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

Input Output

$$(a + b) - \sum_{n=1}^{\infty} n^2 + 2^{a^{b^{c^d}}}$$

`$(a + b) - \sum_{n=1}^{\infty}n^2  + 2^{a^{b^{c^d}}}$`

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

$$\begin{bmatrix} 1 & 2 \\ b & 3 \end{bmatrix} = \begin{bmatrix} a & 5 \\ x & q \end{bmatrix} + \begin{bmatrix} w & m \\ 5 & d \end{bmatrix}$$

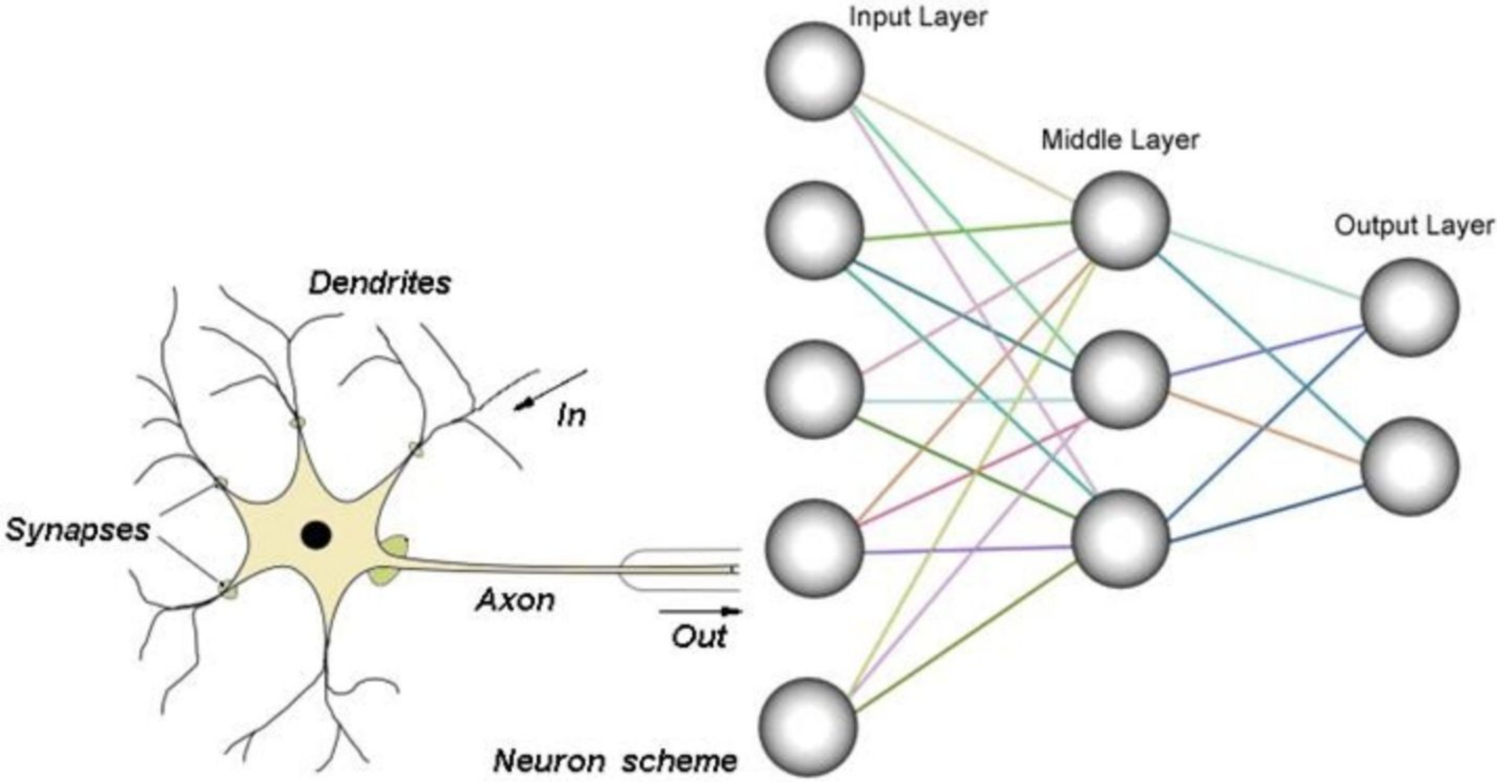
Past work

- 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

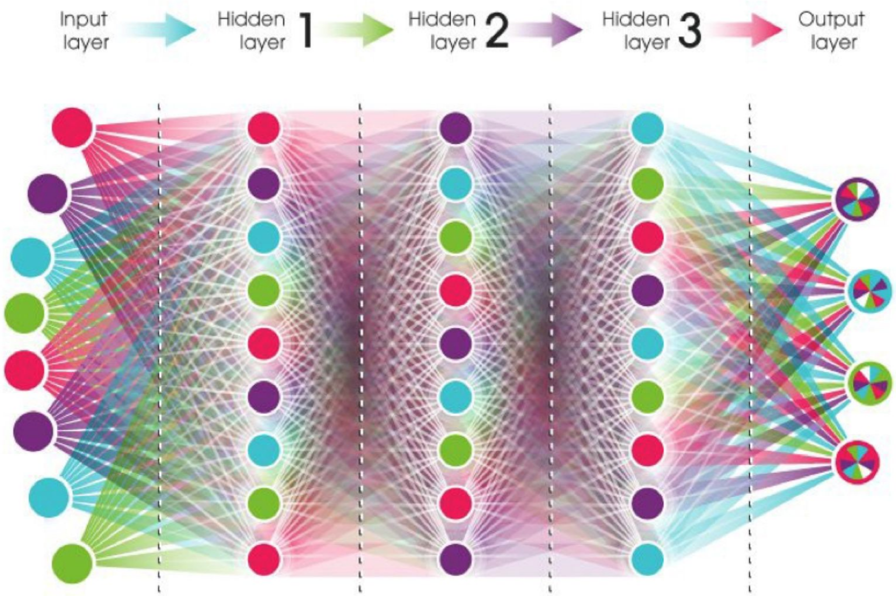
Background

- We are proposing a solution based on neural networks
- We will be specifically focussing on convolutional neural networks



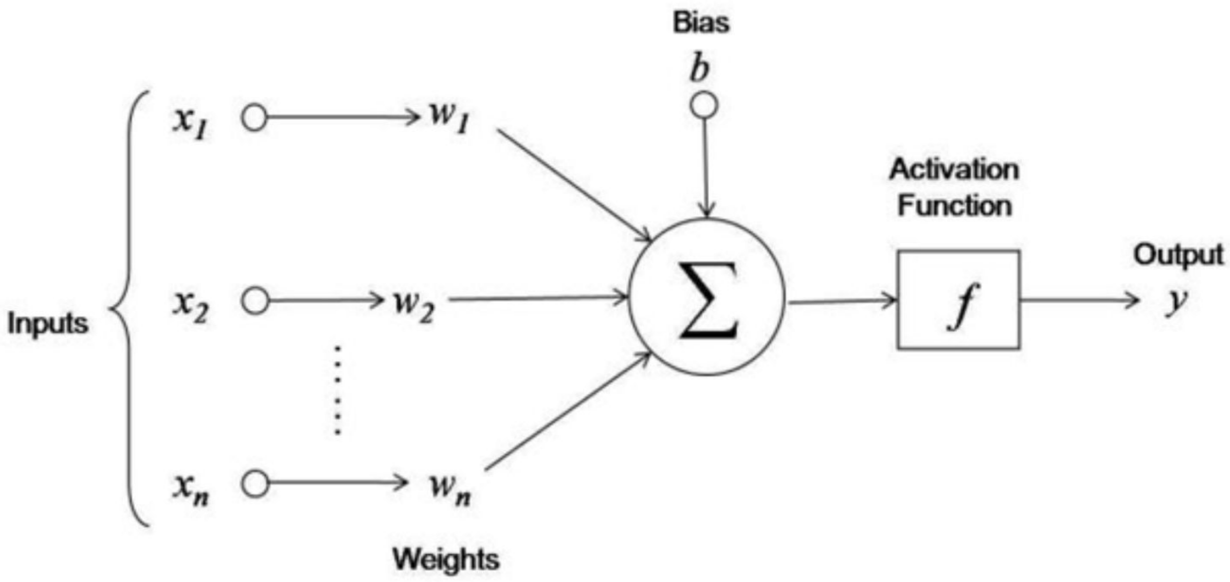
Input Layer
Middle Layer
Output Layer
Dendrites
In
Synapses
Axon
Out
Neuron scheme

Neural Network



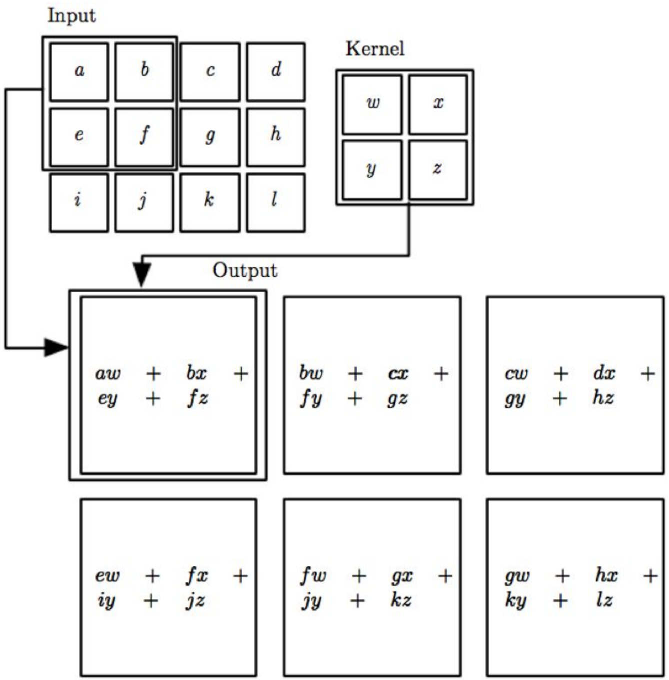
neuralnetworksanddeeplearning.com - Michael Nielsen, Yoshua Bengio, Ian Goodfellow, and Aaron Courville, 2016.

How are neurons activated?

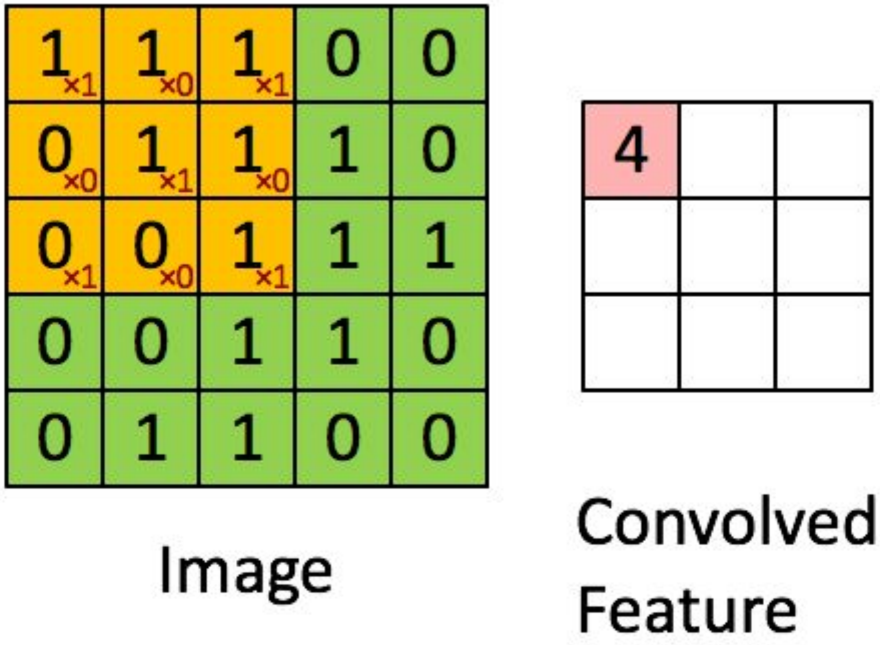


Convolution Operation

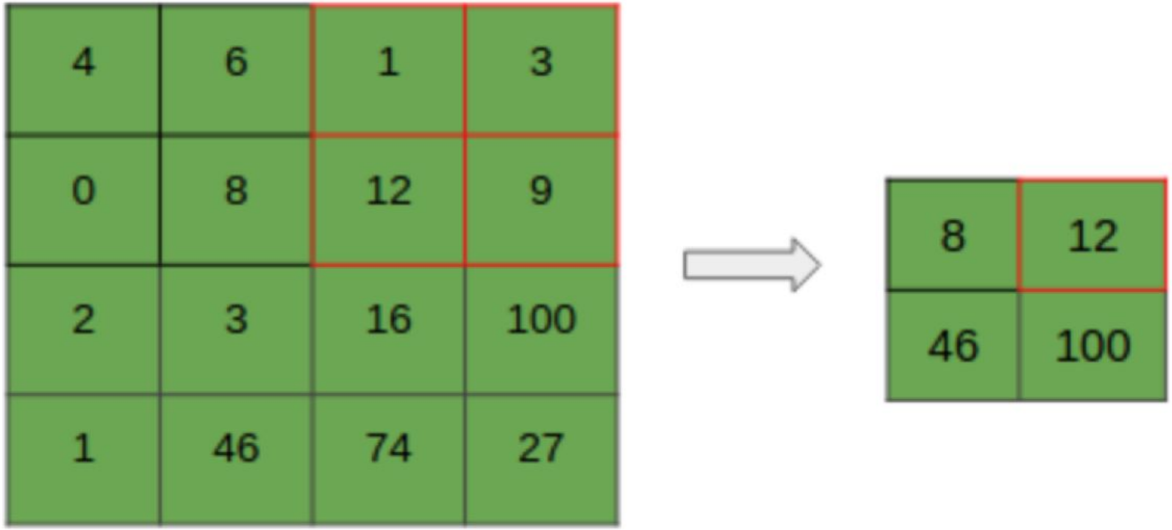
- Convolution
- Kernel
- Feature map
- Filter



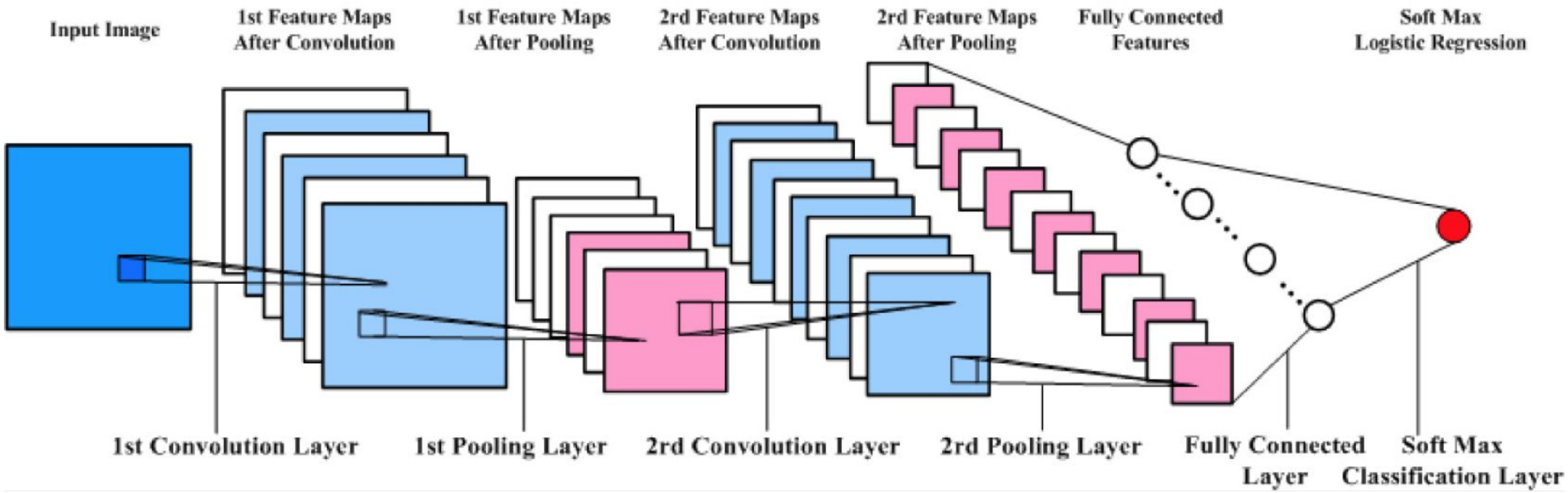
Convolution operation example



Max - Pooling

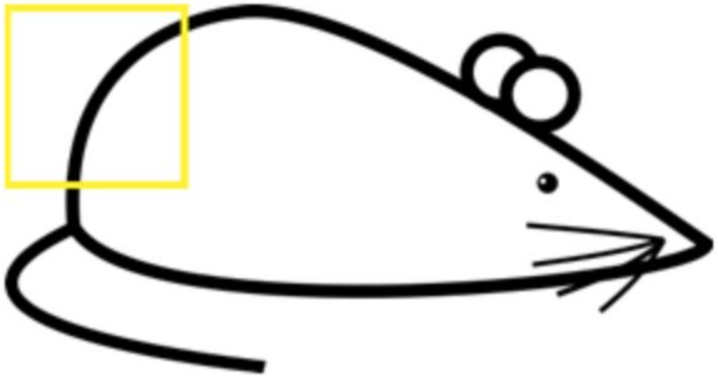


Convolutional neural network example



Convolution Intuition


Original image


Visualization of the filter on the image

Convolution Intuition

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Convolution Intuition

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

$*$

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

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

```
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

```
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 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

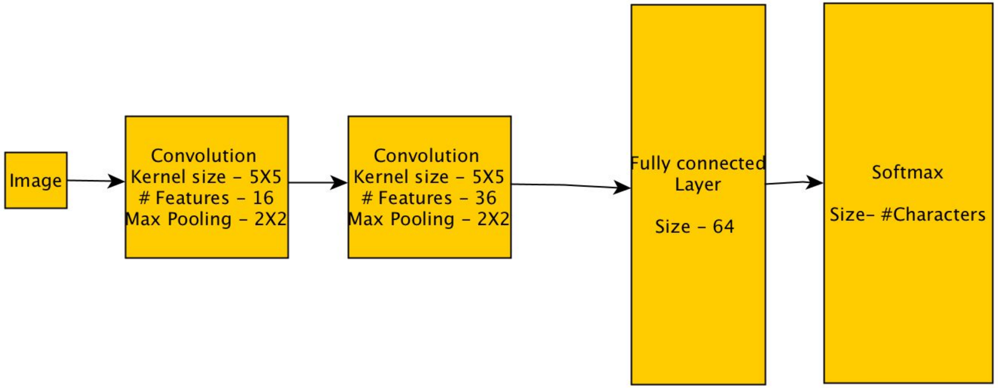
Input for training

- 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

Image	Label corresponding the first character
$\sqrt{c}$	[1,0,0,0]
$\int_q^k a^2 da$	[0,1,0,0]
$\sum_{j=1}^{\infty} 2^j$	[0,0,1,0]
$\frac{a}{b}$	[0,0,0,1]

Predict the LaTeX for the first character

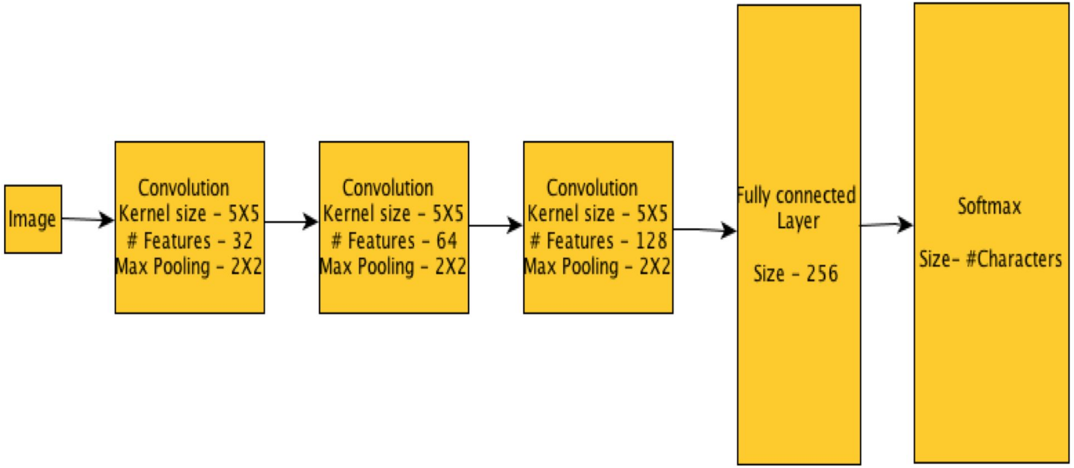


Predicting the LaTeX for Simple Equations

$$2^3 + 5 - a$$

`$ 2^3+5-a$`

Predicting the LaTeX for Simple Equations



Predicting the LaTeX for Complex Mathematical Equations

$$(4 - l) + 2^{1^{8}} * \sum_{w=5}^{\infty} w^2$$

`$(4 - l) +  2^{1^{8}} * \sum_{w=1}^{\infty}w^2$`

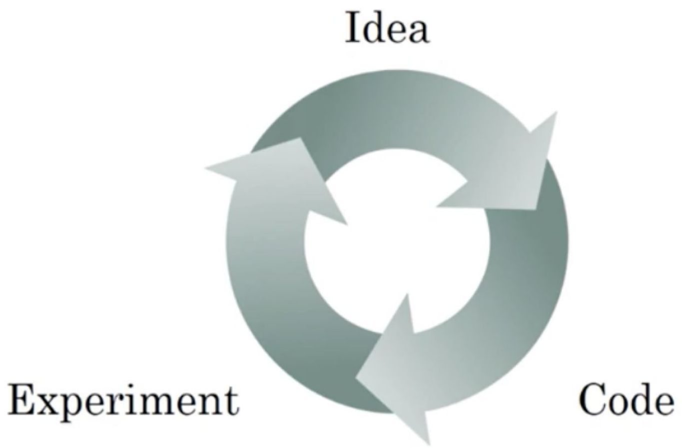
Predicting the LaTeX for an Equation Containing Matrix Operations

$$\begin{bmatrix}w & n\\ 3 & 5\end{bmatrix} = \begin{bmatrix}g & 8\\ q & c\end{bmatrix} * \begin{bmatrix}c & h\\ t & s\end{bmatrix}$$

$ \begin{bmatrix}w & n\\ 3 & 5\end{bmatrix} = \begin{bmatrix}g & 8\\ q & c\end{bmatrix} * \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

Processor	2.5 GHz Intel Core i7
Memory	16 GB 1600 MHz DDR3
Graphics	AMD Radeon R9 M370X 2048 MB Intel Iris Pro 1536 MB

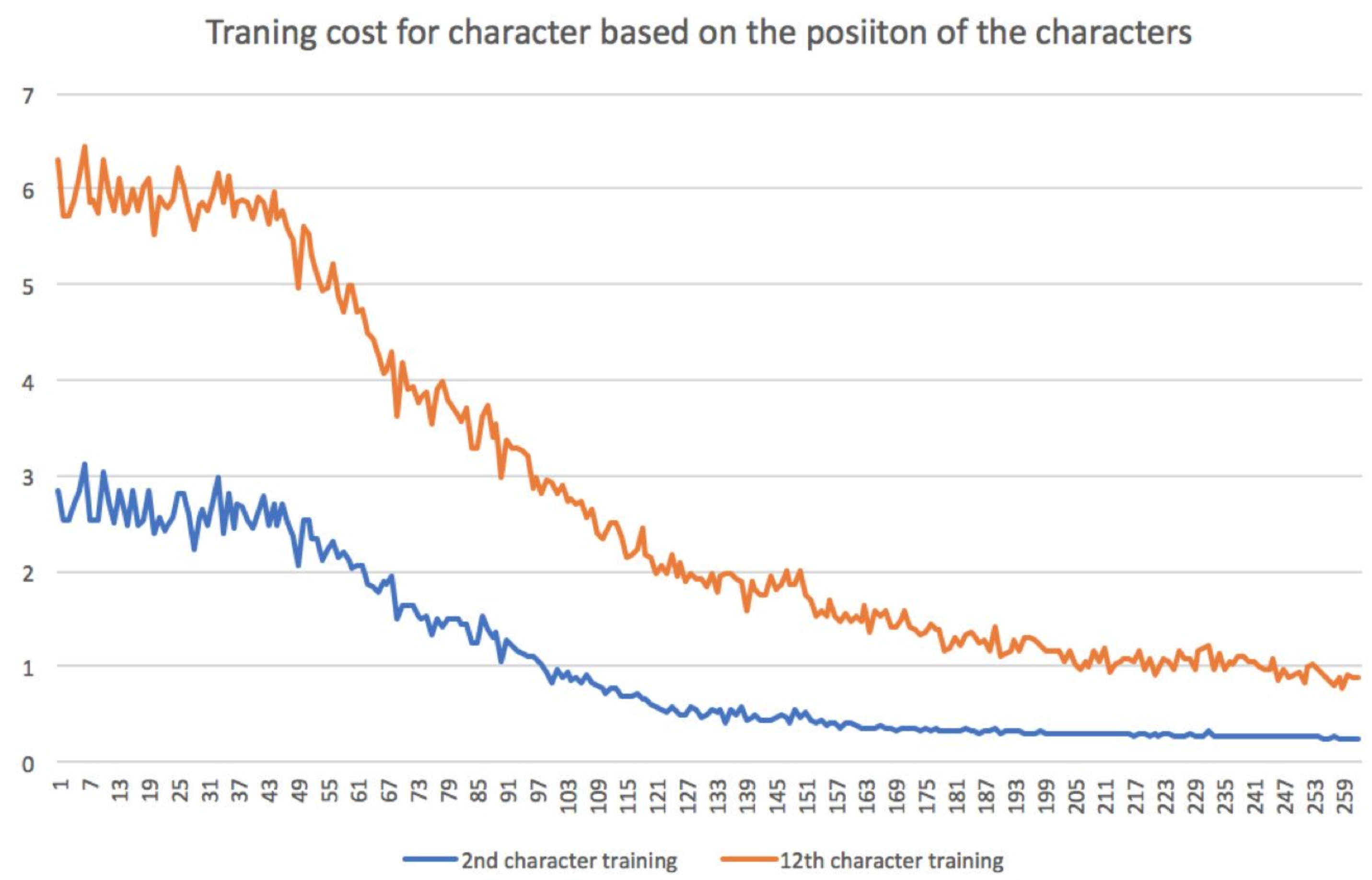
Feature Map

Feature map size	Observation
5X5	It took around 10 seconds to train one batch of 128 images. Model converged and performed well on the test and validation data.
7X7	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.
10X10	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.

Learning Rate

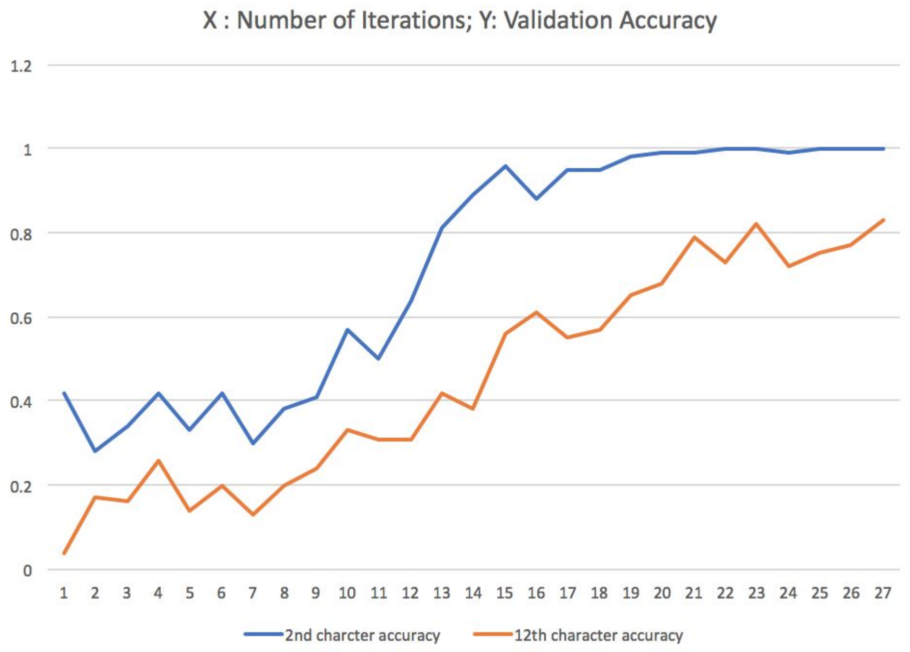
Character Numbers	**Learning rate**
1-4	0.01
5-10	0.0015
10-17	0.001
17-30	0.0001

Oscillations



Traning cost for character based on the posiiton of the characters
7
6
5
4
3
2
1
0
1
7
13
19
25
31
37
43
49
55
61
67
73
79
85
91
97
103
109
115
121
127
133
139
145
151
157
163
169
175
181
187
193
199
205
211
217
223
229
235
241
247
253
259
2nd character training
12th character training

Validation accuracy for different characters



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

[1] “Emanuel Goldberg”, Internet: https://history-computer.com/Internet/Dreamers/Goldberg.html

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

[3] Winkler, H-J., H. Fahrner, and Manfred Lang. "A soft-decision approach for structural analysis of handwritten mathematical expressions." Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on. Vol. 4. IEEE, 1995.

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

References

[5] 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).

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

[7] Goodfellow, Ian, et al. Deep learning. Vol. 1. Cambridge: MIT press, 2016.

[8] Ponnuru Rajeswari, Pookalangara Ajit Kumar “CIFAR-10 Classification using Intel® Optimization for TensorFlow*”, Internet: https://software.intel.com/en-us/articles/cifar-10-classification-using-intel-optimization-for-tensorflow, Dec 12, 2017