Practicum 2 Memo - Team Anonymous

Jason Zhang, Erin Yang, Wenjie(Bianka) Gu, Shaoling Han

Video link:
https://drive.google.com/file/d/1mAJsGexhG-7dH5o3QVrWXSEH5uknhaiL/view?usp=sharing
Github Repository:
https://github.com/hanshaoling/AC295_TeamAnonymous/tree/master/submissions/practicum2_TeamAnonymous

In this practicum, we build a multimodal deep learning model to perform visual question answering tasks. Our teacher model takes advantage of transfer learning on both image and text data, and it achieves a 47% accuracy on the validation data. For additional features, we convert our dataset into **TFRecords**, apply **distillation** and **pruning** to compress the final model.

Data Pre-Processing

After downloading the small dataset, we first read the question and answer json files and merge them together. We then subset our dataset by only choosing the top 10 answers in both the train and validation dataset. Here is the final format of our dataset:

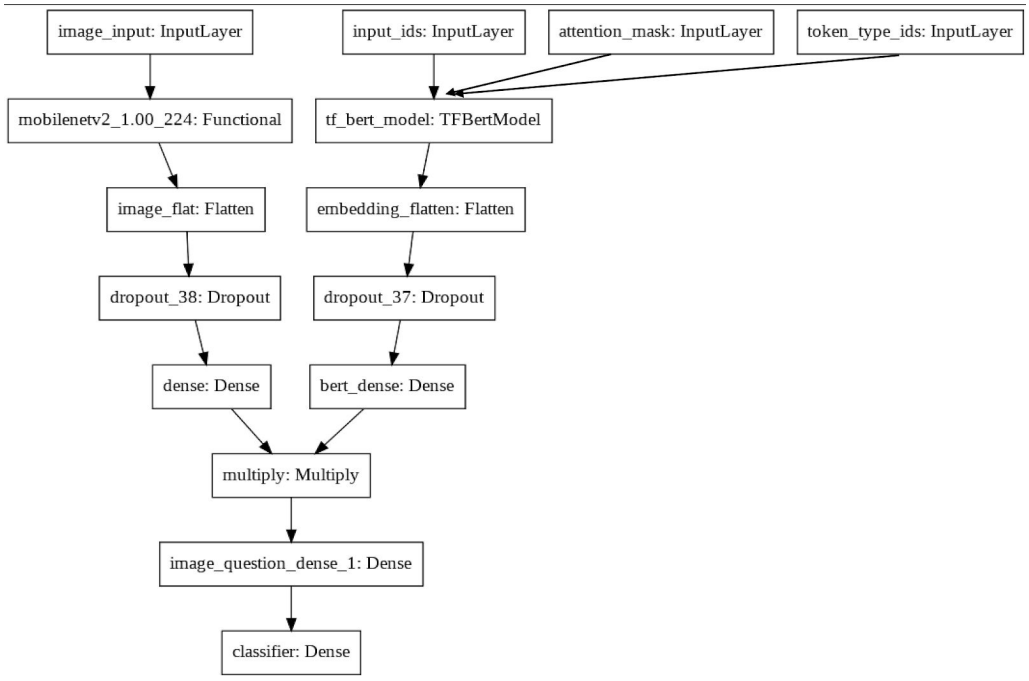
	image_id	question	multiple_choice_answer
3	458752	Is this man a professional baseball player?	yes
4	262146	What color is the snow?	white
6	262146	What color is the persons headwear?	red
8	524291	Is the dog waiting?	yes
10	393221	Is the sky blue?	yes

We then start to create our TF dataset. For the image input, we read in the image by its corresponding image_id and then resize and normalize the image. For the question inputs, we use Pretrained Bert tokenizer to tokenize our input questions and convert them to Bert required input format. We then divide our dataset into batch size of 64 and shuffle them. The graph below is our final input format:

```
train_data <BatchDataset shapes: (((None, 224, 224, 3), {input_ids: (None, 100), token_type_ids: (None, 100), attention_mask: (None, 100)}), (None,)), typ
validation_data <BatchDataset shapes: (((None, 224, 224, 3), {input_ids: (None, 100), token_type_ids: (None, 100), attention_mask: (None, 100)}), (None,))
```

Implementation of Full Model

The full model consists of both pre-trained BERT model to process the question text and mobilenet v2 to extract features from images. We define four inputs here, three of them are input_ids, attention_mask and token_type_ids generated from Bert Tokenizer. We concatenate them and feed it into Bert encoder. The other input is pre-processed images, and we feed it into mobilenetv2. To avoid overfitting, we apply dropout after flatten the output from pre-trained models. We then reduce both layers to the same dimension and apply pointwise multiplication to combine them before adding the classification head.



The model is able to achieve 47% accuracy on the full validation data after 10 epochs.

```
[ ]  model.evaluate(validation_data, return_dict=True)
```

```
1721/1721 [==============================] - 562s 326ms/step - loss: 0.9013 - sparse_categorical_accuracy: 0.4712
{'loss': 0.9013293385505676,
 'sparse_categorical_accuracy': 0.47118231654167175}
```

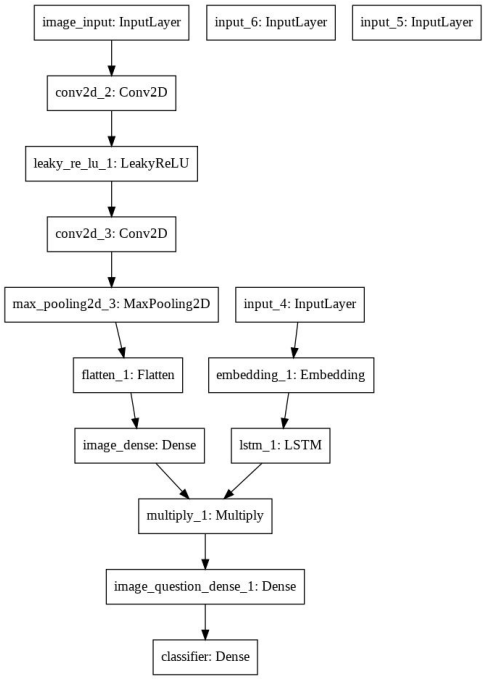
Additional feature 1: TFRecords

Our team also explored using TFRecords to train our model. After preprocessing our image and questions, we tried to directly save the dataset to TFRecords but we encountered a problem that we could not directly convert image and tokenized text tensors to the format that can be accepted by TFRecords. To solve this problem, we used `tf.io.serialize_tensor` to convert tensor to binary strings and then the conversion worked. After that, we directly read our data from our saved TFRecords and used `tf.io.parse_tensor` to convert these binary strings back to the tensor flat. We then shuffled the data and divided our data into batches like before and fed the input into our VQA model. Please refer to our Tfrecords notebook for more details.

We find that the TFRecords is a quite useful tool. We can just easily share the tfrecord file and provide the processing function then everyone can start to train their own model without even downloading the data.

Additional feature 2: Apply distillation to compress the final model

In addition to the original model, our team utilized the distillation method to compress the model. The student model consists of a text model including an embedding layer and a LSTM layer for question text processing, and a 2-layered CNN for the purpose of image processing. Same as the original model, we combined the outputs from image model and text model by pointwise multiplication and then fed the product into a classifier. The student model architecture is shown as follows:



Notice that the input of the student model is the same as the original model for the purpose of distillation. But only the input_id was fed into our student model. The attention_mask and token_type_ids were not integrated into the student model.

We then tuned the hyperparameters alpha and temperature and found the optimal alpha and temperature were 0.9 and 15 respectively. We then built a Distiller class object setting student loss function to be SparseCategoricalCrossentropy, and distillation loss function to KLDivergence. The distilled student model achieved an accuracy of 37% on the full validation data after training for 10 epochs. The model size, accuracy, and inference times of the student model and the teacher model are summarized below:

	Total Parameters	Model Size(MB)	Inference Time(s)	Accuracy(%)
distilled_model	14775626	59.138264	116	36.98
teacher_model	128002218	512.490624	562	47.12

Additional feature 3: Apply pruning to compress the final model

Our team applied pruning to compress the model. Since subclass model is not supported in tensorflow pruning, we try two alternatives: pruning only convolution layers in Mobilenet part, and pruning the whole Mobilenet.

	Model	Model Size (kb)	Model Accuracy	Inference Time (s)
0	Before pruning	474306.05	0.471182	561
1	Only Conv2D pruning	472016.47	0.460029	553
2	Mobilenet pruning	471959.20	0.444254	551

Pruning the mobilenet part does not compress model size or shorten inference time significantly, yet pruned models retain relatively high accuracy. The reason could be that mobilenet is quite optimized and pruning on it doesn't make a big difference.