Predicting SVG codes from Images

Vincent Cai

In this project, the goal to train a model on the training images as inputs and svg codes as outputs. At test time, the trained model is expected to predict the corresponding codes from the test images. I approached this task as a similar problem to image captioning, here is how I solved the problem step by step:

1. Importing images

I imported the training and test images with Matplotlib and resized each of them to the shape of (32, 32, 4) with OpenCV. This has accelerated the training process significantly without substantially compromising the prediction performance.

2. Importing and processing svg codes

I imported the svg codes as text files and read them as strings. Since the first part of each string is identical, only the second part that starts with "*<defs*" and ends with *"/></svg>"* was extracted.
I then used the *Keras Tokenizer* to map each word (with punctuations) to an index, and subsequently converted each extracted string into a sequence of indices.

3. Preprocessing inputs and outputs for training

Let's take a sequence of 20 indices and its corresponding image as an example. The first index will be paired up with the image to predict the second index, the sequence of the first two indices will be paired up with a duplicate of the same image to predict the third index. This operation repeats until the sequence of the first 19 indices is paired up with a duplicated image to predict the last index. Therefore, the inputs consist of pairs of image and sequence while the outputs consist of indices.

4. Building the model

The model consists of a spatial encoder, a temporal encoder, and a combined decoder. For the spatial encoder that encodes the images, two *Conv2D* layers (16 and 32 filters with size of 3 and strides of 2) are used, followed by a *Flatten* and a *Dense* layer (128 neurons). For the temporal encoder that encodes the sequences, an Embedding layer (mapping a dictionary of size 47 to a dimension of 8) is used, followed by two *LSTM* layers (128 neurons each, first layer with *return_sequences* set to *True*) and a *Dropout* layer (rate of 0.05) in between. A decoder is then built to decode the combined outputs of the two encoders into the probabilities of each index, this is done by having two Dense layers (128 and 47 neurons). *Relu* activation is used for all the *Conv2D* and *Dense* layers with an exception of using *softmax* for the last *Dense* layer. The model is then built by chaining the inputs of the two encoders and the outputs of the decoder. Finally, the model is compiled with *categorical_crossentropy* as its loss and *adam* as its optimizer.

5. Training the model

The way I preprocessed the data has resulted in a much larger training set than the original one. This is a limitation of the proposed approach. Initially, I trained the model on Google Colab with TPU, the maximum number of images could be processed was 32000. To use more training data, I then trained the model locally on my MacBook with CPU. The training was done with 7 epochs, each took around 5000 seconds, 46000 samples were used for training and the remaining 2000 for validation.

6. Making predictions

After the model is trained, each pair of image and sequence can be used to predict the next index, this index is then mapped back to its corresponding word and added to the end of the previous sequence for the next prediction. Initially, each image is paired with the word "*<defs"* to predict the second word, and the loop goes on until the word *"/></svg>"* is predicted. A CER of 0.1195 was achieved by this model.

Conclusion

Driven by a lower CER, I trained models with various depths and sizes as well as regularization methods such as *SpatialDropout2D* for the CNNs and *Dropout* and *L2 regularizer* for the Dense layers. Results have shown that the simplest model described above performed the best in solving this particular task.

References

Lamba, H. (2019, February 17). Image Captioning with Keras. Retrieved October 15, 2019, from https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8.

Miteshputhranneu. (n.d.). MITESHPUTHRANNEU/Image-Caption-Generator. Retrieved October 15, 2019, from https://github.com/MITESHPUTHRANNEU/Image-Caption-Generator.