. In 2017, [7] introduced a new attention technique named adaptive attention. Unlike the previous approach that visual attention is activated for every single word, the authors suggested a new thinking that some words might not need visual information to make predictions. For example, non-visual word such as 'a' and 'of' would not require visual information, and visual word such as 'phone' after 'cell' could be predicted from the language model without visual information as well. [7] then presented a new adaptive attention model with a visual sentinel. At each time step, the model will determine whether attend to the image or to the sentinel, and thus, only achieve useful information from the image to generate word sequences. Figure 3 illustrates how the proposed model learn when to attend to the image and when to attend to the sentinel. We believe that it is worth trying to implement the adaptive attention technique to the image-LaTeX translating task because visual information is not always a must to generate LaTeX sequences. For example, we can purely learn from the language model that a '\frac' must be followed by a '{}{}'. In the next section, we will briefly introduce the dataset we used in this project and the preprocessing steps we have done.

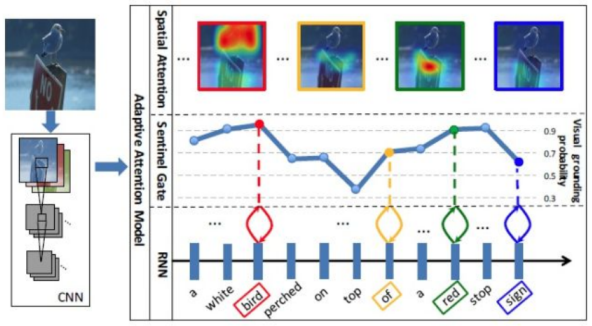


Figure 3. Image from [7] that illustrate the principle of the adaptive attention model.

1.3. Who Cares?

With our model, researchers and students can work more efficiently by saving tons of time from re-coding lengthy equations from text books or websites, and put more energy into research and study. Looking at the bigger picture, image-LaTeX translation is a sub-problem under image captioning. If our adaptive attention based model works well on the LaTeX dataset, we are confident to say that our model can be widely applied to other image-captioning applications with minor modification.

1.4. Data

The dataset we used in this project is named IM2LATEX-100K and created by [6]. This dataset contains a large corpus of real-world mathematical expressions written in LaTeX. The dataset provides 103,556 different LaTeX math equations along with rendered pictures and images that are collected from arXiv. The dataset is also split into train (83,883 equations), validation (9,399 equations) and test (10,354 equations) sets.

In the dataset, each line indicates a separate generated image per formula, including the formula index, image name, and the render type. Before we can input the data for training, it is necessary to perform data preprocessing. We performed similar data preprocessing steps as [6] and [5] that formulas were rendered with *pdflatex* and the images were converted to PNG format with *ImageMagick*. For images, parameters were set as 200 and 100 for density and quality respectively. Moreover, we down sampled images by a factor 2 and used greyscale representation to speed up the process.

For formulas, we tokenized the formulas as it has been observed in [6] that token-based methods are more efficient than character-based. One thing to notice is that one mathematics equation can be expressed in different styles in LaTeX, e.g., $x\_\{0\}$ and x_0 would give the same result x_0. Thus, we further normalized the formulas with KaTeX after we have tokenized the formulas. After the formulas were tokenized, we added four more tokens "PAD" (padding), "START", "END", and "UNK" (unknown) to our vocabulary dictionary. The size of our finalized vocabulary is 509. Padding technique were then performed with the " PAD" token to ensure that all equations have the same length. The maximum length of a formula was set to 150. With the previous step completed, the data is ready to be fit into the network for training.

2. Approach

In this section, we are going to give details about our model. Our model is built based from [5] with Encoder-Decoder structure and we update the Decoder part with an Adaptive Attention introduced by [7]. The main encoder-decoder framework [1] would directly maximizes the following objective:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{I,y} \log p(\mathbf{y}|I, \theta), \tag{1}$$

where I indicates the image, θ is the model parameter, and $\mathbf{y}$ is the corresponding caption.

Past soft attention [12] focus on a weighted sum of information from the image, however we believe that for some token, the contextual information in the memory is same or even more important. This is the basic idea of adaptive attention. For our image-to-latex task, we think this method would be successful, because for LaTeX code generation, it is not necessary to look at the image for information as some LaTeX expressions follow particular patterns, e.g. \mathbb would always be followed by {}.

[1] Code of this part is forked from https://guillaumegenthial.github.io/image-to-latex.html

2.1. Encoder

Since we did not see a better structure to replace the existing ones, we applied the same structure as [5], where we encoded the images with a convolution neural network (CNN). The CNN consists of 6 convolution layers. Each convolution layer has filters of size 3x3, stride of 1 and padding of 1. Max pooling layers were also added in order to extract structure from the characters. Table 1 shows the structure of our encoder. Assume the original image has size H x W, the encoder map the image to a new feature space of size H' x W' x C where C is the number of filters in the last convolution layer. Thus, the original image is split into C regions where each region has a size of H' x W'. In other words, we have obtained $V = [v_1, , v_2, \ldots, v_c]$, where each v_i has size H'xW'.

Input
CNN 3x3 -64 Stride 1 Padding 1
CNN 3x3 -128 Stride 1 Padding 1
CNN 3x3 -256 Stride 1 Padding 1
CNN 3x3 -256 Stride 1 Padding 1
Max-pool (2,1)
CNN 3x3 -512 Stride 1 Padding 1
Max-pool (1,2)
CNN 3x3 -512 Stride 1 Padding 1
Output

Table 1. Structure of the Convolution encoder

2.2. Decoder With Adaptive Attention

After we have obtained the feature map, we plan to decode this new image representation to generate the corresponding LaTeX tokens. To complete this task, we decided to apply a LSTM network as it has shown to be very efficient to capture long term dependencies and handle the exploding/vanishing gradient problem. To be more specific,

$$h_t = \text{LSTM}(x_t, h_{t-1}, m_{t-1}), \tag{2}$$

where h_t is the hidden state of the network at time t, and x_t is the input vector, and m_{t-1} is the memory cell at time $t-1$. With the hidden state h_t and the image vector $\mathbf{V}$, we then need to compute a context vector $\mathbf{c}_t$ as

$$\mathbf{c}_t = g(\mathbf{V}, h_t), \tag{3}$$

where g is the attention function. To be more specific, $\mathbf{V}$ and $\mathbf{h}_t$ are both fed into a single layer neural network followed by a softmax function. The attention distribution over the k regions of the image is computed as follow

$$z_t = w_h^T \tanh(\mathbf{W}_v \mathbf{V} + \mathbf{W}_g \mathbf{h}_t) \quad (4)$$

$$\alpha_t = \text{softmax}(z_t) \quad (5)$$

$$c_t = \sum_{i=1}^{k} \alpha_{ti} v_{ti} \quad (6)$$

where $\mathbf{W}_v$ and $\mathbf{W}_g$ are trainable weights and α is the attention weight over features in $\mathbf{V}$.

Unlike traditional spatial attention model, our decoder implements adaptive attention model which is able to determine whether it needs to attend the image to predict next word with a newly introduced variable $\mathbf{s}_t$ named '**visual sentinel**' [7]. $\mathbf{s}_t$ is computed as follow

$$g_t = \sigma(W_x x_t + W_h h_{t-1}) \quad (7)$$

$$\mathbf{s}_t = g_t \odot \tanh(m_t) \quad (8)$$

where g_t is the gate applied to the memory cell m_t, W_x and W_h are trainable weights, x_t is the input at time t, σ is the logistic sigmoid function, and $\odot$ indicates element wise product. Thus, the new context vector $\hat{\mathbf{c}}_t$ is now computed as

$$\hat{\mathbf{c}}_t = \beta_t \mathbf{s}_t + (1 - \beta_t)\mathbf{c}_t, \quad (9)$$

where β_t is the new sentinel gate at time t range from [0,1]. To compute β_t, a new element is added to z, which indicates how much "attention" the network is placing on the sentinel (as opposed to the image features) to obtain a new $\hat{\alpha}_t$

$$\hat{\alpha}_t = \text{softmax}(z_t; w_h^T \tanh(\mathbf{W}_s \mathbf{s}_t + \mathbf{W}_g \mathbf{h}_t)) \quad (10)$$

where $[\cdot;\cdot]$ simply means concatenation. Thus, our new $\hat{\alpha}_t$ has size $k+1$, and $\beta_t = \hat{\alpha}_t[k+1]$. With $\hat{\mathbf{c}}_t$, the model can adaptively attend to either the image or the visual sentinel when generating the next word while the sentinel vector is updated at each time step.

2.3. Beam Search

At each time step, the LSTM produces a vector $\mathbf{y}_t \in R^V$, where V is the size of the vocabulary, with the distribution probability over the vocabulary for the next word. To determine which token to become the output and feed to the next step, we choose to perform beam search as it has shown more robust performance than the greedy approach. This part of the model was also implemented based on the code from [5]. At each time step, we keep the top K (beam size) tokens with the highest probabilities. While we input the K tokens to the decoder and compute the next K vector $\mathbf{y}_t$, we keep track of the "parents" of the tokens. Then, we calculate the conditional probability of each combination and choose the sequence with the maximum result. Thus, we can reconstruct the optimal sentence at the end of the decoding process and output our best prediction.

2.4. Problems Expected & Encountered

Actually when we start this project, we believed this is a very challenging task and expected many problems. For example, we are not sure how to evaluate the result, how to group different-size input image and how to parse the math equation and so on. Luckily most of the above problems are solved by previous work of [5] and [6].

In our own implementation and experimentation, however, we encounter some other issues unexpected at the first time. We met an OOM(Out-of-memory) issue when training on GPU and finally have to skip some large image to avoid that. Also we met issues where generated Latex code will cause error when generating image. After our hard work, issues are solved and our results are very exciting.

3. Experiments and Results

3.1. Training and Hyperparameters

The experiment was started with similar settings from [6] and [5] with the following hyperparameters:

- Embedding Size: 80
- LSTM Dimension: 512
- Max Length: 150
- Batch Size: 20
- Beam Size: 5
- Learning Rate: [1e-3 - 1e-5]

As mentioned above, the whole training set was trained with 15 epoches and a batch size of 20. Our learning rate was not fixed during the entire training. For the first two epochs, we chose a small warm-up learning rate of 1e-4. Then, we set the learning rate to be 1e-3 from the third epoch to the 10^{th} epoch. Finally, we started to decay the learning rate exponentially from 1e-4 to 1e-5 until the end of the 15^{th} epoch. Last but not least, we set a max size for images to exclude too large images in the training set as we encountered OOM problem at the beginning of our experiment.

3.2. Evaluation

To evaluate the result of our model, we look at this question from two aspects: how well is the LaTeX code reproduced and how close is the reconstructed image to the original one.

From the text perspective, three popular evaluation metrics in NLP were used to evaluate our model, which are

Decoder	EM (I)	BLEU	ED (T)	ED (I)
CNNEnc [6]	0.53	0.75	NR	0.61
Greedy [5]	0.22	0.76	0.76	0.35
Beam [5]	0.32	0.78	0.76	0.62
Our Model	**0.55**	**0.88**	**0.91**	**0.78**

Table 2. Result comparison of our model to the two models purposed in [5] (One with beam search the other with greedy search). EM stands for exact match, ED means edit distance, T and I are text based and image based

exact match, **edit distance for text**, perplexity and **BLEU score**. Those four evaluation metrics are also used in [5], so we decide to use the same metrics for comparison purpose. Exact match measures the quantity of same tokens (same order, same number and same nature) between a predicted formula and the ground truth. BLEU score is a measurement that captures the overlap of n-grams and has been considered as one of the standard metrics in translation (1 is perfect). The perplexity of a language tries to capture the level of confidence of the model while predicting one token, and is computed as

$$\text{perplexity} = e^{-\frac{1}{N}\sum_{i=1}^{N}\log(x_i)} \tag{11}$$

where N is the vocabulary size and x_i is the predicted probability of the true token i. We used perplexity to evaluate our training and validation performances. Edit distance on the text (Levenshtein distance), which compute how many steps (add/remove/change) we have to make to transform the predicted formula to the ground truth. In our experiment, we consider the edit distance as the percentage of the reconstructed text that matches the original. Thus, a perfect match has an edit distance of 1.

From the image perspective, we have decided to use the **edit distance for images** (Levenshtein distance for images). The image is split into several columns and each column is encoded with an integer. Then, the number of columns with the same integer will be counted, and the percentage of matching columns will be calculated, with 1 as a perfect match.

3.3. Results

Quantitative Result The evaluation scores of our model is listed in Table 2 along with the scores from [5] for comparison purpose. We can clearly see that our model has outperformed the models of [5, 6] with the highest scores for all the evaluation metrics. This result shows that we successfully implemented the adaptive attention based decoder. Besides comparing the evaluation scores, we also looked at the perplexities of the training and validation of each epoch. The perplexity curve is shown in Figure 4. From the perplexity curve, we conclude that our model was well trained and did not overfit.

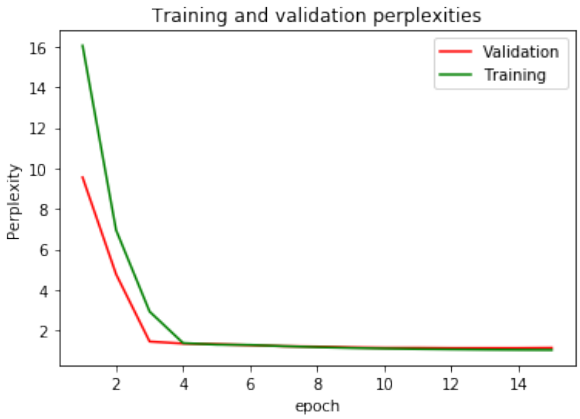


Figure 4. Perplexity Result through training

$$\hat{B}_{(n)} = \sum_{r,s} b_{(n)}^{(r,s)} \hat{a}^{\dagger r} \hat{a}^{s}.$$

Figure 5. Original Formula

$$\hat{B}_{(n)} = \sum_{r,s} b_{(n)}^{(r,s)} \hat{a}^{\dagger} \hat{a}^{s}.$$

Figure 6. Generated Output

Figure 7. Example of Model Output (False Exact Match)

$$I_{J-1} \equiv \int \left[\frac{du}{u}\right]^{J-1} 1 = i^{J-1} \int_0^{\pi} d^{J-1}\theta = \frac{(\pi i)^{J-1}}{(J-1)!},$$

Figure 8. Original Formula

$$I_{J-1} \equiv \int \left[\frac{du}{u}\right]^{J-1} 1 = i^{J-1} \int_0^{\pi} d^{J-1}\theta = \frac{(\pi i)^{J-1}}{(J-1)!},$$

Figure 9. Generated Output

Figure 10. Example of Model Output (True Exact Match)

Qualitative Result We generate 9098 valid math formula out of 9442 test cases and here we can show some example.

3.4. Conclusion & Future Work

In this project, we developed an encoder-decoder translation model with adaptive attention mechanism. Our model shows satisfactory results comparing to the models introduced by [6, 5]. The success of our work indicates that adaptive attention is a promising technique to solve image-captioning problems. In the future, more work can be done on refining the structures of the encoder and the decoder and further fine tuning of the hyperparameters. Furthermore, other dataset can also be applied to the model to test the performance.

4. Work Division

Both team members have contributed equally to this project. Details of what each individual has contributed are shown in Table 3.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 2

[2] Abdelwaheb Belaid and Jean-Paul Haton. A syntactic approach for handwritten mathematical formula recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1):105–111, 1984. 1

[3] Dorothea Blostein and Ann Grbavec. Recognition of mathematical notation. In *Handbook of character recognition and document image analysis*, pages 557–582. World Scientific, 1997. 1

[4] Kam-Fai Chan and Dit-Yan Yeung. Mathematical expression recognition: a survey. *International Journal on Document Analysis and Recognition*, 3(1):3–15, 2000. 1

[5] Guillaume Genthial. Seq2seq for latex generation https://guillaumegenthial.github.io/image-to-latex.html, Nov 2017. 1, 3, 4, 5

[6] Morten Hertzum, Rolf Molich, and Niels Ebbe Jacobsen. What you get is what you see: revisiting the evaluator effect in usability tests. *Behaviour & Information Technology*, 33(2):144–162, 2014. 1, 2, 3, 4, 5

[7] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 375–383, 2017. 1, 2, 3, 4

[8] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015. 2

[9] Erik G Miller and Paul A Viola. Ambiguity and constraint in mathematical expression recognition. In *AAAI/IAAI*, pages 784–791, 1998. 1

[10] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. 2

[11] Masakazu Suzuki, Fumikazu Tamari, Ryoji Fukuda, Seiichi Uchida, and Toshihiro Kanahori. Infty: an integrated ocr system for mathematical documents. In *Proceedings of the 2003 ACM symposium on Document engineering*, pages 95–104, 2003. 1

[12] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015. 1, 2, 3

Student Name	Contributed Aspects	Details
Jiancong Gao	Data Preparation, Implementation, and Analysis	Preprocessed the dataset and trained the model.
Qihang Zhang	Data Preparation, Implementation, and Analysis	Helped train the model and analyzed the results.

Table 3. Contributions of team members.