AML 23/24 - Assignment 3

STFFormer hyperparameter tuning report

Giacomo Bellini, 1970896, bellini.1970896@studenti.uniroma1.it
Ludovica Mazza, 1917778, mazza.1917778@studenti.uniroma1.it
Matteo Rampolla, 1762214, rampolla.1762214@studenti.uniroma1.it
Daniele Solombrino, 1743111, solombrino.1743111@studenti.uniroma1.it

Introduction

This is the final report regarding the “Report and Parameter Fine-Tuning Analysis” section of the third assignment for AML 23/24.

This report illustrates the entire process that we followed to perform hyperparameters tuning, including our thoughts and observations, fuelled by the theoretical AML lectures.
We highlight in **blue** sentences which are asked in section 1 and 4 of the notebook.

For each notable configuration, we link its relative Weights and Biases page (using the run name assigned by WandB), where all details can be found.
All loss plots have the epoch on the x axis and the loss on the y axis.
Unfortunately, WandB doesn’t allow to label the vertical axis.

1. Initial analysis

Before even starting the hyperparameter tuning process, we gathered some **baseline** performances to base our comparisons against.

We started with the following default hyperparameters:

- Batch size → 256
- Total number of trainable parameters → ~28k
- Learning rate → 1e-1
- Learning rate scheduler → StepLR (gamma 0.1, steps [10, 30])
- Optimizer → Adam
- Gradient clipping → None
- Weight decay → 1e-5
- Number of epochs → 40

We tested this default configuration using 10, 25, 50, 75 and 100% of the batches in dulcet-snowflake-12, dauntless-bird-10, firm-grass-35, silvery-microwave-36 and fearless-brook-11.

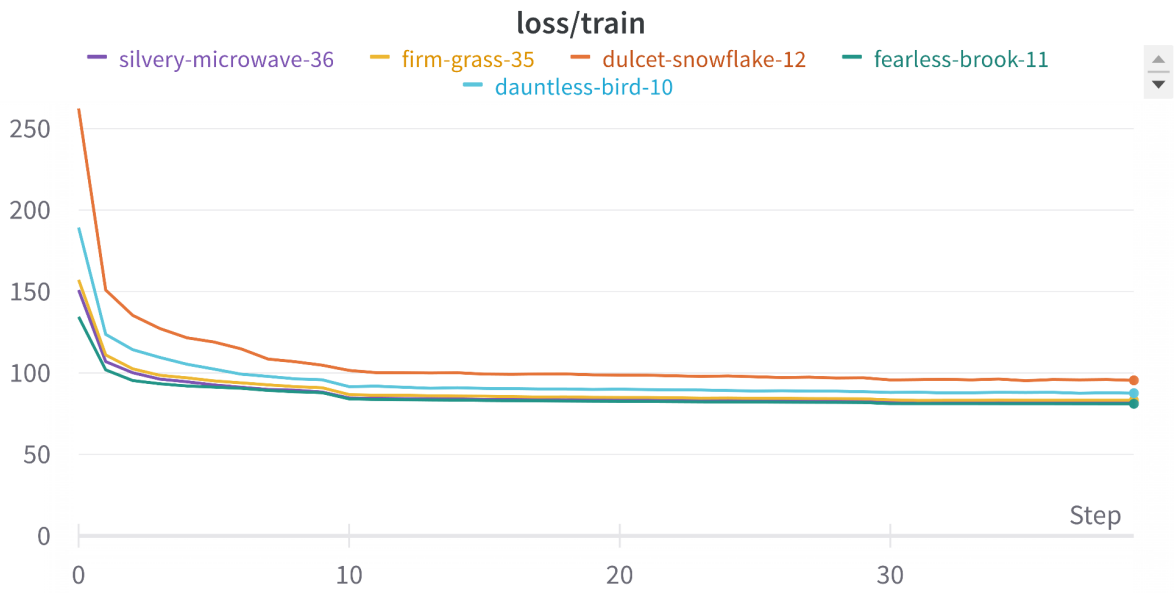

Figure 1. Train loss curves for 10, 11, 12, 35 and 36

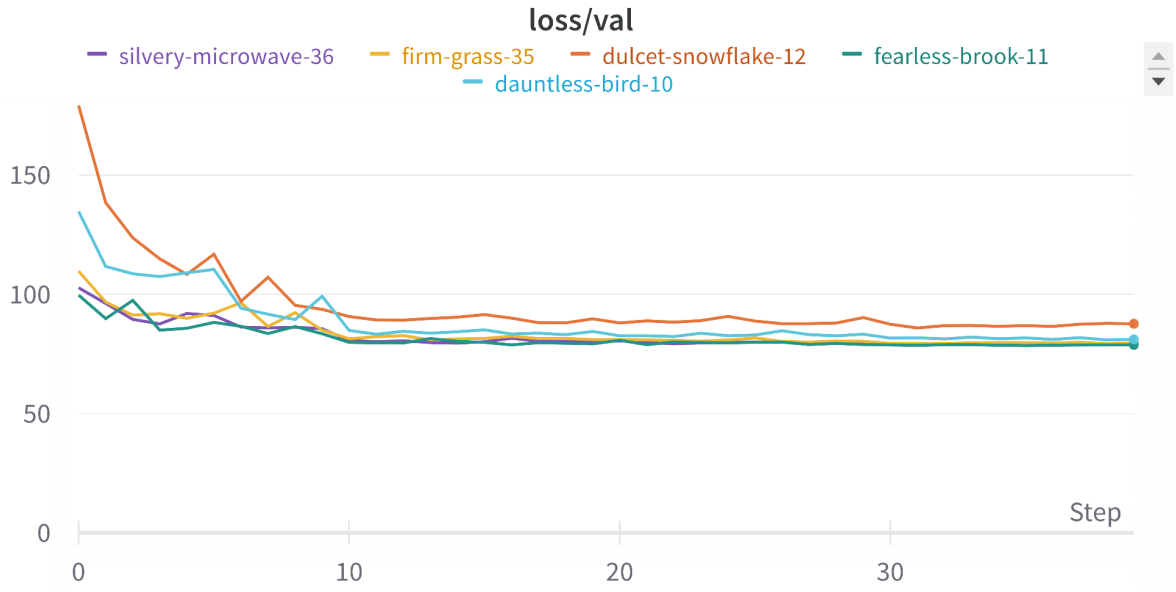

Figure 2. Validation loss curves for 10, 11, 12, 35 and 36

Batches (% of)	Best loss value (A-MPJPE, mm)			Best train loss epoch
	Train	Validation	Test Average actions	
10	95.27	85.93	92	35
25	87.58	80.88	87.7	37
50	83.14	79.3	**84.7**	**31**
75	81.62	78.7	85.2	38
100	**76.08**	**78.54**	85.2	39

Table 1. Effect of using more and more % of batches on train, validation and test losses

Results in table 1 make sense.
More data offers the model more exposure to the underlying **manifold** in which the data is assumed to live in the manifold hypothesis.

In figure 2 we can notice that the noise in the validation data tends to decrease upon increasing the percentage of batches used in training, validation and test.
Using more data is a **trivial regularization** technique and so it may be responsible for the decrease in noise.
Also, the habitual good practice of performing train/validation/test splitting of the data results in validation and test that are performed on **smaller** quantities of data, which may cause some noise in the computation of performance metrics like losses and accuracies.

Plots and data allow us to answer the questions posed for this first section.

"**Is there evidence of overfitting or underfitting in the initial training results?**"
Both training and validation losses are generally going down, so **no overfitting nor underfitting** is happening.
What we mean with "generally" is that, overall, we have a constant downtrend for both curves.

"**Are there fluctuations in training and validation loss or accuracy? If so, what might be causing them?**"
In some specific epochs, we do observe some spikes in the validation loss, which may be due to validation being performed on smaller amounts of data and/or a too fast learning rate.
The latter may make the model reach a local minimum, which corresponds to learning noise in the data as well.

"**What can you infer from the initial learning rate, milestones, and weight decay settings?**"

1. At the end of the fixed epoch budget, learning rate, milestones (so, learning rate scheduler) and weight decay are performing their job, because both losses eventually decrease to some level of **convergence**.
2. **Weight decay** performed at milestone epochs 10 and 30 **help** train and validation losses get out of plateauing situations
3. Nevertheless, train and validation losses have "ideal shapes" across all epochs
4. If one wants to be picky, validation loss could have a better-looking "elbow-like shape", in epochs < 10.
   This may probably depend on too small weight decay.

Overall, we think that default hyperparameters are fine as they are, since they do not yield to overall over or underfit.
Nevertheless, we try to address the aforementioned observations in the next section.

2. Hyperparameters fine-tuning

The objective of this section is to address observations 2 and 4 at the end of the previous section.

Since after each milestone epoch we do get a nice decrease in both losses, in earnest-morning-46 we try milestones of 10, 20 and 30, keeping gamma set to 0.1
The addition of the extra, intermediate milestone made the three losses overall a little bit higher, which is expected, considering that we made the learning rate progressively smaller.
Evidently, the third milestone makes the learning too small.

We notice a pattern where overfit starts to happen approximately after epochs 5, 10 and 20, so we decided to try them as milestones, using the gamma of 0.1, 0.75, 0.5 and 0.25 in genial-snowflake-47, fancy-thunder-48, absurd-music-49 and woven-totem-50

Assuming to have found the right milestones thanks to 46, the idea of testing multiple gammas was to understand whether the usual 0.1 was a too strong decay which then impacted the learning ability of the model, especially after the very last milestone, considering how MultiStepLR decays the learning rate.

Our intuition proved to be correct and 49 consistently manages to get smaller and smaller validation losses, when compared to 46 (figure 3 and 4), but the overall improvement against 36 is basically negligible in validation loss and absent in train and test (table 2).
47, 48 and 50 performances are way worse than 49, confirming the hypothesis that high gamma values do not help across all epochs (figure 5).

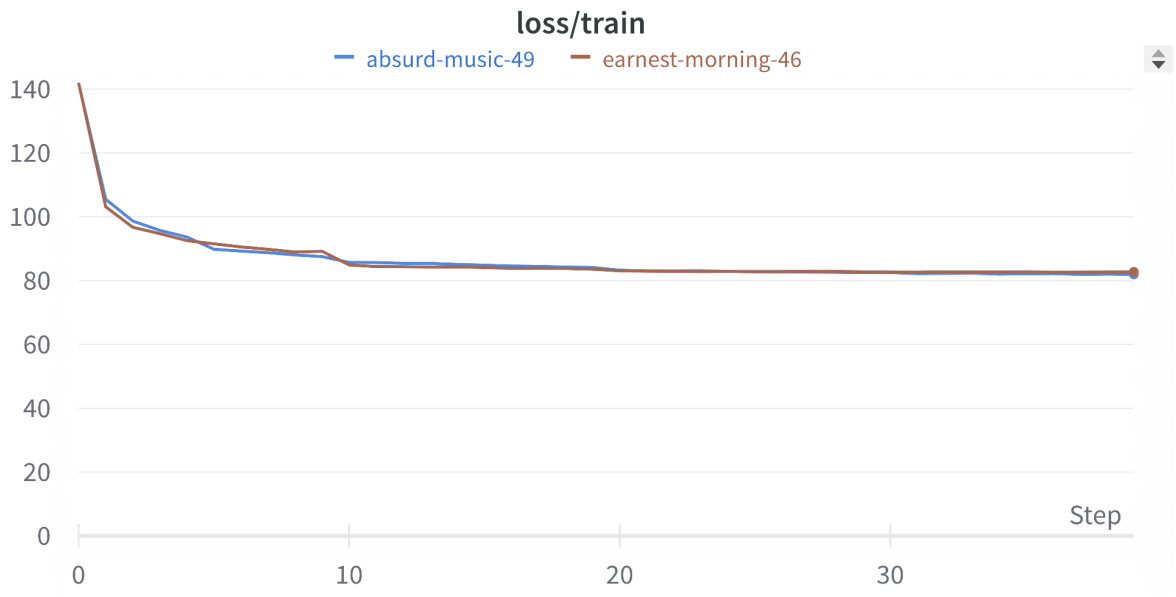

Figure 3. Train loss curves for 46 and 49 basically have the same exact behavior

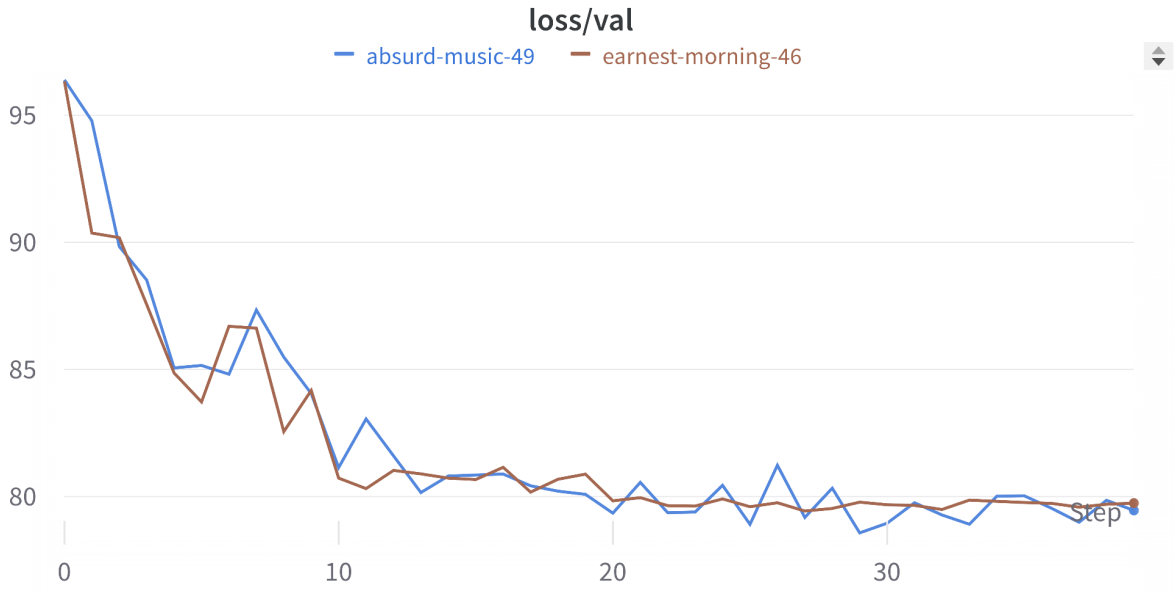

Figure 4. Validation loss curves for 46 and 49.
Smaller gamma learning rate decay in 49 makes validation loss slightly noisier but makes it consistently reach smaller errors in late epochs

Run	Starting LR	Milestones	Gamma	Best loss value (A-MPJPE, mm)		
				Train	Validation	Test avg actions
36	1e-1	10, 30	0.1	76.33	78.71	**85.2**
46	1e-1	10, 20, 30	0.1	**76.27**	79.44	85.8
49	1e-1	5, 10, 20	0.5	81.95	**78.58**	85.9

Table 2. Effect of different milestones and gamma configurations, using 75% of batches. Overall differences are negligible and may be due to stochasticity of batch sampling or other phenomena

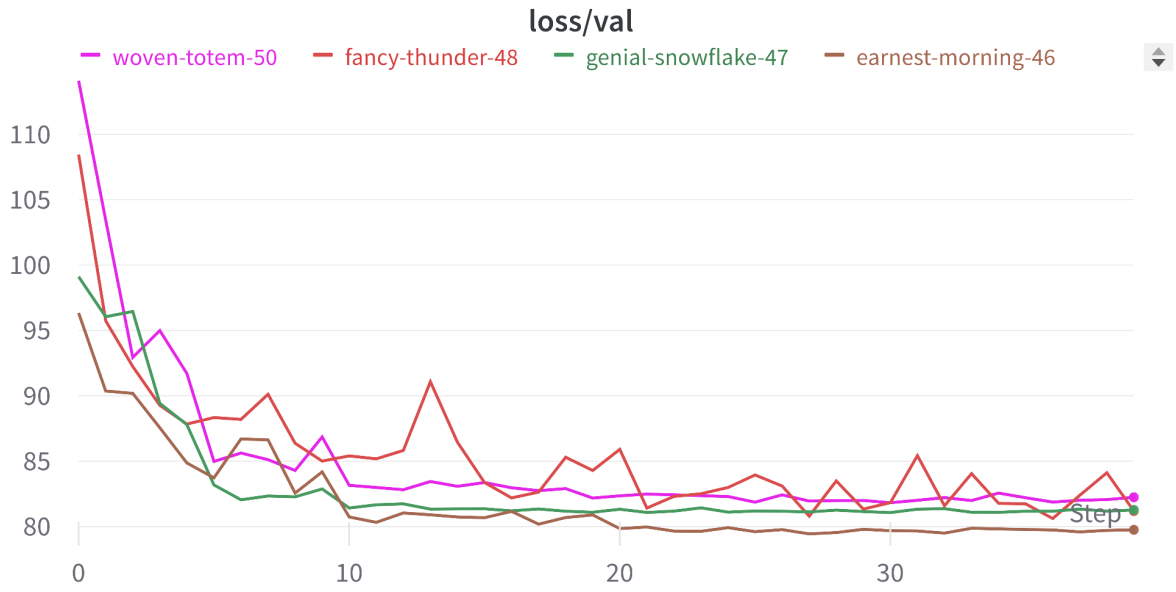

Figure 5. Validation loss of 47, 48 and 50. Higher gamma values do not help across all epochs

Looking at data and plots gathered in 46, 47, 48, 49 and 50 we were pretty sure that starting with a lower learning rate would have not improved the performances, especially because our (negligibly) best configuration 49 almost immediately decays the learning rate at epoch 5.

Weight decay is another important element to combat noise in the validation loss.
Considering the negligible effect of milestones and gamma in previous attempts, we started looking into weight decay taking as reference 49, which was our (negligibly) best configuration yet.

We started by testing multiple weight decay orders of magnitude. Of course, we expect that the stronger the weight decay, the higher the losses, but we still had to somehow "brute-force" a starting order of magnitude value in order to then make finer decisions.

We tested weight decays of 1e-6, 1e-5 (default value), 1e-4, 1e-3 and 1e-2 in different-pine-57, 49 (and 46), lyric-frog-54, eternal-sponge-55 and whole-aardvark-56.
They all performed worse than 49 (and 36).

At this point, we consider having pushed learning rate scheduler and weight decay as much as possible and we argue that original hyperparameters (36) were good enough, since our best configuration (49) yields very marginal improvements only on the validation loss.

Across all runs we almost always got the expected behavior, considering the change in hyperparameter performed (e.g. too strong weight decay led to worse performances, too small gamma increased noise due to larger learning rate being used, etc.)

To conclude, as an extra test, we wanted to work on model-related hyperparameters.

We tested 1, 2 and 4 attention heads in 49, major-surf-58 and crimson-leaf-59, for a total number of 23k, 40k and 68k trainable parameters and in youthful-wildflower-60 we tried to double the number of out channels of convolutions, for a total of 1.2M params.

Improvements were absent in 59.
58 brings negligible improvements to the validation loss, considering the 1.75x more parameters requirement, compared to 49 and 36 (table 3)

60 brings train loss down to by 9 mm (table 3), starting to overfit from epoch 13 (figure 6 and 7)
This discovery is very important, because:

1. It tells us that the model benefits a lot more from larger time dimension "exploding" maps, rather than attending multiple times looking for different patterns (using different heads)
2. Gives us new possible hyperparameter tuning directions

As final test, since overfit starts from epoch 13 in 60, we tried 5, 10, 15 milestones with gamma 0.25 in dandy-grass-61, 0.1 in still-water-62 and 0.05 in stellar-yogurt-63 (table 4).

62 fixes 60 overfit (figure 8) and reaches a lower validation loss than 36 and 49 (figure 9)

Figure 6. Bigger network in 60 manages to bring the train loss down by 9 mm, w.r.t. 36

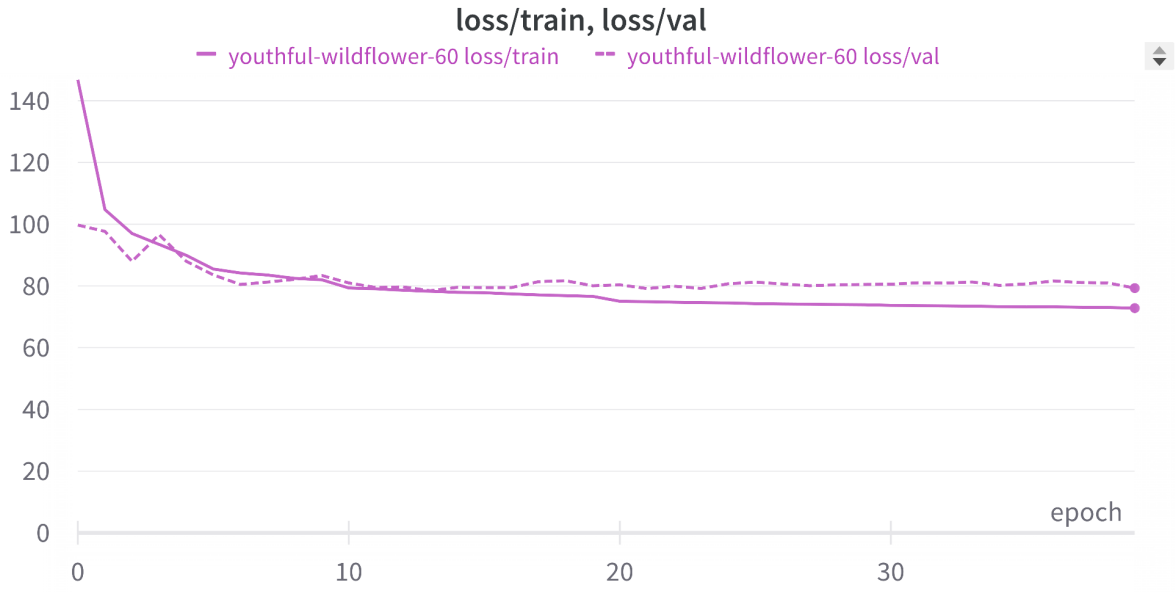

Figure 7. Bigger network in 60 starts to overfit since epoch 13

Run	Attention heads	Hidden channels	Trainable params	Best loss value (A-MPJPE, mm)		
				Train	Val	Test avg actions
36	1	16	23k	81.62	78.71	**85.2**
46	1e-1	16	23k	76.27	79.44	85.8
49	1	16	23k	81.95	78.58	85.9
58	2	16	40k	79.26	**78**	86.3
59	4	32	68k	78.17	79.25	86.4
60	2	32	1.2M	**67.17**	78.45	86.6

Table 3. Effect of increasing number of attention heads and convolution output channels
Strong parameter number increase does not yield considerably better performances, with current regularization and optimizer-related hyperparameters

Run	Milestones	Gamma	Trainable params	Best loss value (A-MPJPE, mm)		
				Train	Val	Test avg actions
36	10, 30	0.1	23k	76.33	78.71	85.2
46	10, 20, 30	0.1		76.27	79.44	85.8
49	5, 10, 20	0.5		81.95	78.58	85.9
60			1.2M	**67.17**	78.45	86.6
61[1]	5, 10, 15	0.25		74.78	78.97	85.7[2]
62[1]		0.1		80.28	**77.32**	**84.7**[2]
63		0.05		80.17	78.4	85

Table 4. Effect of new milestones 5, 10, 15 and different gamma settings to combat overfit witnessed in 60

[1] Training not completed up to all epochs due to system crash 🧨
[2] Test value not uploaded to Weights and Biases due to system crash

3. Conclusion

We performed hyperparameters fine-tuning starting from the provided default configuration to get some baseline result.
We limited our tests to 75% of the batches for time and resources constraints.

Looking at the initial hyperparameters configuration, we observe **absence** of overall **overfit** and **underfit**.
Learning rate decay of epochs 10 and 30 **favors** important train and validation loss decreases.
Further experiments on various **milestone epochs** and learning rate decay strengths yield very **negligible** (0.2 mm) train and validation losses **improvements**, which may be attributed to stochastic phenomena in place in training, like weight initialization and batch sampling.

Weight decay experiments yield no improvement whatsoever, so we move to consider **model-related hyperparameters**, like number of attention heads and output channels of the convolutional layers.
Using 2 attention heads brings an insignificant 0.58 mm validation loss improvement, which, once again, may be attributed to stochastic phenomena in place during training.

Using 32 output convolutional channels (vs. 16) results in a 9 mm lower train loss, but shows signs of overfitting, starting from epoch 13, which is probably normal, considering the usage of 52x more trainable parameters (1.2M vs. 23k).

Refining **milestone epochs** and **gamma** parameters, we manage to **combat** such **overfit** to reach the best absolute validation loss of 77.32 mm (vs. 78.71 using default hyperparameters).

Overall, the proposed architecture does react to learning rate scheduling, weight decay, output convolutional channels and number of attention heads, but the end result is not that far from the original hyperparameters.

We feel like we pushed this architecture to the maximum of its capabilities, considering the already very good performing default hyperparameters and argue that massive performance improvements can be only be achieved by changing the **inductive bias**, by changing model **layers** or data **representation**.

Explicit answers to the questions in the notebook:

“**Has parameter fine-tuning improved model performance?**”
Yes, hyperparameter **fine-tuning** slightly **improved performances**, across all three losses.
Unfortunately, such improvements came in three different models, so there is no unique run with absolutely lowest losses across the board.
Considering the huge amount of extra trainable parameters that the better models have (1.2M vs. 23k), we do **not** consider the improvement **worth it**.

“**Did it mitigate overfitting or underfitting issues?**”
Original hyperparameters had a very “localized” overfit in some epochs and not an overall overfit trend (where val loss continuously grows, while train loss stays flat or decreases).
The only overall overfit case (where validation loss continuously grew, while train loss stayed down) we witnessed happened on a configuration of ours and, in that run, yes, hyperparameter tuning did solve the issue.

“**What can you conclude about the optimal hyperparameters for this task?**”
Original hyperparameter configuration was already very good.
Model positively, although negligibly, **reacts** to different **milestone epochs**, **learning rate decay strength**, **number** of attention **heads** and output **convolutional channels**.
In our tests, original weight decay proved to be the best, probably due to having been tested with other regularizations already in place.

All hyperparameters responded the way we hypothesized, e.g. higher loss if too much weight decay is applied, or faster initial loss, then noise if too strong learning was used, without decay.