7 Guidelines

Neural networks work best on large, accurately labeled datasets having lots of features.

Preprocessing: (Can be very time consuming.)

balance: Is the data set balanced? Are there roughly an equal number of examples per class represented in the data set?

size: Is the data set large enough? Are there more than 1,000 examples for each class? You may be able to increase the size of the training data set by carefully generating additional (artificial) data.

normalization: Is the data properly normalized/standardized?

accuracy: Check/estimate the accuracy of the class labels. The training data class label accuracy rate will limit the prediction accuracy of the trained neural network on new data.

split data: To help prevent over-fitting, split the data set into training and validation data sets. The training set is used to determine the parameters of the network. The validation set is used to determine the hyper-parameters of the network. You may in fact split the data set three ways: a training set, validation set and test set. A separate test set will minimize the danger of over fitting hyper-parameters on the validation set.

encode data: One-hot-encode categorical data. (You may need to use sparse coding methods.)

reshaping: Reshape data to preserve existing structure in the data and/or to make it easier to generate minibatches for stochastic gradient descent.

Baseline Accuracy Rate: A baseline accuracy rate that can be used for comparison purposes should be established.

variance: For regression problems, a simple bias network has MSE equal to the variance of the outputs.

guessing: For classification problems, what is the expected accuracy/error rate for random guessing of class labels?

linear/logistic regression: Linear/logistic regression is sufficiently accurate for many applications. Thus, the accuracy of these methods should be investigated first before spending the time to design, train and test a neural network.

Deliberate Over Fitting: The goal of deliberate over fitting is to check for errors in the training code used to train a neural network. A second objective is to select initial values for hyper-parameters, e.g. learning rate, mini-batch size and weight initialization standard deviations. Try to drive training set error to 0.

seed: Set a seed value for random processes to make outputs reproducible.

reduce data size: If necessary, reduce the size of the data set so that the neural network can be trained quickly and/or the data is easily over fit.

increase size of network: Larger (more nodes/more hidden layers) networks are easier to over fit. Determine the largest neural network that your computing device can train in a “reasonable” amount of time to obtain “frequent” feedback on training performance as you adjust hyper-parameters.

batch size: Determine a good mini-batch size by selecting the mini-batch size that over fits the quickest.

nonlinearity: Is your network nonlinear? Are you using a nonlinear function between linear layers?

batch normalization: Batch normalization can accelerate training, especially for deep networks, making it easier to drive training error to zero.

Regularization: The goal of regularization is to reduce/prevent over fitting the training data and thus improve the accuracy of the neural network on new data (data that was not used to train the network). The validation/test set accuracy rate is an estimate of how well the neural network will perform on new data.

early stopping: Typically, as a neural network is trained, the training error will decrease, but the validation/testing error will initially decrease and then increase. Stopping a training algorithm before a neural network is fully trained is a cheap form of regularization. Simply terminate training when the validation/testing error appears to have leveled off or starts to increase.

dropout: Adding dropout layers is a cheap way to regularize a large neural network.

regularization is failing: You may need a larger and/or more accurate training data set or you may need a larger/more sophisticated neural network architecture