Practical Aspects of Deep Learning

Keith L. Downing

The Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
keithd@idi.ntnu.no

December 21, 2020
Consider simpler techniques first, and if you use DL, try the vanilla versions (e.g. basic Stochastic Gradient Descent with zero or one hidden layer) before adding all the bells and whistles.

For image classification, use convolution; for sequences, use LSTMs or Gated Recurrent Units (GRUs).

Adding more parameters (i.e. layers and weights) can normally reduce training error, but a) increased computational cost, and b) greater risk of overfitting and thus high generalization error.

If you add more parameters, you may have to add regularization.

The Adam Optimizer, Batch Normalization and Dropout are powerful (and popular) regularizers.

But if you have enough data, you may not need any regularizers.

It may be more practical (and effective) to gather more data instead of trying more and more complex DL algorithms.

For hard image-processing or natural-language tasks, pre-trained layers or embeddings can provide a good head start.
Hyperparameters

- Include number and types of layers, learning rate(s), momentum rate(s), initial weight range, dropout rate, regularization penalty, decay rate(s) in optimizers (RMSProp, Adam), etc.
- Select these manually or automatically.
- These have considerable interaction, so you need to think carefully about these relationships when tuning.
- Standard goal: reduce generalization error at lowest computational cost; bells and whistles can be expensive.
- Learning rate is often the most important hyperparameter. Decaying learning rates normally perform better than constant rates.
- If using automated methods, random search in hyperparameter space works better than structured (grid) search, since the former does not force an artificial discretization of the search space.
The number of cases that the network can properly handle.

Methods for Increasing Model Capacity

- Optimally tune the learning rate.
- $\uparrow$ hidden layers and neurons.
- $\uparrow$ convolution kernels.
- $\uparrow$ width of convolution filters/kernels.
- $\uparrow$ zero padding $\rightarrow$ less layer-to-layer size reduction.
- $\downarrow$ weight decay rate. Larger weight range produces more internal patterns.
- $\downarrow$ dropout rate.

Of course, anything that increases model capacity runs the risk of overfitting.
Debugging Tips

- Since NN’s are adaptive, the system may compensate for a bug and thereby hide it.
- Time spent producing informative visualizations is a good investment.
- Monitor changes in weights, internal activations and gradients.
- Start with small datasets.
- Start with small networks and just verify that they are learning (at least a little bit).
- Use comparison of training and testing error to indicate problems such as overfitting.
- 1% Rule - Weight and bias updates should be approximately 1% of the parameter’s magnitude.
- Convince yourself that the representation chosen for input features actually supports learning: the NN can achieve differences (of internal state) that make a difference (in classifications).