Full name : Saad Benslimane
ID : 20228273
Email adress : saad.benslimane@umontreal.ca
March 23th, 2022

Problem 1

1. Figures 1 and 2.

2. From the figure 1, the LSTM with 1 layer trained for 10 epochs and using the optimizers **Adam** and **Adamw** represented by the red and blue curves, gives us the best performance (see table 1) : **Adam** with **4.392%** loss for training and **4.971%** loss for validation, also **Adamw** show almost the same performance with **4.394%** loss for training and **4.974%** for validation. From the figure 2 we can conclude that the performances of **LSTM** with **Adamw** optimizer did not improved by increasing the number of layers, we still have the best performance using only one layer.
   From all the information we have in table 1, based on wall-clock time the **LSTM** (1 layer) with **SGD** optimizer take less time to train than other methods. While based on generalization performance, the **LSTM** (1 layer) with **Adam** we get the best performance (using **Adamw** also give a good performance close to the **LSTM** with **Adam** method).

Optimizer	Layers	Epochs	Train loss	Train PPL	Val loss	Val PPL	train time
Adam	1	10	4.392	80.838	**4.971**	**144.122**	1164.858
Adamw	1	10	4.394	81.003	4.974	144.637	1158.204
SGD	1	10	7.750	2322.226	7.684	2173.959	**1145.895**
Momentum	1	10	7.466	1748.510	7.411	1653.420	1150.707
Adamw	2	10	4.072	58.723	4.995	147.697	1310.358
Adamw	4	10	4.471	87.476	5.092	162.786	1605.471

Table 1: LSTM results for training and validation for all 6 configurations, with lr=0.001, momentum=0.9, weight decay=0.0005, batch size=16, seed=42.

Problem 2

Learnable parameters for Multi-Head Attention are $[\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V, \boldsymbol{W}_O, \boldsymbol{b}_Q, \boldsymbol{b}_K, \boldsymbol{b}_V, \boldsymbol{b}_O]$, each weight and bias corresponds respectively to : Query, Key, Value and Output. The weights $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V$ are multiplied by $\boldsymbol{Q} \in \mathbb{R}^{T\times md}, \boldsymbol{K} \in \mathbb{R}^{T\times md}, \boldsymbol{V} \in \mathbb{R}^{T\times md}$ then we add the bias $\boldsymbol{b}_Q, \boldsymbol{b}_K, \boldsymbol{b}_V$. The input and the output are the same shape as embedding wish is itself $d = \textbf{num\_heads} \times \textbf{head\_size}$. With one attention head ($m = 1$) and $T = 1$ we'll have :

$$[\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m] = \boldsymbol{Q}^{1\times d}\boldsymbol{W}_Q^{d\times d} + \boldsymbol{b}_Q^{1\times d}$$

Same thing for $\boldsymbol{W}_K, \boldsymbol{W}_V, \boldsymbol{W}_O$, we'll have finally the learnable parameters are :

$$[\boldsymbol{W}_Q^{d\times d}, \boldsymbol{W}_K^{d\times d}, \boldsymbol{W}_V^{d\times d}, \boldsymbol{W}_O^{d\times d}, \boldsymbol{b}_Q^{1\times d}, \boldsymbol{b}_K^{1\times d}, \boldsymbol{b}_V^{1\times d}, \boldsymbol{b}_O^{1\times d}]$$

With $d = \textbf{num\_heads} \times \textbf{head\_size}$, the number of learnable parameters is :

$$4 \times (d \times d) + 4 \times (1 \times d) = 4d(d+1)$$

Problem 3

Architecture	Layers	Optimizer	Train loss	Train Acc	Val loss	Val Acc	Train Time
ViT	2	Adam	0.9898	0.6447	1.0656	0.6171	464.63
ViT	2	Adamw	0.9898	0.6446	**1.0655**	**0.6173**	392.75
ViT	2	SGD	2.075	0.2599	2.0708	0.2587	**383.48**
ViT	2	Momentum	1.6449	0.4096	1.6069	0.4212	386.53
ViT	4	Adamw	0.8942	0.6824	1.0017	0.6453	508.62
ViT	6	Adamw	0.8444	0.7020	**0.9490**	**0.6634**	626.91
ViT (Postnorm)	6	Adamw	0.8683	0.6915	0.9984	0.6404	**610.75**

Table 2: ViT results for training and validation for all 7 configurations, with lr=0.0003, momentum=0.9, weight decay=0.0005, batch size=128, seed=42.

1. Figures 3, 4, 5 and 6.

2. Table 2.

3. For the first 4 configurations we trained the **ViT PreNorm** model with 2 layers for 10 epochs using different optimizers. From the tabel 2, we can see that ViT with **SGD** optimizer take less time to train (**383,48 seconds**) but sadly he isn't the best model in term of performance. The next 2 configurations represent the results for ViT PreNorm model with 4 and 6 layers using **Adamw** optimizer, we can see that the model take much time when we increase the number of layers. So in term of wall-clock time, **ViT PreNorm** with **SGD** optimizer is the best model.
   Now talking about performance and not wall-clock time, the **ViT PreNorm** with **Adamw** represent the best results. From the table we can see that this configuration goes from **61,71%** accuracy using 2 layers to **66,43%** using 6 layers.

4. Now if we check the first 4 configurations, we see that only the optimizer change. From figures 3 and 5, we see that only **SGD** performs poorly with an accuracy of **25,87%** but based on what we found on the internet the **SGD** could find a low loss (high accuracy) after a long training. On the other hand, **Momentum** helped us to increase the accuracy compared to **SGD**, and we see that the model still work faster (no big difference between

SGD and Momentum).
Now using **Adam**, we see that the model perform better than **SGD** and **Momentum** but takes more wall-clock time. When using **Adamw** optimizer, we see that the **ViT** model performs a little better and takes less time than when using **Adam** optimizer.

5. For the last two configurations, We trained the **ViT PostNorm** and **ViT PreNorm** for 10 epochs using **Adamw** optimizer. From table 2 we see that **PostNorm** is faster and take less time than **PreNorm**, but in term of performance **PreNorm** gives us the best performance with a higher accuracy **66,34%**.

6. Now in this question, we're asked to compare the **ViT** transformer with **CNN** architectures. As we can see from the results we have a minimum accuracy for the **CNN** architectures of **90,40%** for **GoogleNet**, while in our problem the highest accuracy we get is **66,34%** when using **ViT PreNorm** with 6 layers, **Adamw** optimazer, and trained for only 10 epochs. We can see clearly that the **ViT** perform poorly compared to the **CNN** architectures.
Now due to the nature of our work, we didn't have enough time to train our models so that they converge to a solution like **CNN** architectures. For that we decided to test if by increasing the number of epochs the model will continue to learn or he's already at his limits. By training **config 6** for 50 epochs instead of 10 epochs, we've noticed that the accuracy increase to reach **77,12%**.
To conclude this, we can't really decide if our **ViT** transformer perform better than this **CNN** architectures. Maybe if we trained it for a 100 epochs will get a better accuracy, but definitely this will take a long time.

Config ID	Memory used (Mb)
1	1294
2	1294
3	1286
4	1290
5	1690
6	2086
7	2086

Table 3: GPU memory used for the 7 configurations

7. From the table 3, we see that for the first 4 configurations we trained the same architecture **ViT** with one layer and only the optimizer was changed so we have almost the same number of parameters used, and then our machine will use almost the same GPU memory with a small variation. While for the last 3 configurations, we trained the **ViT** transformer by increasing the number of layers and using the same optimizer, so here it's normal that the number of paramaters will increase and then our machine will use more and more GPU memory. In the last two configurations, the machine use the same GPU memory because we're using the same number of layers and the same optimizer.

8. In our case, we see that all of the configurations we trained : **ViT** transformer don't overfit easily. Now supposing that our model overfit easily, in this case there's some steps tat we can

take to avoid overfitting like :
- First thing to do is to avoid over-parametrized architectures before training.
- During training a learning rate scheduler could help us to achieve a generalizable solutions.
- Using early stopping is a helpful approach (for example monitoring the loss as a stopping criterion and stopping the training before it increases).

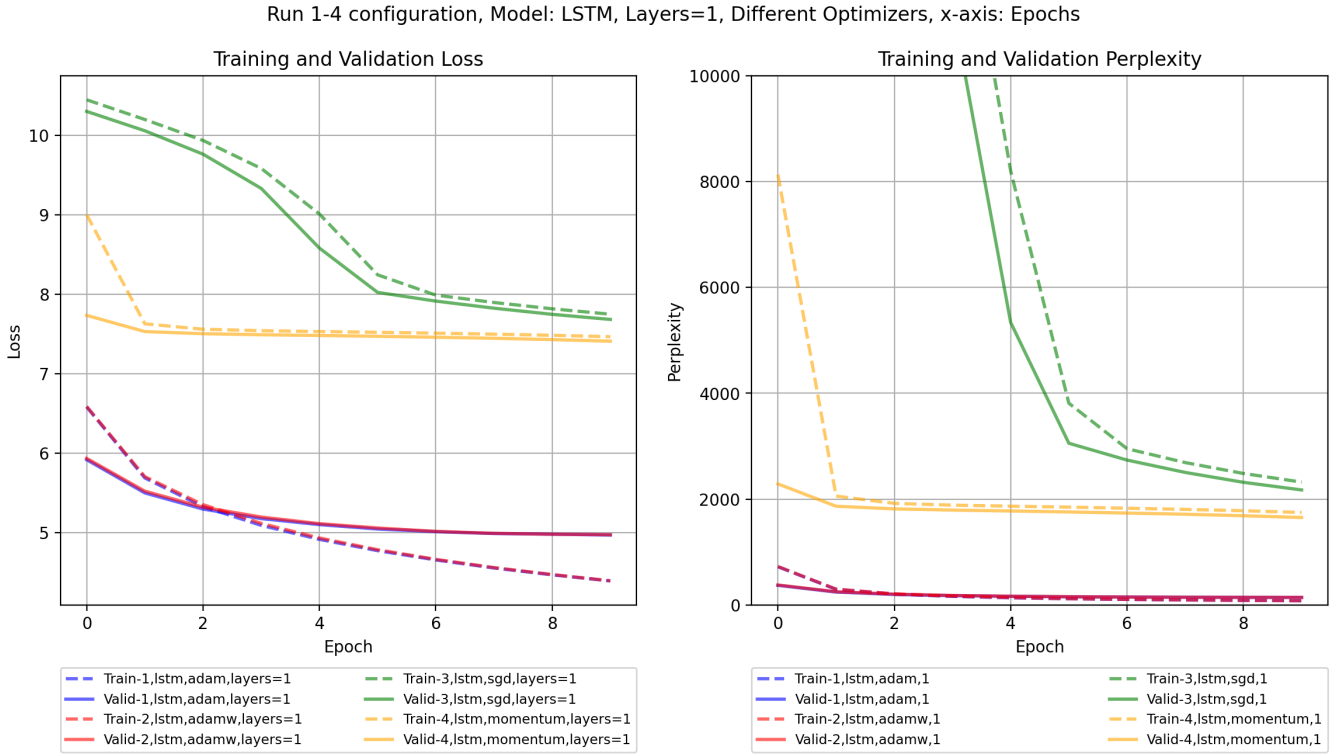

Figure 1: LSTM loss and perplexity for training and validation over epochs for the first 4 configurations with 1 layer

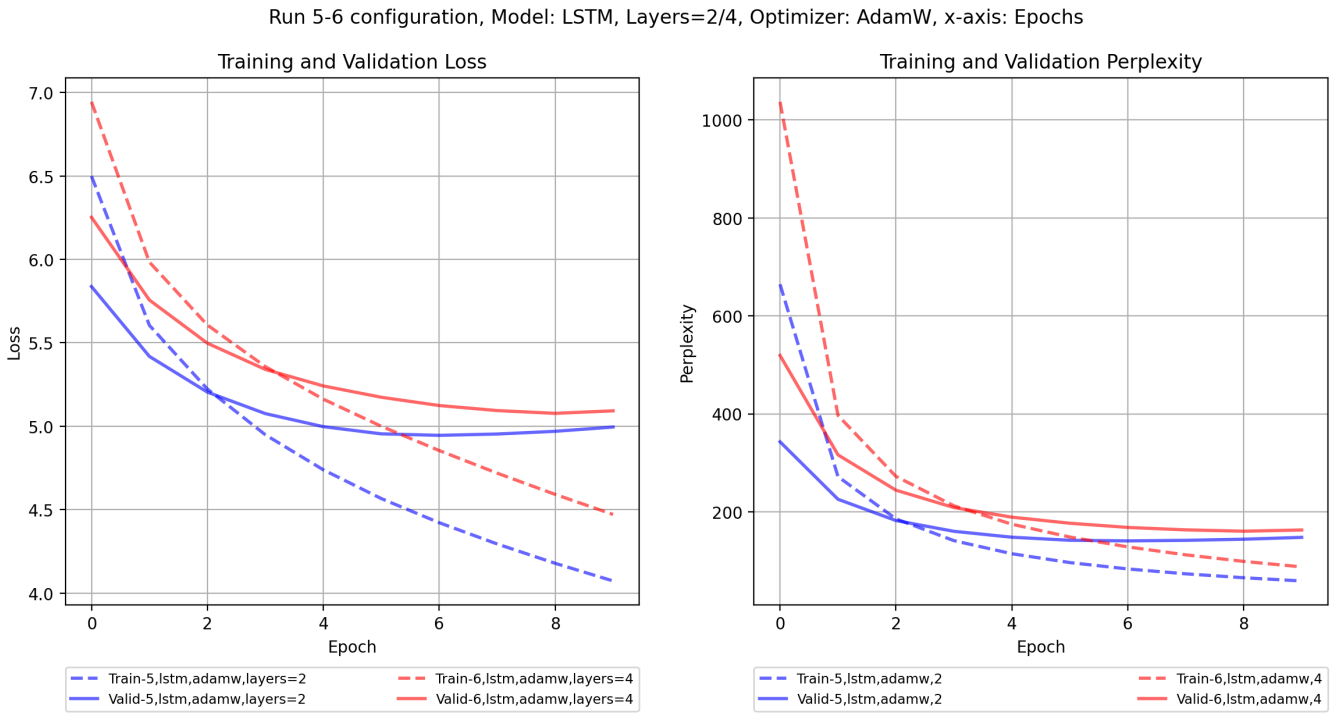

Figure 2: LSTM loss and perplexity for training and validation over epochs for the last 2 configurations with 2 and 4 layers

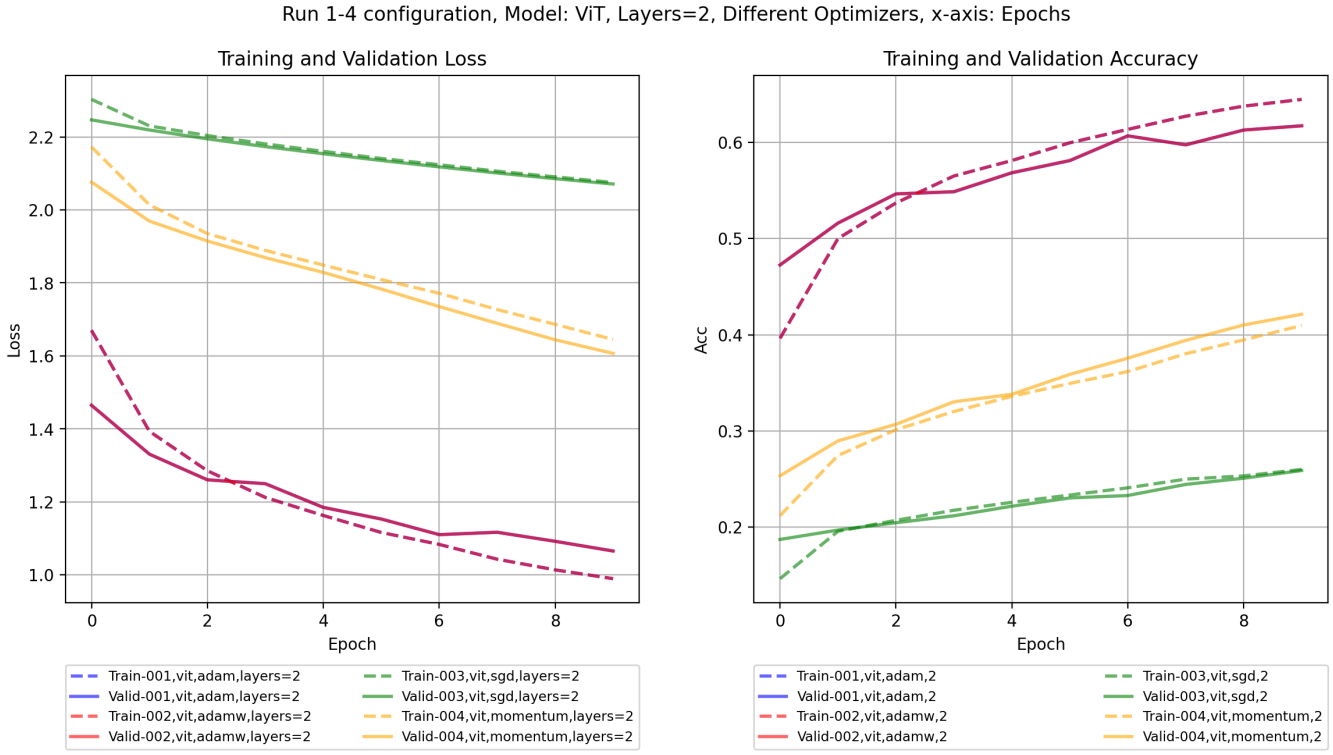

Figure 3: ViT loss and accuracy for training and over epochs for the first 4 configurations with 2 layer

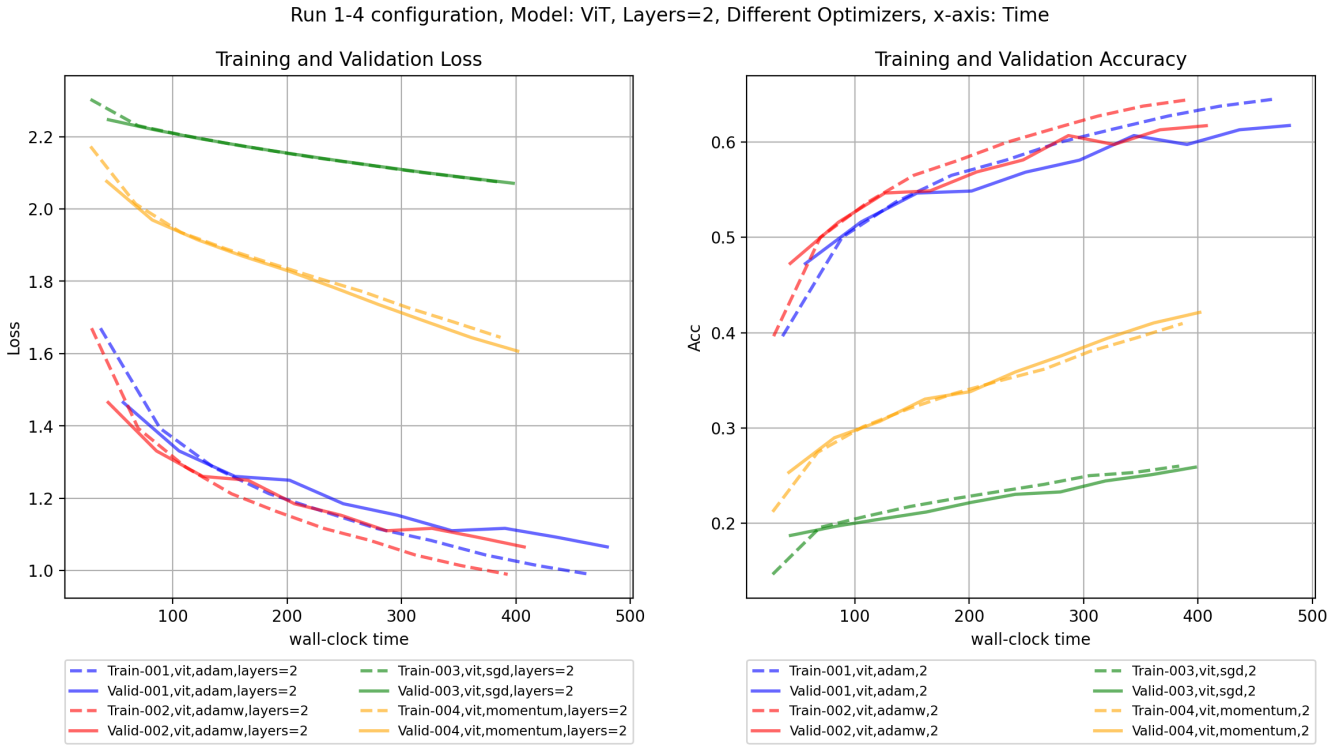

Figure 4: ViT loss and accuracy for training and over wall-clock time for the first 4 configurations with 2 layer

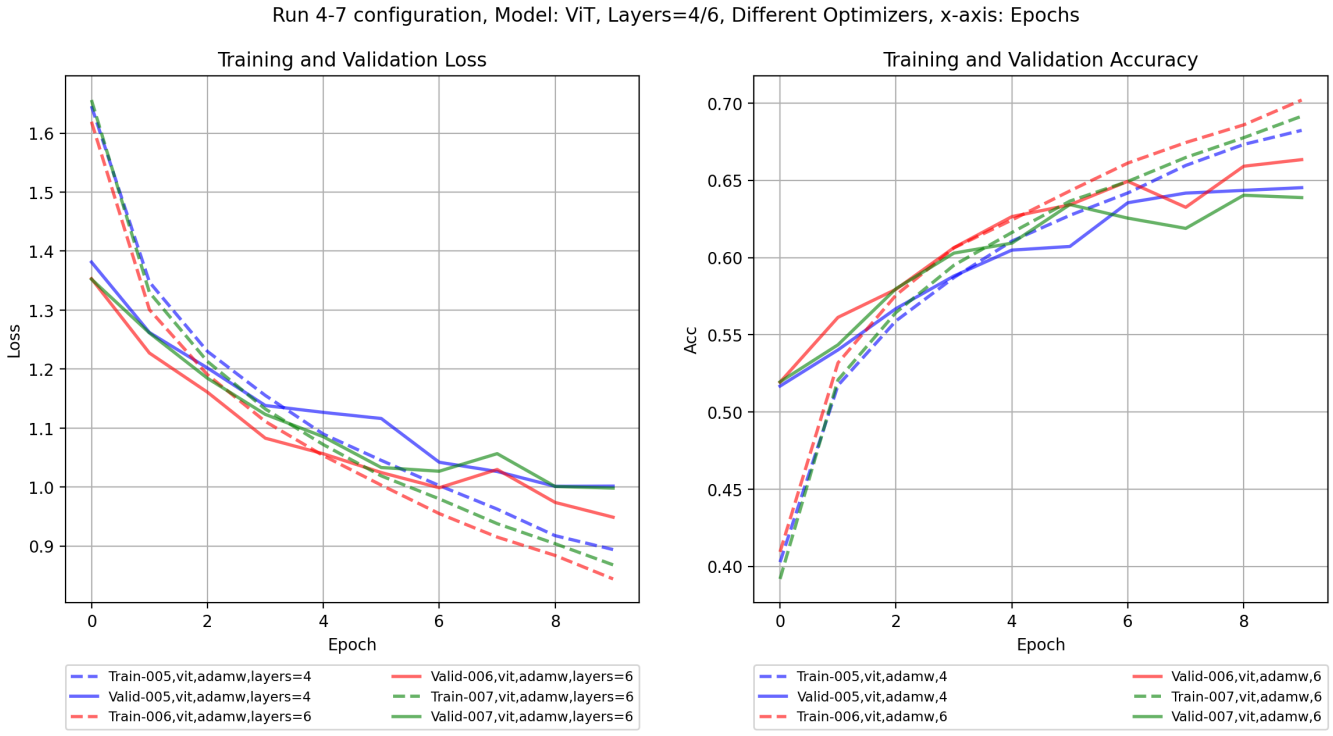

Figure 5: ViT loss and accuracy for training and over epochs for the last 3 configurations with 4 and 6 layers

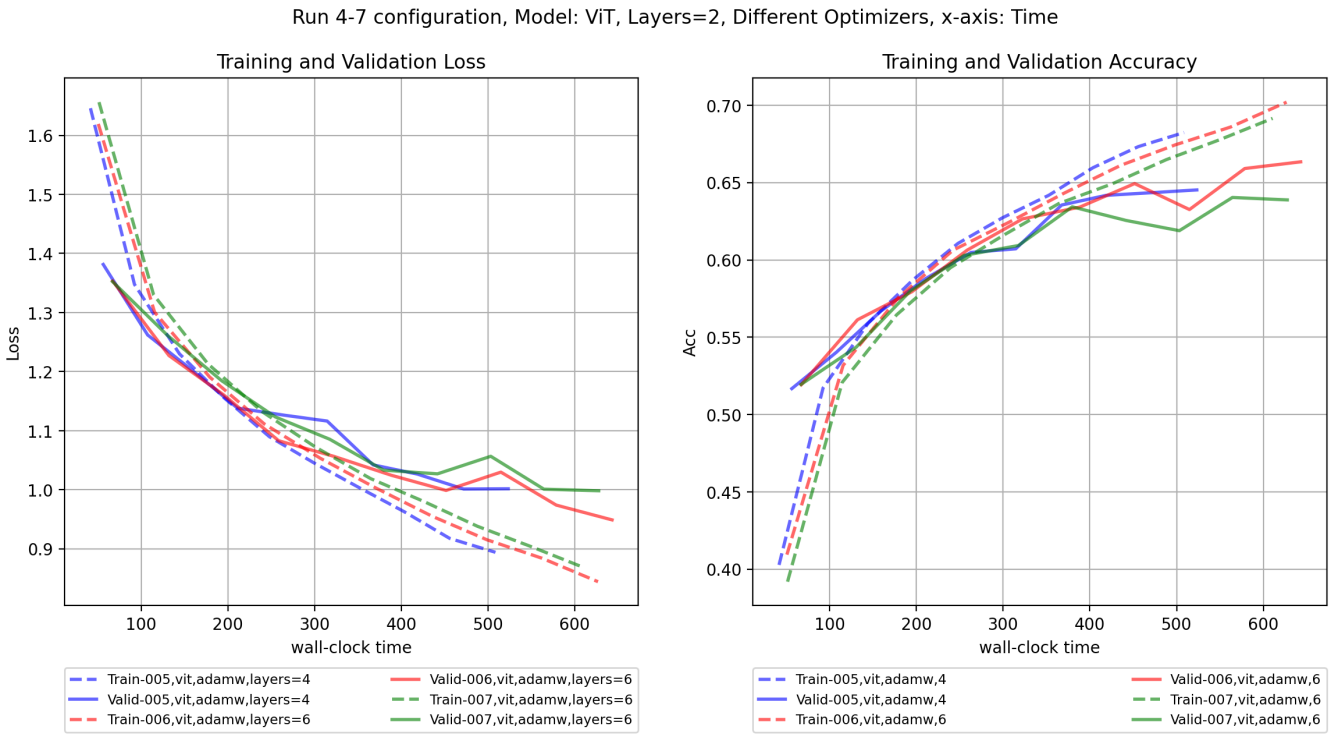

Figure 6: ViT loss and accuracy for training and over wall-clock time for the last 3 configurations with 4 and 6 layers