CSCI 5922 Fall 2022–Lab 4

Abulitibu Tuguluke

October 31, 2022

Validation set

Method

In this lab, we used VizWiz data set[1], which consists of 4319 validation set, 8000 test set, and 20523 training set. Due to memory bottleneck, we wull only use 1500 data from the training, and 600 from validation set for all three models during model selection process, and use 10000 training data + 600 validation data for the final model on 1000 test set. Each data point consists of iamge, question answers, answer_type, and answerable. We used image for the image input and question for the text input, for label, we choose answerable, the problem can be described as 2-modal binary classification with deep learning.
The three neural networks are RNN, LSTM and GRU respectively. After downloading the data, first we extract the text and image information, 2 pre-trained models were selection for this task: spacy's web-md(word2vec) model for text vectorization, and vgg16 from keras for image extraction, each question then tokenized into (96,1) vector, in order for image to 'fit' our model,right after the vgg16 preprocess for each image,]we first resized each image into (1,96,96,3) tensor so that to combine text info and image info into one, we then reshape it to (288,96) matrix, the reason is to make multiplication of image matrix with text vector doable: $(288, 96) \times (96, 1) = (288, 1)$ is the final input we need to feed into the training. For lable, we keep the original $0, 1$ form. Once we got $X\_train, X\_val, y\_trian, y\_val$ ready, we can then design 3 models. Our aim is to study the three deep architecture's effects on VizWiz problem, while keeping the hyperparameters(batch/epoch size, acivation functions and optimizer) the same, keep in mind the neural with each layer differ model to model.
RNN architecture consists of one simplernn and one dense layer, where the hyperparameters are set to 10 batch with 10 epoch, where the activated function for dense layer is sigmoid, and optimizer set as old adam. LSTM has two layer with the same hyperparameters,batch/epoch, activation function along with optimizer. GRU has 2 GRU layer (setting input shape as $(288, 1)$) , the setting is the same as LSTM: 10 batch with 10 epoch, where the activated function for dense layer is sigmoid, and optimizer adam.After 10 epoch of training on 1500 training set ,2e calculated average precision for all three models using sklearn matrix at the end with 600 validation set .
The hardware for this lab is Amazon SageMaker'sG4 instance, 4CPU + 1 GPU with 16G memory. Next let us discuss the results.

Result

In our experiment, we consider unanswerable as 1, and answerwable question as 1. Hence, the average precision calulation has to consider the mean between 0 and 1 lable, after the training with three model, we used average_precision_score methd to calculate the prediction of 600 validation set, what we got is as follow:
At the first glance, we probably will use RNN since it has the highest score, yet if we look at the accuracy

RNN: 67.2353	LSTM: 67.0013	GRU: 66.8333.

and loss for all three after 10 epoch, RNN: loss: 0.5951 acc: 0.7113, LSTM 0.5540 - acc: 0.7313, GRU: loss:

[1] `https://vizwiz.cs.colorado.edu/VizWiz_visualization_img/`

0.5812 - acc: 0.7200. Our bet should be on LSTM since it has lower loss and higher accuracy. Next, let us analyze the results.

Analysis

We conducted these experiments to study how to extract features from text and images for mult-modal classification problem, while also study the effect of different deep architects effects on the same model performances. From the above results and tables, we can see that by simply changing the layer from RNN to LSTM and GRU will not improve the model performance significently. The reason for this is not because of the 'wrong' architectures (as shown in lab 3, where LSTM would improve the accuracy embedding text classification by 2 times), but mostly how we extract the feature from text and image with pre-trained model, and combine them, there are not solid mathematical reason to just combine two input or multiply, simply for the sack of getting on input set, it is totally legit to treat them seperately, e.g.: building 2 seperate models and combine them at the end, or even add them separately during training for different layer. As we can see in Figure 1, that input overwhelmingly learned as a 1(answerable), while ignoring the feature from unanswerable images. We also ruled out the assumption for unbalanced training set, where it did have enough 0 labeled data.
We also learned that training with less batch size and mroe epoch, will initially increase the accuracy, but will lead to no significant improve of learning in the long run. We believe the reason is the same for architecture: enbedding of the inputs is not ideal, causing none change in loss during training by predicting validation as answerables.
Therefore, using the full dataset, plus data augmentation on both the images(flip iamge in 4 direction) and

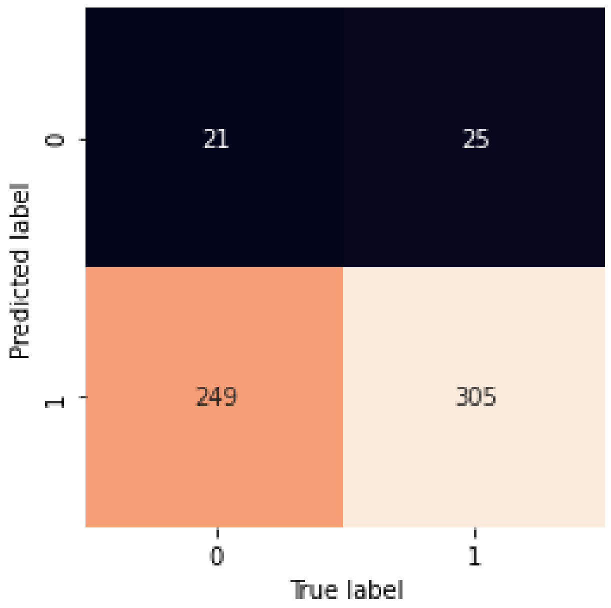

Figure 1: LSTM's confusion matrix.

questions(add new questions by replacing some words with random words in old questions) data, we believe it would immediately creates better results. According to this repo[2], the correct algorithm for this problem is LXMERT, yet I failed at replicating their experiment with keras, taht is why I cling to my old models from previous labs.

[2]`https://github.com/airsplay/lxmert`

Test set

Method

For second part, we used the same VizWiz data, with more training set (10000 in total), due to time/memory, we did not train on all 20523, but we did add 600 validation set on top pf 10000. As for the model, we picked LSTM, yet made changes the architecture, the reason is 'overfitting' issue we found after plotting the loss between train and validation (which I did not save), where val loss line is way higher than training. Therefore, for this LSTM, we added a drop out (rate 0.2) after each LSTM layer, concatenate train and val set while fitting the model. We then define the model fit as the last model, and plot the 'val_loss' vs 'loss' using last model's history attribute. Finally, with this final model, we predict the 1000 test set's output, not by converting to 0,1 lable but confidence between [0,1], this would then be downloaded as a csv file.

Result

the following Figure 2 described the loss curve between our training set (10600) and validation set (600), after adding 2 dropput layers to our LSTM model. The loss did not drop significantly in the later epochs, which is a clear sign for no improvement of learning.

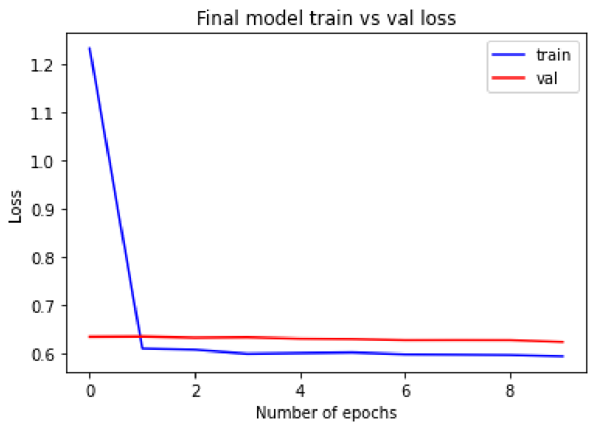

Figure 2: Final model's loss plot.

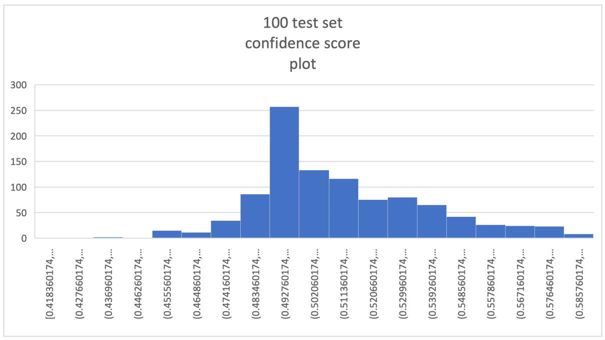

Figure 3: 1000 test set's predictions

Analysis

We concluded this experiment by traing our LSTM model on more data, to study the effect of whether more data would help our model performance (the answer is yes), while also determine the reason for why our model behave this way by plotting the loss comparison, where we can see in Figure 1. The reason for loss to not drop in the later stage of epochs is due to the model predicting most of the class as 1, again, it is because of the embedding multiplication between text vector and image tensor, the learning would tilt toward 1(answerable).
We also realized in the previous LSTM training, there is actually an 'overfitting' happened, by adding two dropput (0.2), it did draw loss curve between train and validation close, yet still not able to increase the accuracy remarkably. We did add one more plot in Figure 3 from our prediction for 1000 test set's confidence. Although we did not the true answer for each of these test sets, we can still observe the abnormality of where the confidence of each data land, there barely any overly 'confident' score, e.g.: 0.9/0.1, most of them lie around the cutting point 0.5. We are sure the reason can still be the combination of embedding from text and images.