Date: March 31, 2022, Name: Hena Ghonia(20213256)

Problem 1

3. (2 pts)Run the 6 configurations listed in `run_lstm.py` .For each of these experiments, plot learning curves (train and validation) of perplexity and loss over epochs. Figures should have labeled axes and a legend and an explanatory caption
Solution: Looking at Figure 1:Loss vs epoch, among 6 configuration; configuration 1, 2, 6

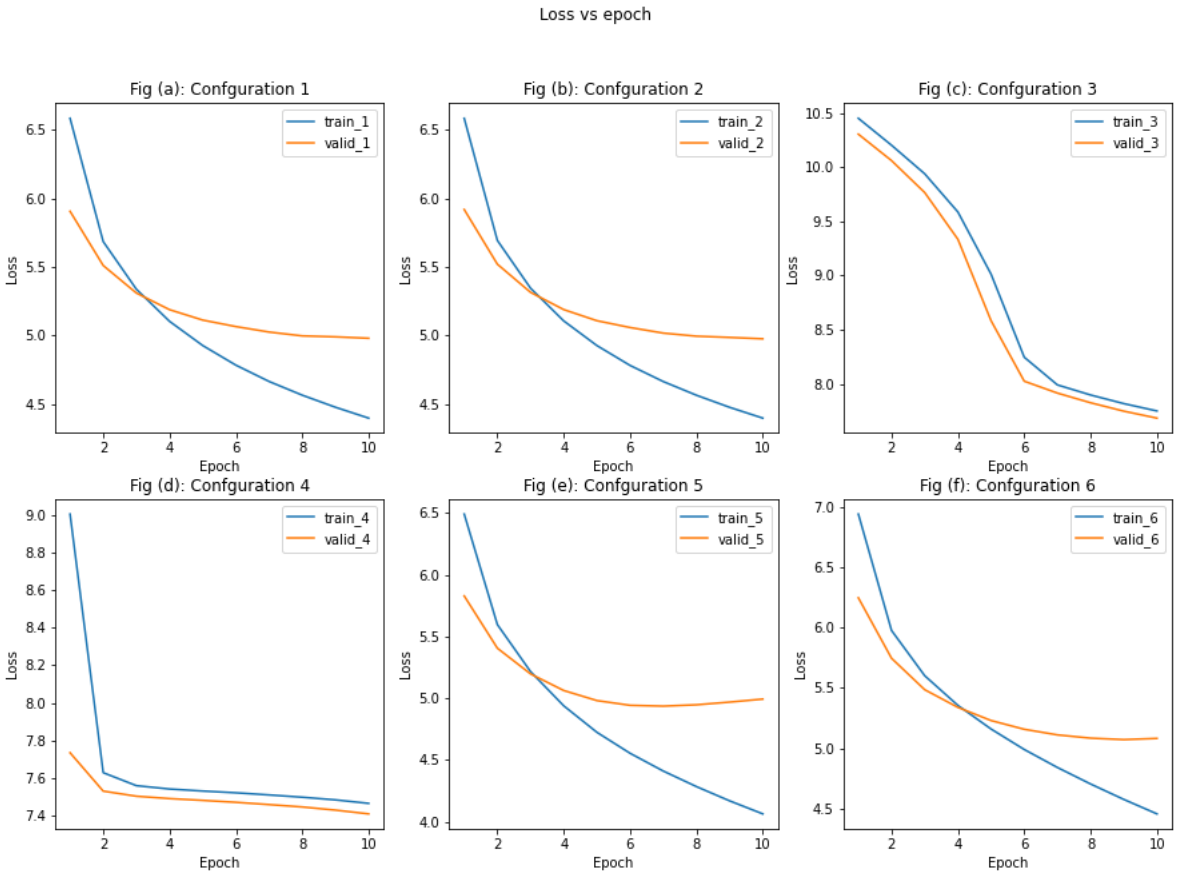

Figure 1: Loss vs epoch for LSTM for 6 configurations

perform in a similar way where optimizer used is adam and adamw. Whereas configuration 3 and 4 does not perform well looking at Loss vs epoch graph as 'sgd' and 'momentum' is used as optimizer in it. Configuration 5 performs the best in terms of reducing loss to 4.06 where layers=2, batch_size=16, optimizer='adamw' was used.

As seen in Figure 2: Perplexity vs epoch for 6 configurations, least perplexity was obtained and lesser the perplexity value better the model. Configuration 5 has least perplexity score while

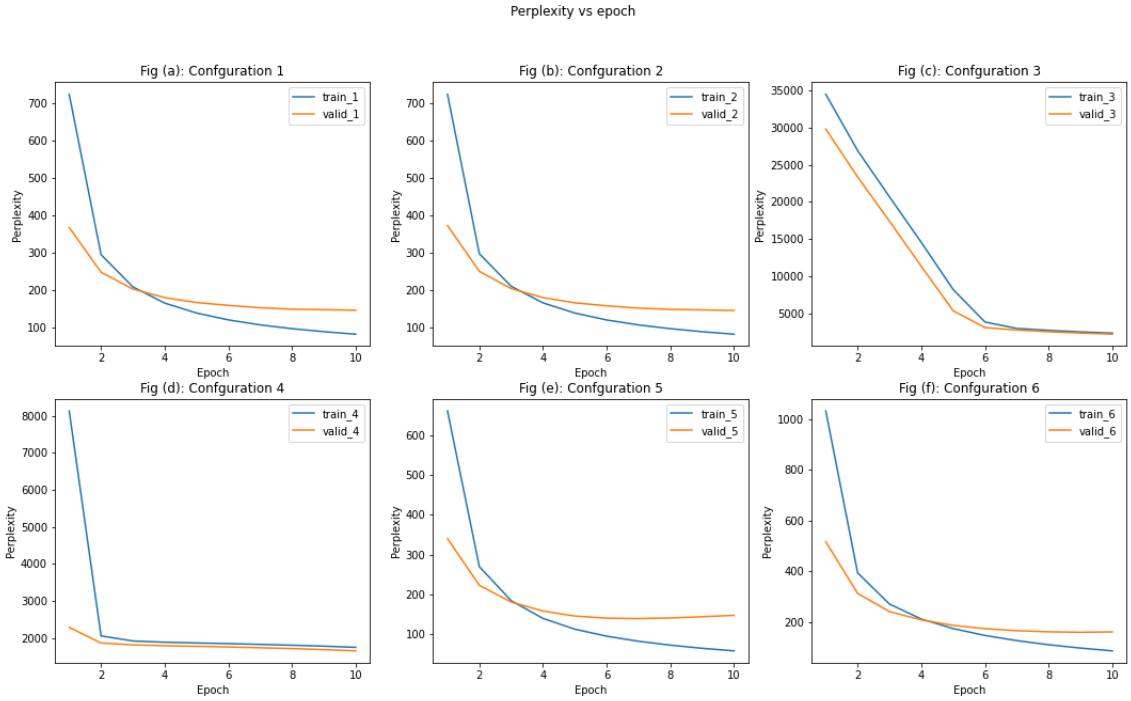

Figure 2: Perplexity vs epoch for 6 configurations

training and validating model. Best train perplexity score is 58.10 and best valid perplexity score is 139.17. Configuration 1, 2 and 5 performs similarly in terms of perplexity score.

4. (2 pts)Among the 6 configurations, which hyperparameters + optimizer would you use if you were most concerned with wall-clock time? With generalization performance?
Solution:

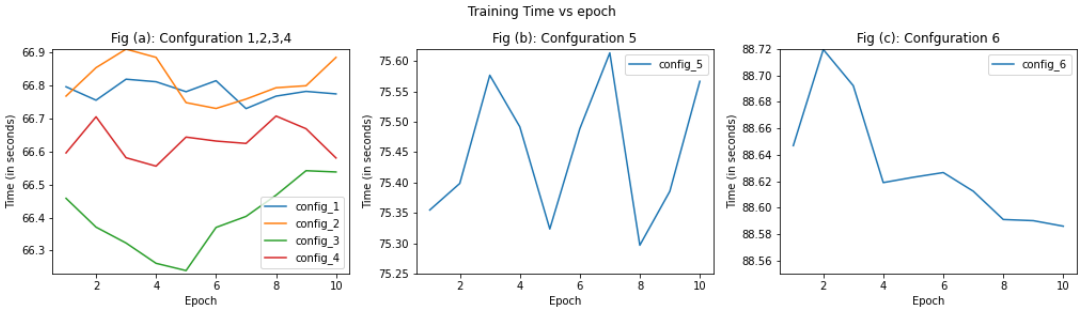

Figure 3: Wall clock time vs epoch for LSTM for 6 configurations for training

In Figure 3, Fig(a): Configuration 1,2,3,4 it is seen that config_3 which means configuration 3 has least wall clock time among 6 configurations but its performance in terms of perplexity and loss value is not satisfying so if we were most concerned with only wall clock time we would use configuration 3.

- If we were most concerned with wall clock time but to achieve better generalization we would **use Configuration 1** with hyperparameters layers=1, batch_size=16, log=True, epochs=10, optimizer='adam' since its wall clock time is less than configuration 6, 5 and 2 and has descent perplexity score(81.06 while training 145.21 while validation) and loss value(4.97 while validation and 4.39 training) compared to configuration 3 and configuration 4.
- If we were not concerned with wall clock time and to achieve better generalization performance, we will **use Configuration 6** with hyper-parameters layers=4, batch_size=16, optimizer='adamw' since there is least difference between validation loss and training loss in Figure1 and there is less difference in perplexity score between training and validation as seen in Figure 2.

Problem 2.5 Number of Multihead attention learnable parameters = 4 * (num_heads*num_heads x head_size*head_size + num_heads x head_size).

Problem 3

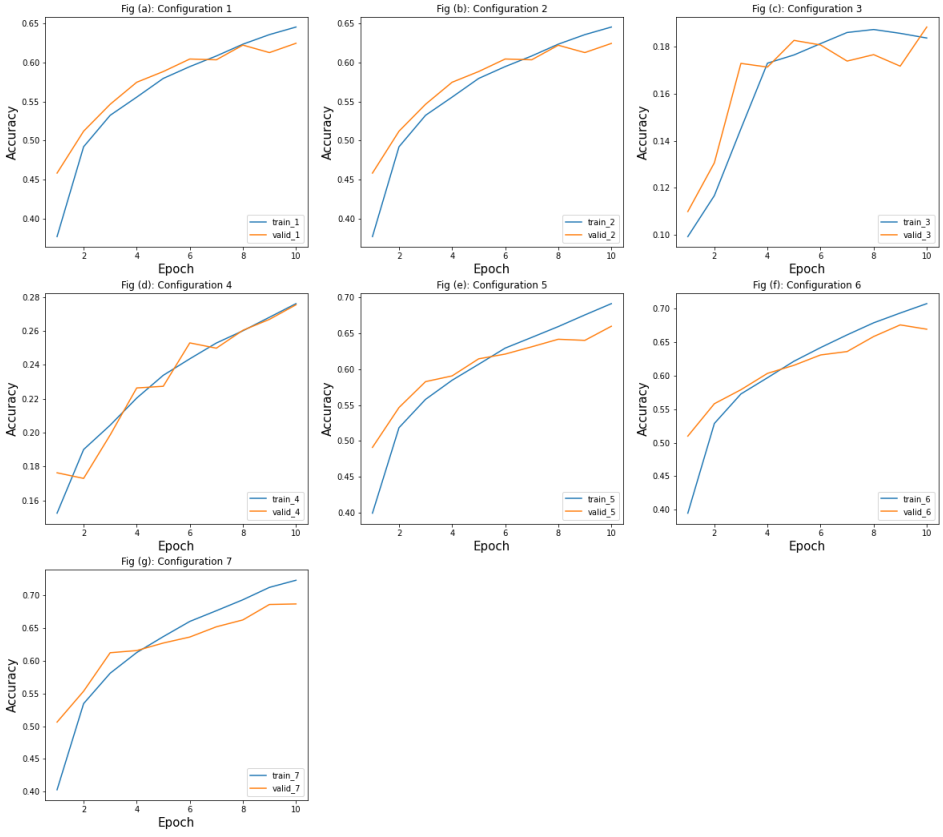

Figure 4: Accuracy vs epoch for 7 configurations

1.

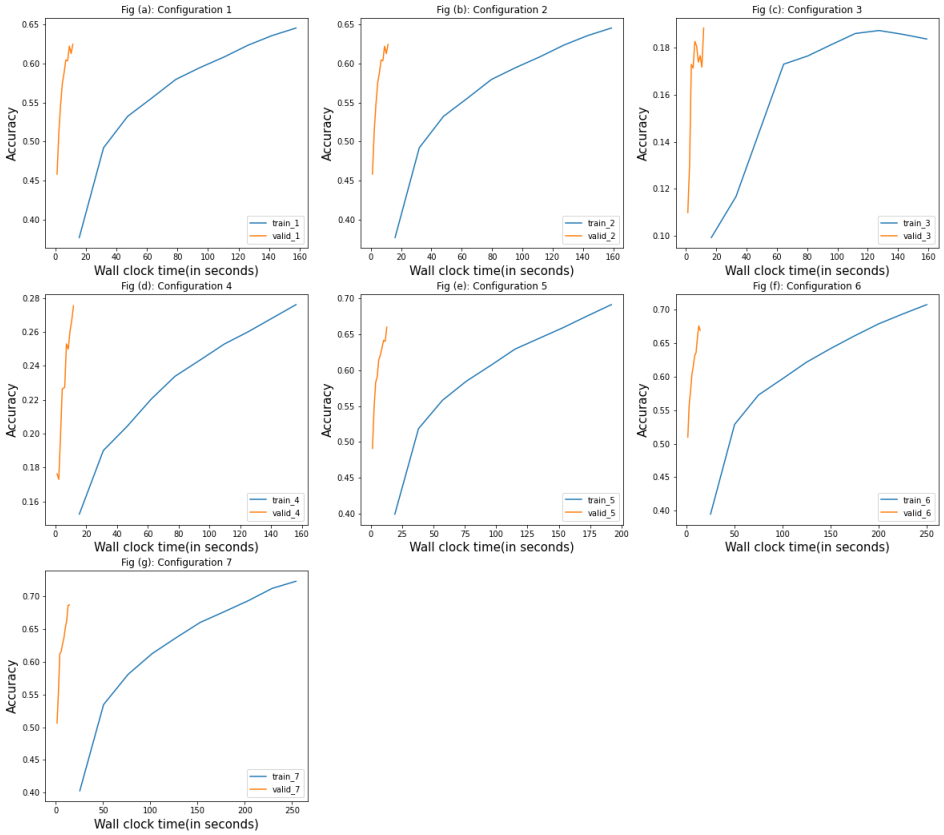

Figure 5: Accuracy vs cumulative wall clock time for 7 configurations

2. **Solution:** Table 1 shows performance of hyperparameter and optimizer for VIT Architechture and configuration 7 performs best.

Architecture	layers	optimizer	block	Parameters	Train accuracy	Validation accuracy	Training Loss	Validation Loss
1	2	adam	prenorm	1086730	64.5143	62.4414	0.991229	1.044897
2	2	adamw	prenorm	1086730	64.5121	62.4219	0.991289	1.044927
3	2	sgd	prenorm	1086730	18.7233	18.8281	2.245391	2.244044
4	2	momentum	prenorm	1086730	27.6064	27.5391	1.972686	1.969948
5	4	adamw	prenorm	2140938	69.1506	65.9961	0.870379	0.947713
6	6	adamw	prenorm	3195146	70.7755	67.5977	0.824738	0.923701
7	**6**	**adamw**	**postnorm**	**3195146**	**72.3179**	**68.6914**	**0.778364**	**0.883504**

Table 1: Performance of hyperparameter and optimizer for VIT Architechture

Figure 6: Table 1

3. (2pts)Among the first 6 configurations, which hyperparameters + optimizer would you use if you were most concerned with wall-clock time? With generalization performance?
Solution: If I was most concerned with wall clock time, out of 7 configuration I would use configuration 3, since it takes least wall clock time to fit.
-If I was concerned with generalization performance I would use Configuration 7 since it has less difference between train and validation accuracy and has best accuracy among other configurations.

4. (3pts)Between the experiment configurations 1-4 at the top of `run_exp_vit.py`, only the optimizer changed. What difference did you notice about the four optimizers used? What was the impact of weight decay, momentum, and ADAM?
Solution: Configuration 3 and 4 does not perform well i.e accuracy in configuration 3 and 4 drops significantly compared to configuration 1 and 2 as seen in Figure 4, which means optimizer Adam and weight decay perform better than sgd and momentum.

5. (3pts)Compare experiments 6 and 7. Which model did you think performed better (PreNorm or PostNorm)? Why?
Solution: PostNorm performed better than Prenorm for 10 epochs and for 6 layers. If model is trained for 30 epochs for configuration 6 and 7, Post-Norm overfits more than Pre-norm configuration. Also, if reduced the number of layers to 4 in configuration 6 and 7, PreNorm performed better than Post-Norm. In paper On Layer Normalization in the Transformer Architecture, it shown that with Pre-Norm layer, learning rate warm up can be removed which eases the hyperparameter tuning and at initialization, the gradients are well-behaved without any exploding or vanishing for the Pre-LN Transformer. Pre-Norm converges faster than Post norm and since we trained only for 10 epochs, we can see that Pre-Norm performes better than Post-norm.

6. (3pts)In configurations 1- 7, you trained a transformer with various hyper-parameter settings. Given the recent high profile transformer based models, are the results as you expected? Speculate as to why or why not. How do they compare with CNN architectures given below.
Solution: Number of parameters were 3195146 for configuration 7 which gave best accuracy result. Validation accuracy was 68.69 and training accuracy was 72.31 for configuration 7 which is less than CNN architecture given in table. One reason could transformers do not

Model	Val Accuracy	Test Accuracy	Num Parameters
GoogleNet	90.40%	89.70%	260,650
ResNet	91.84%	91.06%	272,378
ResNetPreAct	91.80%	91.07%	272,250
DenseNet	90.72%	90.23%	239,146
VIT.	68.69%	68.64 %	3195146

perform well on less training data. Here we trained VIT on small amount of data which 6000 images per 10 class. In Vision Transformer- it is shown that they trained model on large amount of data and transfered to small image recognition benchmarks which has similar results as CNN state of art results.

7. (2pts)For each of the experiment configurations above, measure the average steady-state GPU memory usage (`nvidia-smi` is your friend!). Comment about the GPU memory footprints of each model, discussing reasons behind increased or decreased memory consumption where applicable.
Solution: I used 'pynvml' library to monitor gpu performance for each 7 configuration. It was seen that configuration 7 uses gpu memory of 1328MiB which used more memory than other configurations. However it was observed that for GPU was utilized more for more number of layers. So as number of layers increases GPU utilization increases too.

8. (2pts)Comment on the overfitting behavior of the various models you trained, under different hyperparameter settings. Did a particular class of models overfit more easily than the others? Can you make an informed guess of the various steps a practitioner can take to prevent overfitting in this case?

Solution: - Looking at Figure 4: Accuracy vs epoch for 7 configurations, it is observed that configuration 3 has more difference between validation accuracy curve and training accuracy curve, hence configuration 3 easily overfits compared to other where optimizer used was sgd.
- However in configuration 2, 5, 6, 7 optimizer used was Adam weight decay which reduces overfitting and gives better generalization.
- To reduce overfitting, weights of network is preferred small which is referred as regularization methods. Several methods such as early stopping and dropout can be used with weight constraint(to limit magnitude of weights within certain range). Activity regularizer such as L2 or L1 norm can be added to