IMAGE TO LATEX VIA NEURAL NETWORKS

CS 297 Report

Presented to

Dr. Chris Pollett

Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Class

CS297

By

Avinash More

December 2017

TABLE OF CONTENTS

I. Problem Statement

Many research papers in mathematics, computer science, and physics are written in LaTeX format. While writing technical papers or articles, there are some scenarios where the text to be written is a mathematical equation. Writing a mathematical equation in LaTeX format takes a lot more time compared to writing the same equation on a paper. The time-consuming approach of converting the equation written on paper to LaTeX format can be automated and optimized. In this project, I am exploring an approach to convert an image of a mathematical equation to its corresponding LaTeX.

Reading text from books or images is not a new problem. Problems like this have been attempted since early the 1900s. In the year 1914, an analog machine was developed by a Russian researcher Emanuel Goldberg to read the characters and convert those to telegraph code. In 1974 researchers from a company called Kurzweil Computer Products, Inc. started working on a product which should be able to recognize character written in any font also called Omni-font OCR problem. In the year 1978, selling of commercial version of this OCR computer software was started in the market. In the 2000s, with the popularity of the internet, many online cloud-based services are available for OCR. With the use of smartphones, OCR applications have become commonplace. Various OCR applications try to solve specific OCR problems. Common examples include reading financial documents such as Cheque, automatic character recognition from the vehicle number plate, reading text from the images of books (Google books) etc.

All the solutions to the above application specific OCR problem were based on the technology trends during that period. In recent years because of advancement in the hardware technologies training of neural networks has become more feasible and easier. In this project, we are trying to solve this problem with the neural networks. Neural networks we are exploring to solve this problem are convolutional neural networks (CNN) and recurrent neural networks (RNN).

Coincidently, in the year 2016 same problem statement was also posted on the OpenAI's requests to research section. One of the approaches used to solve this problem was using a concept called coarse-to-fine attention by Harvard researchers. In this project, dataset of images

used was the dataset formed by extracting the mathematical equations from the different technical publications [1].

The same problem statement of converting an image to LaTeX can be modified a little bit and made as converting an image to a corresponding HTML. Another paper explored an approach to solve this problem with a model which does not need to have any knowledge of the underlying markup language. This model is an extension of attention based encoder-decoder model [2].

Another similar problem statement is a problem of image caption generation. This problem involves identifying the objects in the image and establishing the relationship between them. Even though this problem does not have an exactly same description as our problem statement, the approach used to solve this problem is quite interesting. In this approach, a convolutional neural network is used get the data from the image. After multiple layers of convolution and max pooling, the flattened output of the final max-pooling layer is given as an input to the recurrent neural network [3].

In this report, I will discuss four deliverable and why those deliverables are important parts of this project. The first deliverable of the project is understanding the markup language tags and exploring how to write various types of markup tags. In the deliverable 2 part of this report, I will explore various aspects of the of deep learning library TensorFlow which are required to train deep neural networks. The deliverable 3 part of this report will deal with the data generation part which is needed to train the deep neural network. In the deliverable 4, I will discuss the possible architecture of deep neural network which can be used to convert an image to corresponding LaTeX. In the last section of the report, I will conclude with the possible CS298 work and deliverables.

II. Deliverable 1

The first deliverable for my project work was to identify most famous mathematical equations and getting the corresponding LaTeX. The objective of this deliverable was to understand the working of markup language such as LaTeX. For this purpose, I wrote LaTeX for mathematical equations involving fractions, exponents, trigonometric functions, log functions, roots, limits, differentials, integrals, summation, matrix, and multiline equations.

I learned the following things about the representation of mathematical equations from this exercise:

1. **Size**: Size of the literals varies depending on the position of the literal in the mathematical equation.
   Example:
   a. Size of the literal in the exponent part is smaller than size of the literal in the mantissa part. Example: a^a, in this example size of literal a in the exponent is smaller than the size of literal a in the mantissa.
2. **Greek letters**: Usually while writing an English document, we rarely use Greek letters. In contrast, in mathematical equations Greek letters are used more frequently.
   Example:

$$f(\zeta) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \zeta} dx$$

Figure 1: Equation involving Greek symbols

3. **Fonts**: In a mathematical equation, the font of the variable and the font of the functions can be different.

   Examples:

   a. Log functions:

$$\log xy = \log x + \log y$$

Figure 2: Log function

   b. Limits:

$$\frac{df}{dt} = \lim_{h \to 0} \frac{f(t+h) - f(t)}{h}$$

Figure 3: Limits

4. **Multiline equations**: Single equation can have multiple lines.

   Example:

$$n = \cfrac{1}{\frac{2n-1}{n} + \cfrac{1}{\frac{2n-3}{n} + \ddots + \cfrac{1}{\frac{2n-n}{n}}}}$$

Figure 4: Multiline mathematical equation

III. Deliverable 2

The second deliverable of preparation for writing project was to explore a popular deep learning library by Google called TensorFlow. The following are some reasons to use an already existing popular library:

- **Code reuse:** It is better not to reinvent the wheel.
- **Less error prone:** As the libraries are used and tested by multiple users it is usually less error prone.
- **Level of abstraction:** Using a library provides a level of abstraction because of which programmer does not have to worry about minor details of the code.
- **Efficiency:** Libraries are written by the expert programmers. They are tested for performance. Therefore, libraries are efficient.
- **Online community support:** Because popular libraries are used by large community of

Python programming language has libraries such as NumPy and SciPy to make it more efficient for numerical and scientific computing. NumPy has its subroutines compiled in a more efficient the low-level programming language. Therefore, using NumPy subroutines instead of Python looping constructs with lists makes Python programs more efficient. Transferring data from one programming environment to other is an expensive operation. This overhead gets even more when we are transferring data to graphics processing unit (GPU) or when we are transferring the data to a distributed environment where there is a high cost associated with the data transfer. To handle this overhead TensorFlow has an interesting approach. Instead of frequently switching between different language environments, TensorFlow lets a user define the complete computation. This complete computation is represented by computation graph of interacting operations. Once complete graph is defined, TensorFlow executes the complete computation graph by establishing a session.

There might be scenario when we do not know the input at the time of deciding the computation graph. To handle such scenarios TensorFlow has a concept called placeholders. Using placeholders, input can be fed to the computation graph.

TensorFlow also has various methods for doing common operations related to multilayer perceptron such as follow:

1. **Calculating the error or the cost**:

   *cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))*

2. **Applying various matrix operations such as multiplication, addition, inverse, etc**.:

   y = tf.matmul(x,W) + b

3. **Applying different activation functions:**

   h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)

4. **Deciding and applying the backpropagation algorithms:**

   train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

5. **Defining a neural network:**

```
def conv2d(x, W):
  return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')


def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                        strides=[1, 2, 2, 1], padding='SAME')
```

TensorFlow has provides a way to save and restore the model.

Following code can be used to save the model:

```
with tf.Session() as sess:
    sess.run(init_op)
    # Do some work with the model.
    inc_v1.op.run()
    dec_v2.op.run()
    # Save the variables to disk.
    save_path = saver.save(sess, "/tmp/model.ckpt")
    print("Model saved in file: %s" % save_path)
```

Following code can be used to restore the model:

```
with tf.Session() as sess:
    # Restore variables from disk.
    saver.restore(sess, "/tmp/model.ckpt")
    print("Model restored.")
```

In this deliverable, I explored these methods of TensorFlow with the classic MNIST dataset.

IV. Deliverable 3

In this deliverable, the objective was to find ways to generate data. In machine learning, finding data for training is one of the most important tasks. The importance of finding the correct and appropriate dataset which will be closer in representation to the real-world data is not often recognized. But, there have been many instances where the same algorithm with a better set of data has given much better results [4]. Therefore, it was important to explore multiple approaches to generate data.

For this deliverable, I tried two approaches more prominently. The first approach was creating a portable network graphics (PNG) image file and the second approach was for generating portable document format (PDF) file using postscript. The approach of generating PDF uses PNG as base file and embeds it with more wrappers and creates an equivalent PDF file. Therefore, using PNG file seems like more appropriate way.

The PNG image file can be generated with a code in different ways but for our dataset generation, there are multiple conditions which need to be satisfied. One of the most important condition while generating a data set for training is that it should have image representation as well as the label for the same. The label for this problem is a LaTeX representation of the image. Having an exact representation of the LaTeX for all the images in the dataset is a critical requirement for our problem-solving approach.

While exploring possible libraries for generating data library modules Matlab seemed suitable. As Matlab is a proprietary, its corresponding library in Python programming language called Matplotlib was chosen. Using Matplotlib images can be created with an equation in LaTeX form as the text.

Following code snippet shows the way to generate images:

```
def render_latex(formula, fontsize=10, dpi=300, format_='svg'):
   fig = plt.figure(figsize=(0.3, 0.25))
   fig.text(0.05, 0.35, u'${}$'.format(formula), fontsize=fontsize)
    buffer_ = StringIO()
   fig.savefig(buffer_, dpi=dpi, transparent=True, format=format_, pad_inches=0.0)
   plt.close(fig)
   return buffer_.getvalue()
```

```
image_bytes = render_latex(expression, fontsize=5, dpi=200, format_='png')
image_name = './data/' + 'name.png'
with open(image_name, 'wb') as image_file:
    image_file.write(image_bytes)
```

V. Deliverable 4

The objective of Deliverable 4 was to come up with an architectural diagram for the implementation of the solution in the next semester. The main idea of this architecture is inspired by the paper which provided an approach for the image captioning problem [3]. In this paper, language translation problem was discussed. In earlier approaches of language translation, the problem was divided into multiple components where each component would work on subproblems. Those subproblems included translating words individually, aligning words, reordering, etc. But, in recent times, it has been realized and proven that this problem can be solved in the much easier way.

Machine translation or language translation problem can be solved using recurrent neural networks with the same state of the art performance. In this approach, an “encoder” RNN reads the source sentence and transforms it into a rich fixed-length vector representation, which in turn in used as the initial hidden state of a “decoder” RNN that generates the target sentence [3]. For the image captioning problem this approach can be used which in turn can be used for our problem with a slight modification. This slight modification is replacing the encoder RNN by a deep convolution neural network (CNN).

Over the last few years, it has been seen that that convolutional neural networks can produce a rich representation of an image by embedding it to a fixed-length vector, such that this representation can be used for a wide variety of vision [3]. Hence, it was an obvious choice to use a CNN as an image “encoder”. In our approach we will be first pre-training an image for classification task and use the last hidden layer as an input to the recurrent neural network decoder that generates the corresponding LaTeX.

The RNN encoder which we will be using for the project will be long short term memory (LSTM) provided by TensorFlow. RNN has a vanishing gradient descent problem. This problem is not only with RNNs but compared to other networks RNNs tend to be very deep (as deep as the equation length in our case), which makes the problem a lot more common. We will be using LSTM because it can address this problem.

Following is the block diagram of the proposed solution:

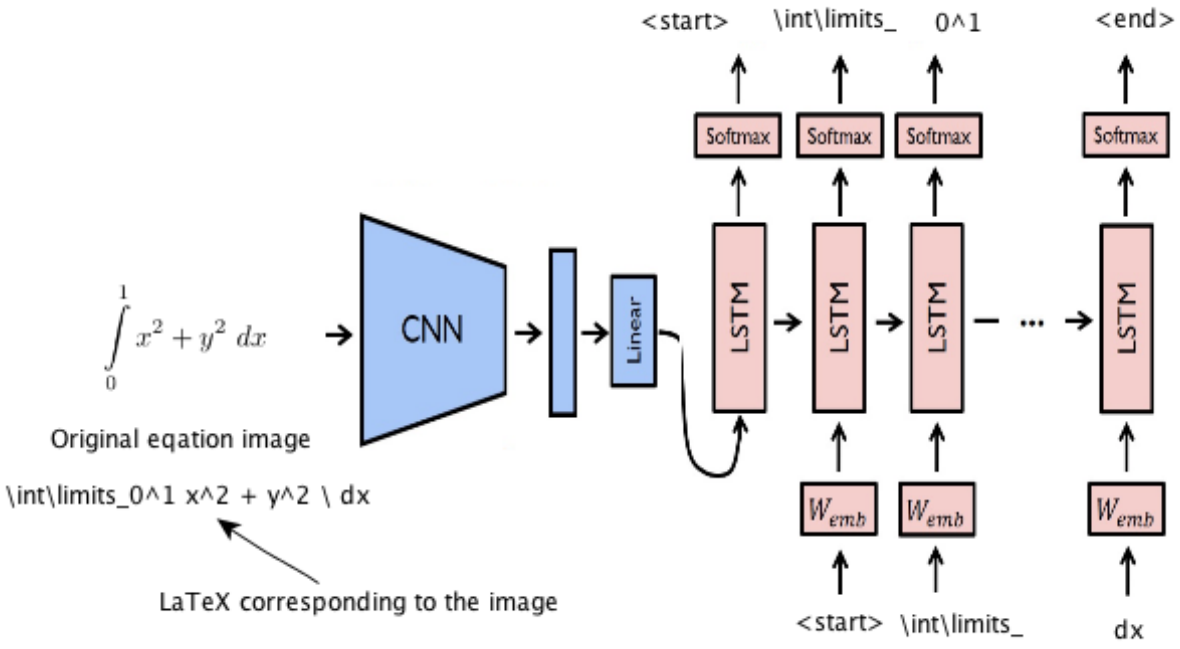

Figure 5: Architecture diagram with a sample equation

VI. Conclusion

In conclusion, this semester work was extremely important for the project implementation in the next semester. All the deliverables added valuable information and skills which will be needed for the project implementation.

The deliverable 1 which was to write LaTeX for different types of equations helped me understand the complexity of the project better. The second deliverable gave an overview of which deep learning concepts have an implementation in deep learning library TensorFlow. In the third deliverable, I found a way to get the dataset for the training the model. The fourth deliverable dealt with the deciding the architecture diagram involving the CNN and RNN.

In this semester, apart from all the deliverable, I also studied multiple deep learning concepts such as feedforward networks, backpropagation, multilayer perceptron, convolutional neural networks, and recurrent neural networks.

Having gained the knowledge and exposure to deep learning concepts in this course, in the CS 298 I would like to get started with the implementation. There can be multiple deliverables which will have incremental implementations. The first deliverable can be finding the first character of the image and the last deliverable can be getting the complete LaTeX of corresponding to mathematical equation image.

References

[1] Deng, Yuntian, et al. "Image-to-Markup Generation with Coarse-to-Fine Attention." *International Conference on Machine Learning*. 2017.

[2] Deng, Yuntian, Anssi Kanervisto, and Alexander M. Rush. "What You Get Is What You See: A Visual Markup Decompiler." *arXiv preprint arXiv:1609.04938* (2016).

[3] Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[4] Sun, Chen, et al. "Revisiting unreasonable effectiveness of data in deep learning era." *arXiv preprint arXiv:1707.02968* 1 (2017).