Nuts and Bolts of Applying Deep Learning

Sep 26, 2016

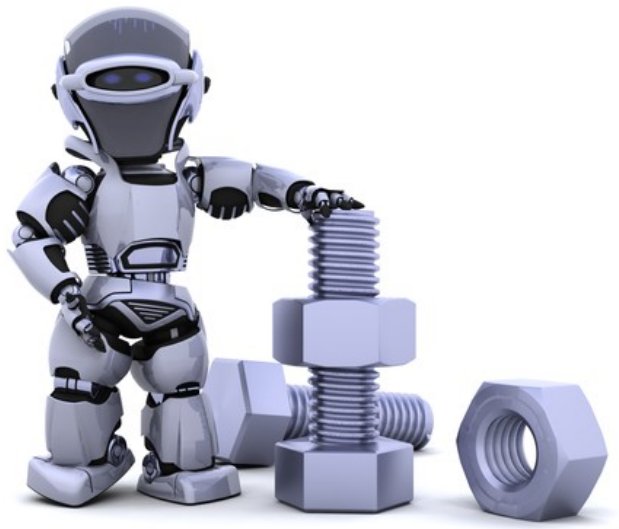


Image Courtesy

This weekend was very hectic (catching up on courses and studying for a statistics quiz), but I managed to squeeze in some time to watch the Bay Area Deep Learning School livestream on YouTube. For those of you wondering what that is, BADLS is a 2-day conference hosted at Stanford University, and consisting of back-to-back presentations on a variety of topics ranging from NLP, Computer Vision, Unsupervised Learning and Reinforcement Learning. Additionally, top DL software libraries were presented such as Torch, Theano and Tensorflow.

There were some super interesting talks from leading experts in the field: Hugo Larochelle from Twitter, Andrej Karpathy from OpenAI, Yoshua Bengio from the Université de Montreal, and Andrew Ng from Baidu to name a few. Of the plethora of presentations, there was one somewhat non-technical one given by Andrew that really piqued my interest.

In this blog post, I'm gonna try and give an overview of the main ideas outlined in his talk. The goal is to pause a bit and examine the ongoing trends in Deep Learning thus far, as well as gain some insight into applying DL in practice.

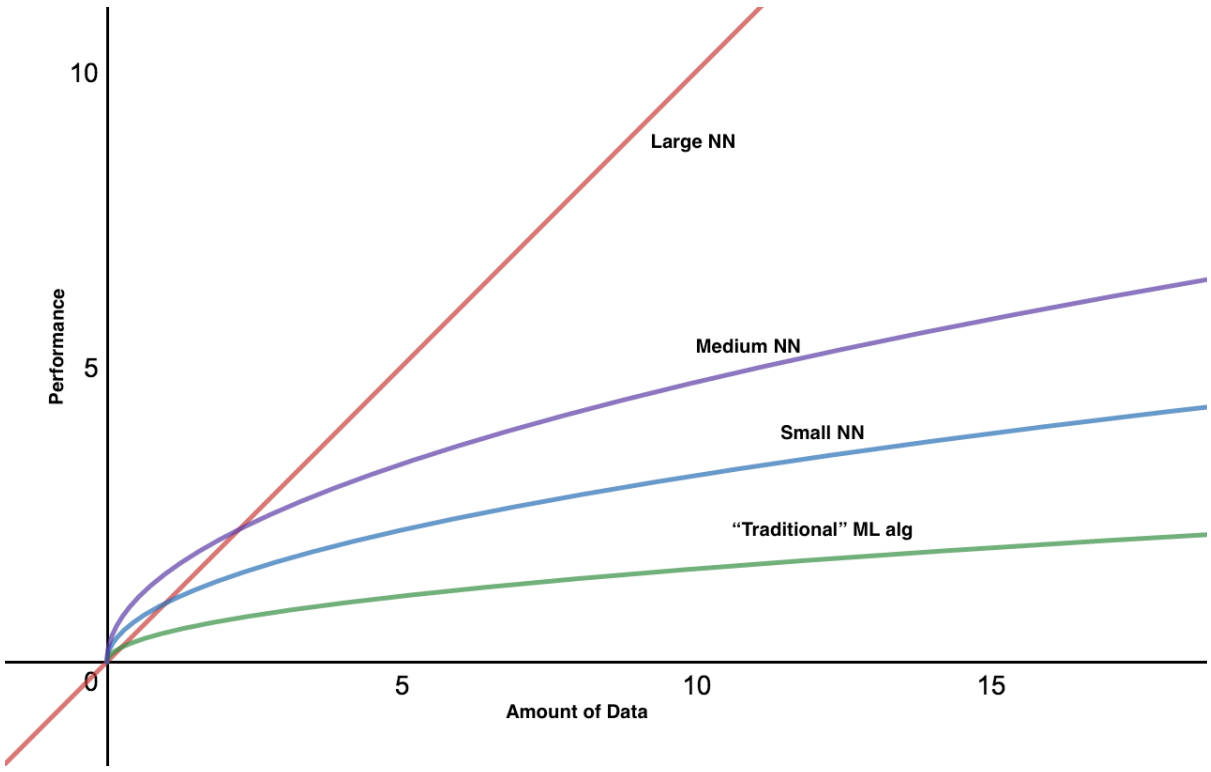
By the way, if you missed out on the livestreams, you can still view them at the following: Day 1 and Day 2.

Table of Contents:

- Major Deep Learning Trends
- End-to-End Deep Learning
- Bias-Variance Tradeoff
- Human-level Performance
- Personal Advice

Major Deep Learning Trends

Why do DL algorithms work so well? According to Ng, with the rise of the Internet, Mobile and IOT era, the amount of data accessible to us has greatly increased. This correlates directly to a boost in the performance of neural network models, especially the larger ones which have the capacity to absorb all this data.

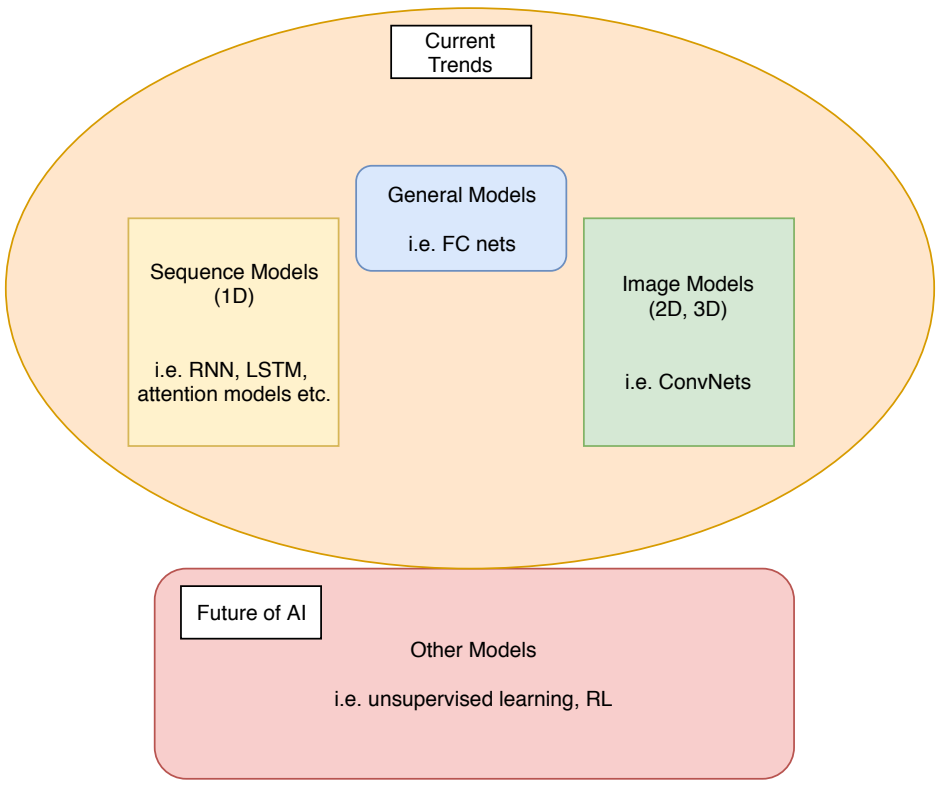


However, in the small data regime (left-hand side of the x-axis), the relative ordering of the algorithms is not that well defined and really depends on who is more motivated to engineer their features better, or refine and tune the hyperparameters of their model.

Thus this trend is more prevalent in the big data realm where hand engineering effectively gets replaced by end-to-end approaches and bigger neural nets combined with a lot of data tend to outperform all other models.

Machine Learning and HPC team. The rise of big data and the need for larger models has started to put pressure on companies to hire a Computer Systems team. This is because some of the HPC (high-performance computing) applications require highly specialized knowledge and it is difficult to find researchers and engineers with sufficient knowledge in both fields. Thus, cooperation from both teams is the key to boosting performance in AI companies.

Categorizing DL models. Work in DL can be categorized in the following 4 buckets:

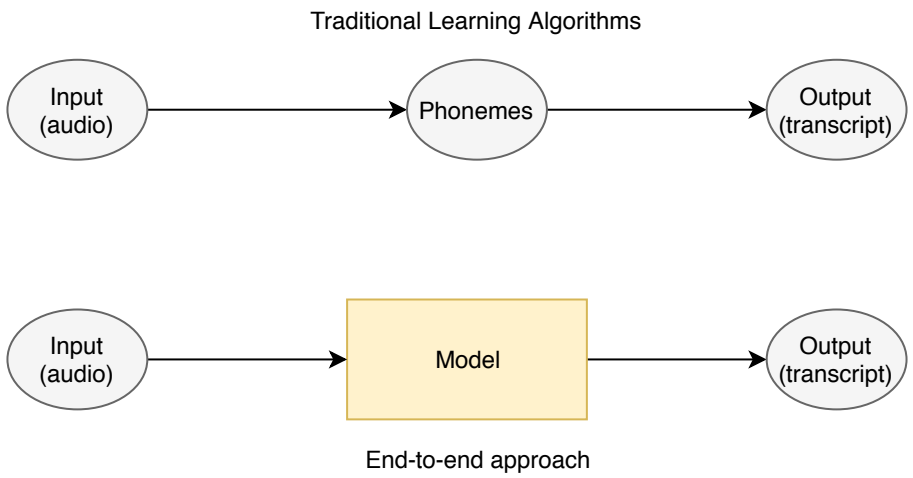


Most of the value in the industry today is driven by the models in the orange blob (innovation and monetization mostly) but Andrew believes that **unsupervised deep learning** is a super-exciting field that has loads of potential for the future.

The rise of End-to-End DL

A major improvement in the end-to-end approach has been the fact that outputs are becoming more and more complicated. For example, rather than just outputting a simple class score such as 0 or 1, algorithms are starting to generate richer outputs: images like in the case of GAN's, full captions with RNN's and most recently, audio like in DeepMind's WaveNet.

So what exactly does end-to-end training mean? Essentially, it means that AI practitioners are shying away from intermediate representations and going directly from one end (raw input) to the other end (output) Here's an example from speech recognition.



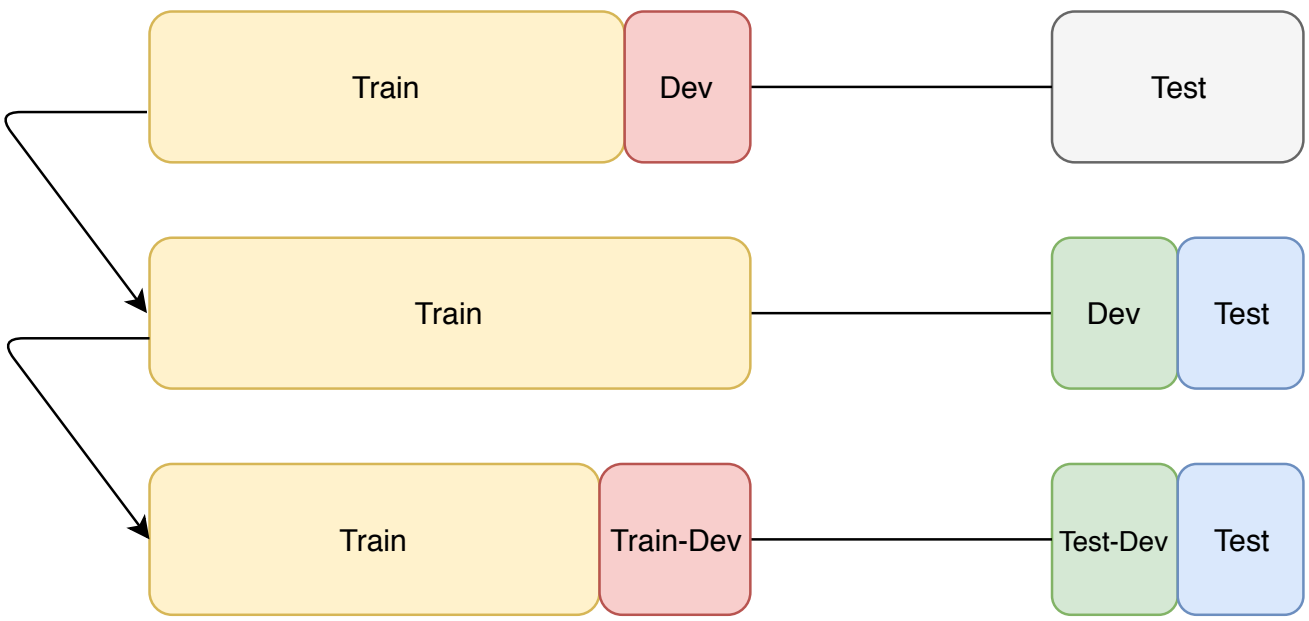
Are there any disadvantages to this approach? End-to-end approaches are data hungry meaning they only perform well when provided with a huge dataset of labelled examples. In practice, not all applications have the luxury of large labelled datasets so other approaches which allow hand-engineered information and field expertise to be added into the model have gained the upper hand. As an example, in a self-driving car setting, going directly from the raw image to the steering direction is pretty difficult. Rather, many features such as trajectory and pedestrian location are calculated first as intermediate steps.

The main take-away from this section is that we should always be cautious of end-to-end approaches in applications where huge data is hard to come by.

Bias-Variance Tradeoff

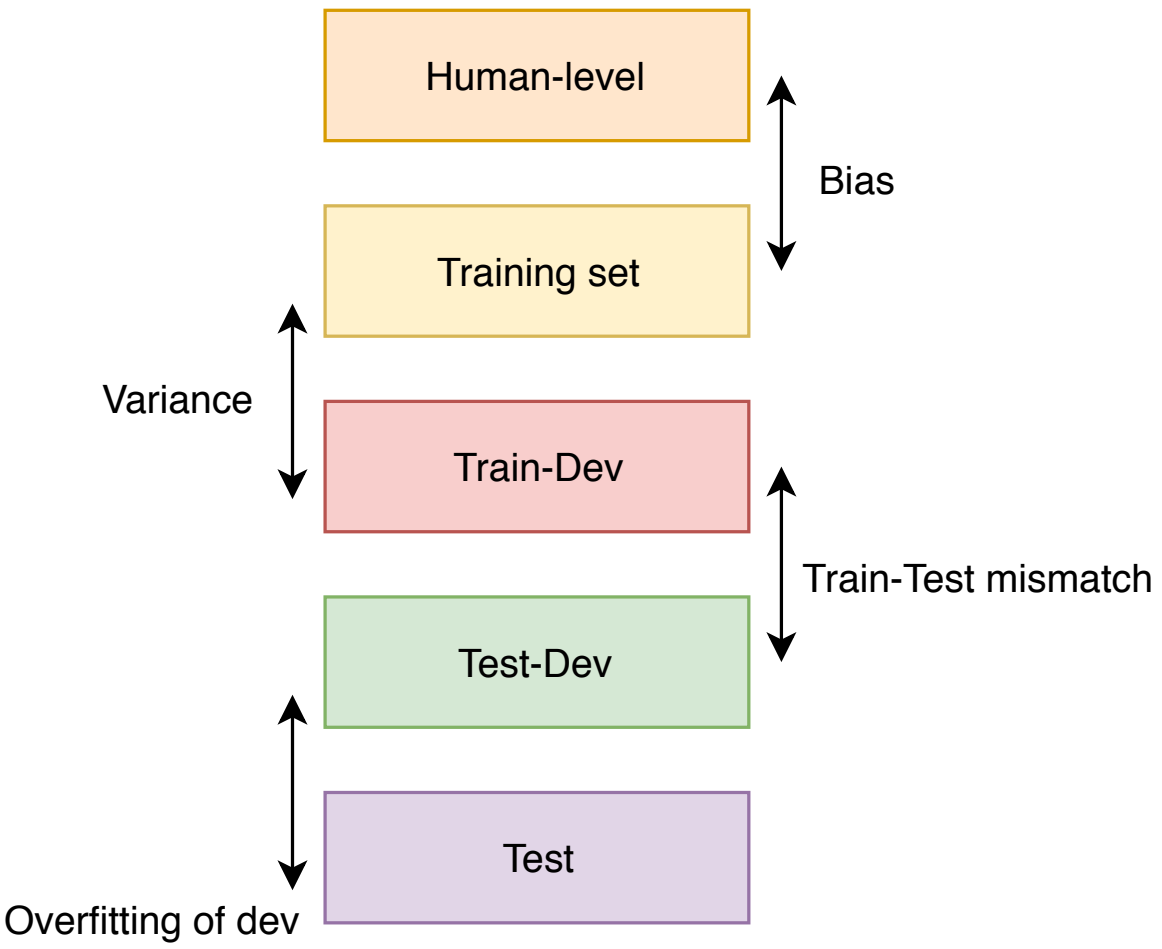
Splitting your data. In most deep learning problems, train and test come from different distributions. For example, suppose you are working on implementing an AI powered rearview mirror and have gathered 2 chunks of data: the first, larger chunk comes from many places (could be partly bought, and partly crowdsourced) and the second, much smaller chunk is actual car data.

In this case, splitting the data into train/dev/test can be tricky. One might be tempted to carve the dev set out of the training chunk like in the first example of the diagram below. (Note that the chunk on the left corresponds to data mined from the first distribution and the one on the right to the one from the second distribution.)

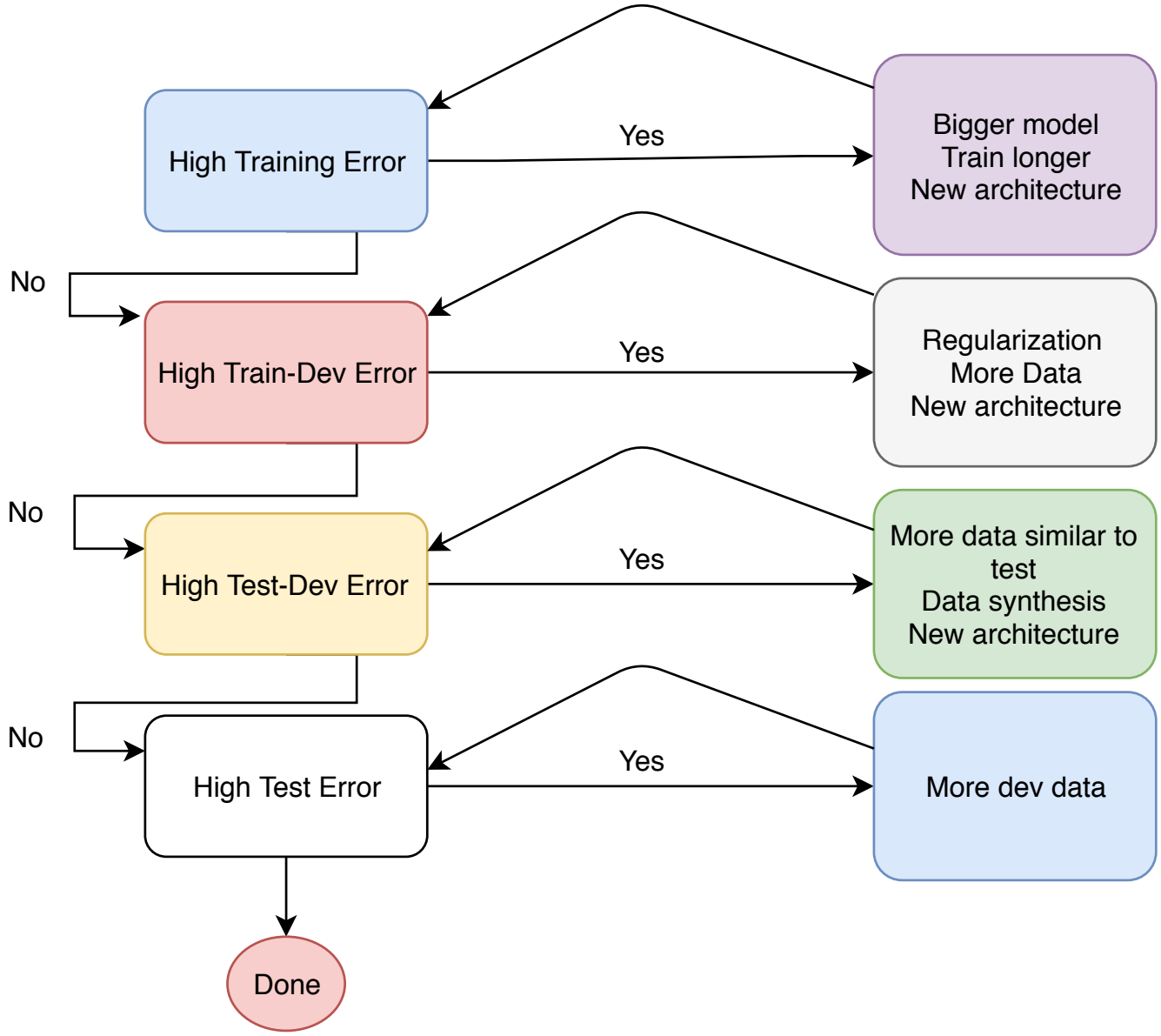


This is bad because we usually want our dev and test to come from the same distribution. The reason for this is that because a part of the team will be spending a lot of time tuning the model to work well on the dev set, if the test set were to turn out very different from the dev set, then pretty much all the work would have been wasted effort.

Hence, a smarter way of splitting the above dataset would be just like the second line of the diagram. Now in practice, Andrew recommends creating dev sets from both data distributions: a train-dev and test-dev set. In this manner, any gap between the different errors can help you tackle the problem more clearly.



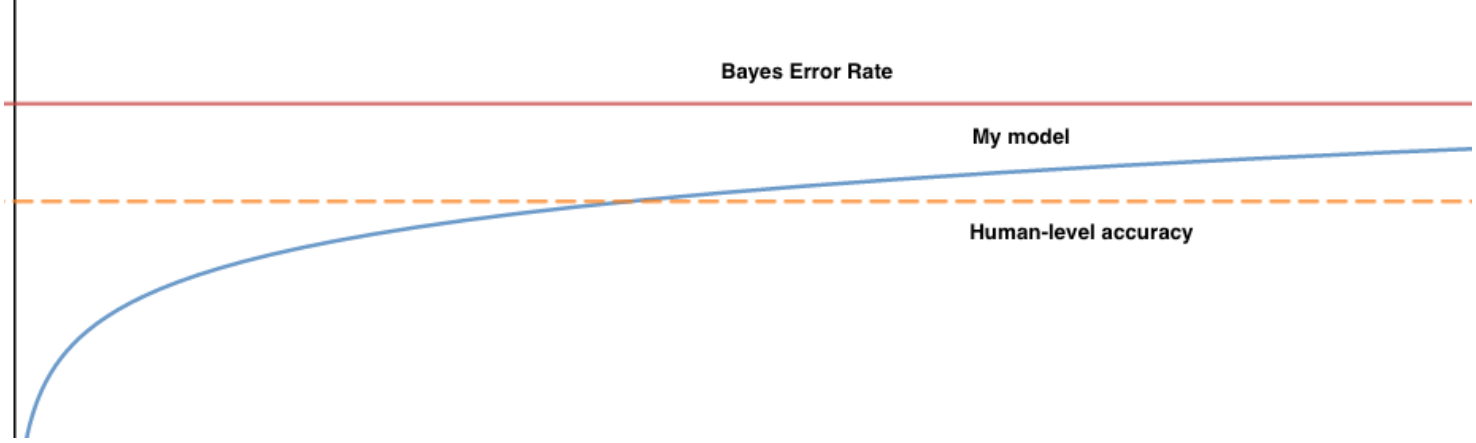
Flowchart for working with a model. Given what we have described above, here's a simplified flowchart of the actions you should take when confronted with training/tuning a DL model.



The importance of data synthesis. Andrew also stressed the importance of data synthesis as part of any workflow in deep learning. While it may be painful to manually engineer training examples, the relative gain in performance you obtain once the parameters and the model fit well are huge and worth your while.

Human-level Performance

One of the very important concepts underlined in this lecture was that of human-level performance. In the basic setting, DL models tend to plateau once they have reached or surpassed human-level accuracy. While it is important to note that human-level performance doesn't necessarily coincide with the golden bayes error rate, it can serve as a very reliable proxy which can be leveraged to determine your next move when training your model.



Reasons for the plateau. There could be a theoretical limit on the dataset which makes further improvement futile (i.e. a noisy subset of the data). Humans are also very good at these tasks so trying to make progress beyond that suffers from diminishing returns.

Here's an example that can help illustrate the usefulness of human-level accuracy. Suppose you are working on an image recognition task and measure the following:

- **Train error**: 8%
- **Dev Error**: 10%

If I were to tell you that human accuracy for such a task is on the order of 1%, then this would be a blatant bias problem and you could subsequently try increasing the size of your model, train longer etc. However, if I told you that human-level accuracy was on the order of 7.5%, then this would be more of a variance problem and you'd focus your efforts on methods such as data synthesis or gathering data more similar to the test.

By the way, there's always room for improvement. Even if you are close to human-level accuracy overall, there could be subsets of the data where you perform poorly and working on those can boost production performance greatly.

Finally, one might ask what is a good way of defining human-level accuracy. For example, in the following image diagnosis setting, ignoring the cost of obtaining data, how should one pick the criteria for human-level accuracy?

- **typical human**: 5%
- **general doctor**: 1%
- **specialized doctor**: 0.8%
- **group of specialized doctor**s: 0.5%

The answer is always the best accuracy possible. This is because, as we mentioned earlier, human-level performance is a proxy for the bayes optimal error rate, so providing a more accurate upper bound to your performance can help you strategize your next move.

Personal Advice

Andrew ended the presentation with 2 ways one can improve his/her skills in the field of deep learning.

- **Practice, Practice, Practice**: compete in Kaggle competitions and read associated blog posts and forum discussions.
- **Do the Dirty Work**: read a lot of papers and try to replicate the results. Soon enough, you'll get your own ideas and build your own models.