Lab 4 | Submission | Deep Learning & Neural Nets

Submitted by: Anusha Basavaraja
(anba9017@colorado.edu)

Part 1: Report

Methods – Validation Dataset:

The dataset used for this lab is from VizWiz VQA website. It has pre-defined training, validation and test split. The original image repository consists of 20523 examples in the train dataset, 4319 examples in the validation dataset and 8000 examples in the test dataset. The website also provides annotation file which is in json format with all needed information pertaining the images for this lab. In this lab, the goal is to use the image data to predict whether a given a question on the image is answerable or not – visual question answering binary classification task. So, out of all information available in the annotation file on the image, the focus is on the question and the target values with binary classes (0 – unanswerable and 1-answerable). To conduct the experiment, I am considering first 2000 examples in training dataset, 800 examples in validation dataset and 1000 examples in test dataset by implementing three neural network models that uses combined features from image and question as input to perform a binary classification task of predicting whether the question is answerable or not upon a given image.

To conduct the experiment, first I am extracting the features from image data and question. For extracting image learned features for each image in the dataset, I am using VGG16 model with Imagenet weights by not including the top layer and to extract the text features from each question in the dataset, I am using CountVectorizer() function from the scikit-learn module with all default values specified in it. This in-built function converts the sequence of text in the question to a vector with numerical values. These resultant features are then concatenated into 1D array from each example before inputting it into the models. I have implemented 3 neural network models which has the following hyperparameters.

Model-1 has one hidden layer and an output layer. Hidden layer has 8228 nodes with ‘ReLU’ activation function and output layer has 1 node with ‘sigmoid’ activation function. The optimizer used is ‘SGD’ with learning rate 0.01 and there is a dropout layer between the hidden layer and the output layer with the value 0.25. The loss used is ‘binary_crossentropy’ and this model is trained with 6 epochs.

Model-2 has two hidden layers and an output layer. Both the hidden layers have 128 nodes with ‘tanh’ activation function and output layer has 1 node with ‘sigmoid’ activation function. The optimizer used is ‘SGD’ with learning rate 0.01 and there is no dropout regularization used in this model. The loss used is ‘binary_crossentropy’ and this model is trained with 6 epochs.

Model-3 almost follows the architecture of Model-2 with two hidden layers and an output layer but it has 64 nodes in both the hidden layers with ‘tanh’ activation function and the output layer has 1 node with ‘sigmoid’ activation function. The optimizer used is ‘Adam’ with default learning rate of 0.001 and there is dropout layer with value 0.1 after each hidden layer. The loss used is ‘binary_crossentropy’ and this model is trained with 6 epochs.

All three models are trained on the first 2000 examples in the training dataset and validated on first 800 examples in the validation dataset of VizWiz VQA dataset. Rest all of the hyperparameters have default values available in the tensorflow module. All the models are executed on the Google Colab without the GPU support. I have chosen these set of hyperparameters explicitly as I believe this might yield good observation with respect to learning and identifying good trends from the execution of these models.

Results – Validation Dataset:

The table shown below shows the results from three different models implemented that uses multimodal features from image and text to perform the VQA binary classification task of predicting whether a given question on an image is answerable or not. It has results from training the three models on the training dataset and prediction on the Validation dataset.

Model	***# of Hidden layers***	***# of neurons per layer***	***Activation function***	***Optimizer & learning rate***	***Average Precision***	***Best Training Accuracy***	***Best Validation Accuracy***
Model-1	1	8228	Relu	SGD, 0.01	57.97	97.65	59.62
Model-2	2	128, 128	Tanh	SGD, 0.01	57.36	92.55	59.50
Model-3	2	64, 64	Tanh	Adam, 0.001	59.61	79.30	62.25

Table: Table showing the results from the execution of 3 models.

The table above shows that Model-3 have better results in terms of Average Precision and validation accuracies in comparison with Model-1 and Model-2 which seems like causing overfitting when we observe the higher training accuracies in them.

Analysis – Validation Dataset:

From the table of the results, we can observe that Model-3 performs better than Model-1 and Model-2. If we further observe the accuracies observed on the training and validation datasets, we can notice that both the Model-1 and Model-2 has great accuracies on the training dataset but not on the validation dataset. Whereas in case of Model-3, the gap between the accuracies of training and validation datasets is not that huge compared to that of Model-1 and Model-2. The possible reason I could think of is the case of overfitting. Model-1 and Model-2 might have become too complex and started to learn the noise as well so that they can't understand the simple trends in the validation dataset thereby failing to generalize. But, in case of Model-3, which is much simpler in its architecture tries to avoid overfitting scenario thereby have the ability to identify the simple trends in the validation dataset to generalize it.

In terms of Average precision, again Model-3 has very good performance in comparison with Model-1 and Model-2 even though Model-3 is much simpler than other two models. Also, one more difference in the architecture of Model-3 is, it uses 'Adam' optimizer with smaller learning rate of 0.001 which is the default learning rate in 'Adam' whereas other two models use 'SGD' optimizer with little higher learning rate of 0.01 which is set explicitly. The use of optimizers with different learning rate could be one of the major reasons why Model-3 might be performing better in terms of average precision when compared to other two models. The reason could be because SGD is naturally much bouncy in nature compared to Adam which makes it locally unstable and with SGD having slightly higher learning rate than the Adam makes it much bouncier thereby causing the Model-1 and Model-2 to overfit and skipping the phase of reaching the optimal weights. But in case of Adam with much lower learning rate, it maintains the stable slow learning steps over the training iterations thereby allowing the Model-3 to generalize it much compared to other two models.

In all the three models, one common observation that can be made is the models performs very well during training but not in case when tested on the validation datasets. This could be possibly because of the input that is fed to these models. The VGG16 model without the top layer is being used to extract the features from the images. This VGG16 model is mainly used for image identification task. In case of extracting the features from the question, a vectorizer is used from the scikit-learn module which simply convert the sequence of words into vectors. Later these two features are concatenated. Looks like the feature extraction process is not very accurate or adequate enough for very good prediction in this lab. Possible contributions could be from missing the position info when vectorizing the question, lots of absence or input values being zero after zero padding stage and after flattening the features from VGG16 model prediction during feature extract from image and question, VGG16 could be a wrong choice here as it gives more info towards identifying the main object present in the image rather than the task of answering a question which is based on the fact that the focus here could be very small part of the image rather than the actual object present in it, vectorizing the question might not be sufficient here as it only maps set of unique words in the text to bunch of integers.

I had to intentionally choose either ‘tanh’ or ‘relu’ as activation functions in the hidden layers of all the three models rather than ‘sigmoid’ function. When I executed Model-2 with hidden layers having ‘sigmoid’ activation function, I noticed that the learning from the training dataset was way too slow. For using 6 epochs, this model showed that the accuracy during epoch 1 was in 50’s and during epoch 6, it was still in 60’s with best and worst accuracies for validation dataset being the same of 54.5%. But when same model executed using ‘tanh’ activation function in the hidden layers, the learning during the training iterations was really fast with the accuracy during the epoch 1 being in 60’s which spiked to 90’s during epoch 6 in the training phase and also the best accuracy for validation dataset almost reached 59.5%. The only possible reason I can think of to reason this observation is ‘sigmoid’ function has vanishing gradients problem which is slightly better in case of activation functions like ‘tanh’ and ‘relu’.

Methods – Test Dataset:

Model-3 is my finalized model that I used to make predictions on the Test dataset having first 1000 examples from the VizWiz VQA dataset. For this experiment, I have not made any modifications. I did not train the model again with additional data. My experiment still includes training the Model-3 on training dataset with the size of 2000 and validation dataset with size of 800 and predicting it on the Test set with the size of 1000 examples.

Results – Test Dataset:

The loss curves from finalized Model-3 on the training and the validation dataset looks like the following shown below. The blue colored curve represents the loss vs no. of training iterations on the training dataset with 2000 examples and the red colored curve represents the loss vs no. of training iterations on the validation dataset with 800 examples. Then the model is used to perform the prediction on the 1st 1000 examples from the test repository in the VizWiz VQA datset.

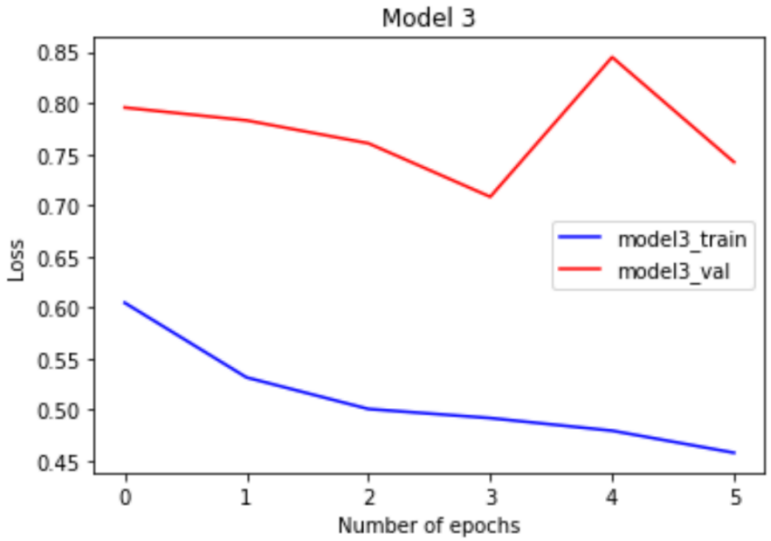

Fig: Loss curves showing loss vs no. of training iterations on training and validation datasets

Inference – Loss Curves from final model:

The figure from the above results section shows the loss curves from the Model-3 on training and validation datasets respectively. If we observe the loss curve during training, we can notice that the loss reduces over the no. of training iterations whereas during validation, the loss reduces up until certain point till epoch 3, then spikes at epoch 4 and again reduces at epoch 5 during the course of the training iterations. From both the curves we can observe that the model is learning over the no. of training iterations. Also, we need to find the optimal no. of training iterations such that neither the model overfits or underfits and also the gap between these loss curves during the training and validation phases is minimal for better prediction on the Test dataset. As per the Model-3 performance, the accuracies on the training dataset is around 79% and on validation dataset is around 62-63% which almost looks like comparable. And also, the no. of training iterations used is 6 seems like a good optimal case in this situation as the loss curves almost seems like they converge up until epoch 3 and later there is some bouncing characteristics shown by validation dataset but in case of training dataset, the loss keeps reducing smoothly. If I would have trained for longer iterations, then the model would have overfit I guess which is not good in terms of prediction results and observations.