Image To LaTeX: A Neural Network Approach

Zhengqing Zhou [*1] **Junwen Zheng** [*2] **Zhongnan Hu** [*3]

Abstract

Formulas are essential gradients in most of the research articles. Among various of documentation written software, LaTeX provides several options to handle images and make them look exactly what you need. Hence it would be exciting if we could build an OCR that converting image of formula to latex code. To construct and train such a model requires techniques in Computer Vision and Neural Language Processing. In this project, we addressed this task with a Densely Connected Network encoder, an RNN decoder with gated recurrent unit (GRU) and an attention model. This model is trained on a open source dataset called IM2LATEX-100K. We evaluate our model against several metrics, and compiling the predicted code to generate new formulas and comparing them to the ground truth. We achieve very good performance with a BLEU score of 84% and an Edit distance of 85%.

1. Introduction

Consider the situation that you are writing a paper and you want to quote some math formulas from other papers, it would be quite time-consuming if you write the LaTeX code of the formulas by hand. Motivated by this scenery, a very natural question is that if there is way to convert math formula images to LaTeX code. This lead us to the area of image captioning, which is the process of generating textual description of an image. Optical character recognition (OCR) is most commonly used to recognize natural language from an image. Problems like OCR that require joint processing of image and textual data have recently seen increased research interest due to the development of deep learning in these two domains. For instance, recent advances have been made in the areas of handwriting recognition (Jaderberg et al., 2014), OCR in natural scenes (Ciresan et al., 2010), (Wang et al.) and image caption generation (Karpathy & Li, 2014).

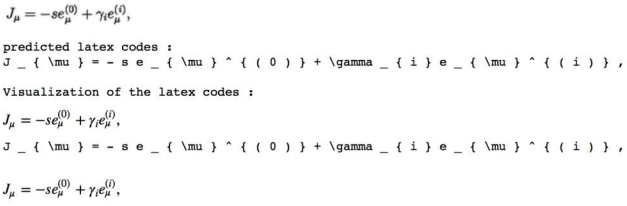

Figure 1. Example of input image and output LaTeX codes

In this project, our main focus is to build an OCR for math expressions. Unlike the traditional approaches that assumed expertise in the language from the images, we are seeking for a supervised model that can learn to produce correct LaTeX code from an image, without requiring any knowledge of the underlying language, and is simply trained end-to-end on real-world data.

2. Related Work

Recent years, many breakthrough has been made in Computer Vision and Neural Language Processing. For instance, papers like (Karpathy & Li, 2014) and (Karpathy et al., 2015) proposed data driven approach to learn an encoded version of the input image which is then decoded to generate a textual output. Those promising data-driven methods gained can be applied to a wide range of datasets, since they do not require heavily preprocessing inputs or professional knowledges of a specific domain.

When narrowing down to the problem of "Image to Latex", it is important to combine sequence-to-sequence model with the techniques in Image Captioning. One exited breakthrough has been made by (Xu et al., 2015), they encoded the image with a CNN, and then decoding the CNN output step by step. At each step, they generated a new word of caption and then fed it to the next step. Another enlightening technique was that they introduced an attention model, which enables the decoder at each step to take a look at the encoded image.

[*]Equal contribution [1]Department of Mathematics, Stanford University, CA, USA [2]Department of Civil and Environmental Engineering, Stanford University, CA, USA [3]Department of Mechanical Engineering, Stanford University, CA, USA. Correspondence to: Zhengqing Zhou <zqzhou@stanford.edu>, Junwen Zheng <junwenz@stanford.edu>, Zhongnan Hu <chrisnan@stanford.edu>.

Finally, a recent work from (Deng et al., 2016) took the similar approach as (Xu et al., 2015) and successfully applied it to the generation of Latex code from images of formulas. They built a nice dataset IM2LATEX-100K (Kanervisto, 2016) and achieve a very promising performance in test results. Our work is based on their paper, and we were trying to change the architecture of CNN encoder and RNN decoder so as to achieve a better performance.

3. Data Set and Features

3.1. Raw Data

We used an open source dataset called IM2LATEX-100K (Kanervisto, 2016) as our raw dataset. This is a prebuilt dataset with a lots of real-world images of mathematical formulas in LaTeX for OpenAI's task for "Image to Latex system". Specifically, the IM2LATEX-100K dataset provides 103,556 different LaTeX math equations along with rendered pictures. The formulas were extracted from papers in arXiv, which was a repository of electronic preprints approved for publication after moderation. LaTeX sources were obtained from tasks 60,000 papers. The around 100 thousand formulas were rendered in a vanilla LaTeX environment. Rendering was fulfilled with pdflatex and formulas that failed to compile were excluded. Then the rendered PDF files were converted to PNG format. The raw dataset was separated into training set (83,883 equations), validation set (9,319 equations) and test set (10,354 equations) for standardized experiment setup.

3.2. Data Preprocessing

The images in the raw dataset IM2LATEX-100K (Kanervisto, 2016) contain a LaTeX formula rendered on a full page. To accelerate training, we need to pre-process the images. Firstly, we cropped the formula area, and group images of similar sizes to facilitate batching. The bucket sizes were (160, 64), (224, 32), (320, 64), (128, 64),(160, 32), (480, 64), (384, 64), (224, 64), (128, 32),(480, 32), (384, 96),(384, 32), (192, 32), (320, 32), (256, 32), (256, 64), (192, 64), (480, 128), (480, 160). Then we prepared train, validation and test files. We excluded large images from training and validation set, and we also ignored formulas with too many tokens or formulas with grammar errors.

After that, we tokenized the formulas into vocabulary dictionary. We also added the special tokens PAD, START, END, and UNK. PAD token was added to pad the formulas to have the maximum lenght of 150 to faciliate training.

4. Methods

Our model combined several components from computer vision and natural language processing. First of all it extracted formula images by using a convolutional network, and then encoded the output of convolutional network by a RNN. Finally, it fed the encoded feature to a RNN decoder with a visual attention mechanism.

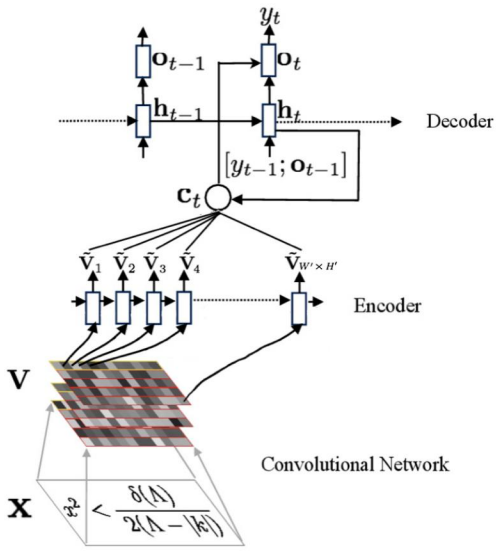

Figure 2. Network Architecture

4.1. Convolutional Network

4.1.1. Baseline CNN Model

For the baseline model, we aimed to reproduce the model in (Deng et al., 2016). The visual features of an image were extracted with a multi-layer convolutional neural network interleaved with max-pooling layers. We did not use final fully-connected layers, since we wanted to preserve the locality of CNN features in order to use visual attention. The CNN took the raw input $R^{H \times W}$ and produced a feature grid V of size $D \times H_0 \times W_0$, where c denotes the number of channels and H_0 and W_0 were the reduced sizes from pooling. More details of the CNN architecture are shown in Table 1.

Type	Filter Size	# channels	Stride	Padding	Batch Normal
Convolution 1	3x3	64	(1, 1)	SAME	No
Max pool 2	2x2	64	(2, 2)	VALID	No
Convolution 2	3x3	128	(1, 1)	SAME	No
Max pool 2	2x2	128	(2, 2)	VALID	No
Convolution 3	3x3	256	(1, 1)	SAME	Yes
Convolution 4	3x3	256	(1, 1)	SAME	No
Max pool 3	1x2	256	(1, 2)	VALID	No
Convolution 5	3x3	512	(1, 1)	SAME	Yes
Max pool 4	2x1	256	(2, 1)	VALID	No
Convolution 6	3x3	1024	(1, 1)	SAME	Yes

Table 1. Baseline CNN Architecture

4.1.2. DenseNet model

DenseNet (Huang et al., 2016) is a network that connects each layer to every other layer. For each layer, we used the

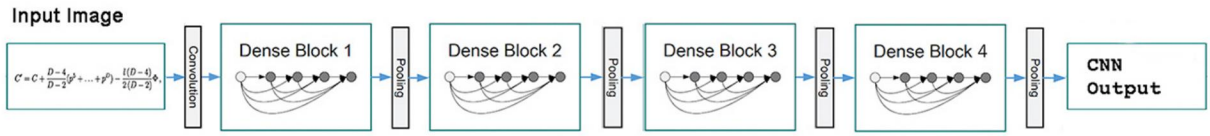

Figure 3. Deep DenseNet with four dense blocks

feature maps of all previous layers as inputs, and we also used its own feature maps as inputs of all subsequent layers. One motivation of introducing DenseNet was that we found it hard for the Baseline CNN model to distinguish the difference of some tiny symbols (say subscript or superscript, see Figure 4). DenseNet have the following advantages:

(1) Strengthen the feature propagation.

(2) Alleviate the vanishing gradient problem.

(3) Substantially reduce the number of parameters.

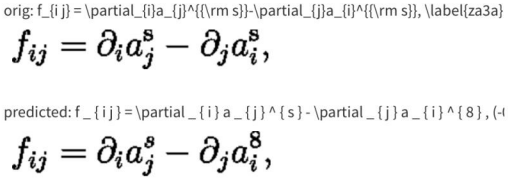

Figure 4. Fail to distinguish superscript s and 8

There are two substantial gradients in DenseNet model. The first one is called the "Dense Block", there are four bottleneck layers in a "Dense Block", each layer took all preceding feature maps as input and concatenated into a tensor. The second gradient is called the "Transition Layer", which are in between two adjacent dense blocks. In this layer, we applied Batch Normalization on the output of the preceding "Dense Block", and then we applied Max-Poolings to obtain the input of the subsequent "Dense Block". In (Huang et al., 2016), researchers also compressed feature-map via convolution in the "Transition Layer". However, due to small model size, we don't have to do this. The network structure of DenseNet is shown in Figure 3.

4.2. Encoder

We used a recurrent neural network (RNN), which recursively maps a input vector and a hidden state to a new hidden state. Among different variants of RNN, gated recurrent unit networks (GRUs) were chosen, because it has been proved to be effective for many NLP tasks, especially for small dataset. The size of feature map from CNN output was 512, and the size of GRU cell size was 256. Word embedding was also applied, which represents different aspect of a token, so that similarity feature of tokens could be learned by the network. The dimension of word embedding was 80. The input of encoder will be the output of CNN network with the shape of (batch size, $\frac{1}{8}\times$ original height, $\frac{1}{8}\times$ original width, 512). The maximum length of RNN encoder was size of output height times width ($W' \times H'$)

4.3. Decoder

Among sequence-to-sequence models, we chose the attention model to focus on contents which might be useful for prediction. As shown in Figure 7, the target markup tokens were generated by a decoder that built by a sequence of annotation V. We were able to produce the LaTeX tokens with a recurrent neural network once receiving the feature map of size (N, H, W, C). This decoder was trained as a conditional language model to give the probability of the next token given the history and the annotations.

After computing an attention vector at each time step by the decoder, it produced a distribution with a recursive formula:

$$p(y_t) = f(y_{t-1}, h_{t-1}, c_t)$$

where p is the distribution, y_{t-1} is the input previous token, h_{t-1} is the hidden state of the GRU, c_t is an attention vector.

Moreover, another output state o_{t-1} was used to compute the distribution probability over the formula as follows.

$$h_t = \text{GRU}(h_{t-1}, [Ey_{t-1}, o_{t-1}])$$

$$c_t = \text{Attention}(h_t, \tilde{V})$$

$$o_t = \tanh(W^c[h_t, c_t])$$

$$p(y_{t+1}|y_1, \cdots, y_t, V) = \text{softmax}(W^{\text{out}} o_t)$$

where E is an embedding matrix and W are matrices.

Since c_t only stands for the attention vector, it can not show the cell state of the GRU. Therefore, we had to seek it by researching the interior structure of GRU.

After applying the knowledge of attention model, we were able to enhance the performance of the decoder. In this project, soft attention mechanism was chosen. According to the attention mechanism published from (Luong et al.,

2015), we applied the formulas below to finish the decoding process.

$$e_i^t = \beta^T \tanh(W_h h_{t-1} + W v_i)$$
$$\alpha^t = \text{softmax}(e^t)$$
$$c^t = \sum_{i=1}^{W' \times H'} \alpha_i^t v_i$$

4.4. Training

The complete model was trained end-to-end without given any knowledge about the markup language or generation process. Cross-entropy loss for a sequence of logits were used for our model. Learning rate was initialed to be 0.001 and decayed by one-tenth if the loss didn't reduce for two epoch. In total, we ran for 40 epochs. For training set, the loss for 4-block DenseNet model was decreased from 3 to 0.008.

5. Experiments / Results / Discussion

5.1. Hyperparameters

We kept most of the hyperparameters the same as those in (Deng et al., 2016) model for RNN part. The batch size was set to be 20 due to memory in GeForce GTX 1080 Ti GPU. To compare the baseline CNN model, the output of CNN was controlled to be in the same height, width and channels. Given the output constrain and keeping the number of bottlelayer same in each dense block, we have tuned the number of dense block and growth rate in DenseNet model. 3-block densenet model was developed with growth rate of 16/32/64; 4-block densenet model was developed with growth rate of 16/32/32/32.

Model	BLEU score	Edit distance
(Deng et al., 2016) CNNEnc	0.75	0.61
Baseline CNN	0.77	0.76
DenseNet(3 blocks)	0.80	0.79
DenseNet(4 blocks)	0.84	0.85

Table 2. Experiment results

5.2. Accuracy

To evaluate our result, we tracked the accuracy of predicted latex codes and the true latex codes by the BLEU score (Papineni et al., 2002). We also calculated the edit distance on the text (Levenshtein distance), between the predicted latex code and the true label. Edit distance is the percentage of the reconstructed text that matches the original.

Both the BLEU score and edit distance of DenseNet models were better than CNNEnc (Deng et al., 2016) and baseline CNN model, because bottleneck layers were closely connected and more features from shallower layers could be directly transferred into deeper layers. Please see more details in Figure 5 and Table 2. The results of 4-block densenet model was slightly better than 3-block one.

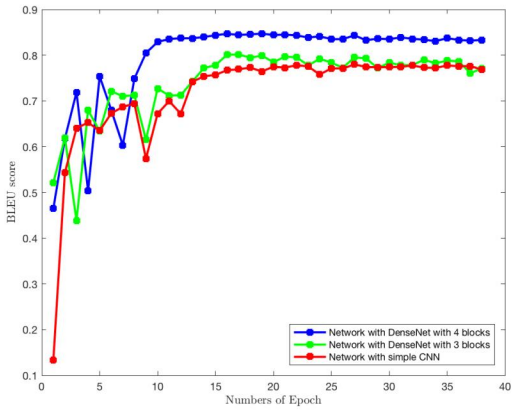

Figure 5. BLEU score

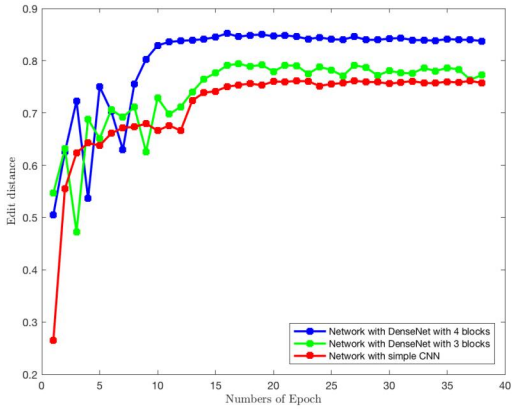

Figure 6. Edit distance

5.3. Training Efficiency

Showing in Figure 5 and Figure 6 that Densenet model reached best BLEU sore and edit distance earlier, we found that training for DenseNet model was more efficient compared with baseline CNN model. There were close connections between bottleneck shallow and deep layers in dense block. During the back propagatation, the weights of layers could have more updates compared with traditional convolution. Thus, within one epoch, more information could be captured by densenet model, resulting in more iteration of training and tunning cycle in same period of time.

```
Input Image :
```

$$f_{ij} = \partial_i a_j^s - \partial_j a_i^s,$$

```
predicted latex codes :
f _ { i j } = \partial _ { i } a _ { j } ^ { s } - \partial _ { j } a _ { i } ^ { s } ,

Visualization of the latex codes :
```

$$f_{ij} = \partial_i a_j^s - \partial_j a_i^s,$$

```
Input Image :
```

$$J_\mu = -se_\mu^{(0)} + \gamma_i e_\mu^{(i)},$$

```
predicted latex codes :
J _ { \mu } = - s e _ { \mu } ^ { ( 0 ) } + \gamma _ { i } e _ { \mu } ^ { ( i ) } ,

Visualization of the latex codes :
```

$$J_\mu = -se_\mu^{(0)} + \gamma_i e_\mu^{(i)},$$

Figure 7. Some predicted results of DenseNet(4 blocks). Comparing the first result to Figure4, we found that our new model could come up with the correct superscript s. The second example also showed that our new model had reasonable well performance in dealing with tiny symbols in subscript and superscript.

5.4. Learning Rate

From Figure 5 and Figure 6, we observed that DenseNet has a larger drop in both BLEU performance and edit distance, when learning rate changed, compared with baseline CNN, so DenseNet seemed to be more sensitive in learning rate decay. Therefore, choosing a more suitable decay function for DenseNet might enhance the training efficiency. We also found that 4-block densenet model was slightly more sensitive than 3-block one. More attention on selecting appropriate learning rate function shall be paid to deeper densenet model.

5.5. Overfitting

From the data collected from model output, the edit distance of DenseNets was around 85%, which was 10% more than normal CNN, which means by using DenseNets, we were able to achieve a lower error. We assumed that it was due to the good usage of raw data and interior connections among different layers, a tendency of DenseNet-BC bottleneck and compression layers were less overfitting and able to reduce error dramatically when compared with normal constitutional neural networks.

6. Conclusion / Future Work

This paper presents an improved implementation of image captioning applied to Latex generation from raw image. A new convolutional network using Densenet was introducted and it enhanced the performance in terms of prediction accurancy and computational efficiency.

For future work, firstly, we'd like to apply beam search to improve the RNN network, which as an approximate search often works far better than the greedy approach. The result is predicted to be no worse than current performance at the expense of computational effort. Secondly, our research can be scaled from printed mathematical formulas images to the hand written mathematical formulas images. Surprisingly, we were overwhelmed by the request for this function from audiences during our presentation.

7. Team Contribution

All three members of this team work together and contribute equally to this project in data prepossessing, algorithm designing, model designing, model training and report writing.

References

Ciresan, Dan Claudiu, Meier, Ueli, Gambardella, Luca Maria, and Schmidhuber, Jürgen. Deep big simple neural nets excel on handwritten digit recognition. *CoRR*, abs/1003.0358, 2010. URL http://arxiv.org/abs/1003.0358.

Deng, Yuntian, Kanervisto, Anssi, and Rush, Alexander M. What you get is what you see: A visual markup decompiler. *CoRR*, abs/1609.04938, 2016. URL http://arxiv.org/abs/1609.04938.

Huang, Gao, Liu, Zhuang, and Weinberger, Kilian Q. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL http://arxiv.org/abs/1608.06993.

Jaderberg, Max, Simonyan, Karen, Vedaldi, Andrea, and Zisserman, Andrew. Deep structured output learning for unconstrained text recognition. *CoRR*, abs/1412.5903, 2014. URL http://arxiv.org/abs/1412.5903.

Kanervisto, Anssi. im2latex-100k , arxiv:1609.04938, June 2016. URL https://doi.org/10.5281/zenodo.56198.

Karpathy, Andrej and Li, Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014. URL http://arxiv.org/abs/1412.2306.

Karpathy, Andrej, Johnson, Justin, and Li, Fei-Fei. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078, 2015. URL http://arxiv.org/abs/1506.02078.

Luong, Minh-Thang, Pham, Hieu, and Manning, Christopher D. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. URL http://arxiv.org/abs/1508.04025.

Papineni, Kishore, Roukos, Salim, Ward, Todd, and Zhu, Wei-Jing. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pp. 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

Wang, Tao, Wu, David J., Coates, Adam, and Ng, Andrew Y. End-to-end text recognition with convolutional neural networks.

Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Cho, Kyunghyun, Courville, Aaron C., Salakhutdinov, Ruslan, Zemel, Richard S., and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015. URL http://arxiv.org/abs/1502.03044.