The periphery

configuration trackers, hypertuning, schedulers, linting, typehinting

deep learning 4

R. Grouls, 22 mei 2023

Manual search

Manual tuning is necessary to develop some intuition about complexity of models.

When encountering a problem, you will be able to make a guess:

- This problem seems more complex than MNIST, but not as complex as the flower photos;
- Let’s start with settings somewhere between what worked for those two

However, *after* you have an intuition, it can be helpful to make the searching less random

The size of the search space

That escalated quickly

A typical mistake of a beginner is to make the search space way to big.

Let’s say you consider amounts of:

- Linear units (between 16 and 256)
- Depth of linear layers (between 1 and 10)
- Convolution filters (between 16 and 64)
- kernel size (between 3 and 5)
- dropout (4 options)
- depth of convolutions (between 1 and 5)
- learning rate (4 options)
- optimizers (3 options)
- activation functions (3 options)
- batchsize (between 16 and 128)

Can you calculate how big the search space is?

The curse of dimensionality

Finding a grain of sand

This search space has almost 2.7×10^{10} options. This is roughly the amount of sand when you fill your $40m^2$ living room with 50 cm of sand.

If you test at random 50 options, you are just scanning a very small fraction of all options.

You shouldn’t be surprised if the result of random hypertuning is worse than the result of your manual hypertuning!

Keeping track

I can’t keep track of all the trackers

There are a lot of ways to keep track of your experiments.

- **Gin-Config**: A simple configuration library for Python that uses dependency injection to simplify the process of configuring methods.
- **TensorBoard**: A visualization toolkit for tracking and visualizing metrics such as loss and accuracy, view the architecture of your model, and visualize embeddings. An unsustainable but common hack is to add parameters to the name of the model.
- **MLflow**: An open-source platform for managing the end-to-end machine learning lifecycle, including experimentation, reproducibility, and deployment.
- **Ray**: A framework-agnostic, open-source, high-performance distributed computing library that provides a simple, universal API for building distributed applications. The ‘tune’ submodule has extensive options for distributed hyperparametertuning experiments.

Hypertuning

Algorithms for tuning algorithms

While hypertuning manually is a great way to obtain

intuitions and gutfeelings

of what works, we need to move to algorithmic hypertuning when the searchspace is

getting too complex for gutfeelings.

You should learn to work together with the hypertuning algorithms, and understand which part they can take over and which part you should (learn) to do.

Tuning frameworks

Algorithms for tuning algorithms

HYPEROPT

- we will use Ray tune
- HyperOpt is another alternative
- You want a framework that is agnostic to ML frameworks; this way you dont have to switch if your framework changes.
- Ray offers parallel computing, also for training, data preprocessing, etc.

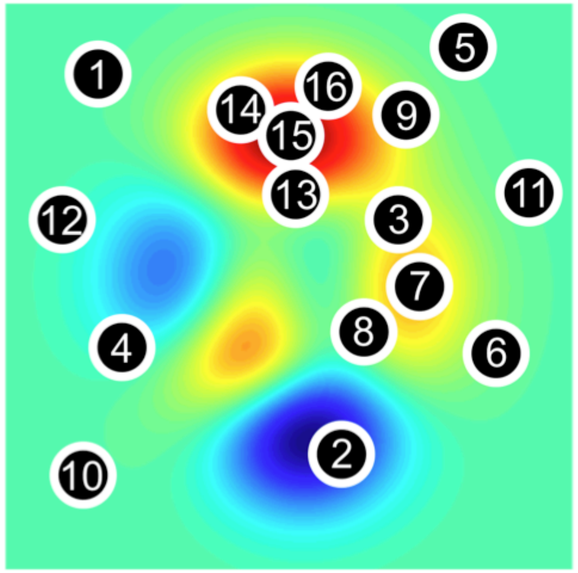
	HyperOpt	Optuna	Keras-Tuner	HpBandSter	Tune
Distributed Execution	✅	✅	☐	✅	✅
Fault Tolerance	☐	☐	☐	☐	✅
Search algorithms	2	4	3	3	6
Framework Support (TF/Keras, Sklearn, PyTorch)	✅	✅	☐	☐	✅

Baysian Hyperband

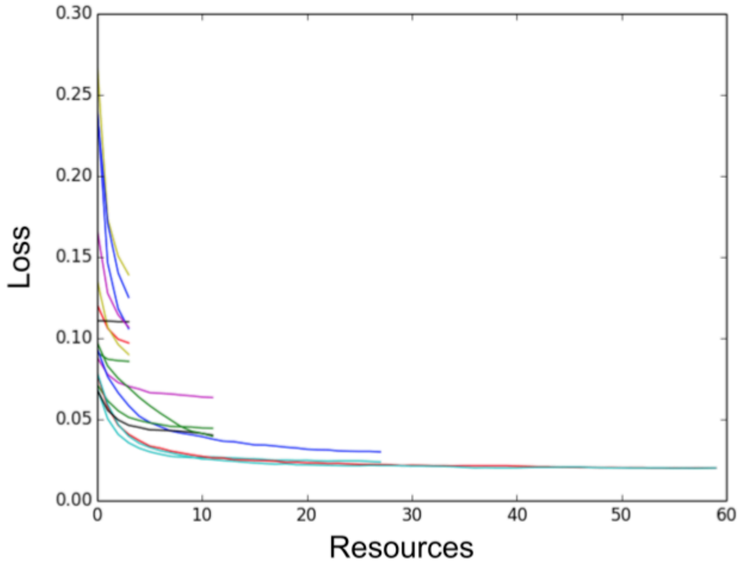
Algorithms for tuning algorithms

Some nice solutions are:

- bayesian search (a)
- Tree Parzen Estimators (bayesian optimization over awkward search spaces with real, discrete and conditional parameters)
- hyperband (b)
- Bayesian Optimization and Hyperband (BOHB) combines bayesian & hyperband techniques.



(a) Configuration Selection



(b) Configuration Evaluation

Learning rate schedulers

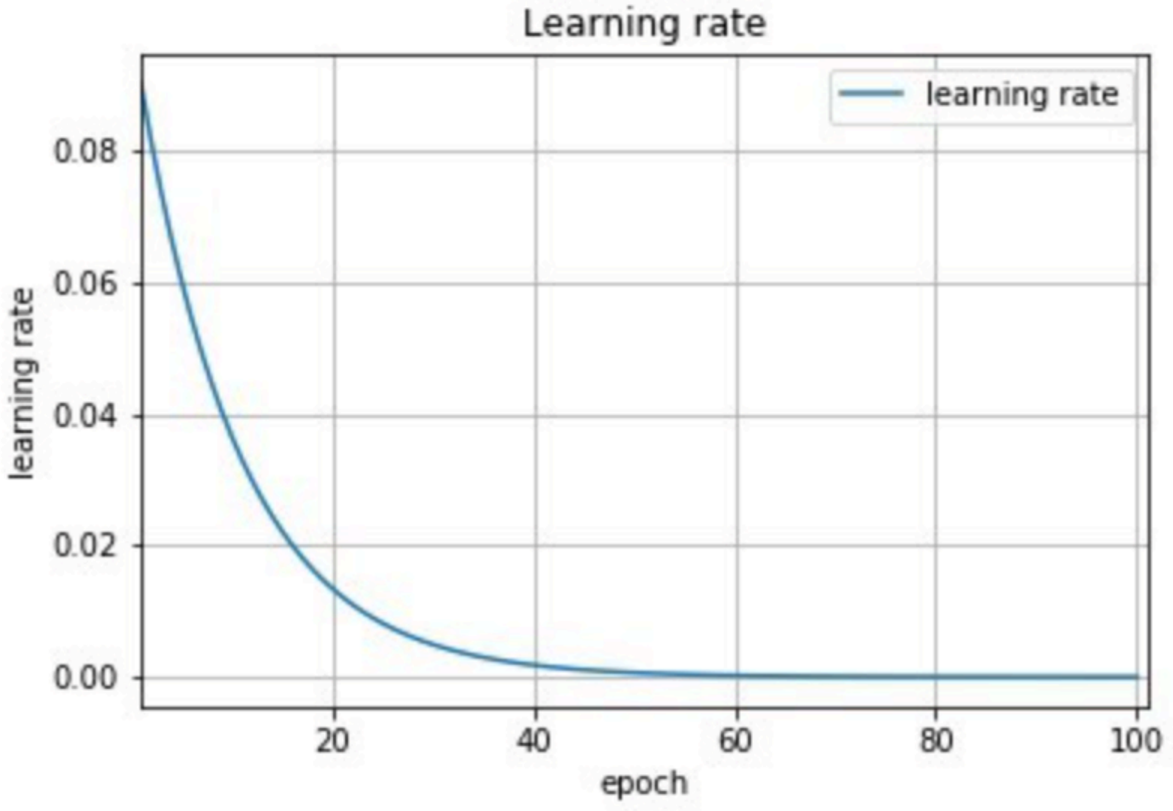
When to use a chainsaw or a pocket knife

The learning rate η tells how fast weights are adjusted

$$W_{t+1} \leftarrow W_t - \eta \times gradient$$

The learning rate can be compared to different sizes of tools to carve wood. Big tools go fast, but miss the details.

So, often a decay of the learning rate is a good idea.

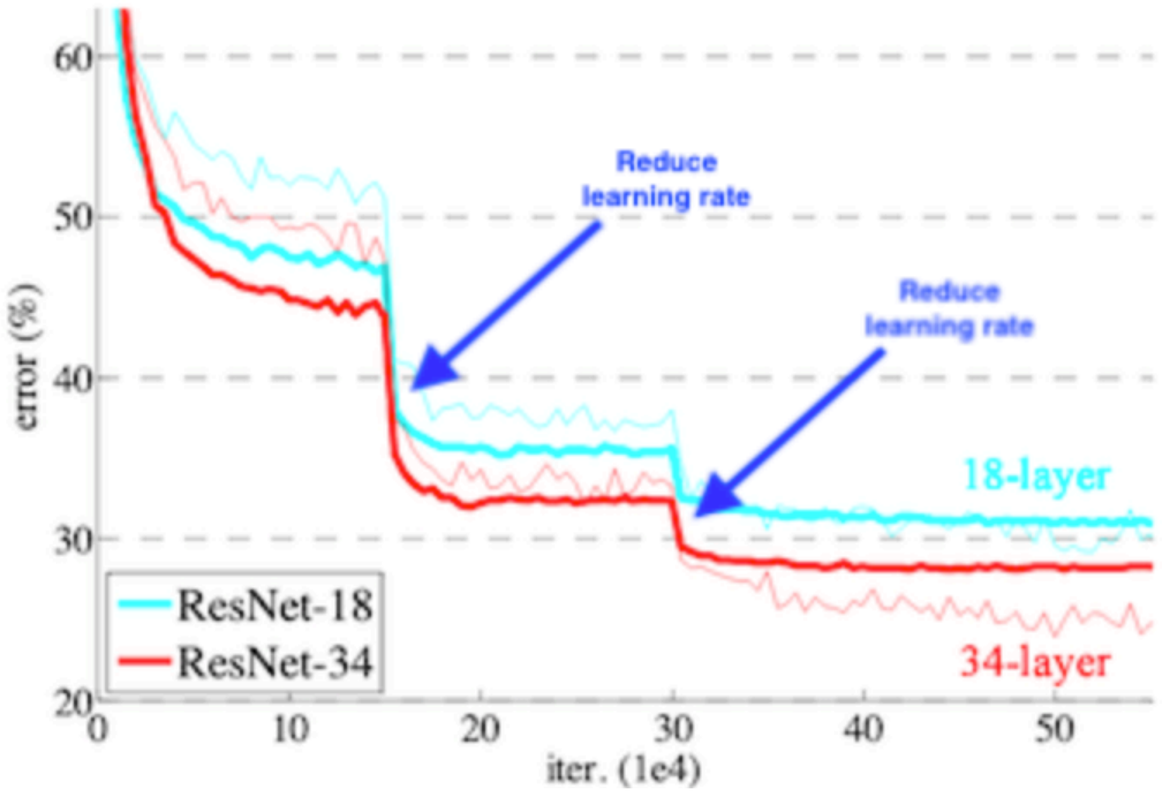


Learning rate schedulers

ReduceLROnPlateau

ResNet can be trained for very long. From the paper:

> *"The learning rate starts from 0.1 and is divided by 10 when the error plateaus, and the models are trained for up to* 60×10^4 *iterations"*



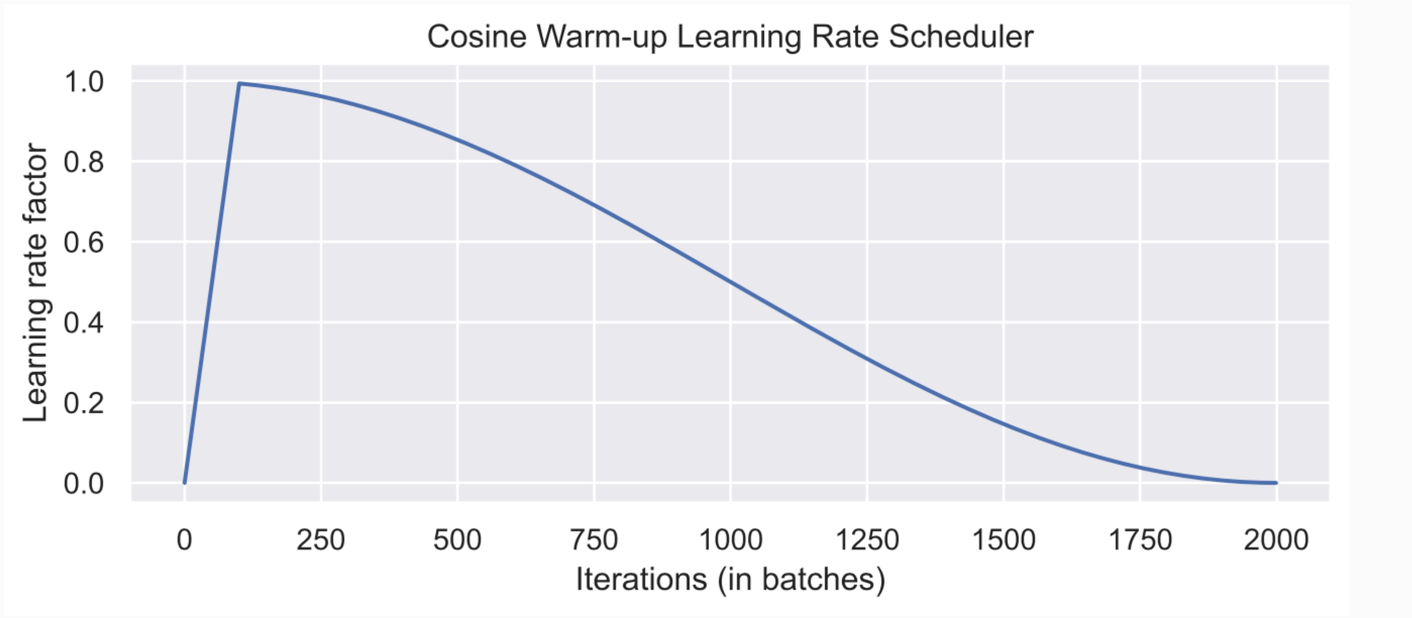
Learning rate schedulers

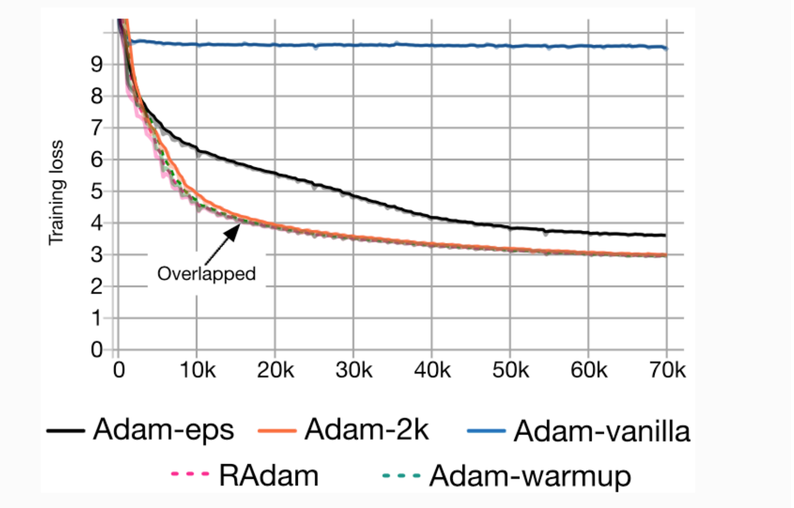
Learning rate warm up

The first iterations can have very high gradients, and could “lock” the model in the wrong direction.

A solution is to gradually increase the learning rate from 0 on to the originally specified learning rate in the first few iterations.

This is typically used with transformer architectures.





Code formatter

Any color you like as long as it is black

From the *black* github readme:

> *Black* is the uncompromising Python code formatter. By using it, you agree to cede control over minutiae of hand-formatting. In return, *Black* gives you speed, determinism, and freedom.
>
> You will save time and mental energy for more important matters.
>
> Watch the PyCon talk for more

Typehinting

Typehinting was introduced in Python with PEP 484

The mypy package checks your type hinting:

- Makes it easier to find bugs, with less debugging
- Typing acts as machine-checked documentation
- Makes your code easier to understand

Simply run `mypy src`

Highly recommended

In professional environments, this is often mandatory.

- This is the minimal setup:
  - `poetry add black flake8 isort mypy pep8-naming flake8-annotations --group=dev`
- You are done when you can run this on your folder (eg `/src`) without errors:
  - `poetry run isort src`
  - `poetry run black src`
  - `poetry run flake8 src`
  - `poetry run mypy src`
- See the `Makefile` for easy shortcuts.