Author: Yatharth Kumar
Entry Number: 2020CS10413

Competitive Part

For this part I have trained Donut model available on HuggingFace Transformers library. I have fine-tuned this model on both Synthetic and Handwritten Data provided to us. Donut is a model that is adept at visual document understanding tasks, such as visual document classification or information extraction (a.k.a. document parsing). I have modified the ground_truth appropriately for the text_generation task. Note that model has not been specifically trained for handwritten images to latex task.

The BLEU score obtained in this part is **0.28274.**

Acknowledgements and References for Competitive Part

I have taken reference from the following sources:

https://github.com/clovaai/donut
https://www.kaggle.com/code/s17elgho/fine-tuning-donut-for-document-classification
https://www.philschmid.de/fine-tuning-donut

Apart from few code snippets that I have used from the above acknowledged sources all the code have been written by me only.

Non Competitive Part

Encoder

The encoder uses a CNN to compute out the context vector. The input to this CNN is an image. Before feeding it into the network, the image is resized to 224×224 pixels and normalized. The output is the context vector which is fed to the decoder which is an LSTM cell that outputs tokens one after another.

The architecture is as follows.

- CONV1: Kernel Size → 5x5, Input Channels → 3, Output Channels → 32
- POOL1: Kernel Size → 2x2
- CONV2: Kernel Size → 5x5, Input Channels → 32, Output Channels → 64
- POOL2: Kernel Size → 2x2
- CONV3: Kernel Size → 5x5, Input Channels → 64, Output Channels → 128
- POOL3: Kernel Size → 2x2
- CONV4: Kernel Size → 5x5, Input Channels → 128, Output Channels → 256
- POOL4: Kernel Size → 2x2
- CONV5: Kernel Size → 5x5, Input Channels → 256, Output Channels → 512
- POOL5: Kernel Size → 2x2
- AvgPool2D: Window Size → 3x3 (Output size : 1x1x512)

I used ReLU as the activation function for all the layers apart from the Pooling layers.
For all pool and conv operations I have used the default size with no zero padding.

Decoder

The decoder is a LSTM cell that works by outputting the next token i.e x_{t+1} when x_t is given. An LSTM, or Long Short-Term Memory, is a type of recurrent neural network (RNN).

LSTMs have feedback connections(hidden layers) that make them capable of processing not just individual data points, but entire sequences of data. This makes them particularly well-suited for tasks involving sequential data, such as time series analysis, natural language processing, and speech recognition

I have used a single layer LSTM as the architecture of choice that takes the context vector as input and generates the latex formula. I have set the dimensions of LSTM class of pytorch with the following:

- Embedding Layer: → 512 (A learnable embedding for the output vocabulary)
- Hidden Layer: → 512 dimensions
- Output Layer: → Output Vocabulary size; transforms the hidden representation into the vocabulary space.

I created a vocabulary from the formulas in the training dataset and then initialised an embedding for each word/character in the vocabulary.

Due to upcoming placements and unability to find a suitable teammate, I could not complete Non Competitive part. I implemented Decoder and Encoder class only.