Neural networks: practical guide

Pavel Ostyakov

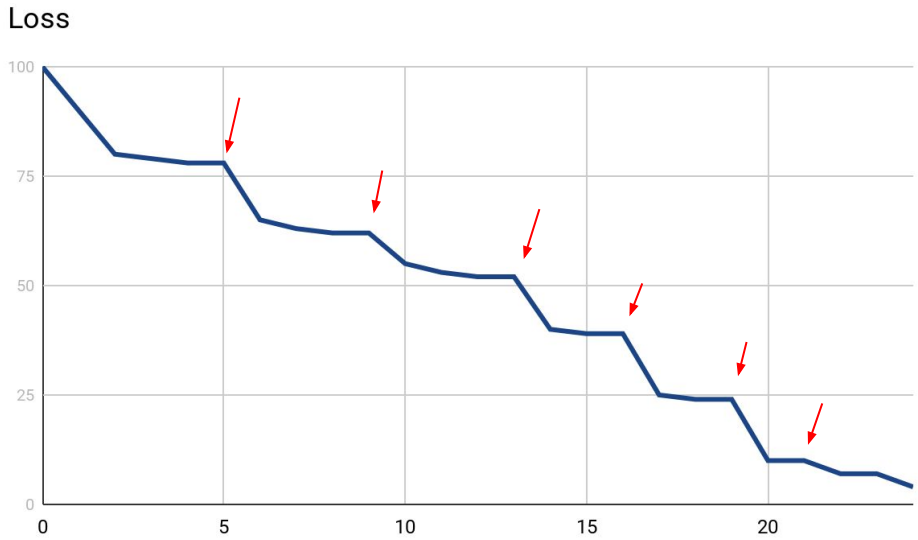
How to improve training of NNs?

Learning Rate Scheduling

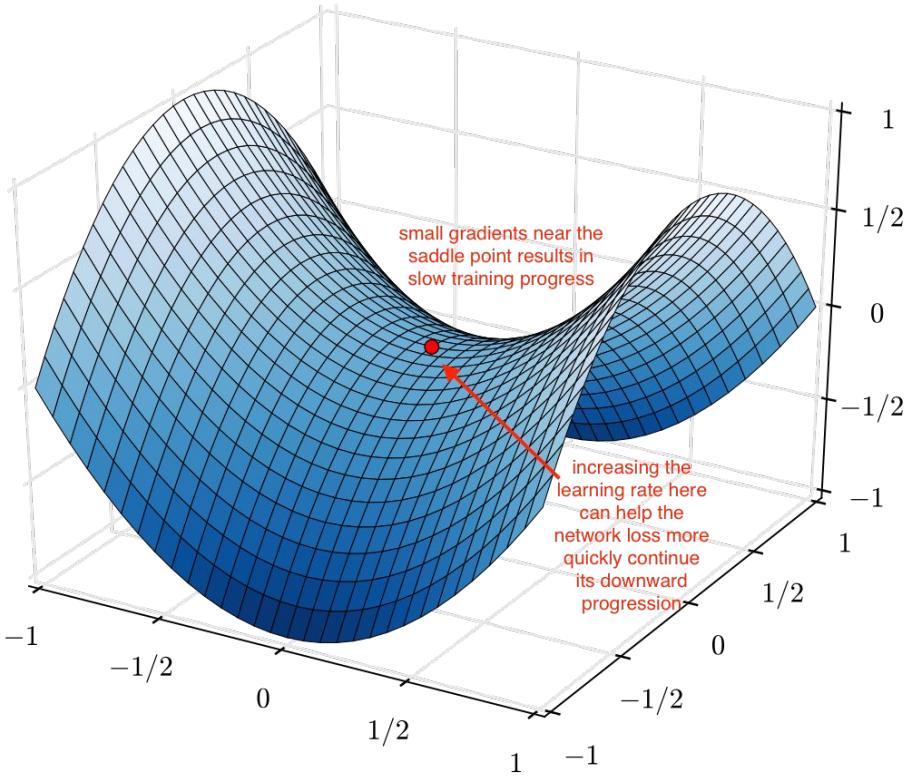
ReduceLROnPlateau

Reduce learning rate when a metric has stopped improving.

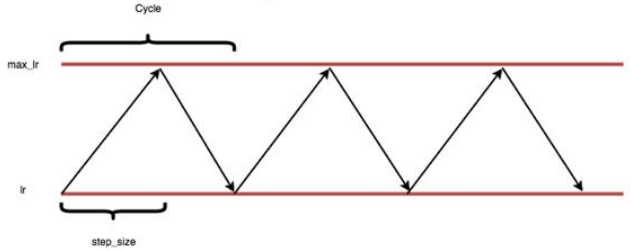
- It’s easy to tune scheduling parameters
- It works always



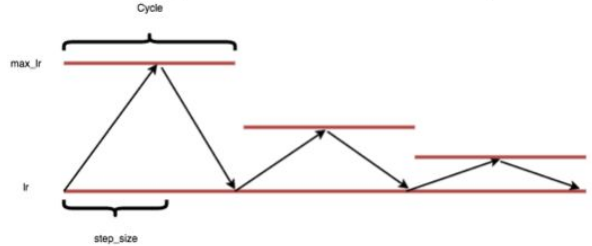
CyclicLR



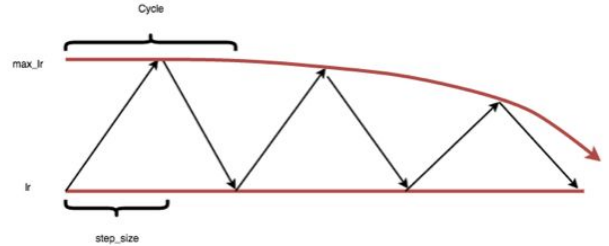
Triangular schedule



Triangular schedule with fixed decay

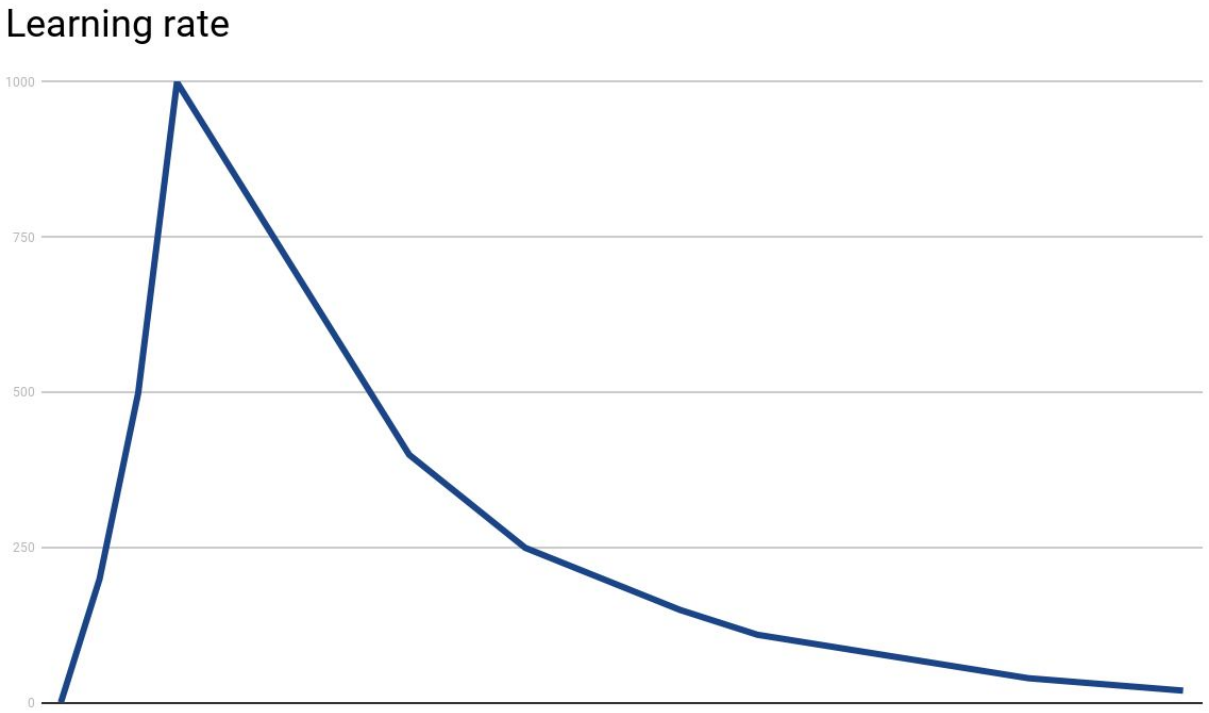


Triangular schedule with exponential decay



How to train a large network in 5 minutes?

Warmup



Optimizers

Adam

- Usually starting with lr=1e-4 works well => no parameters for tuning
- Better for training from scratch
- Faster convergence

SGD

- More parameters for tuning: lr, momentum, is nesterov
- Usually better for small tuning
- Sometimes gives better accuracy

Adam

- Usually starting with lr=1e-4 works well => no parameters for tuning
- Better for training from scratch
- Faster convergence

SGD

- More parameters for tuning: lr, momentum, is nesterov
- Usually better for small tuning
- Sometimes gives better accuracy

Conclusion: use Adam

The main secret


jupyter


jupyter

A good deep learning engineer
is
a good coder

An example of a well-organized pipeline

436 commits | **1** branch | **0** releases | **1** contributor

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

PavelOstyakov Fix make lgbm — Latest commit `d98f5f1` 6 days ago

bin	Fix train script	a month ago
doodle	Some fixes from doodle	a month ago
inclus	Fix & added new config	a month ago
protein	Fix make lgbm	6 days ago
solver	Fix input_channels	15 days ago
.gitignore	Working on Pipeline	3 months ago
README.md	Initial commit	3 months ago

How to speed up convergence

How to speed up convergence

- Start from a small image size
- Gradually increase it during training
- Coordinate this procedure with lr scheduling

Data augmentation

- Works always and everywhere
- To speed up convergence increase probability of augmentations step-by-step during training
- Again, coordinate it with lr scheduling
- At the finish, turn off augmentations and tune your model a little bit

MixUp

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j,$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j,$$
$$\lambda \sim \text{Beta}(\alpha, \alpha)$$

We already have a model, how to improve an accuracy of predictions?

1. Use test time augmentation (TTA)

- Use augmentations which were used during training
- Simply average all predictions
- TTA-10 is very fast but it can significantly improve accuracy
- TTA-128 still makes sense

2. Apply k-nearest neighbors algorithm

1. Extract features from the network for validation set
2. Build index (use NMSLIB)
3. During inference extract features for the test data and find their neighbors
4. Use some heuristics to improve score

3. Adapt predictions for a test distribution

- Tuning the last layer to perform better on another target’s distribution
- Fix predictions having prior knowledge

Key takeaways from supervised tasks

- Image size 224x224 usually is enough
- Usually training two models and ensemble them is faster than training one strong single model
- However, to get a high result we need several **diversified strong** models

To be continued...