IFT6135 - Homework 2 - Practical Report - Lucas Maes

Problem 1: LSTM models

1.3

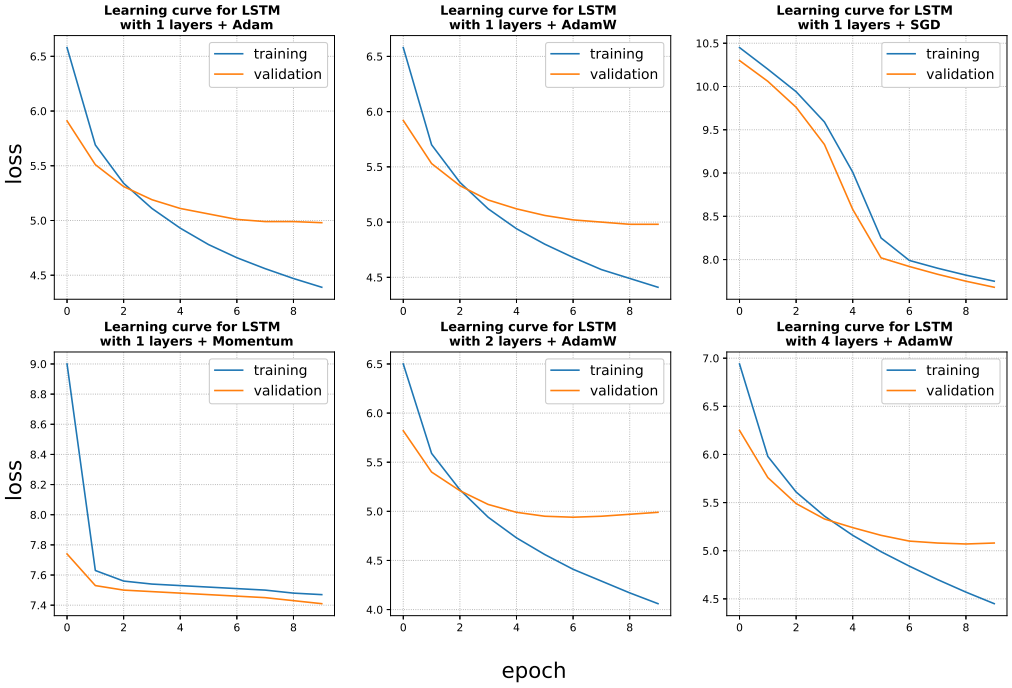


Figure 1: Learning curves of the 6 different LSTM configurations, representing the training and validation loss over epochs.

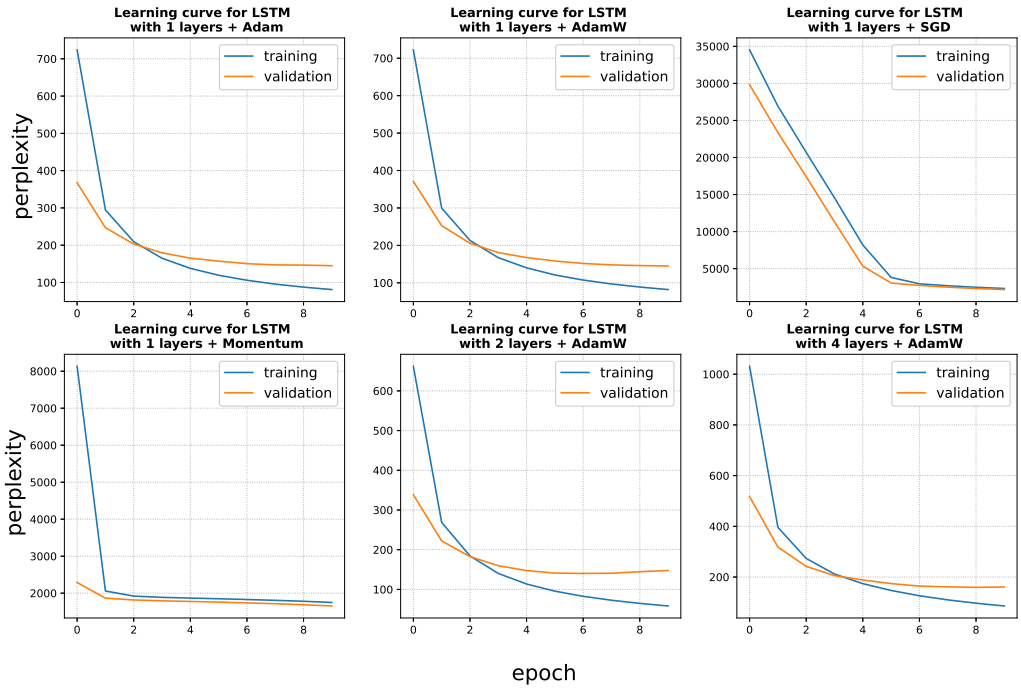
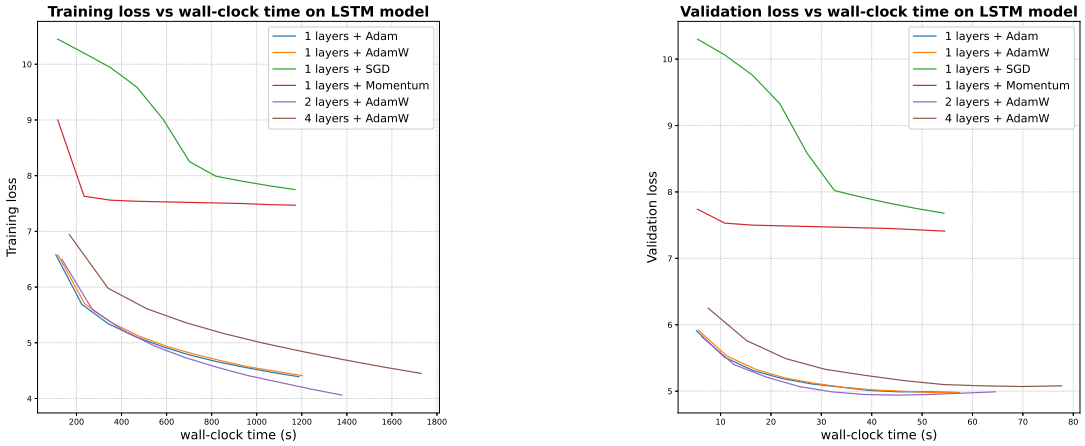


Figure 2: Learning curves of the 6 different LSTM configurations, representing the training and validation perplexity over epochs.

1.4

Most concern by the wall-clock time:



(a) **(b)**

Figure 3: Training and validation loss versus wall-clock time for each of the 6 configurations.

If we are most concern by having the best trade-off between wall-clock time and performance, the best configuration is definitively to use 1 LSTM layer and AdamW/Adam (it doesn't make a big change in something) optimizer.

Most concern by the generalization performance:

Looking at Fig.1, if we are most concern by have the best generalization, the best configuration is definitively to use 1 LSTM layer, AdamW optimizer. That configuration leads to the best generalization performance of 205.85 perplexity (using early stopping).

Problem 2: ViT

2.5

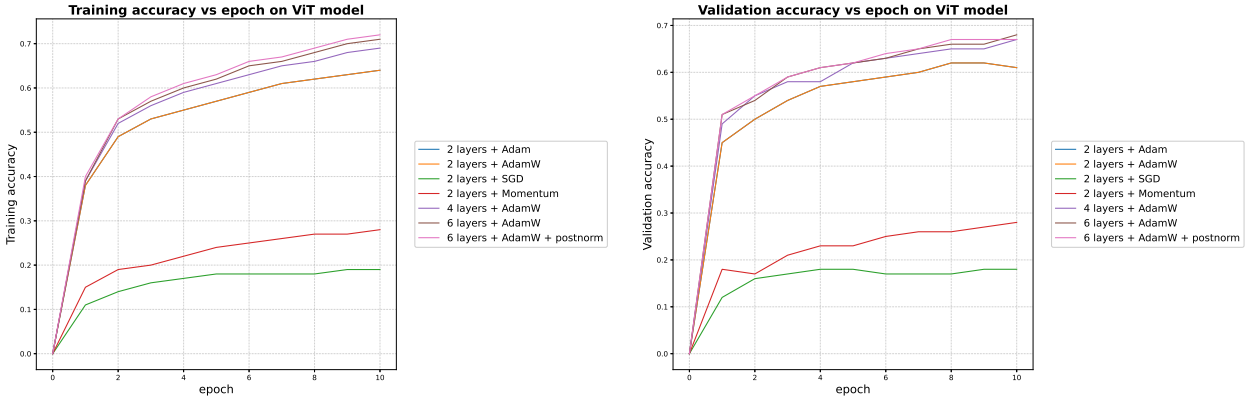
How many learnable parameters does your module have, as a function of num_heads and head_size?

In the Multi-head attention operation, 4 learned matrices are used to make linear projections of embeddings W_Q, W_K, W_V, W_Y. Those matrices have all the same dimensions (number_heads^2 $\times$ head_size^2). The operation also has 4 biases b_Q, b_K, b_V, b_Y with also same dimensions (number_heads$\times$ head_size). Therefore, as each element of those matrices and bias are learnable the function that gives the number of learnable parameters for the Multi-head attention is by the function f:

$$f(\text{number\_heads}, \text{head\_size}) = 4(\text{number\_heads}^2 \times \text{head\_size}^2 + (\text{number\_heads} \times \text{head\_size}))$$

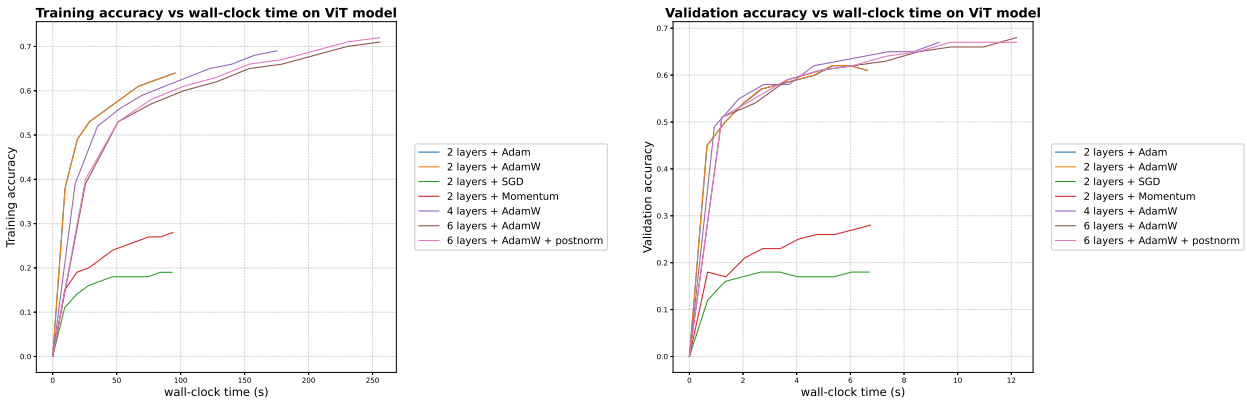
Problem 3: ViT models

3.1



(a) **(b)**

Figure 4: Training accuracy and validation accuracy performance as function of epoch number. Configurations were both trained using defaults hyperparameter values, see Appendix for more details. With no surprise, the best configurations are the ones using 6 layers of transformer encoder in addition to using AdamW as optimizer with post-normalization or pre-normalization layer giving training accuracy of respectively 72 and 71% . One may notice that, as expected, the worst results are 2 layers using SGD or momentum with training accuracy respectively under 20 and 30%. However, the training accuracy of configurations using 2 layers of transformer encoder as well as Adam and AdamW are much more important that the configs using SGD or momentum, resulting in accuracy of 64 for both of the configurations. It seems that using Adam or AdamW as optimizer doesn't really change the outcome of the training with few layers (i.e 2).



(a) **(b)**

Figure 5: Training accuracy performance and validation accuracy as function of wall-clock time (in seconds). Configurations are the same in Fig.4. The total training time seems to depend on the number of transformers layers. When the number of layers is 2 the total training time never exceed 100 seconds, for 4 layers it is under 180 seconds and above 250 seconds for training using 6 layers. As the optimizer alter the accuracy of configurations given specific transformer encoder layer number, it doesn't seem to have a significant impact on the training time.

3.2

Experiment n°	Model	Layers	Optimizer	Block	Training	Validation
1	ViT	2	Adam	prenorm	0.64	0.61
2	ViT	2	AdamW	prenorm	0.64	0.61
3	ViT	2	SGD	prenorm	0.19	0.18
4	ViT	2	Momentum	prenorm	0.28	0.28
5	ViT	4	AdamW	prenorm	0.69	0.67
6	ViT	6	AdamW	prenorm	0.71	**0.68**
7	ViT	6	AdamW	postnorm	**0.72**	0.67

Table 1: Result of both training and validation accuracy for different Vision Transformer model configurations. The best configuration is the experiment 6 with a training accuracy of 71% and a validation accuracy of 0.68%

3.3

Most concern by the wall-clock time:

If we are most concern by have the best trade-off between performance and wall-clock time, the best configuration is definitively to use 2 transformers layers and AdamW optimizer (experiment n°2). Those configurations lead to a validation performance of 61% for a total training time under 100 seconds.

Most concern by the generalization performance:

If we are most concern by have the best generalization, the best configuration is definitively to use 6 transformers layers, AdamW optimizer, and post-layer-normalization block. That configuration lead to a test performance of 69%.

3.4

The usage of weight decay on top of Adam doesn't make a huge difference (probably due to the short number of epoch and data) but gives 1% better accuracy on validation set. Besides that, Adam was far away the best optimizer to use, as it nearly triples the accuracy (both on training and validation) in comparison to the momentum or SGD among the experiment's run. Momentum was the best second optimizer to use, but with much poor result (nearly 2 times less accuracy than when using Adam). Stochastic Gradient Descent (SGD) was the worst optimizer to use, as it doesn't exceed the 20% of accuracy, where Adam exceeded 60%. One may also notice that configuration based on Adam converge way much faster than other method. Momentum also converge a bit faster than SGD.

3.5

On same model configuration and optimizer. The usage of prenorm (experiment 6) perform slightly better than the postnorm (experiment 7) on the validation set. More experiment should be done to make sure that this is true. Indeed, here experiment 6 and 7 where only using one seed and averaging more than one training with those configurations could give less biases results. Also, the training was done using a small dataset and only 10 epochs maybe with more data and using more epochs we could see a large better performance of using the prenorm block. In fact, according to [Xiong et al., 2020] using prenorm layer instead of postnorm makes the gradient well-behaving at initialization which prevents from exploding or vanishing gradient. This result has been proved theoretically and empirically. This result is related to the fact that we are normalizing before heavy operation on multihead attention, and also allows the skip-connection to bypass the layer-normalization step.

3.6

The results for a state-of-the-art model were not the ones that I was expecting. In my opinion, this can be caused by different factors. The biggest one is the number of learnable parameters. The Vision Transformer model that has the best performance has, 3195146 learnable parameters and the smallest one as 1086730 learnable parameters which is huge compared to what the best CNN model have (maximum of 260650). Other factors may be that those model could require more data to learn properly. In the original paper, the authors used 12 transformers layers, as well as more attention heads (12), where here the maximum was 6 transformers layers.

3.7

Experiment n°	Model	Layers	Optimizer	Block	Avg. GPU Mem. usage (GB)	%
1	ViT	2	Adam	prenorm	1.46	9
2	ViT	2	AdamW	prenorm	1.44	9
3	ViT	2	SGD	prenorm	1.42	9
4	ViT	2	Momentum	prenorm	1.43	9
5	ViT	4	AdamW	prenorm	1.86	11
6	ViT	6	AdamW	prenorm	2.27	14
7	ViT	6	AdamW	postnorm	2.25	14

Table 2: Average GPU memory consumption during the training of each of the experiments about ViT models.The GPU used was a Tesla-P100 and the total available memory was 16.28 GB.

What seems to have the greatest impact on memory consumption is the number of layers, as this necessarily leads to more computation (because of more matrix multiplication, model take more GPU memory space etc ...). Unsurprisingly, the model configurations that used the most memory during their training were those used in experiments 6 and 7 with a slight increase in experiment 6 probably due to the usage of the prenorm instead of postnorm. For experiments 1 to 4, we can see that there is a slight difference depending on the optimizer used. Indeed, the configuration 4 is the one that used the least memory because it uses SGD which is less expensive to calculate the momentum which is less expensive than Adam/AdamW which has to calculate the first and second order moments and then correct them.

3.8

By comparing the validation and the train accuracy of configurations by looking both at Fig.4 and Table. 1, we can conclude that configurations using Adam and AdamW leads to overfitting.

My advices for a practitioner will be to be sure to have enough data in function of the capacity of the model he wants to train. Then select a good optimizer such as Adam as well as a good regularizer such as weight decay (see AdamW). Another thing to not neglect is choosing the right hyperparameters values and using a method to avoid overfitting, such as early stopping. On classification tasks, label smoothing can also be a good trick to use.

Note: When dealing with images, data augmentation methods such as scaling, cropping, saturation, MixOut, CutMix, CutOut, etc... could be a good practical tip to use.

Appendix

Default configuration used to trained ViT models:

```
epochs = 10
batch size = 128
learning rate = 3e-4
weight decay = 5e-4
```

```
momentum term = 0.9
seed = 42
```

References

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.