Deep Learning Project 2 - MSCOCO

Goh Yu Jin (1002157)
Singapore University of Design and Technology
yujin_goh@mymail.sutd.edu.sg

Clemence Goh (1002075)
Singapore University of Design and Technology
clemence_goh@mymail.sutd.edu.sg

Ronald Heng (1002094)
Singapore University of Design and Technology
Ronald_Heng@mymail.sutd.edu.sg

ABSTRACT

Image captioning trained on captioned images from MSCOCO 2014 dataset. The MSCOCO 2014 dataset which contains images and annotations of questions and corresponding answers for each image. The goal of the project would be to generate captions on new unseen images using a deep learning model. We used a Densenet-161 architecture and fed its outputs into a single layer Long Short Term Memory (LSTM) Cell. Through training a deep learning model, we have been able to successfully generate relevant captions on new images.

KEYWORDS

image, mscoco2014, caption, generation, LSTM, Densenet, Embedding

1 INTRODUCTION

We have been tasked to develop a deep learning model capable of captioning images from the MSCOCO 2014 dataset, which contains images and annotations of questions and corresponding answers for each image. Through the use of transfer learning, we are to develop a model that could accurately caption a given image.

We are therefore, required to produce the following items:

(1) a dataloader for MSCOCO train and val datasets
(2) an Encoder CNN and a Decoder LSTM
(3) loss for minimizing
(4) average accuracy measure on validation set
(5) a GUI which allows an input image to be captioned

2 LIBRARIES

Developing the model has required us to use the following libraries:

(1) pytorch 1.0.1.post2
(2) numpy 1.16.2
(3) Pillow 6.0.0
(4) torchvision 0.2.2.post3
(5) nltk 3.4.1
(6) pycocotools 2.0.0
(7) matplotlib 3.0.3

3 VOCABULARY

Utilizing the vocabulary extractor created by yunjie, we were able to customize our own vocabulary.[1] The file took in the captions of all images in the annotation files and counted the frequency of every word that appeared. We were given the option of selecting the threshold frequency for a word to be included in our vocabulary. Thus, we had set the threshold value to the default of 4. This left us with a vocabulary size of 9000 to be embedded in the embedding layer.

4 RESIZING

All images are resized using the Resize script provided by yunjie with minor edits.[1] Instead of resizing to a square, all images in the training and validation set are resized such that the shorter dimension is 256 pixels while keeping the same aspect ratio. This ensures that the image is not distorted and the object of interest is more likely to be present after a random crop.

5 DATALOADER AND DATASET CLASS

For the purpose of our project, a Pytorch Dataset class developed by yunjie was used for it. [1] This dataset class will take in the json files (already split by training and validation) containing annotations from the MSCOCO dataset. The annotations then contain the file name of the respective image and its caption. At initialization, the dataset will extract out all of the file names listed in the annotation file and its respective caption. When the data loader is called, a few modifications is performed to the retrieved data. Firstly, Transformations and augmentations will be made to the PIL image retrieved. Next, the vocabulary index for a start symbol, "<start>" will be appended to the caption list before appending the rest of the vocabulary indexes for the words in the captions. Afterwards, an "<end>" vocabulary index will be appended to the end of the caption. Finally, both the image and caption list containing vocabulary indexes are returned. The dataloader class will load in the data from either the training or validation set. When called, it will shuffle the data to ensure a more representative set of datapoints is used each time.

6 MODEL

6.1 Model Architecture

From our experiments we have decided to use a Densenet 161 model for the Encoding CNN architecture which takes in an input size of 224 by 224 with a RGB channel. The final layer of the Densenet will have its weights reset and output dimension set to an embedding of 256 features. The 256 features will be passed into the Decoding single layer LSTM architecture with its hidden state and memory reset. Afterwards, the LSTM layer will be trained to generate the caption provided. This is done by feeding the output of the LSTM into a word embedding layer and then back into the inputs. The weights of the LSTM are trained based by minimizing the loss function of the output word and the word that was provided in the ground truth caption.

6.2 Data Augmentation

Data Augmentation was used to make the model more robust to similar inputs. Thus for the training set two different augmentations

Channel	Mean	Standard deviation
1	0.485	0.229
2	0.456	0.224
3	0.406	0.225

Table 1: Normalization parameters

were included, specifically being random crop and horizontal flips. Random crop and horizontal flips were adjusted so that the classifier would be able to learn from target classes at different positions on the inputs.

Both training and validation images were randomly cropped at different positions of the image to a size of 224 by 224 pixels.

Both images were transformed into tensors before applying a normalization with parameters according to the MSCOCO 2014 dataset as listed in Table 1

6.3 Optimizer and learning rate scheduler

The ADAM adaptive gradient descent algorithm was used to optimize the models parameters by updating the weights in the direction of the steepest gradient descent. A initial learning rate of 0.001 was set.

6.4 Loss

The Cross Entropy Loss was the loss we had used for our model to determine how accurate our model was in predicting the output. The Cross Entropy Loss was used as the outputs were categorical, since a word vector could only represent a single word at anytime. We apply a cross entropy loss function on the individual outputs of the softmax outputs on the Decoder RNN together with the ground truth labels. The loss function is then obtained to be minimized by the optimizer.

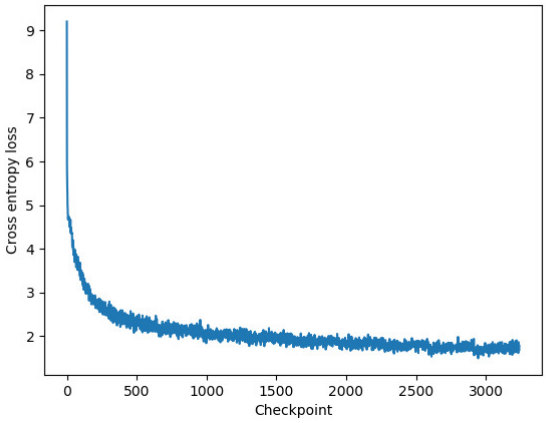

Figure 1: train loss vs checkpoints

6.5 Bleu Score

Bilingual Evaluation Understudy Score, for comparing a candidate translation of text to one or more reference translations. The Bleu score allows us to better calculate how well the decoder is producing captions as compared to the ground truth, with a min score of 0 and max score of 1. Since the ground truth captions are not classification problems, precision and accuracy are less relevant and bleu scores have to be used instead. The average Bleu score from eval.py based on this encoder and decoder stands at 0.22.

6.6 Failed Captions

Figure 2: Failed Caption 1 - a bench sitting in a flooded area next to a tree.

The original caption - "wintery scene of a bench by a mountain lake"

Figure 3: Failed Caption 2 - a man in a wheelchair is talking on his cell phone.

The original caption - "a man and woman riding on the back of a motorcycle."

7 GUI

7.1 Single Image Captioning

The GUI accepts an uploaded image from the user and runs it through the models loaded with the state dict from the trained Encoder. It then makes use of the trained Decoder as well as the vocab from training to produce a caption that best represents the image given in the form of a sentence, and displays it on the right of the image on the GUI. For the purposes of the demonstrator, the encoders and decoders used are as follows:

(1) encoder-10-3000.ckpt
(2) decoder-10-3000.ckpt
(3) vocab.pkl

An example of how to use the GUI is shown below:

Figure 4: Initial GUI

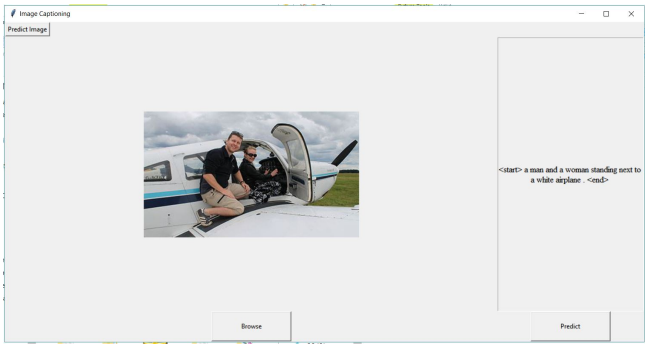

Figure 5: Caption with pilots

7.2 Contributions

Modifying the model - Yu Jin, Clemence, Ronald
Evalution script for trained model - Yujin, Ronald
Graphical User Interface - Clemence
Report - Yu Jin, Clemence, Ronald

REFERENCES

[1] Yunjey. 2018. Image-captioning. (Jun 2018). https://github.com/yunjey/pytorch-tutorial/tree/master/tutorials/03-advanced/image_captioning