Deep Learning Performance Cheat Sheet

Simple and complex tricks that can help you boost your deep learning models accuracy

Christopher Dossman
Oct 15, 2018 · 7 min read

The question that I get the most from new and experienced machine learning engineers is “how can I get higher accuracy?”

Makes a lot of sense since the most valuable part of machine learning for business is often its predictive capabilities. Improving the accuracy of prediction is an easy way to squeeze more value from existing systems.

The guide will be broken up into four different sections with some strategies in each.

- Data Optimization
- Algorithm tuning
- Hyper-Parameter Optimization
- Ensembles, Ensembles, Ensembles

Not all of these ideas will boost performance, and you will see limited returns the more of them you apply to the same problem. Still stuck after trying a few of these? That

indicates you should rethink the core solution to your business problem. This article is just a cheat sheet, so I'm linking you to more detailed sources of information in each section.

Data Optimization

Balance your data set

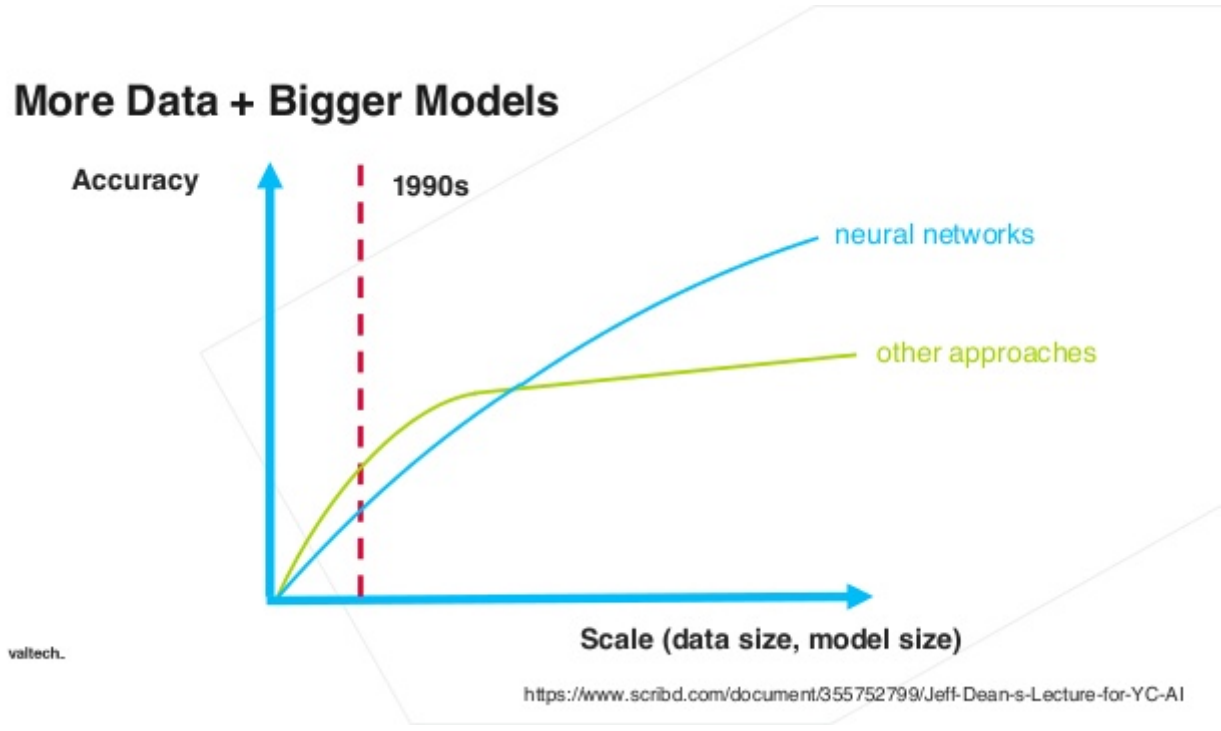
One of the easiest ways to increase performance for underperforming deep learning models is to balance your dataset if your problem is classification. Often real-world data sets are skewed, and if you want the best accuracy you want your deep learning system to learn how to pick between two classes based on the characteristics not by copying its distribution

Common methods include:

- **Subsample Majority Class:** You can balance the class distributions by subsampling the majority class.
- **Oversample Minority Class:** Sampling with replacement can be used to increase your minority class proportion.

Here is a nice write up for detailing this issue in more detail

More Data



Many of us are familiar with this graph. It shows the relationship between the amount of data and performance for both deep learning and classical machine learning approaches. If you are not, then the lesson is clear and straightforward. If you want better performance for your model, you need more data. Depending on your budget you might opt for creating more labeled data or collecting more unlabeled data and training your feature extraction sub-model more.

Open Source Labeling Software

- Images
- Audio
- Video

Generate More Data

Or Fake it, till you make it. An often ignored method of improving accuracy is creating new data from what you already have. Take for example photos; often engineers will create more images by rotating and randomly shifting existing images. Such transformations also increase the reduce overfitting of the training set.

An excellent resource on creating more data for image problems

Algorithm Tuning

Copy The Researchers

Are you working on a problem that has lots of research behind it? You are in luck because 100's of engineers might have already put a bunch of thought in how to get the best accuracy for this problem. Read some research papers on the topic and take note of the different methods they used to get results! They might even have a git-hub of their code for you to sink your teeth in to.

Google Scholar is an excellent place to start your search. They offer many tools to help you find related research as well.

For storage and organization of research papers, I use Mendeley

Algorithm spot check

Don’t let your ego get the best of you. Its impossible to know which machine learning algorithm will work best for your problem. Whenever I attack a new problem, with not much in the way of research behind it, I look at a few methods available and try all of them. Deep learning (CNN’s, RNN’s, etc.) and classical machine learning approaches (Random Forests, Gradient Boosting, etc.)

Rank the results of all your experiments and double down on the algorithms that perform the best.

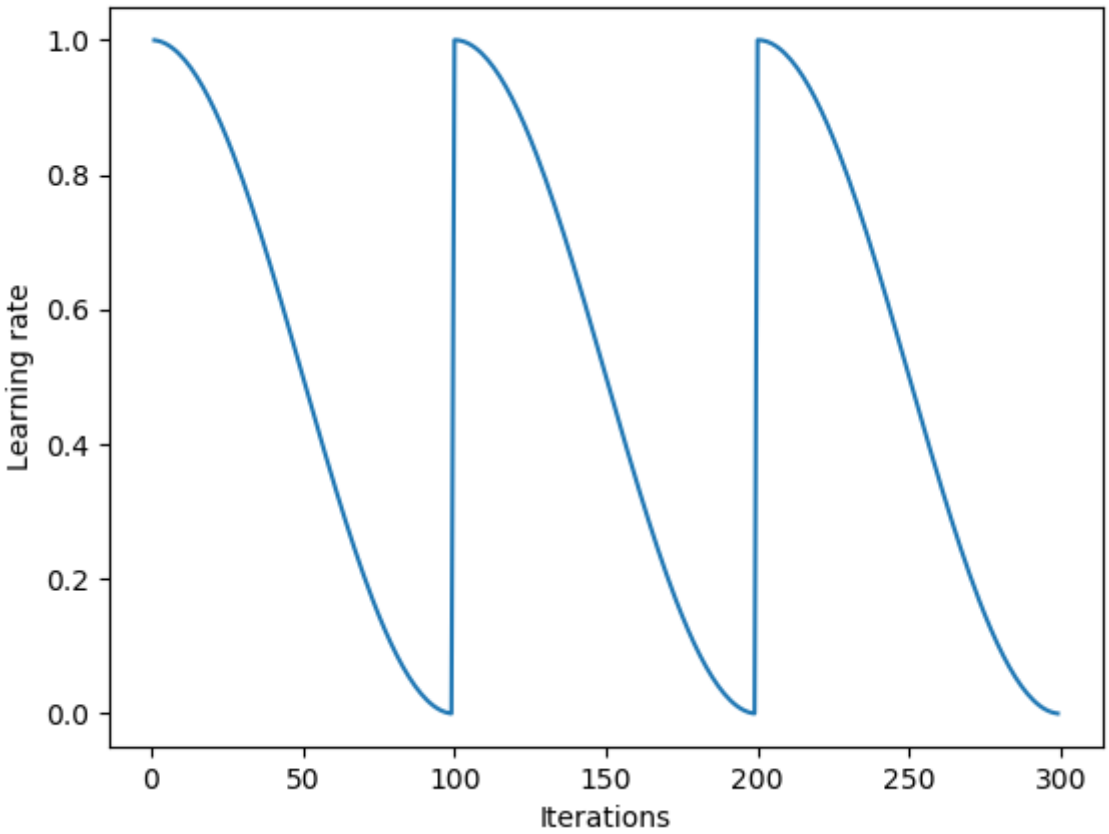
Check out a data-driven approach to choosing machine learning algorithms

Hyper-Parameter Optimization

Learning rates

The Adam optimization algorithm is tried and true. Often giving amazing results on all deep learning problems. Even with Its fantastic performance it still can get you stuck in a local minimum for your problem. An even better algorithm which has the benefits of Adam and helps eliminate the chance of getting stuff in a local minimum is Stochastic Gradient Descent with Warm Restarts.

Great write up on learning rates here



Batch size and number of epochs

A standard procedure is using large batch sizes with a large number of epochs for modern deep learning implementations, but common strategies yield common results. Experiment with the size of your batches and the number of training epochs.

Early Stopping

This is an excellent method for reducing the generalization error of your deep learning system. Continual training might improve accuracy on your data set, but at a certain point, it starts to reduce the model's accuracy on data not yet seen by the model. To improve real-world performance try early stopping.

Example of early stopping

Network Architecture

If you want to try something a little more interesting, you can give Efficient Neural Architecture Search (ENAS) a try. This algorithm will create a custom network design that will maximize accuracy on your dataset and is way more efficient than standard Neural architecture search that cloud ML uses.

Regularization

A robust method to stop overfitting is to use regularization. There are a couple of different ways to use regularization that you can train on your deep learning project. If you haven't tried these methods yet I would start to include them into every project you do.

- **Dropout:** They randomly turn off a percentage of neurons during training. Dropout helps prevent groups of neurons from all overfitting to themselves.
- **Weight penalty L1 and L2:** Weights that explode in size can be a real problem in deep learning and reduce accuracy. One of the ways to combat this is to add decay to all weights. These try to keep all of the weights int he networks as small as possible unless there are large gradients to counteract it. On top of often increasing performance, it has the benefit of making the model easier to interpret.

Ensembles, Ensembles, Ensembles

Having trouble picking the best model to use? Often you can combine the outputs from the different models and get better accuracy. There are two steps for every one of these algorithms.

- Producing a distribution of simple ML models on subsets of the original data
- Combining the distribution into one “Aggregated” model

Combined Models/Views (Bagging)

In this method, you train a few different models, which are different in some way, on the same data and you average out the outputs to create the final output. Bagging has the effect of reducing variance in the model. You can intuitively think of it as having multiple people with different backgrounds thinking about the same problem but with different starting positions. Just as on a team this can be a potent tool for getting the right answer.

Stacking

Its similar to bagging the difference here is that you don’t have an empirical formula for your combined output. You create a meta-level learner that based on the input data chooses how to weight the answers from your different models to produce the final output.

Still having issues?

Reframe Your Problem

Take a break from looking at your screen and get a coffee. This solution is all about rethinking your problem from the beginning. I find it helps to sit down and start brainstorming of different ways that you could solve the problem. Maybe start by asking your self some simple questions:

- Can my classification problem become a regression problem or the reverse?
- Can you break down your problem any smaller?
- Are there any observations that you have collected about your data that could change the scope of the problem?
- Can your binary output become a softmax output or vice versa?
- Are you looking at this problem in the most efficient way?

Rethinking your problem can be the hardest of the methods to increase performance but it often the one that yields the best results. It helps to chat with someone that has experience in deep learning and can give you a fresh take on your problem.

If you would like to chat with someone, *I am making my self-available for the next month to have a 30-minute conversation with you about your project. I'm charging 5 dollars for this 30-minute call as a barrier to keep those out who are not serious from wasting our time.*

Sign up for a time slot

Thanks for reading :) If you enjoyed it, hit that clap button below as many times as possible! It would mean a lot to me and encourage me to write more stories like this

Let's also connect on Twitter or LinkedIn