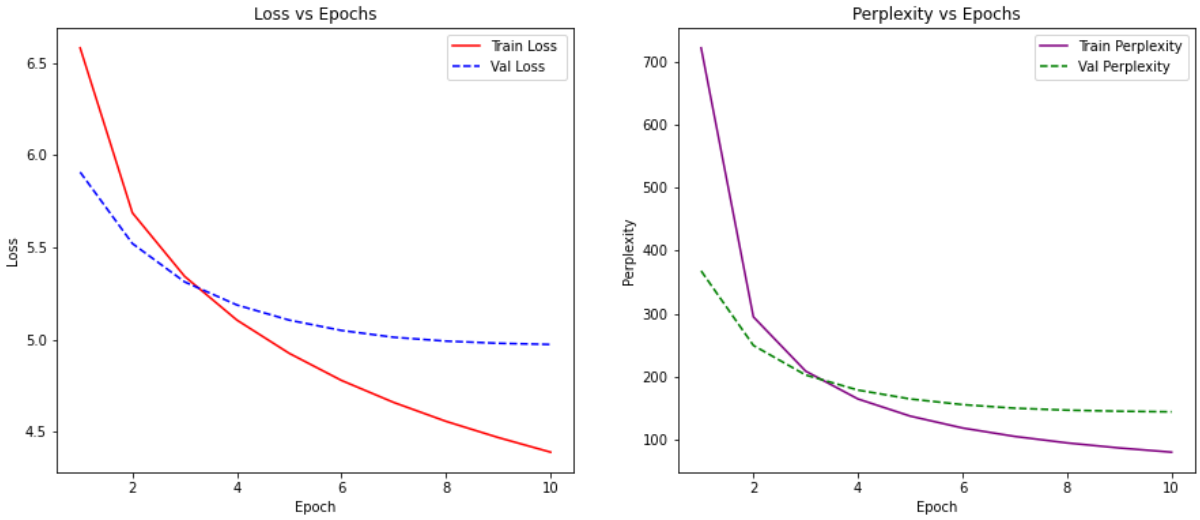
6135 - Assignment 2 - Practical Report

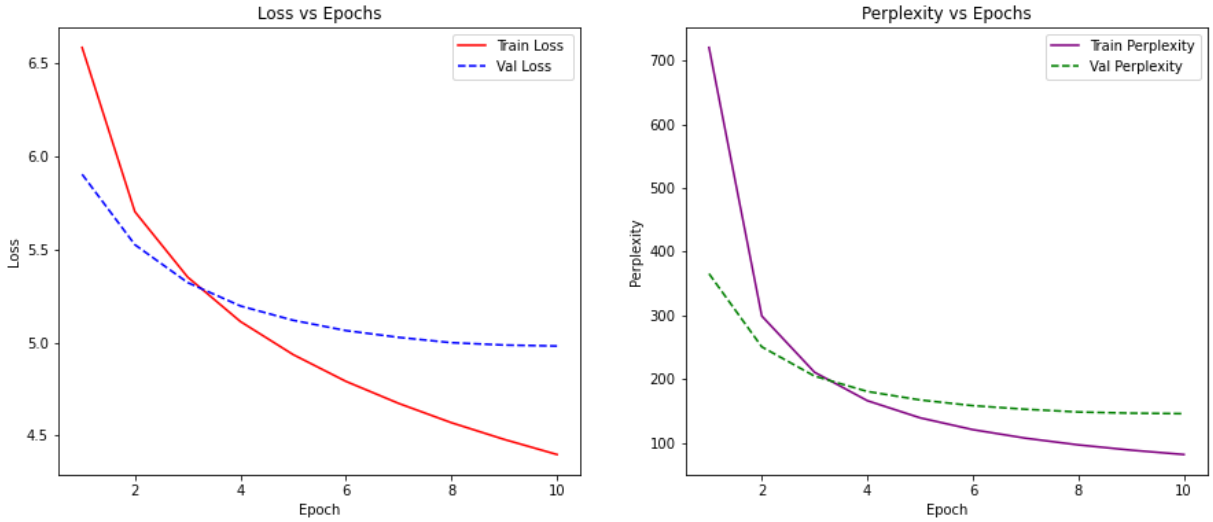
Problem 1

Answer 1.3: Learning curves (train and validation) of perplexity and loss over epochs for the given 6 configurations.

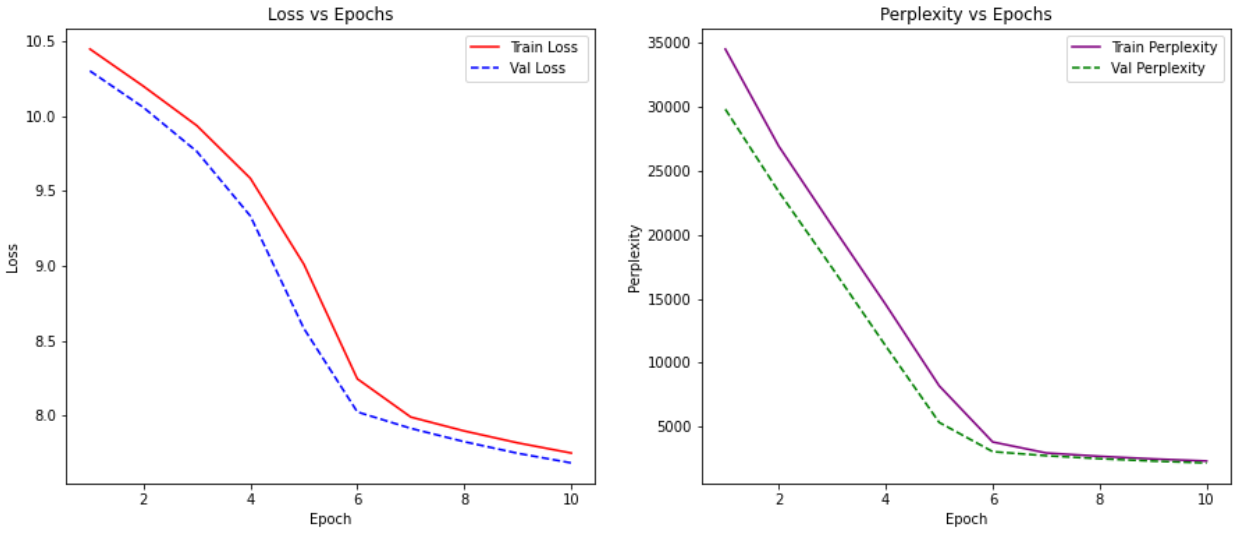
Config 1: layers : 1, epochs : 10, batch size : 16, optimizer : adam



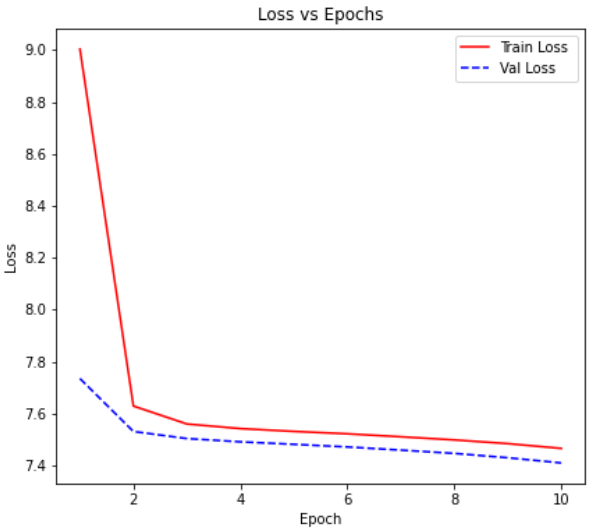
Config 2: layers : 1, epochs : 10, batch size : 16, optimizer : adamw

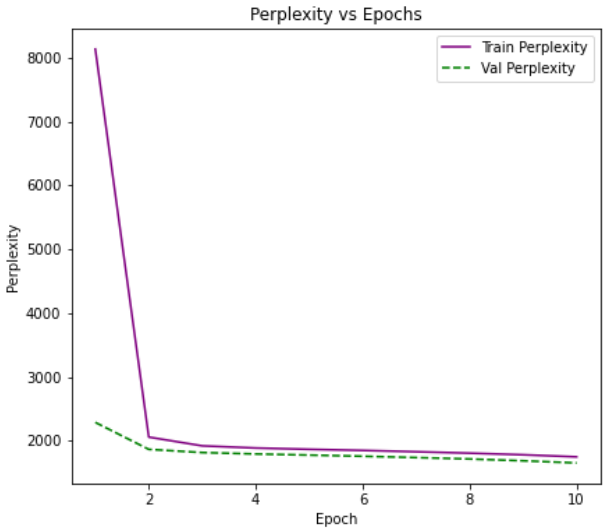


Config 3: layers : 1, epochs : 10, batch size : 16, optimizer : sgd

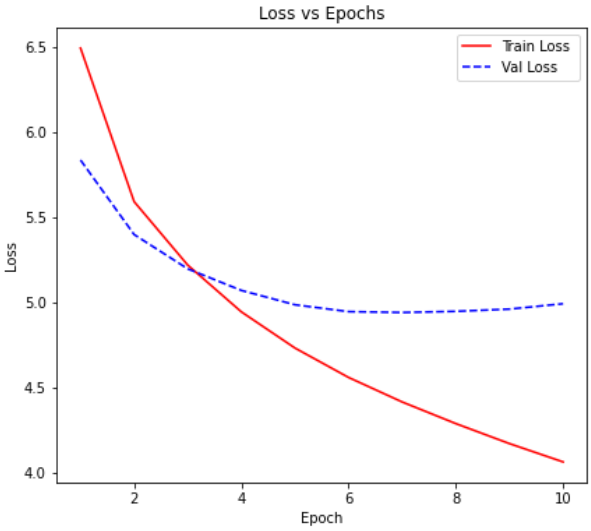


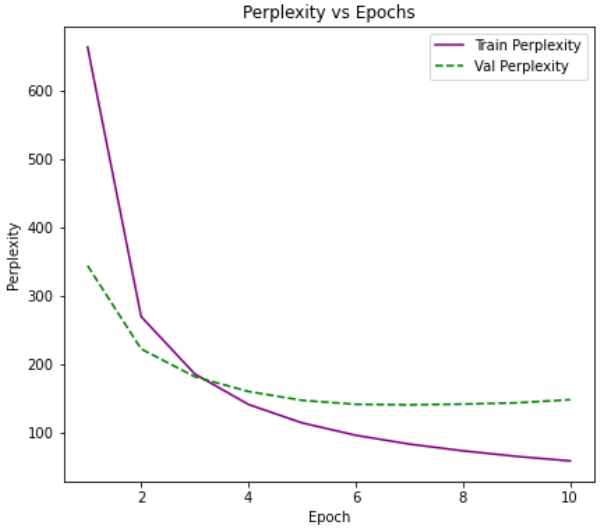
Config 4: layers : 1, epochs : 10, batch size : 16, optimizer : momentum



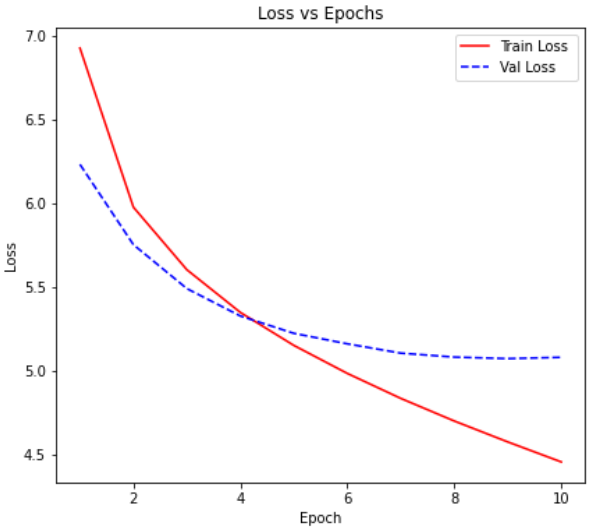


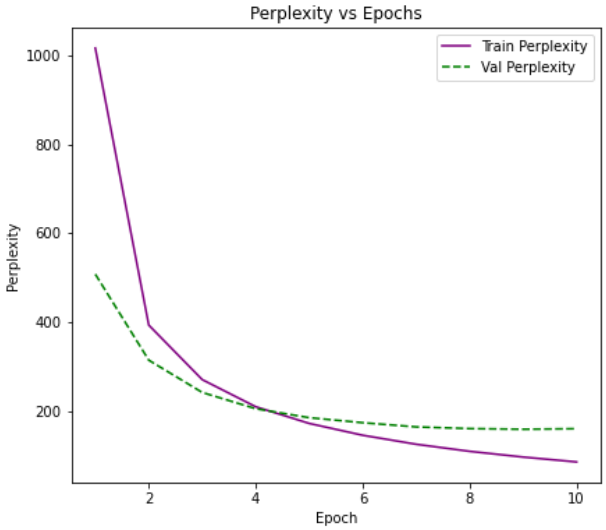
Config 5 Details: layers : 2, epochs : 10, batch size : 16, optimizer : adamw





Config 6: layers : 4, epochs : 10, batch size : 16, optimizer : adamw





Answer 1.4: Average Wall-clock time per configuration:

Config 1	Config 2	**Config 3**	Config 4	Config 5	Config 6
35.57	35.54	**35.17**	35.23	45.66	58.73

- If we are most concerned with wall-clock time, according to the above table, we should select the hyperparameters + optimizer configuration 3**.** As its average wall-clock time (35.17) is less as compared to other configs. But,
- If we look at the perplexity of these models from the above graphs, config 3 with SGD has a low perplexity value and will take way more time than Adam to reach convergence. Configuration 5 is a two-layer network with adamw which leads to higher wall clock time with slightly better performance.
- Also, note that Config 5 may start to overfit soon as the gap between the losses is way more than Config 1 and Config 2.
- To get a better generalization we need a lower loss value and less gap between the train and valid loss. Looking at the graphs from answer 3, we can see that configuration 5, and 6 fits this definition. Moreover, configs 1 and 2 also perform nearly the same.

Configuration	Train Perplexity	Val Perplexity	Train Loss	Val Loss
1	80.68	144.63	4.97	4.39
2	81.39	145.59	4.39	4.98
5	58.39	148.55	4.06	5.01
6	86.31	160.96	4.46	5.08

- If we take wall-clock time and generalization both into account, then, we can see that config 6 gives better generalization but is very slow. And it is generally the case that simpler models generalize better i.e. models that are less complex, have fewer layers.
- On the other hand, configs 1 (Adam with 1 layer) and 2 (Adam with 2 layers) give almost similar generalization performance and are also faster. Thus, considering everything config 1 generalizes the most.

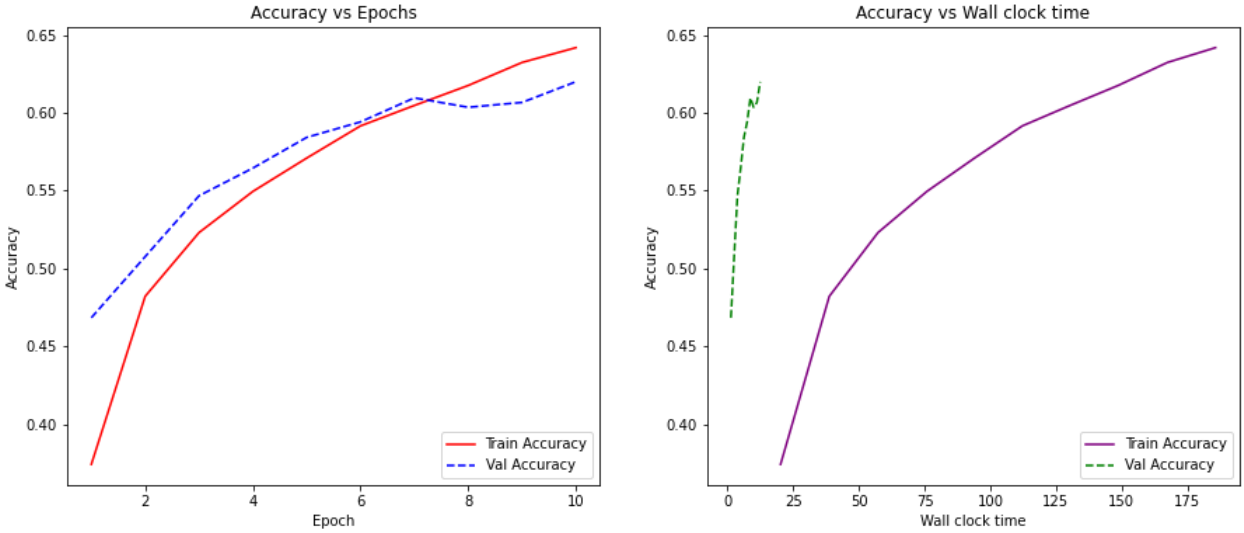
Problem 2

Answer 2.5: The number of params is: $4 * (\text{num\_heads}^2 \times \text{head\_size}^2 + \text{num\_heads} \times \text{head\_size})$

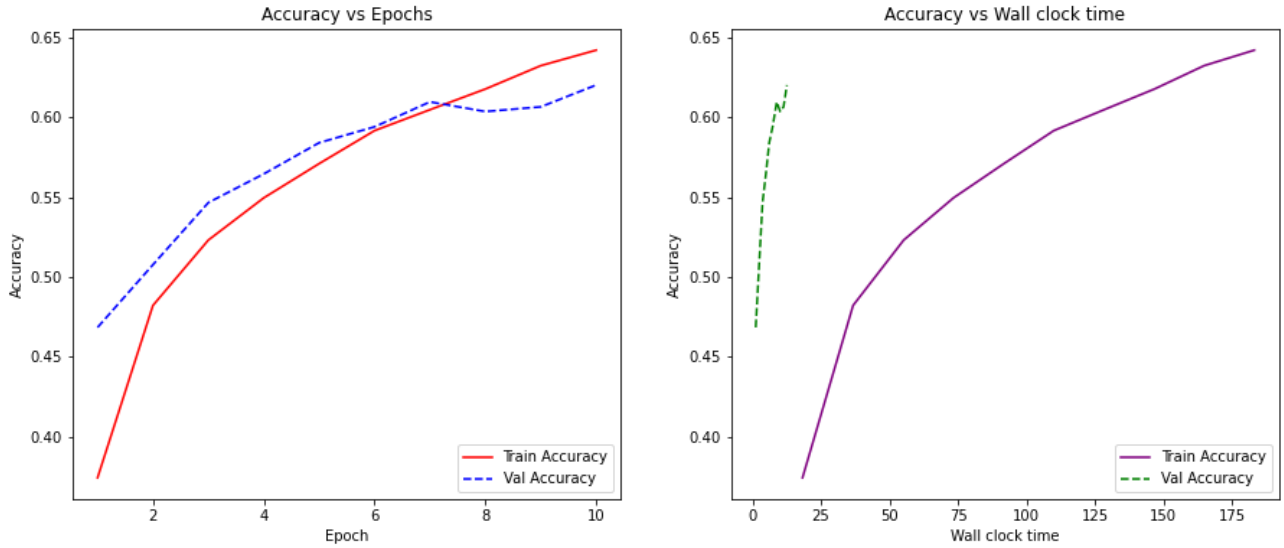
Problem 3

Answer 3.1: Learning curves (train and validation) of perplexity and loss over epochs for the given 6 configurations.

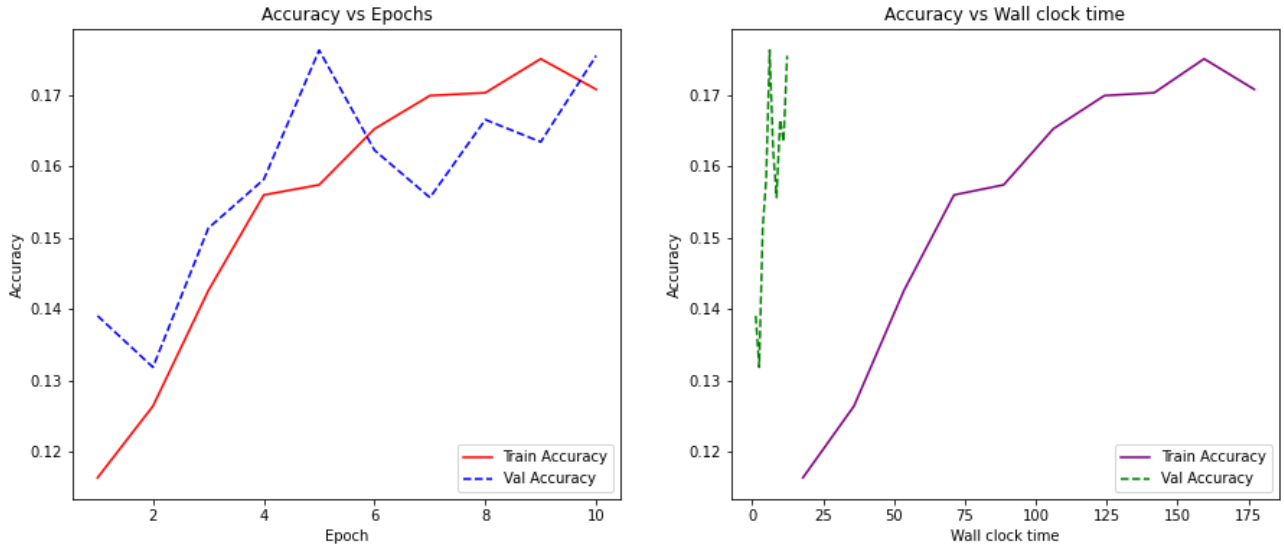
Config 1 : num_layers=2, batch_size=128, epochs=10, optimizer='adam'



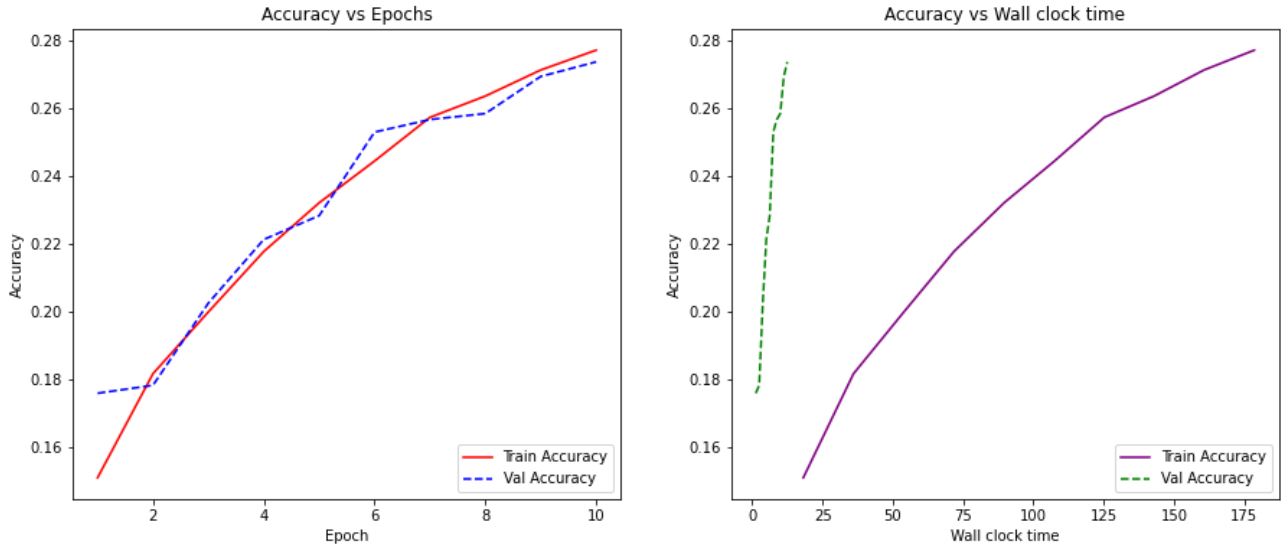
Config 2 : num_layers=2, batch_size=128, epochs=10, optimizer='adamw'



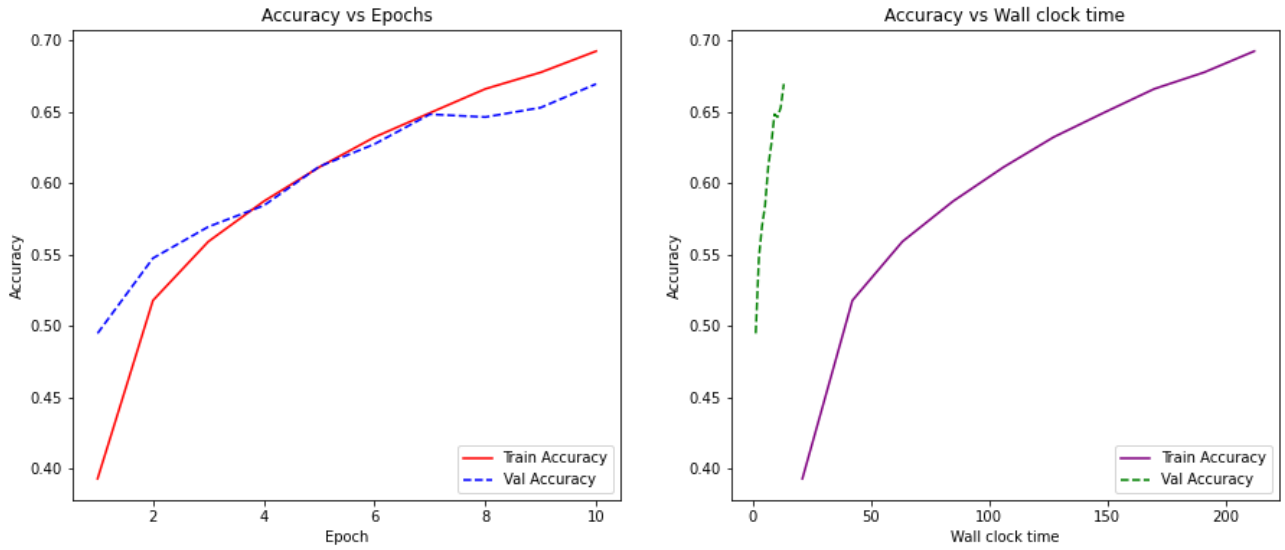
Config 3: num_layers=2, batch_size=128, epochs=10, optimizer='sgd'



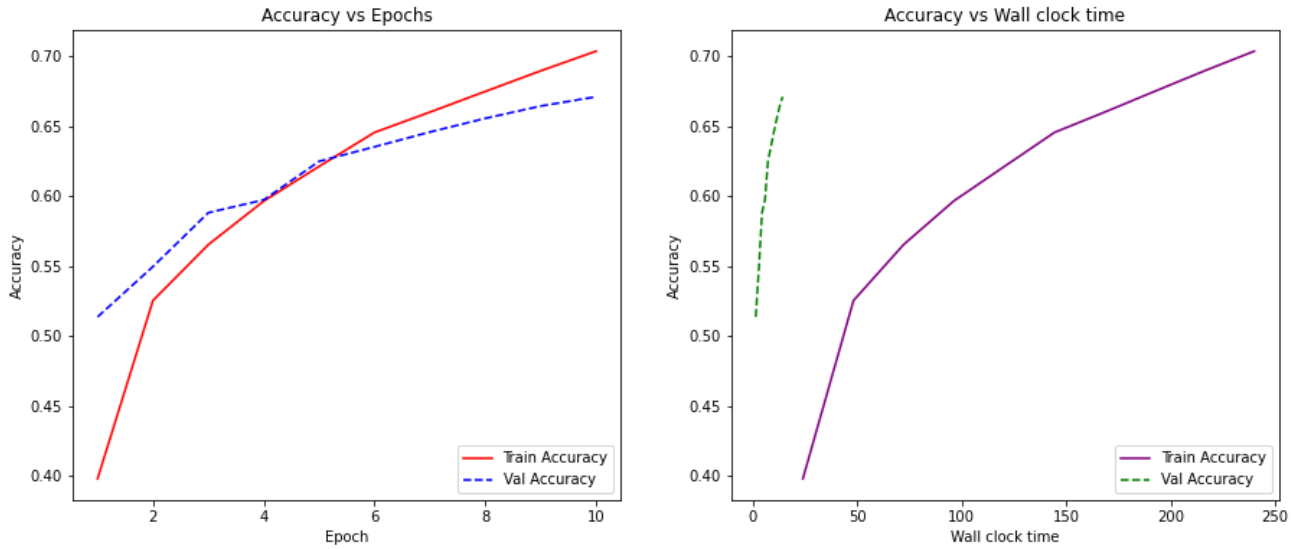
Config 4: num_layers=2, batch_size=128, epochs=10, optimizer='momentum'

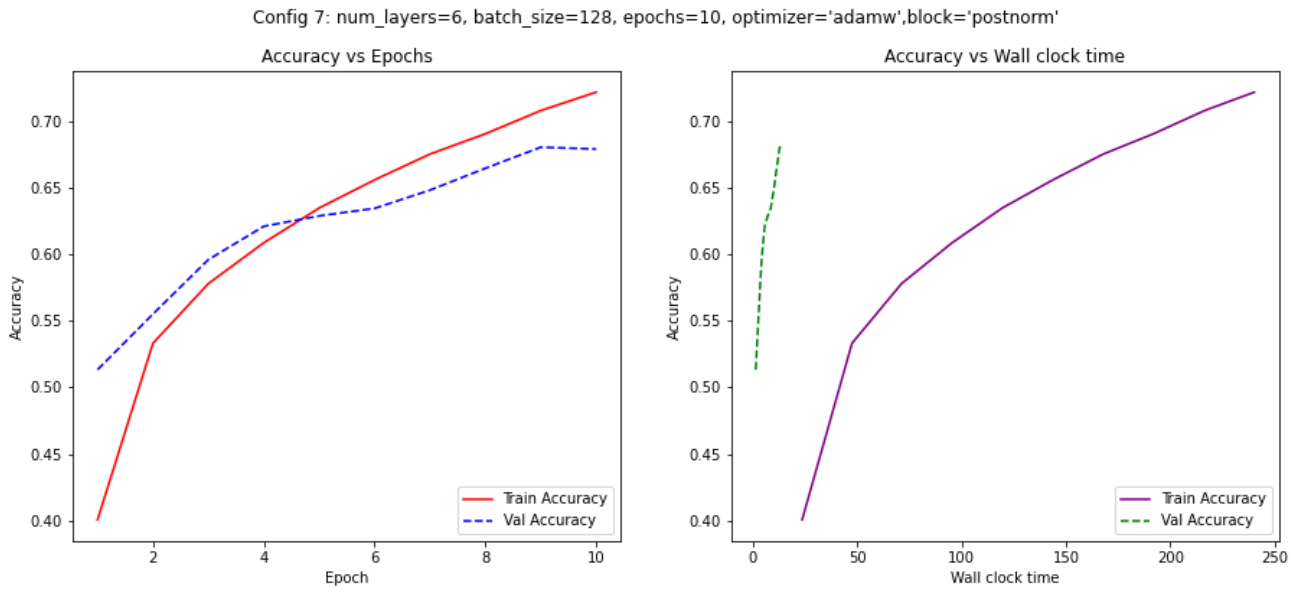


Config 5: num_layers=4, batch_size=128, epochs=10, optimizer='adamw'



Config 6: num_layers=6, batch_size=128, epochs=10, optimizer='adamw'





Answer 3.2: Table of results summarizing the train and validation performance for each experiment, indicating the architecture and optimizer. Sorted in ascending order of architecture, the number of layers, and optimizer.

Architecture	Config #	Layers	Optimizer	Train Accuracy (%)	Validation Accuracy (%)	Train Loss	Validation Loss
ViT-PostNorm	**7**	**6**	**adamw**	**72.15**	**67.89**	**0.78**	**0.89**
ViT-PreNorm	1	2	adam	64.18	61.99	1	1.05
ViT-PreNorm	2	2	adamw	64.19	62.01	1	1.05
ViT-PreNorm	4	2	momentum	27.78	27.36	1.96	1.96
ViT-PreNorm	3	2	sgd	17.08	17.55	2.24	2.24
ViT-PreNorm	5	4	adamw	69.27	66.97	0.86	0.93
ViT-PreNorm	6	6	adamw	70.34	67.08	0.83	0.92

- Insights:
  - Configuration 7 trained on ViT-Postnorm architecture performs the best as it has the highest train and validation accuracy and lowest train and validation loss.
  - It is built with 6 layers and adamw optimizer.

Answer 3.3: Average Wall-clock time per configuration:

Config 1	Config 2	**Config 3**	Config 4	Config 5	Config 6	Config 7
18.57	18.32	**17.71**	17.85	21.22	24.01	24.03

- If we are most concerned with wall-clock time, according to the above table, we should select the hyperparameters + optimizer configuration 3. As its average wall-clock time (17.71) is less as compared to other configs.
- For generalization performance Vision Transformer with PostNorm - Config 7 and PreNorm Config 6 performs the best as seen from the graphs in Answer 3.1. To get a better generalization we need a lower loss value and less gap between the train and valid accuracy. Looking at the graphs from answer 1, we can see that configuration 6 and 7 fits this definition.
- But, if we look at the gap between tarin and valid accuracy, we can understand that PreNorm is a bit more stabilized than of PostNorm. Also, according to Wang et al., 2019 and He et al., 2016 PreNorm is more stable than PostNorm as inserting LayerNorm in between Residual Path causes of Transformer's instability.
- So, considering everything config 6 - PreNorm architecture generalizes better in less wall-clock time.

Answer 3.4: Interpretation of results of configurations 1-4

- Config 1, 2 use Adam and AdamW optimizers, 3 and 4 use SGD and momentum respectively.
- Overall, configs 1 and 2 performed much better as compared to configs 3 and 4 based on the accuracy and loss mentioned in the above tables. Whereas, wall-clock time for configs 3 and 4 is lesser than of 1 and 2.
- Train accuracy for config 1-4 is approx. 64%, 64%, 17%, 28% and valid accuracy is 61%, 62%, 17%, 27%. Thus, config 3 with SGD performs the worst, and config 2 with AdamW performs the best achieving better convergence. This is because AdamW uses weight decay resulting in better performance and generalization.
- From the table of Answer 3 above, we can notice that configs 3 and 4 perform much faster as compared to configs 1 and 2. As the implementation of Adam and AdamW is slightly complex resulting in slower computation but achieving better overall performance.

Answer 3.5: Interpretation of results of configurations 6 and 7

Architecture	Config #	Layers	Optimizer	Train Accuracy (%)	Validation Accuracy (%)	Train Loss	Validation Loss	Avg Wallclock time
ViT-PostNorm	**7**	**6**	**adamw**	**72.15**	**67.89**	**0.78**	**0.89**	**24.01**
ViT-PreNorm	6	6	adamw	70.34	67.08	0.83	0.92	24.03

- From the table above, we can notice that there is just a slight difference between the accuracy and loss values of these 2 configs. Overall config 7 (PostNorm) performed better than config 6 (PreNorm) while running the ViT model for 10 epochs based on solely the accuracy value.
- If we look at the gap between tarin and valid accuracy, we can understand that PreNorm is a bit more stabilized than of PostNorm. Also, according to Wang et al., 2019 and He et al., 2016 PreNorm is more stable than PostNorm as inserting LayerNorm in between Residual Path causes of Transformer's instability. Hence, PreNorm is implemented in popular toolkits (tensor2tensor, fairseq, sockeye) etc.
- There can be some factors leading to this result such as smaller data size, a lesser number of epochs, smaller batch size. If we tune them properly, we may get reverse and more stable results without having instability in the network and problems of gradient vanishing or exploding.

Answer 3.6: Comparision of ViT vs CNN Architectures

Model	Val Accuracy	Test Accuracy	Num Parameters
GoogleNet	0.904	0.897	260650
ResNet	0.9184	0.9106	272378
ResNetPreAct	0.918	0.9107	272250
DenseNet	0.9072	0.9023	239146
ViT with 2 layers	0.6201	-	1086730
ViT with 6 layers	0.6789	-	3195146

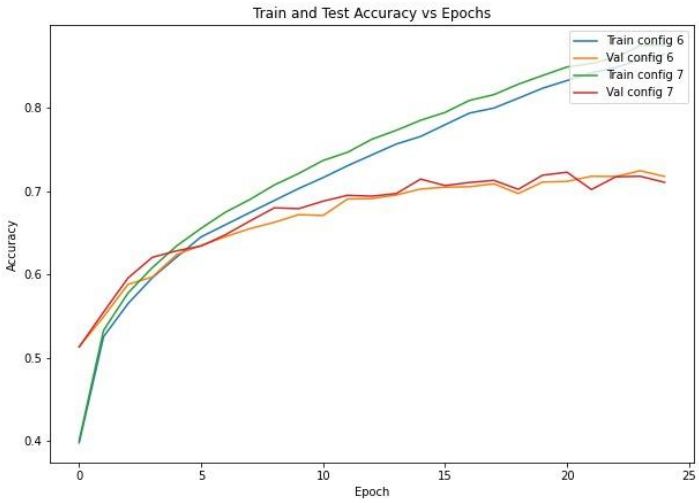
- As we can see from the above table, Transformers performed worse than CNN architectures even after having more parameters.
- ViT with 6 layers has 13X more parameters than of DenseNet even ViT with 2 layers has 2X more params than that still there is a gap of nearly 30% in accuracy.
- Stating the reason for that, these networks are required to do pretraining on huge datasets (i.e. JFT-300M dataset) and then fine-tune on downstream tasks. But we have not done that and trained from scratch for just 10 epochs. Hence, the poor results.

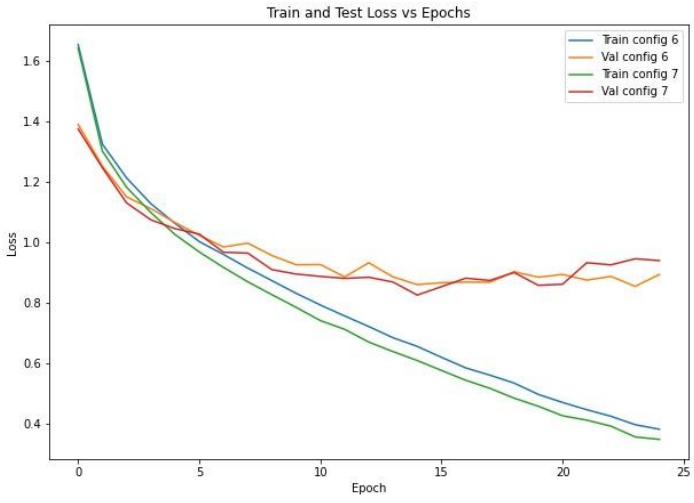
Answer 3.7: measure the average steady-state GPU memory usage

Configuration	Memory Usage
Config 1 (num_layers=2, optimizer='adam')	7222MiB / 16160 MiB
Config 2 (num_layers=2, optimizer='adamw')	7430MiB / 16160 MiB
Config 3 (num_layers=2, optimizer='sgd')	7430MiB / 16160 MiB
Config 4 (num_layers=2, optimizer='momentum')	7434MiB / 16160 MiB
Config 5 (num_layers=4, optimizer='adamw')	7592MiB / 16160 MiB
Config 6 (num_layers=6, optimizer='adamw')	8066MiB / 16160 MiB
Config 7 (num_layers=6, optimizer='adamw',block='postnorm')	8114MiB / 16160 MiB

- Based on the above table, we can note that the configurations from 1- 4 have the same (i.e. 2) layers and the same number of parameters (i.e. 1086730) and only differ in the case of optimizers. So, there is only a slight difference between the memory consumption among these.
- Config 5 has more layers than 1-4, hence, the increase (i.e. 150MiB) in memory usage is seen.
- Moreover, configs 6 and 7 use an ever higher number of layers (i.e. 6) and have a jump of more than 500MiB than the previous config.

Answer 3.8: Comments on the overfitting behavior of the models





- From the answer 2 table, we can note that for all of the configs, validation accuracy is small as compared to train accuracy but there is not much difference in the value i.e. no sign of the major overfitting up till we train the model for 10 epochs.
- But to validate this, I trained the config 6 and 7 (best performing models) models for 25 epochs as shown above and plotted the accuracy and loss curves over epoch. As these two are complex models i.e. have more parameters there is a possibility for them to overfit.
- Overfitting is when train loss gets lower and lower but validation loss drops initially and then does not drop i.e. either remains the same or increases. This is exactly what is happening for configs 6 and 7 when trained for 25 epochs i.e. both are overfitting on the given dataset.
- Moreover, config 7 (PostNorm) has a higher gap between train and validation accuracy as compared to config 6 (PreNorm) thus, it shows that PreNorm is more stable than PostNorm.

- Furthermore, to reduce overfitting there are various approaches we can try like increasing the size of the dataset, introducing regularization in the model, using dropout layers, trying data augmentation techniques, building a less complex model, etc.