ML – Image Captioning

In this notebook, a deep learning image captioning algorithm was implemented. The final model is capable of generating descriptive captions for a given image input.

Data Processing

The dataset used in this notebook was the Flick8k dataset from Kaggle [1]. This consisted of 8000 images with varying depictions, each described by a set of five captions. This required data processing for both captions and images.

Text Processing

Once loaded from file, captions were converted to lowercase. Any punctuations and excess whitespace were removed, and '<start>' and '<end>' tokens were added to the beginning and end of each caption, respectively. Similar to what was done in [3], a TextVectorization layer was initialized and adapted on the captions, with a maximum vocabulary size of 5000 words and a maximum caption length of 30 words.

Image Processing

To vectorize the images, the EfficientNetV2B3 CNN model [2] was initialized using pre-trained imagenet weights, and the output layer was removed.

Building the Dataset

For each image-caption pair, an image was loaded, and the pre-trained model was used to generate a vector representation of the image. Using a dictionary format the image vectors were assigned to the encoder input, and the input and label captions were first tokenized using the initialized TextVectorization layer before being assigned to the decoder input and output respectively. The only difference between the tokenized input and label captions was that the '<end>' tag was removed from the input caption and the '<start>' tag from the label.

The data dictionary was then converted into a tensor dataset using the Tensorflow data API, and the data was shuffled, batched and prefetched. This was done after the data was split using scikit-learn's train_test_split with an 80-20 split for the training and validation, respectively. This ensured that images in the training set did not appear in the validation set.

Model Structure

As shown in Figure 1 below, the model is made up of an encoder-decoder structure. The encoder is made up of an input, dropout, and dense layer. The core decoder layers consist of five successive Gated Recurrent Unit (GRU) layers, preceded by an input, embedding and dropout layer, and followed by two dense layers using 'tanh' and 'linear' activation functions respectively.

The model is then built with a Functional API approach similar to what is employed in [4] using the 'encoder_in' and 'decoder_in' layers as inputs and the 'decoder_out_l2' layer as output.

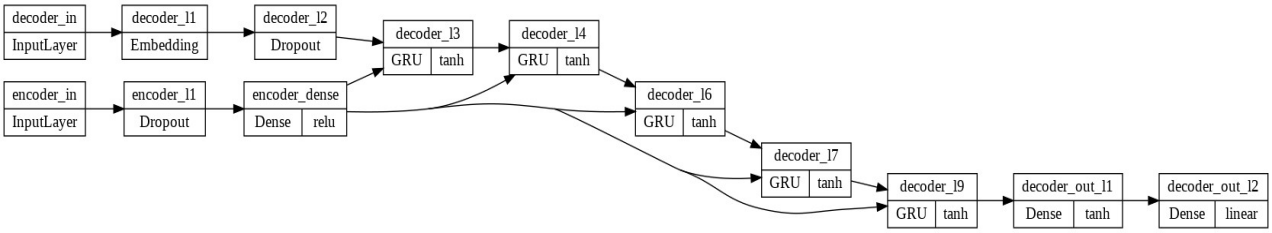

Figure 1: Encoder-decoder structure used for image captioning

Training

The final model was compiled using the Adam optimizer with a learning rate of 0.001, the sparse categorical accuracy metric provided by Tensorflow [5], and a custom sparse cross-entropy loss function [6] which employed sparse softmax cross-entropy. The model was then fit on the data with 20 epochs and using Keras's learning rate reduction [7] and early stopping [8] callbacks.

Conclusion

In summary, the notebook includes:

- Use of the Flick8k dataset,
- Use of a pre-trained CNN model, namely EfficientNetV2B3 with imagenet weights,
- A novel five GRU approach for building a model trained with 20 epochs,
- Use of two Tensorflow callbacks, namely learning rate reduction and early stopping,
- Use of Tensorflow's data API for building, shuffling, batching, and prefetching the dataset,
- Use of scikit-learn's train_test_split function for splitting the dataset with 80% for training and 20% for validation,
- A plot of both accuracy and loss over time for the training and validation data,
- Use of a dropout of 0.5 in the network,
- Model validation at each epoch,
- Captions generated for five non-dataset images.

References

[1] https://www.kaggle.com/adityajn105/flickr8k

[2] https://www.tensorflow.org/api_docs/python/tf/keras/applications/efficientnet_v2/EfficientNetV2B3

[3] https://www.tensorflow.org/tutorials/text/image_captioning#preprocess_and_tokenize_the_captions

[4] https://blog.paperspace.com/image-captioning-with-tensorflow/

[5] https://www.tensorflow.org/api_docs/python/tf/keras/metrics/SparseCategoricalAccuracy

[6] https://androidkt.com/image-captioning-using-tensorflow-high-level-api/?fbclid=IwAR0SMjYSUsTZ-_1tZMR1rPsLYk82xaVQTLTo8sHVMKiSg32Zx_fs7EB-nRs

[7] https://keras.io/api/callbacks/reduce_lr_on_plateau/

[8] https://keras.io/api/callbacks/early_stopping/