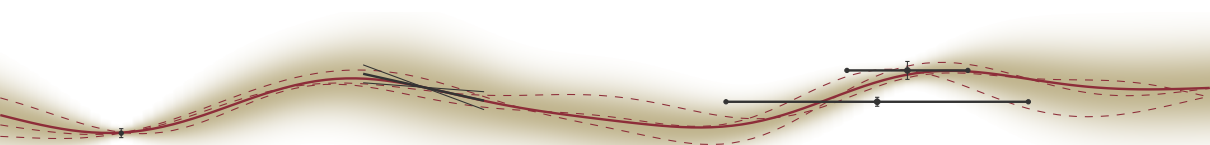
Numerics of Machine Learning
Tutorial 11
Optimization for Deep Learning

Frank Schneider

Eberhard Karls
Universität
Tübingen

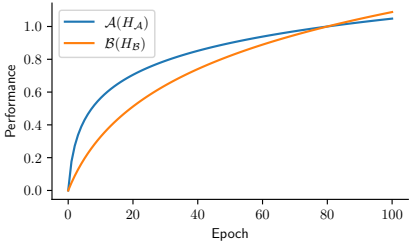
Faculty of Science
Department of Computer Science
Chair for the Methods of Machine Learning



EXAMple Question

What is wrong with this statement?

Assume we have two update rules $\mathcal{A}$ and $\mathcal{B}$ which each have a set of hyperparameters $\mathcal{H}_{\mathcal{A}}$ and $\mathcal{H}_{\mathcal{B}}$. In the following, we consider the performance of both methods only on a single problem (e.g. a single neural network using a fixed dataset).
Both methods are now tuned using random search with 20 tuning trials. The hyperparameters are tuned for $\mathcal{A}$ and $\mathcal{B}$ independently with the search spaces given by $\Phi(\mathcal{A})$ and $\Phi(\mathcal{B})$. The specific hyperparameter setting that reached the best-observed performance after 100 epochs are denoted by $H_{\mathcal{A}}$ and $H_{\mathcal{B}}$. The performance curves (higher means better) of both training algorithms $\mathcal{A}(H_{\mathcal{A}})$ and $\mathcal{B}(H_{\mathcal{B}})$ are shown in the figure.

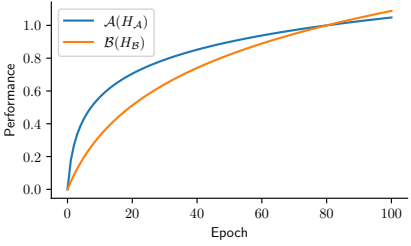


What is wrong with the following statement:

"On this problem, although $\mathcal{B}$ provides a slightly better final performance, $\mathcal{A}$ offers faster training, and should be preferred if trained for 80 *epochs or less."*

EXAMple Question

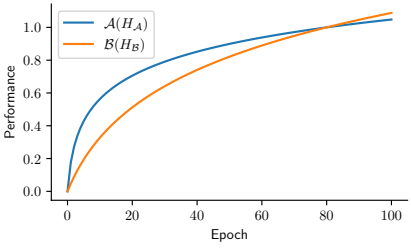
Some possible answers (non-exhaustive!)



- **You cannot make a statement about which algorithm should be preferred for 80 epochs, if you tuned for performance at 100 epochs.**

EXAMple Question

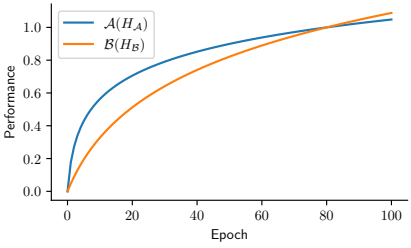
Some possible answers (non-exhaustive!)



- **You cannot make a statement about which algorithm should be preferred for 80 epochs, if you tuned for performance at 100 epochs.**
  - Saying $\mathcal{A}$ is "better" for 80 epochs is incorrect, since tuning B for best performance at 80 epochs could in fact be better.

EXAMple Question

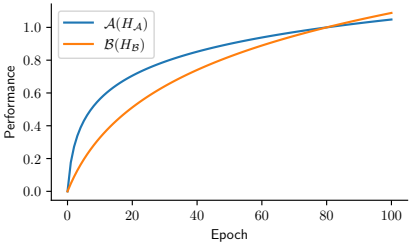
Some possible answers (non-exhaustive!)



- **You cannot make a statement about which algorithm should be preferred for 80 epochs, if you tuned for performance at 100 epochs.**
  - Saying $\mathcal{A}$ is "better" for 80 epochs is incorrect, since tuning B for best performance at 80 epochs could in fact be better.
  - This is most visible when you think about learning rate schedules. Typically they are designed to provide the best performance at the end.

EXAMple Question

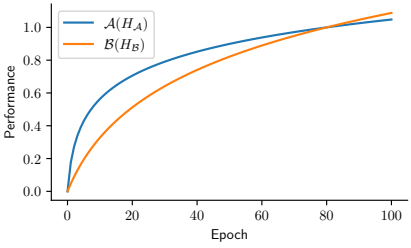
Some possible answers (non-exhaustive!)



- **You cannot make a statement about which algorithm should be preferred for 80 epochs, if you tuned for performance at 100 epochs.**
  - Saying $\mathcal{A}$ is "better" for 80 epochs is incorrect, since tuning B for best performance at 80 epochs could in fact be better.
  - This is most visible when you think about learning rate schedules. Typically they are designed to provide the best performance at the end.
- **It is dangerous to draw conclusions from just a single run.** Are the differences significant?

EXAMple Question

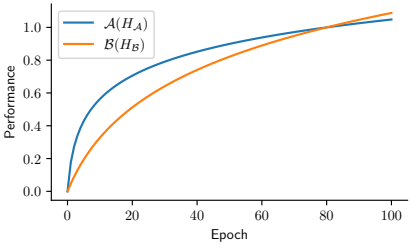
Some possible answers (non-exhaustive!)



- **You cannot make a statement about which algorithm should be preferred for 80 epochs, if you tuned for performance at 100 epochs.**
  - Saying $\mathcal{A}$ is "better" for 80 epochs is incorrect, since tuning B for best performance at 80 epochs could in fact be better.
  - This is most visible when you think about learning rate schedules. Typically they are designed to provide the best performance at the end.
- **It is dangerous to draw conclusions from just a single run.** Are the differences significant?
- **It is imprecise to talk about $\mathcal{A}$ and $\mathcal{B}$.** Instead something like $\mathcal{A}(\Phi, H)$ would be more precise as $\mathcal{A}$ and $\mathcal{B}$ could be the same.

EXAMple Question

Some possible answers (non-exhaustive!)



- **You cannot make a statement about which algorithm should be preferred for 80 epochs, if you tuned for performance at 100 epochs.**
  - Saying $\mathcal{A}$ is "better" for 80 epochs is incorrect, since tuning B for best performance at 80 epochs could in fact be better.
  - This is most visible when you think about learning rate schedules. Typically they are designed to provide the best performance at the end.
- **It is dangerous to draw conclusions from just a single run.** Are the differences significant?
- **It is imprecise to talk about $\mathcal{A}$ and $\mathcal{B}$.** Instead something like $\mathcal{A}(\Phi, H)$ would be more precise as $\mathcal{A}$ and $\mathcal{B}$ could be the same.
- Phrasing such as "on this problem" or "performance" might be unclear or ill-defined.

Coding Question

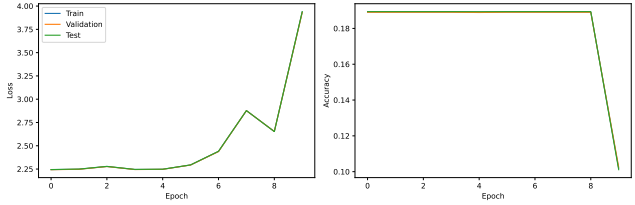
An example debugging scenario

Assume your colleague is trying to train a neural network but it is clearly not getting the results they think are achievable (as measured by the accuracy on unseen data). They hand over their current code to you and you can find the code in the notebook file `Exercise_11.ipynb`. Your task is to debug the neural network training process in the provided notebook file, improve the network's performance as much as you can, and create a logbook detailing your process.

Running the code once

The first impression of the debugging and tuning task

- Code takes a long time to run!
- Network doesn't train at all!



Debugging Strategy I

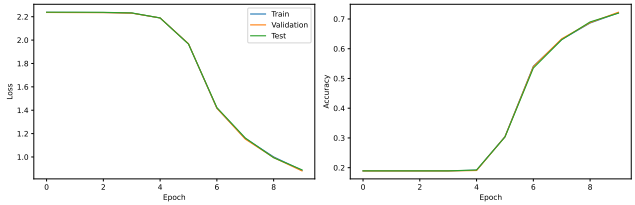
Reduce the runtime for faster iterative debugging and tuning

- Try to run it on a GPU/cluster/etc.
- Reduce the runtime of the script:
  - Turn off COCKPIT.
  - Reduce the tracking frequency of COCKPIT.
  - Reduce or skip the evaluation part (and just focus on the mini-batch training loss).
  - Reduce the number of epochs (or even train for just a few steps).
- Improve the performance and efficiency of the existing code (e.g. max out batch size, improve data loading, etc.)
- Look at individual modules (e.g. data, model, etc.) in isolation.

Debugging Strategy II

Carefully examine the code

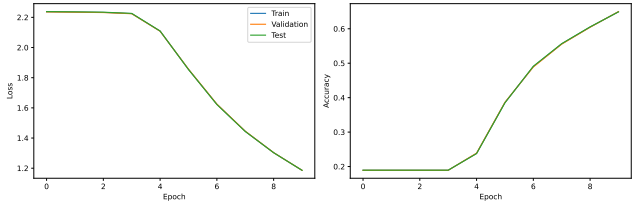
- A valid strategy: Carefully look through the code
- Doesn't require running the script at all
- **Bug: No zero_grad()**
  - In every step (i.e. every mini-batch) we need to set the gradient to zero
  - Otherwise, we are doing gradient accumulation (i.e. simulating a larger mini-batch)
  - With this fix alone we can already (somewhat) train our network. Success!



Debugging Strategy III

Turning off those bells and whistles

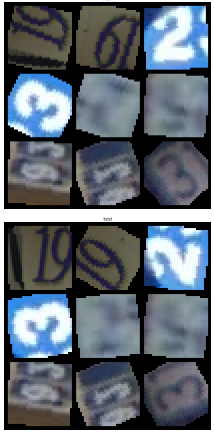
- Start with a simple thing that works and then add fancy stuff on top (augmentations, schedules, etc.)
- In this case, I turned off:
  - Data augmentation (is it necessary? Training is faster without it)
  - LR schedule



Debugging Strategy IV

Isolate modules

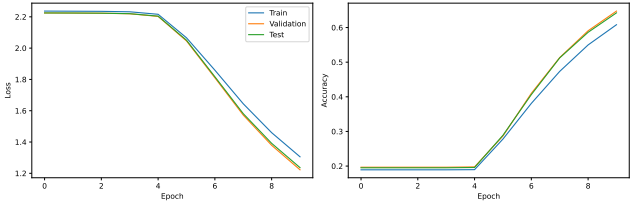
- Try to look at each module (e.g. data, model, training loop, etc.) in isolation.
- For example, debugging the data pipeline (mostly) doesn't require you to actually perform a training run!
- Visualize your data:
  - Show train/valid/test set
  - Turn off shuffle and random split for debugging purposes
  - Add back the data augmentation for debugging purposes
- **Bug: Same data for train/valid/test**
- **Bug: Inappropriate data augmentation**
  - Rotations are dangerous as "6" and "9" can be confused
  - Currently we don't seem to be limited by data, so we don't need augmented data
  - Another common but inappropriate augmentation would be vertical shifts



Debugging Strategy IV

Isolate modules

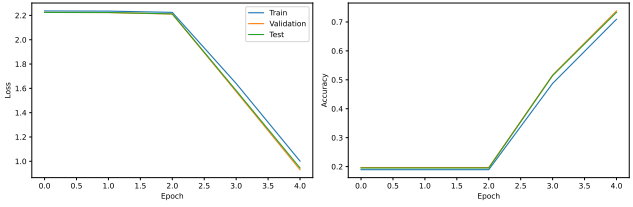
- After fixing both bugs (same data for train/valid/test and the inappropriate data augmentation)



Debugging Strategy V

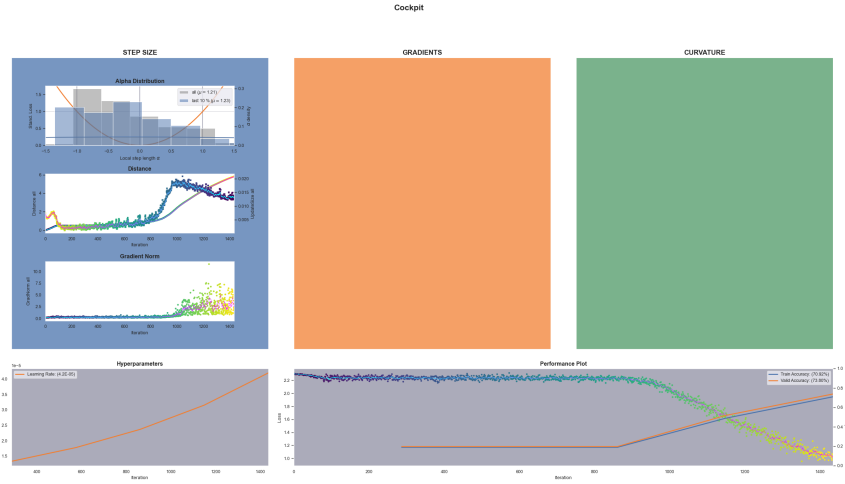
Cockpit

- As an additional debugging and tuning strategy we can have a look at COCKPIT
- Uses the already setup quantities and tracking frequencies
- We can observe:
  - Learning rate schedule looks weird (increases?)
  - Learning rate seems to be too small (initially, without the increase)



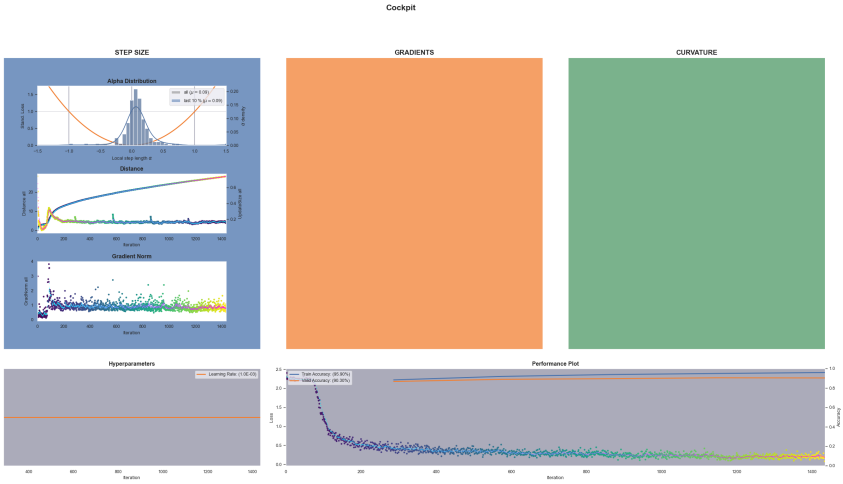
Debugging Strategy V

Cockpit: Before



Debugging Strategy V

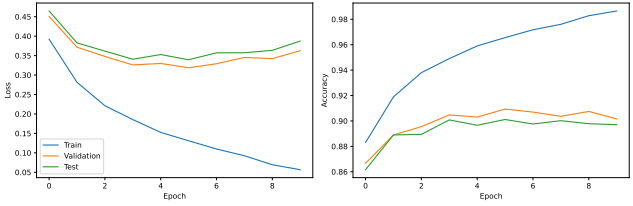
Cockpit: After (constant learning rate schedule)



Debugging Strategy V

Cockpit

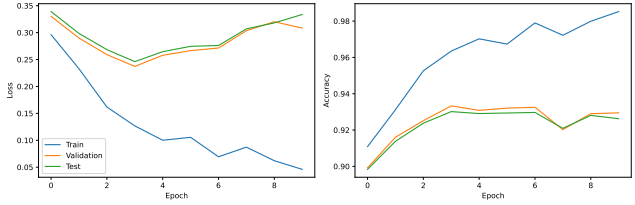
- As an additional debugging and tuning strategy we can have a look at COCKPIT
- Uses the already setup quantities and tracking frequencies
- We can observe:
  - Learning rate schedule looks weird (increases?)
  - Learning rate seems to be too small (initially, without the increase)



Debugging Strategy VI

Tuning

- We can further tweak our training algorithm hyperparameters, models, etc. to improve the performance.
- Ideally, we would do most of this jointly, but this is not always feasible.
- In this case, I did
  - Change from tanh to ReLU activation (decreased the train performance, but increased on validation/test)
  - Added batch normalization layers
  - Increased the learning rate further
- This can certainly be done more extensively!



Debugging Strategies

- Carefully examine the code
- Turning off bells and whistles
- Isolate modules & visualize the data
- Cockpit
- Tuning

Bugs

- No zero_grad()
- Loading train data twice (for test set as well)
- Inappropriate data augmentation
- Learning rate schedule
- Learning rate
- Model inefficiency (tanh activation)

Additional tips

- In a more realistic (production) scenario, I would:
  - Train for longer
  - Tune the learning rate with every modification I do (e.g. model changes)
  - Average all results across multiple seeds (e.g. reruns) to show whether they are significant
  - More carefully organize my experiments (e.g. some experiment tracking)
- **"A Recipe for Training Neural Network"** by Andrej Karpathy
- **Deep Learning Tuning Playbook**

