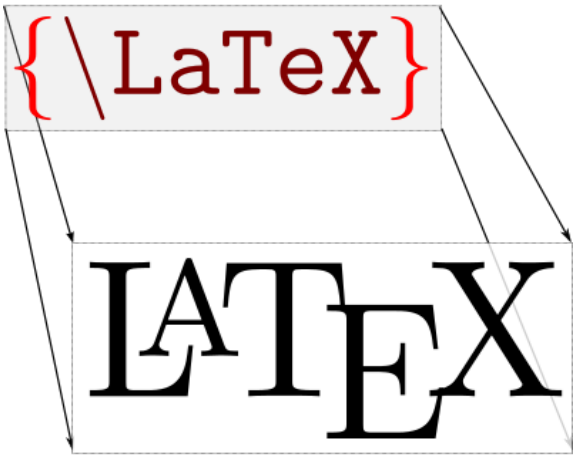


Image to Latex Conversion

Date: 24/08/2021

Jahnvi Kumari

Foreword

Hi readers,

I interned with the Data Science team of Extramarks EdTech in the Summer of 2021 working on the problem of Image to Latex Conversion. Throughout the journey, Tensorflow and Google Colab were my companions and I referred many online resources to learn ML and build solutions for ML problems. This is a document to guide beginner level interns on how to approach the Image to Latex conversion (or any other ML) problem. I've curated a list of articles that I have read and found useful during my internship[1].

Best,

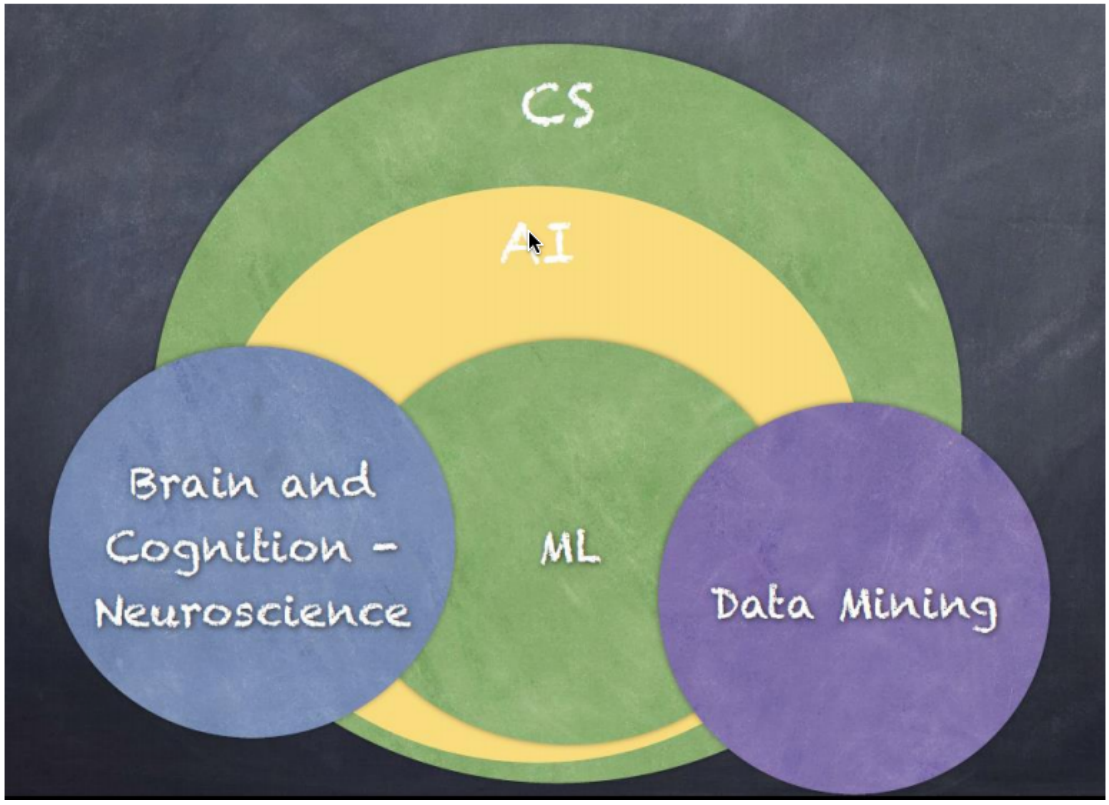
Jahnvi

1. My guide is not exhaustive. This document will be slightly biased towards resources that I found first. You are free and encouraged to explore on your own too!

Prerequisites

What is ML?

- An excellent resource to study the machine learning landscape is Chapter 1 of [Hands on Machine Learning by Oreilly](). This book is the most practical one I found online. Though the first chapter is a sufficient overview of ML, the latter chapters can be read side-by-side a college course etc to enhance your practical skills.
- Below is a venn diagram showing where ML, AI fall. You can consider DL as a subset of ML.



- **Good linear algebra** basics are essential to understand the working of algorithms. For beginner level projects, you only need to get comfortable with matrix representation and multiplication. For more advanced stuff, you need to know matrix decomposition techniques (like SVD), finding inverses etc.

- Is mathematical knowledge sufficient? No, **coding experience** is equally required; you would want to build amazing projects too, won't you? Watch this video to know more.
- **Pro tip**: It's essential to understand the algorithms you would be using in your project. Once you've understood it, oftentimes, it is easier to write your own algorithm code from scratch than tailor one found on some Github repo.

Language, Libraries and Environment

- **Language**: People use R and Python for Data Science. To be honest, I have never used R but I am quite sure that it cannot beat the versatility of Python. So if you don't know Python, then invest some time in a good course on Python(there are many free/paid/YouTube courses. Also, learn some Object Oriented Programming concepts in Python.
- **Environment**: I used Google Colaboratory because my laptop did not have a GPU. If yours does, you can install Jupyter Notebook. Both are similar. GPU or Graphical Processing Unit was developed originally for games which required fast rendering of images (recall the stunning graphics of GTA) and is fast with matrix multiplication- something used extensively in ML. This means that your training time significantly decreases when your calculations happen on a GPU instead of a CPU. To change the default runtime of CPU to GPU, go to Runtime->Change runtime->select GPU from Hardware Accelerator dropdown.

- **Tensorflow and Pytorch** are the best libraries for building ML models. Both have detailed tutorials. I used Image captioning with visual attention tensorflow tutorial as the basis for my project but found it a bit difficult to grasp probably because CNN, RNN and attention sounded foreign to me. But it's easy to follow after understanding these concepts (through blogs, videos etc.). Hence, tip: understand the components and the model would start seeming obvious to you.

- The tutorials have been prepared by experts. So, skim through them(pytorch or tensorflow). **Note** how they are downloading files into the runtime, using classes and inheritance to build models, preprocessing the dataset and the pipeline they follow.

Additional Sources

- You can do a **short course** (~4-6 hrs) on ML/DL/CV. I did one on DataCamp. Pro tip: If you have an institution linked Github account, then you can take a 3 month full subscription for free from the Github Student Developer pack.

- **Learn to Google.** It's the fastest way to learn! The Indian education system follows the top-down approach. You learn stuff and you give the test and that's what we are used to doing. I call this the learner-first approach which gels well where the overall intention is to learn. But when you are on a college or a project deadline, a better approach is the work-first approach. In this, you study and gain an 'idea' of the work you are going to do and then start. As you go along, you can study/refer etc as and when you encounter a problem. Googling, documentations, short youtube tutorials etc are meant for this bottom-up (ie work-first) approach. Refer to them as and when you require. Pro tip: If you find documentation hard to comprehend, open the documentation of a function you already know side by side; this would help you make an analogy with the new function you want to refer to.

Image to Latex conversion Problem

Study Materials

- How OCR (Optical Character Recognition) works https://moov.ai/en/blog/optical-character-recognition-ocr/
- https://wandb.ai/site/articles/image-to-latex
- Word-embedding: https://www.youtube.com/watch?v=5PL0TmQhItY
- Encoder-decoder model:
    - https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346
    - https://www.youtube.com/watch?v=BUqXqoUEmGI
- CNN: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
- RNN:
    - https://builtin.com/data-science/recurrent-neural-networks-and-lstm
    - https://machinelearningmastery.com/exploding-gradients-in-neural-networks/
- Attention:
    - https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html
    - https://blog.floydhub.com/attention-mechanism/
- Andrew NG courses are a real Gem! Greedy and Beam Search: https://www.youtube.com/watch?v=RLWuzLLSIgw&t=373s
- Implement the tutorial https://www.tensorflow.org/tutorials/text/image_captioning afterwards.
    - Implement variations of the tutorials like https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/
    - https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/

(my implementation is given in the progress report below)

Code out these variations if you have confusion regarding what-code-cell-does-what in the tensorflow tutorial. The code given for these two tutorials have bugs. Feel free to explore some better tutorials.

Useful Code snippets

- My GitHub repo containing some code snippets: https://github.com/Jahnvi13/Image-to-Latex
- Word-error Rate: https://holianh.github.io/portfolio/Cach-tinh-WER/
- Alternate pytorch model: https://github.com/whywhs/Pytorch-Handwritten-Mathematical-Expression-Recognition is the SOTA (I think) which achieves a WER of 0.17.

My Notebooks etc.

Tip: Do “Save a copy in Drive” to edit the notebooks.

- This Notebook achieves a WER of 0.6 (bad, I know) only because of the longer captions (>15 characters). https://colab.research.google.com/drive/1dFk2Jnmx2Qp0UoUtzhJkNuHSeSDeWz_L?usp=sharing
- This NB was used to generate some stats on the data https://colab.research.google.com/drive/14Wb5vpc-Gp0TTD6u7DomCI7eu6H0WU2H?usp=sharing
- A progress report done earlier by me containing various notebooks. https://docs.google.com/document/d/15yfciKdb_rS878HhhzHkPxmo3HchCmxEIH9jyOAaiuQ/edit?usp=sharing
- A presentation I made (yes, I am giving out everything to make you lazy :)) https://docs.google.com/presentation/d/1U9EmP2XO5Eb4nJZzCTnUnznSJ6ktMIPMmfm151rlR2c/edit?usp=sharing

Additional Tips

- You need not train the model everytime you start working. Save the model parameters as Checkpoints (refer the tensorflow tut) and resume from there.

- Just to make you aware, some functions have default parameters eg. `tf.keras.preprocessing.text.Tokenizer(num_words=top_k, oov_token="<unk>", filters='', lower=False)` gave us a hard time when it by default lowered all the characters!
- You can run the below loop to keep your colab runtime open for 5-6 hrs.

```
while(1):

    continue
```

Ending Note

Hope you enjoyed reading! Feel free to open a GitHub issue on https://github.com/Jahnvi13/Image-to-Latex if you have any questions, comments or suggestions.

A huge shout out to my mentor Tanay Karve for encouraging me and being patient in our debugging sessions. I thank him for his guidance and making the internship memorable!