Transforming Hand-Written Equations to Latex

COL774: Assignment 4

Sem I, 2023-24

Due Date: Monday November 27 (2023), 11:59 pm. Total points: 100

Notes

- This assignment has two parts: non-competitive and competitive.
- You should submit all your code (including any pre-processing scripts written by you) and any graphs that you might plot.
- Include a write-up (PDF) file, consolidated for both parts, which includes a brief description for each question explaining what you did. Include any observations or model details required by the question in this single write-up file.
- **You should use Python as your programming language and PyTorch as your deep learning framework.**
- **Your code should have appropriate documentation for readability.**
- You will be graded based on what you have submitted as well as your ability to explain your code. Additionally, in competitive part, you will be graded based on your model performance relative to the class
- Refer to the Piazza for assignment submission instructions.
- This assignment should be done in groups of in groups of 2. If you are really keen on doing it individually, talk to us. You should carry out all the implementation by yourself (i.e., in your group).
- Since the assignment consists of a competitive part, no late submissions are allowed.
- For the competitive part, you will submit your predictions on a leaderboard, which will display your score and your ranking with respect to other submissions in the class.
- We plan to run Moss on the submissions. We will also include solutions from the internet to maintain integrity. Any cheating will result in a zero on the assignment, an additional penalty equivalent of the weight of the assignment and possibly much stricter penalties (including a fail grade and/or a DISCO).

1 Problem Statement

You are given two datasets of images which contain mathematical expressions and its corresponding latex formula. The first dataset contains handwritten mathematical expression in the images and the second contains the images which showcase mathematical expressions that have been generated from LaTeX-based code(synthetic dataset).You are provided with both train and test set for both the datasets. Your task is to train an ML model that takes the image of the mathematical expression into input and outputs the corresponding latex code.

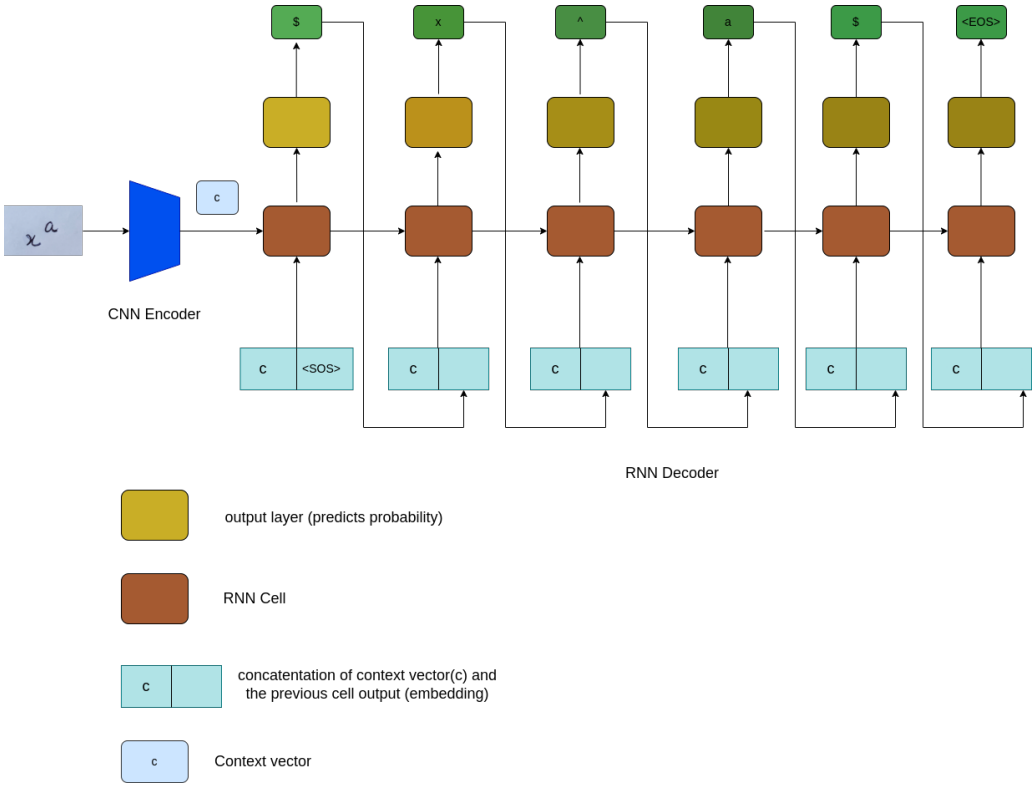

Figure 1: Model architecture

2 Non-competitive part (40 marks)

In this part,you will be using an **Encoder-Decoder architecture** for modelling this problem. An encoder is used to encode the given input into a context vector which is further used for decoding by the decoder. The decoder is applied on this context vector to generate the sequence auto-regressively(one word/character at a time). Your task is to implement an encoder which will take the image of the mathematical expression as input and the decoder which will output the latex formula for the provided expression.

You have to implement this part of the problem in two subparts:

- **Part-a** You should only use the synthetic training dataset for your training and report the BLEU scores on validation set of handwritten and both test and validation of the synthetic dataset.
- **Part-b** You should train your model on the synthetic dataset and then finetune the same trained model on the handwritten dataset. Report the BLEU scores of the model on validation set of handwritten and both test and validation of the synthetic dataset.

You can use the following as a starting point:

1. **Encoder:** In this part you have to implement a simple CNN which takes as input an image and returns a context vector to be used by the Decoder. Make sure to resize the image to (224, 224) and normalise it.

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

   Use ReLU as the activation function for all the layers apart from the Pooling layers. For all Pool and Conv operations use the default size with no zero padding.

2. **Decoder:** Use a single layer LSTM as the architecture of choice that takes the context vector as input and generates the latex formula. Set the dimensions of LSTM class of pytorch with the following:

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

   You will first have to create a vocabulary from the formulas in the training dataset and then initialise an embedding for each word/character in your vocabulary. Since each formula can be of varying length, use padding to make all list sizes consistent.

3. **Training strategy:** Use cross-entropy as your loss function and teacher-forcing for training the decoder. Don't forget to use a **START** and **END** token to allow for variable length formula generation from the decoder. When passing input to the LSTM cell, you need to concatenate the context vector with the learned embedding of the output of the previous timestep. This can be done in two phases:

    - First, you can use teacher forcing where you will concatenate the context vector with the embedding of the ground truth label of the previous timestep.
    - Second, you can use the learned embedding of the output of the previous timestep and concatenate it with the context vector and treat it as the input to the current timestep.
    - Use teacher forcing for 50% of the time during training.
    - Hint: Embedding Class of pytorch can be used to maintain and learn the embeddings of labels in the output vocab.

4. **Metric Scores** You will be graded on BLEU score. You can read about it in Section 5.3 of this pdf document.

3 Competitive part (60 marks)

In this part, you are free to train any model architecture of your choice. The score for this part will depend on your performance relative to other groups in the class. **You can use both(handwritten and synthetic) datasets provided for training**. You should be submitting your predictions only for the `sample_submission.csv` on kaggle in the same order. You are free to use any generic pre-trained models or embeddings, but you are not allowed to use any task-specific models available on the web. You can use standard open-source libraries like Torchvision, Torchtext or Hugging Face. **For using other libraries or if in doubt, please clarify with the instructor or TAs**.

You are not allowed to download any additional training examples from the internet, you should only work with the given dataset. **Any augmentations/preprocessing you do must be in your training scripts**. **We may train your model again using your submitted script and any significant deviations will be severely penalised**. (Please make sure to minimize any randomness in your models by initializing seeds etc. so that the models can be replicated if needed)

The kaggle contest is hosted here. To access the competition, please look out for the invitation link on piazza. The final score will depend on ranking in both the public leaderboard (consisting of 45% test set) and private leaderboard (consisting of 55% test set). The private leaderboard will be visible only after the assignment deadline.

4 Submission instructions

The submission will consist of

1. **Kaggle competition**: Register for the competition and form a team. Upload the test predictions directly as per the instructions given on the competition page. Please note, you will not receive a score in the competitive part if your submission is not visible on the leaderboard.

2. **Moodle submission**: Only one submission per group is needed. Upload all python files along-with the writeup in a zip file named `<entrynum1>_<entrynum2>.zip` or `<entrynum1>.zip` depending on the group size. There should be the following files (alongwith the writeup pdf, and any other additional python files as needed):

    a. *requirements.txt*: Make sure to include the pytorch version, torchvision version etc here. Any non-trivial libraries you use should be here to make sure there is no version mismatch while running your code. This file would be run as `pip install -r requirements.txt`.

    b. *part1a.py*: Would be run as `python3 part1a.py <dataset_dir_path>`. This should train your encoder-decoder architecture and generate a file `<pred1a.csv>` and `<pred1b.csv>` containing two columns **image_id** and **formula** containing the test predictions for handwritten test dataset and synthetic dataset respectively.

    c. *part1b.py*: Would be run as `python3 part1b.py <dataset_dir_path>`. This should train your encoder-decoder architecture and generate a file `<pred1a.csv>` and `<pred1b.csv>` containing two columns **image_id** and **formula** containing the test predictions for handwritten test dataset and synthetic dataset respectively.

    d. *comp.py*: Would be run as `python3 comp.py <dataset_dir_path>`. This should train the model used in the competitive part and generate a file `<comp.csv>` containing competitive test set predictions.

5 Important links

- Kaggle link: https://www.kaggle.com/t/67386ef9d1d04b1aa84cac6d79aec269