Train model

From cs231n

August 2024

Things you might try

To train a ConvNet model, you can experiment with architectures, hyperparameters, loss functions, and optimizers to train a model.

Things you might try:

- **Filter size**: Above we used 5x5; would smaller filters be more efficient?
- **Number of filters**: Above we used 32 filters. Do more or fewer do better?
- **Pooling vs Strided Convolution**: Do you use max pooling or just stride convolutions?
- **Batch normalization**: Try adding spatial batch normalization after convolution layers and vanilla batch normalization after affine layers. Do your networks train faster?
- **Network architecture**: The network above has two layers of trainable parameters. Can you do better with a deep network? Good architectures to try include:
  - conv-relu-pool $\times N \rightarrow$ affine $\times M \rightarrow$ softmax or SVM
  - conv-relu-conv-relu-pool $\times N \rightarrow$ affine $\times M \rightarrow$ softmax or SVM
  - batchnorm-relu-conv $\times N \rightarrow$ affine $\times M \rightarrow$ softmax or SVM
- **Global Average Pooling**: Instead of flattening and then having multiple affine layers, perform convolutions until your image gets small (7x7 or so) and then perform an average pooling operation to get to a 1x1 image picture (1, 1, Filter#), which is then reshaped into a (Filter#) vector. This is used in https://arxiv.org/abs/1512.00567Google's Inception Network (See Table 1 for their architecture).
- **Regularization**: Add l2 weight regularization, or perhaps use Dropout.

Tips for Training

For each network architecture that you try, you should tune the learning rate and other hyperparameters. When doing this there are a couple important things to keep in mind:

- If the parameters are working well, you should see improvement within a few hundred iterations.
- Remember the coarse-to-fine approach for hyperparameter tuning: start by testing a large range of hyperparameters for just a few training iterations to find the combinations of parameters that are working at all.
- Once you have found some sets of parameters that seem to work, search more finely around these parameters. You may need to train for more epochs.
- You should use the validation set for hyperparameter search, and save your test set for evaluating your architecture on the best parameters as selected by the validation set.

Going Above and Beyond

If you are feeling adventurous there are many other features you can implement to try and improve your performance. You are **not required** to implement any of these, but don't miss the fun if you have time!

- Alternative optimizers: you can try Adam, Adagrad, RMSprop, etc.
- Alternative activation functions such as leaky ReLU, parametric ReLU, ELU, or MaxOut.
- Model ensembles.
- Data augmentation.
- New Architectures:
  - https://arxiv.org/abs/1512.03385ResNets where the input from the previous layer is added to the output.
  - https://arxiv.org/abs/1608.06993DenseNets where inputs into previous layers are concatenated together.
  - https://chatbotslife.com/resnets-highwaynets-and-densenets-oh-my-9bb15918ee32This blog has an in-depth overview.

Have fun and happy training!