Folder Name: Image_Captioning

Overview:

Keras/Tensorflow Image Captioning application using CNN and Transformer as encoder/decoder. In particular, the architecture consists of three models:

- A CNN: used to extract the image features. In this application, we used *EfficientNetB0, EfficientNetB1, EfficientNetB2, EfficientNetB3, EfficientNetB4, EfficientNetB5, ResNet50, ResNet101*, and *MobileNetV2* models which are pre-trained on "*imagenet"* dataset.
- A TransformerEncoder: the extracted image features are then passed to a Transformer encoder that generates a new representation of the inputs.
- A TransformerDecoder: it takes the encoder output and the text data sequence as inputs and tries to learn to generate the caption.

Dataset:

The model has been trained on the "*2014 Train/Val COCO*" dataset. You can download the dataset from (*https://cocodataset.org/#download*). Note that test images are not required for this code to work. We will later split the valid data into valid and test data. Original dataset has **82783 train** images and **40504 validation** images; for each image there are a number of captions between 1 and 6. Preprocessing is done on the dataset to keep only images that have **exactly 5 captions**. After this filtering, the final dataset has 68363 train images and 33432 validation images. The training and validating dataset is saved as json file which can be found here:

training json = "*COCO_dataset/train.json*"
validating json = "*COCO_dataset/valid.json*"

Each element in the "*train.json*" file has the following structure:

"COCO_dataset/train2014/COCO_train2014_000000318556.jpg": ["caption1", "caption2", "caption3", "caption4", "caption5"], ...

Same structure goes for "*valid.json*" file:

"COCO_dataset/val2014/COCO_val2014_000000203564.jpg": ["caption1", "caption2", "caption3", "caption4", "caption5"], ...

Dependencies:

We have used the following versions for the code to work:

- python==3.8.8
- tensorflow==2.4.1
- tensorflow-gpu==2.4.1
- numpy==1.19.1
- h5py==2.10.0

Training:

The training can be performed with following steps:

- First it is needed to make sure that the training images are in the folder "*COCO_dataset/train2014/*" and that validation images are in "*COCO_dataset/val2014/*".
- Modify the parameters in the "*settings.py*" file according to your need.
- Start the model training with *python training.py --model "model_used"*
- Different models can be selected and passed as *--model* arguments from the above mentioned models. Default model is set to "*EfficientNetB0*".
- The trained weights and relevant info are saved in the "*save_train_dir*" directory.

Data Augmentation:

Data augmentation is performed on the training set to better generalize the result. It can be found in "*dataset.py*" file.

```
trainAug = tf.keras.Sequential([
    tf.keras.layers.experimental.preprocessing.RandomContrast(factor=(0.05, 0.15)),
    tf.keras.layers.experimental.preprocessing.RandomTranslation(height_factor=(-0.10, 0.10), width_factor=(-0.10, 0.10)),
    tf.keras.layers.experimental.preprocessing.RandomZoom(height_factor=(-0.10, 0.10), width_factor=(-0.10, 0.10)),
    tf.keras.layers.experimental.preprocessing.RandomRotation(factor=(-0.10, 0.10))
])
```

You can customize your data augmentation by changing this code or disable data augmentation by setting *TRAIN_SET_AUG = False* in "*settings.py*".

Inference:

For inferencing you can train your own model or download our pre-trained weights from here "*https://www.kaggle.com/datasets/alifrahman/image-captioning-trained-model*" and place the contents inside the "*save_train_dir*" directory. All the models and their weights are stored here. **Remember to pass the used model name as argument while running the inferencing script.**

You can perform the inferencing in two ways. If no specific image file path is passed as argument, the code will load the "*test.json*" file, perform prediction on each image and store the prediction as "*prediction.json*" for later use.

And if an image file path is passed then the trained model will predict the caption as show the output in terminal.

Inferencing steps :

- Check the parameters in the file "*settings_inference.py*". Default values are okay if your trained weight is not saved in another location.
- Run *python inference.py --image="image_path_file" --model "model_used"*

Evaluation Metrics:

Three evaluation metrics are used here. They are *"BLEU", "ROUGE"*, and *"METEOR"*. For running the ROUGE evaluation metric you need to install a library "*pip install rouge_score*" and for running the BLEU and METEOR, run "*pip install nltk*". Now run the evaluation scripts using following commands:

- *python BLEU.py*
- *python ROUGE.py*
- *python METEOR.py*

Training Environment:

- CPU: Amd 7745hx
- RAM: 16 GB
- GPU: Nvidia 4060 8 GB VRAM