Image to LaTeX via Neural Networks

Avinash More
Agenda

- Introduction
  - LaTeX
  - Problem Statement
  - Past work
- Background
  - Neural Networks
  - Convolutional Neural Networks
- Data Generation
- Implementation
- Experiments
- Results
- Conclusion
- Future work
LaTeX

- What is LaTeX?
  - Short for Lamport TeX (Leslie Lamport)
  - LaTeX is a set of macros built on top of TeX (Typesetting system by D. Knuth).
  - LaTeX is used as a document preparation system
  - Markup language that also handles typesetting and rendering

- Why is LaTeX used?
  - Design philosophy of separating presentation from content
  - Focus on the content without attending simultaneously to its visual appearance
\[(a + b) - \sum_{n=1}^{\infty} n^2 + 2^{abc^d}\]
Why is this problem hard?

- This problem is different from the usual OCR
  - Different font size in a single line
  - Multiline equations
  - Equation reading order

\[
\begin{bmatrix}
1 & 2 \\
\color{red}b & \color{red}3
\end{bmatrix} = \begin{bmatrix}
\color{red}a & 5 \\
\color{red}x & \color{red}q
\end{bmatrix} + \begin{bmatrix}
w & m \\
5 & \color{red}d
\end{bmatrix}
\]
Past work

- Winkler, et al. (1995) - handwritten mathematical expression images based on Hidden Markov Model (HMMs)
- OpenAI’s requests to research
- Deng, Yuntian, et al. (2016) - coarse-to-fine attention
- Deng, Yuntian, Anssi Kanervisto, and Alexander M. Rush. (2016) - HTML instead of LaTeX using attention based encoder-decoder model

We are proposing a solution based on neural networks
Background

- We are proposing a solution based on neural networks
- We will be specifically focusing on convolutional neural networks
Neural Network
How are neurons activated?
Convolution Operation

- Convolution
- Kernel
- Feature map
- Filter
Convolution operation example
Max - Pooling
Convolutional neural network example
Convolution Intuition

Original image

Visualization of the filter on the image
Convolution Intuition

**Visualization of the receptive field**

**Pixel representation of the receptive field**

**Pixel representation of filter**

**Multiplication and Summation**:

\[
(50 \times 30) + (50 \times 30) + (50 \times 30) + (20 \times 30) + (50 \times 30) = 6600 \text{ (A large number!)}
\]
Convolution Intuition

Visualization of the filter on the image

Pixel representation of receptive field

Pixel representation of filter

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 40 & 0 & 0 & 0 & 0 \\
40 & 40 & 0 & 0 & 0 & 0 \\
40 & 20 & 0 & 0 & 0 & 0 \\
0 & 50 & 0 & 0 & 0 & 0 \\
0 & 0 & 50 & 0 & 0 & 0 \\
25 & 25 & 0 & 50 & 0 & 0 \\
\end{array}
\times
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 30 \\
0 & 0 & 0 & 30 & 0 & 0 \\
0 & 0 & 0 & 30 & 0 & 0 \\
0 & 0 & 0 & 30 & 0 & 0 \\
0 & 0 & 0 & 30 & 0 & 0 \\
0 & 0 & 0 & 30 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[\text{Multiplication and Summation} = 0\]
Data Generation

- Deep learning works well with a lot of data
- Finding labeled data for our problem was difficult
- We are proposing a way to generate labeled data for Image to LaTeX problem
Data Generation Approach

- Matlab
- Matplotlib
- Mathtex

```python
from matplotlib import rcParams

rcParams['text.usetex'] = True

rcParams['text.latex.preamble'] = r'\usepackage{amsmath}'

os.environ['PATH'] += os.pathsep + '/Library/TeX/texbin'
```
Sample code snippet for data generation

```python
import matplotlib.pyplot as plt

def render_latex(formula, image_name, fontsize=10, dpi=300, format_='png'):
    fig = plt.figure(figsize=(1.30, 0.20))
    fig.text(0.01, 0.20, formula, fontsize=5)
    fig.savefig(image_name, dpi=dpi, transparent=True, format=format_, pad_inches=0.0)
    plt.close(fig)

image_name = './' + data_folder + '/' + image_name + '.png'
render_latex(expression, image_name, fontsize=8, dpi=200, format_='png')
```
Implementation

1. The first deliverable: Predicting the LaTeX for the first character
2. The second deliverable: Predicting the LaTeX for simple equations
3. The third deliverable: Predicting the LaTeX for complex mathematical equations
4. The fourth deliverable: Predicting the LaTeX for matrix operations [2 x 2]
Input

Input for training

- Image - is a grayscale image containing zeros and ones
- Label - One hot vector representation for each character in the LaTeX representation of the equation in the image
Predict the first character

<table>
<thead>
<tr>
<th>Image</th>
<th>Label corresponding the first character</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqrt{c}$</td>
<td>[1,0,0,0]</td>
</tr>
<tr>
<td>$\int_{q}^{k} a^2 da$</td>
<td>[0,1,0,0]</td>
</tr>
<tr>
<td>$\sum_{j=1}^{\infty} 2^j$</td>
<td>[0,0,1,0]</td>
</tr>
<tr>
<td>$\frac{a}{b}$</td>
<td>[0,0,0,1]</td>
</tr>
</tbody>
</table>
Predict the LaTeX for the first character
Predicting the LaTeX for Simple Equations

$2^3 + 5 - a$

$\$ 2^3+5-a\$\$
Predicting the LaTeX for Simple Equations
Predicting the LaTeX for Complex Mathematical Equations

\[(4 - 1) + 2^{18} \cdot \sum_{w=5}^{\infty} w^2\]

\$$(4 - 1) + 2^{18} \cdot \sum_{w=1}^{\infty} w^2\$$
Predicting the LaTeX for an Equation Containing Matrix Operations

\[
\begin{bmatrix}
w & n \\
3 & 5
\end{bmatrix} = \begin{bmatrix}
g & 8 \\
q & c
\end{bmatrix} \times \begin{bmatrix}
c & h \\
t & s
\end{bmatrix}
\]

\[
\begin{bmatrix}
w & n \\
3 & 5
\end{bmatrix} = \begin{bmatrix}
g & 8 \\
q & c
\end{bmatrix} \times \begin{bmatrix}
c & h \\
t & s
\end{bmatrix}
\]
Experiments

1. Mini batch size
2. Size of feature map
3. Number of feature maps
4. Changing the value of learning rate
Mini batch size

- Stochastic Processing
- Batch processing
- Mini-Batch processing
- 64, 128, 256, 512 or 1024 data samples
- As the computer memory is laid out and accessed in the memory blocks of the power of two, sometimes having the batch size power of 2 gives provides faster training
## Feature Map

<table>
<thead>
<tr>
<th>Feature map size</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5X5</td>
<td>It took around 10 seconds to train one batch of 128 images. Model converged and performed well on the test and validation data.</td>
</tr>
<tr>
<td>7X7</td>
<td>It took around 15 seconds to train on one batch of 128 images. Model worked fairly well on test and validation data but the efficiency was not as good as it was when we were using 5X5 filter size.</td>
</tr>
<tr>
<td>10X10</td>
<td>It took around 25 second to trains on one batch of 128 images. Even after training for more than 10 hours the model did not perform well on test data.</td>
</tr>
</tbody>
</table>
# Learning Rate

<table>
<thead>
<tr>
<th>Character Numbers</th>
<th>Learning rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>0.01</td>
</tr>
<tr>
<td>5-10</td>
<td>0.0015</td>
</tr>
<tr>
<td>10-17</td>
<td>0.001</td>
</tr>
<tr>
<td>17-30</td>
<td>0.0001</td>
</tr>
</tbody>
</table>
Oscillations
Validation accuracy for different characters

X : Number of Iterations; Y: Validation Accuracy

- Blue line: 2nd character accuracy
- Orange line: 12th character accuracy
Conclusion

- We achieved an accuracy of more than 90% for the images containing up to 35 characters involving numbers, letters, and a few mathematical symbols.
- Our experiments involving the prediction of LaTeX corresponding to matrix operations of size 29 were also successful where we achieved an accuracy of more than 85%.
- While training the equations involving more than 35 characters we realized the limitations of using CNN for the classification of complex objects when there is a hard relationship between different objects in an image.
- Using an encoder–decoder model involving CNN for encoding and recurrent neural network for decoding could be a better approach for the prediction of LaTeX corresponding to an image.
Thank you!

Questions?
References


References


