Deep learning

5.6. Architecture choice and training protocol

François Fleuret

https://fleuret.org/dlc/

UNIVERSITÉ DE GENÈVE

Choosing the network structure is a difficult exercise. **There is no silver bullet.**

- Re-use something "well known, that works", or at least start from there,
- split feature extraction / inference (although this is debatable),
- modulate the capacity until it overfits a small subset, but does not overfit / underfit the full set,
- capacity increases with more layers, more channels, larger receptive fields, or more units,
- regularization to reduce the capacity or induce sparsity,
- identify common paths for siamese-like,
- identify what path(s) or sub-parts need more/less capacity,
- use knowledge about the "scale of meaningful context" to size the filters,
- grid-search all the variations that come to mind (and hopefully have farms of GPUs to do so).

We will re-visit this list with additional regularization / normalization methods.

Notes

To assess that the capacity of the network is sufficient, it should overfit at least on a small subset of the training data. If it does not, the capacity should be increased, e.g. more layers, more channels.
Some paths of the network might need more capacity. For instance, if the samples come as pairs of images and text, one may require more or less capacity for the text part than for the image part, depending on the expected complexity of the signal.
The network should be designed by taking into account the "scale of meaningful context": for instance for object detection, if we have an idea of the size of a face or a car to detect, this should be reflected in the network. The last layer making the prediction should have access to a sufficient amount of information from the input signal (its receptive field) to make that prediction.
Once the model is designed, a grid-search over these hyper-parameters can be performed. Among others:

- the number of filters, of layers,
- the learning rate(s),
- the optimizer,
- changing the backbone (architecture on top of which we extend a new network),
- etc.

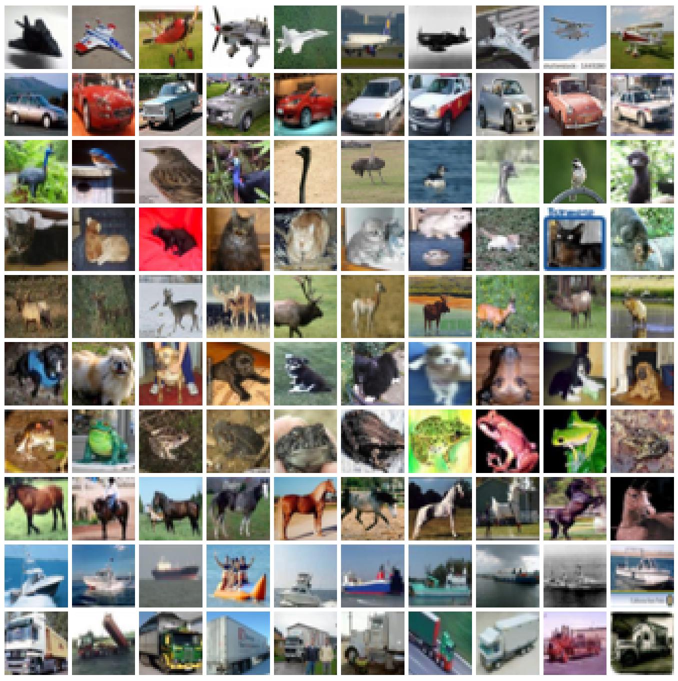
Regarding the learning rate, for training to succeed it has to

- reduce the loss quickly $\Rightarrow$ large learning rate,
- not be trapped in a bad minimum $\Rightarrow$ large learning rate,
- not bounce around in narrow valleys $\Rightarrow$ small learning rate, and
- not oscillate around a minimum $\Rightarrow$ small learning rate.

These constraints lead to a general policy of using **a larger step size first, and a smaller one in the end.**

The practical strategy is to look at the losses and error rates across epochs and pick a learning rate and learning rate adaptation. For instance by reducing it at discrete pre-defined steps, or with a geometric decay.

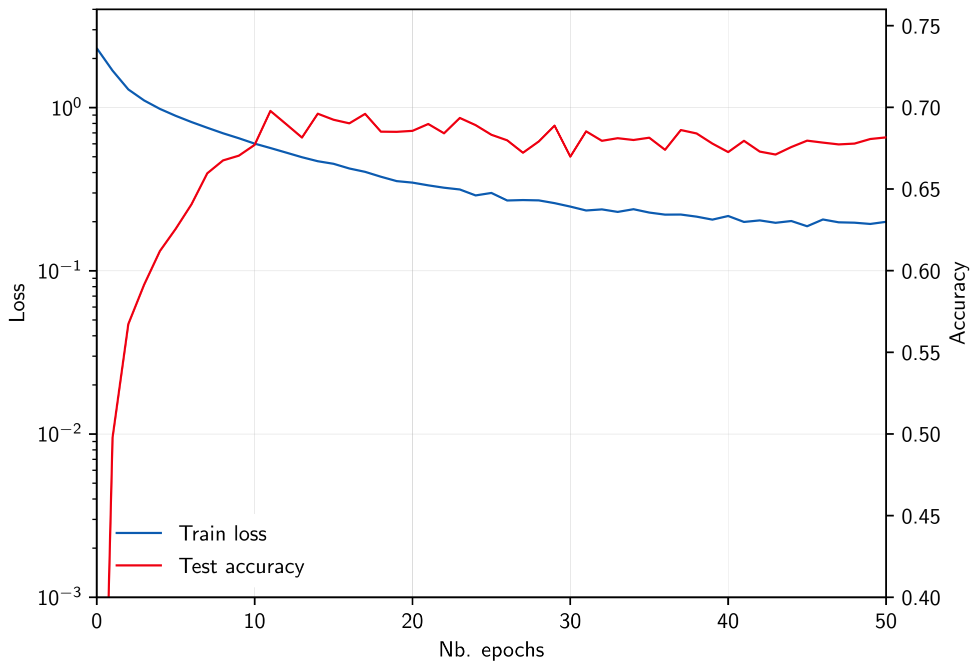
CIFAR10 data-set



32×32 color images, $50,000$ train samples, $10,000$ test samples.

(Krizhevsky, 2009, chap. 3)

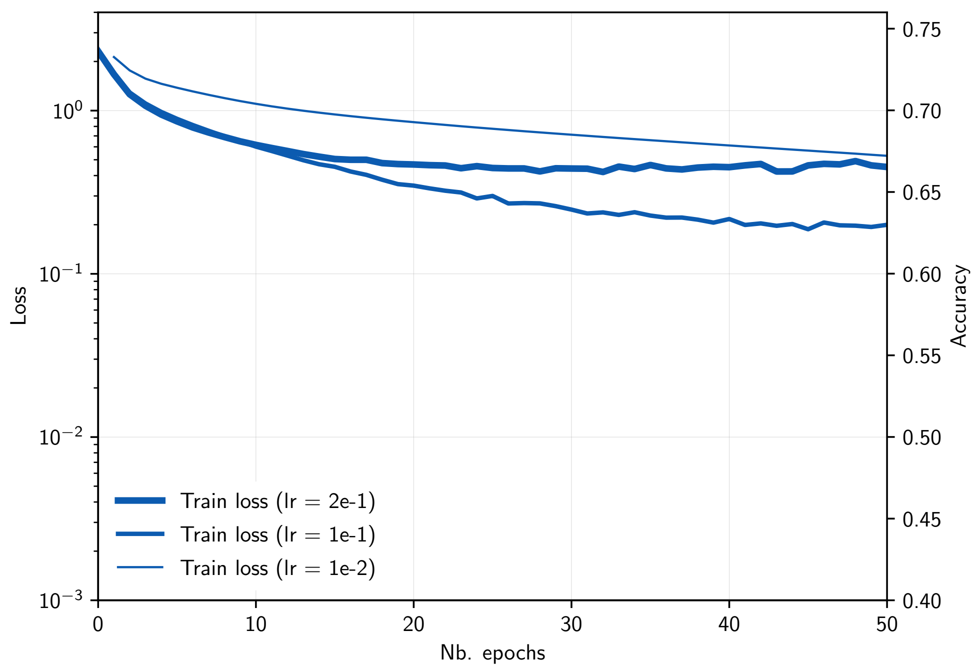
Small convnet on CIFAR10, cross-entropy, batch size 100, $\eta = 1e-1$.



Notes

We train a small convnet on the CIFAR10 dataset and show the training loss (left axis) and the test accuracy (right axis).
This baseline result is obtained with a step size of 0.1 with standard stochastic gradient descent.
Note that these results are used to illustrate the learning rate policies and are not state-of-the-art.

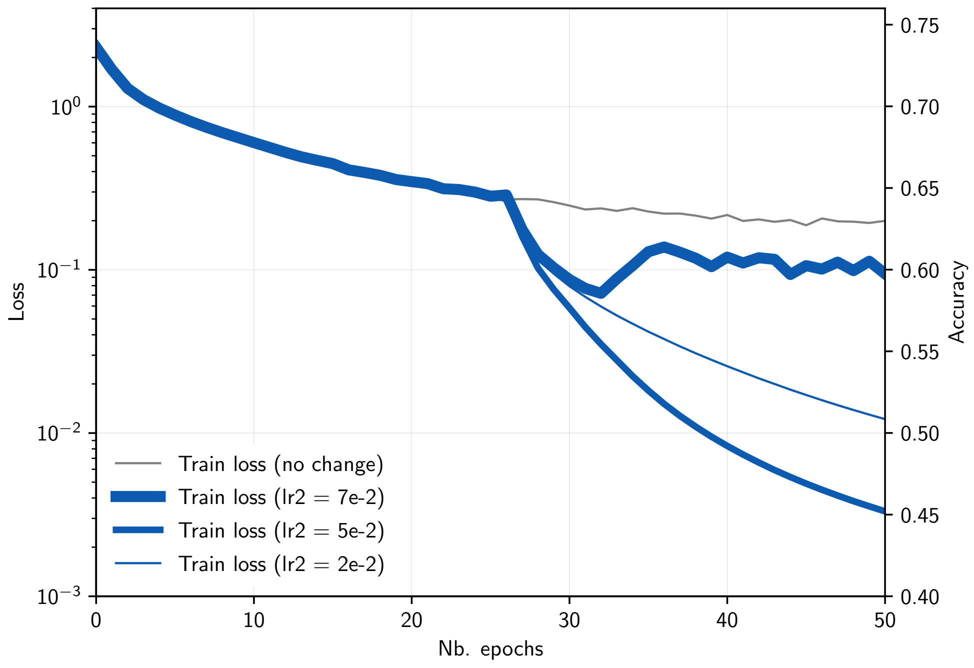
Small convnet on CIFAR10, cross-entropy, batch size 100



Notes

This plot shows the influence of the learning rate on the training loss. We try 3 rates: 0.01, 0.1, and 0.2.
The learning rate of 0.01 decreases the loss more slowly than the other two.
The learning rate of 0.2 does as well as 0.1 at start, but eventually does worse, probably because it is trapped in a local minimum, or cannot go into a "valley".

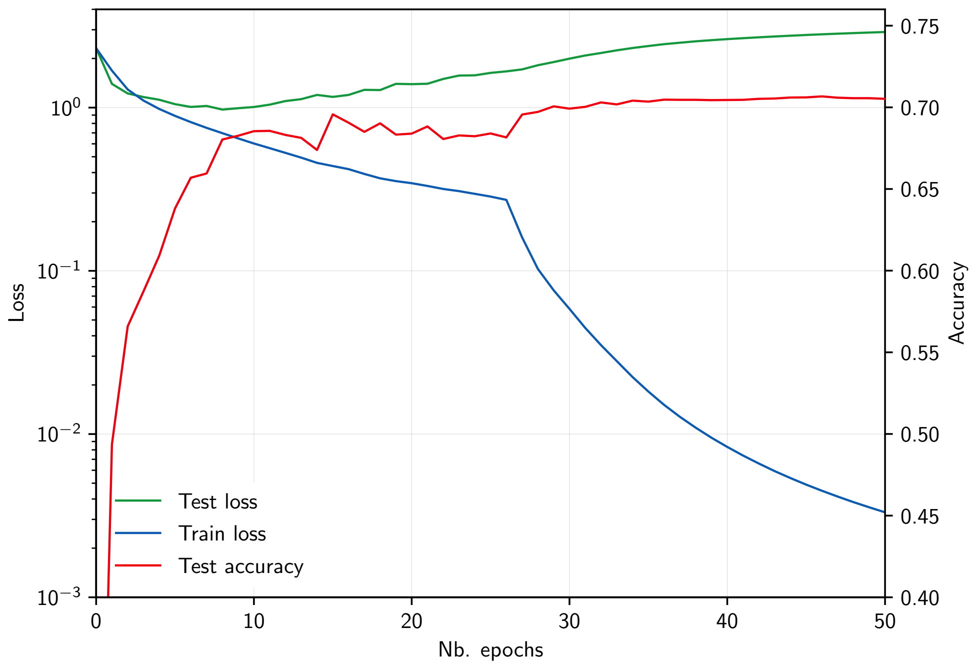
Using $\eta = 1e-1$ for 25 epochs, then reducing it.



Notes

Here we illustrate the effect of reducing the learning rate during training.
The gray curve correspond to our baseline with a constant learning rate of 0.1 all along.
After 25 epochs, we decrease the learning rate down to 0.07, 0.05, or 0.02. In all cases, this reduction has a dramatic effect on the loss which from there decreases faster.

While the test error still goes down, the test loss may increase, as it gets even worse on misclassified examples, and decreases less on the ones getting fixed.



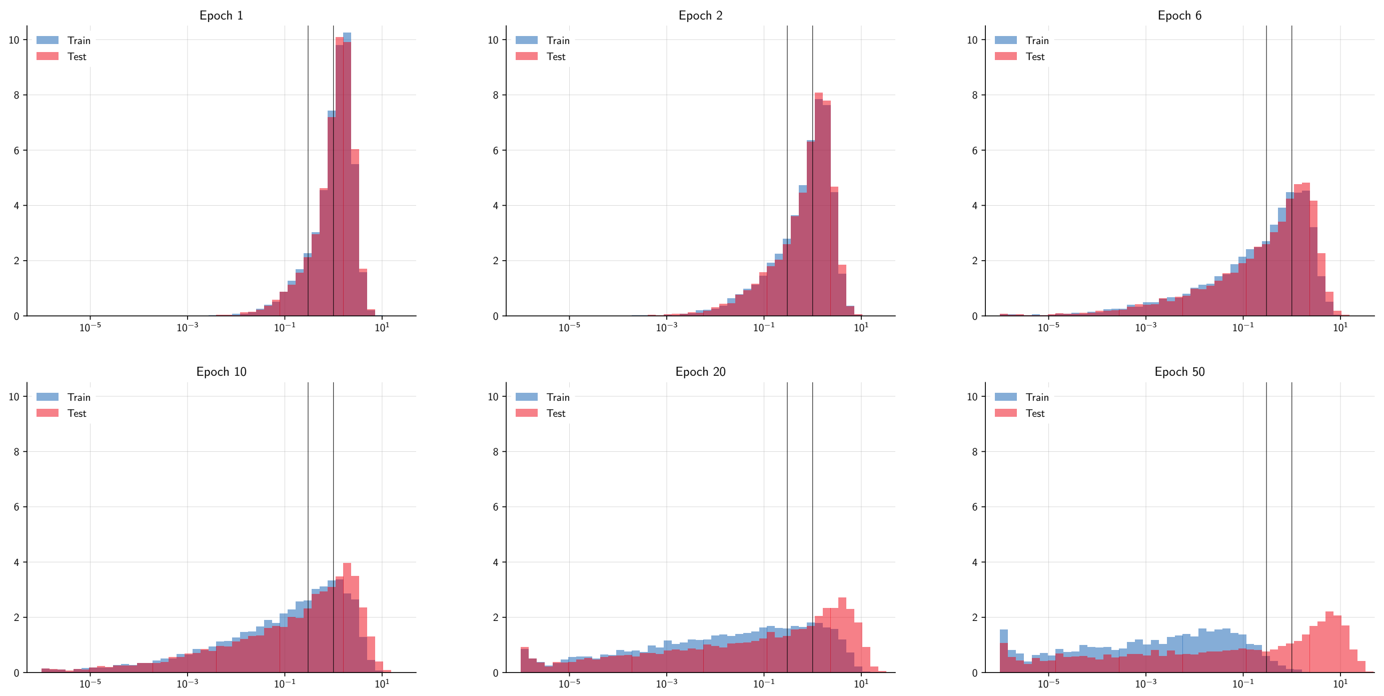
Notes

Here we look at the test accuracy in the best case we saw before: a learning rate of 0.1 during 25 epochs, then 0.05 for 25 more epochs.
After 50 epochs, the test accuracy is above 70%, which is better than in our baseline which was below.
Interestingly, the test loss increases, although the test accuracy still improves over the epochs, which is counter intuitive.

We can plot the train and test distributions of the per-sample loss

$$\ell = -\log\left(\frac{\exp(f_Y(X; w))}{\sum_k \exp(f_k(X; w))}\right)$$

through epochs to visualize the over-fitting.



Notes

ℓ is the loss on sample X given the current model w. We plot the distribution of this loss over the epochs. Y is the real class of X.
The red histogram is the histogram of the train loss per sample while the blue histogram is the histogram of the test loss per sample.
With

$$\hat{P}_y(x) = \frac{\exp(f_y(x; w))}{\sum_k \exp(f_k(x; w))}$$

the left black vertical line corresponds to the loss value for which $\hat{P}_Y(X; w) > 0.5$, hence all samples on the left of that line are necessarily well classified.
The right black vertical line is the loss value for which $\hat{P}_Y(X; w) < \frac{1}{C}$ (where C is the number of classes), hence all samples on the right of that vertical line are necessarily misclassified: there is at least one other class the score of which is greater than that of the true class.
In the middle of the two lines, we cannot say anything whether the samples are well classified or not.
As the training goes on, the distribution of the training set is pushed to the left, meaning that the training samples are well classified. We can slowly see over-fitting occurring.
After epoch 15, we clearly see that some of the test samples are really misclassified: the histogram has drifted on the right part. This explains why the test loss may increase: as the training goes on, the already misclassified samples get even more misclassified, causing the test loss to increase.

References

A. Krizhevsky. **Learning multiple layers of features from tiny images**. Master's thesis, Department of Computer Science, University of Toronto, 2009.