

Now we have learned how to implement a neural network. In this week we will learn the practical aspects of how to make our neural networks work well (ranging from hyperparameter tuning to how to set up the data to how to make sure that the optimization algorithm works quickly…).



Making good choices in how to set up train, dev and test set can make a huge difference for finding a good high-performance neural network.

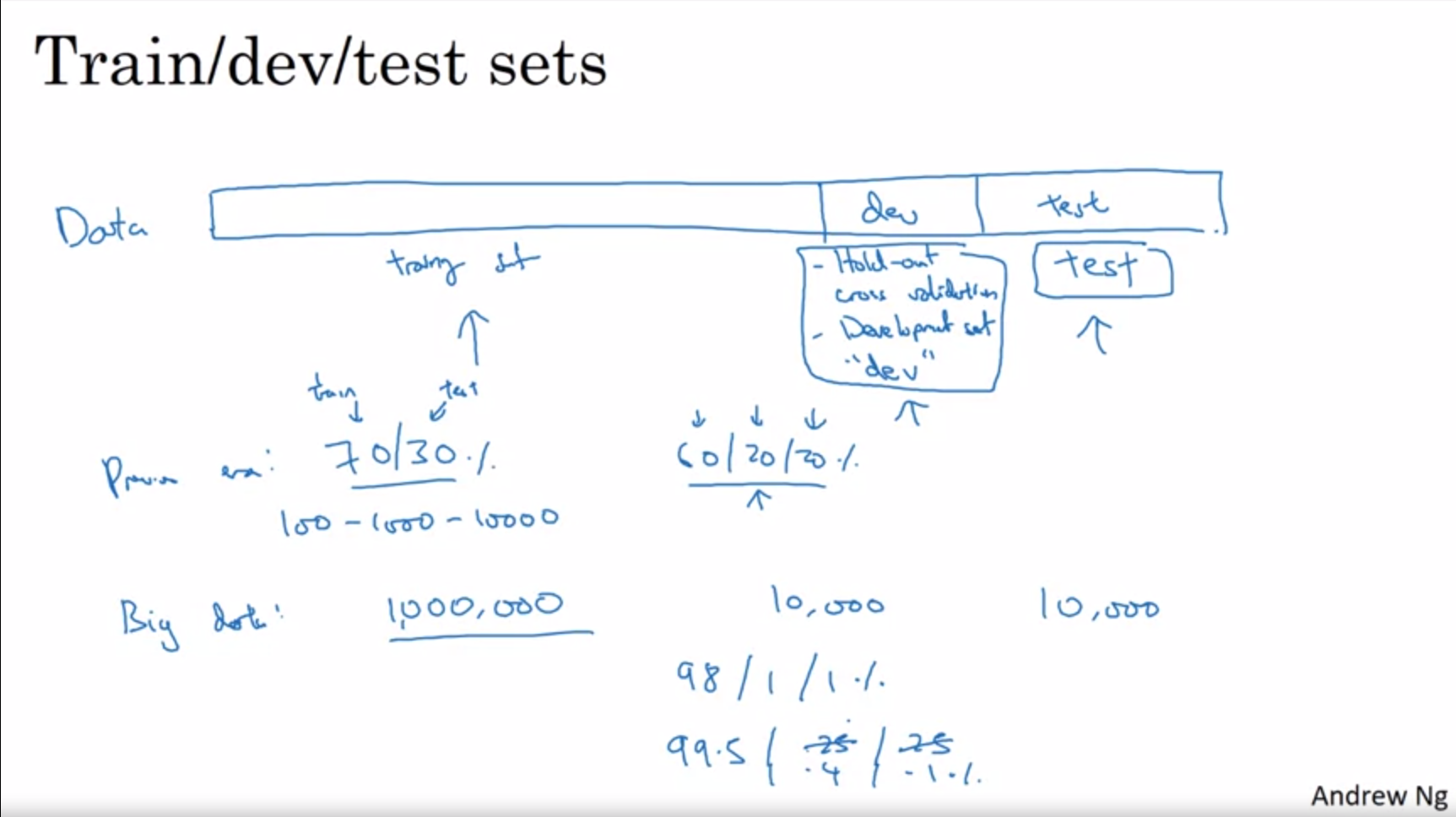
There are many decisions to be made.

When starting out with a new application it is impossible to guess the hyperparameters at first attempt. This is a very iterative process. You run, experiment, and see the outcome, then refine your idea.

Intuitions from one domain / application area do often not transfer to another one. It depends on a lot of things (also on hardware configuration e.g.).

One of the things that determine how fast you make progress is how fast you can cycle through iterations.

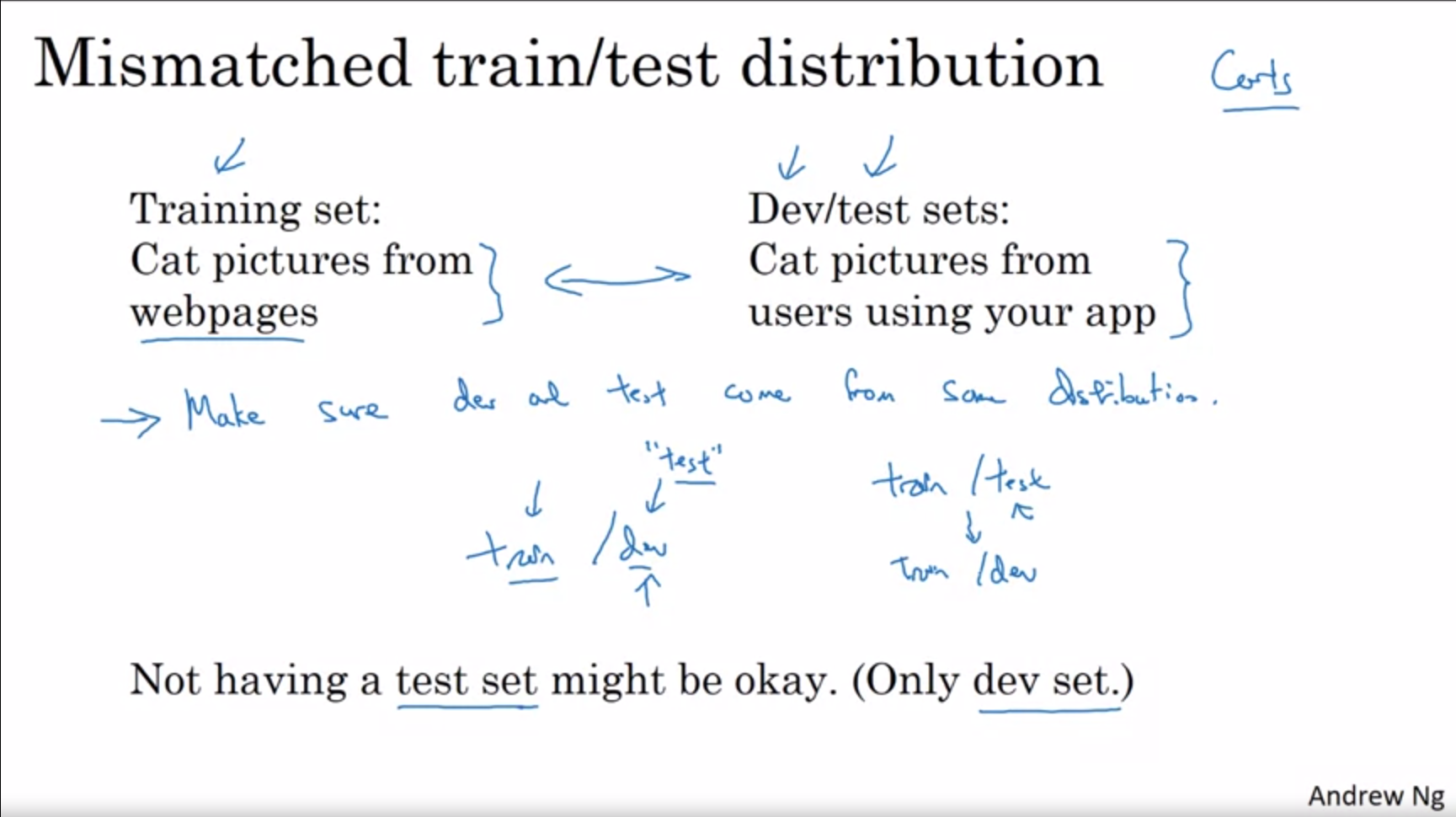
Setting up the data correctly can make you much more efficient at that.



Dev set to evalute different models and hyperparameters and decide which works best.

At the very end evaluate with the test set in order to get an unbiased estimate on how well the model is doing.

Previously (pre deep learning era) it was common to split e.g. 60-20-20. Now dev and test set are much smaller because that is sufficient to fulfill the goal. The more data we have for train the better. Of course this depends on how much data you have. We will see more specific guidelines later.

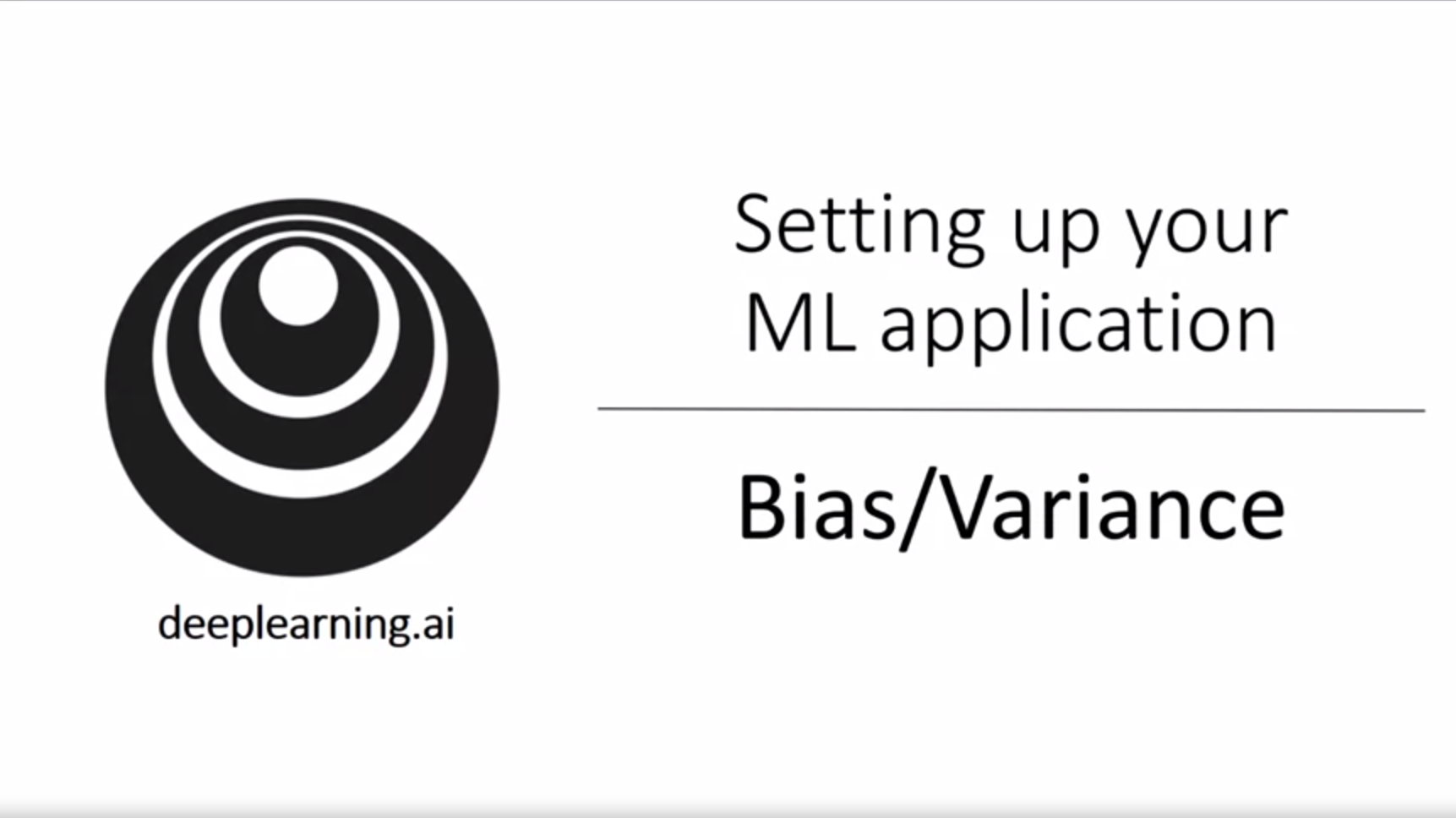


Another observation in the deep learning era is that more and more people train on a mismatched train/test distribution (because deep learning applications have such a big hunger for data).

Webpages might have high quality images while users upload somehow blurry more casual images.

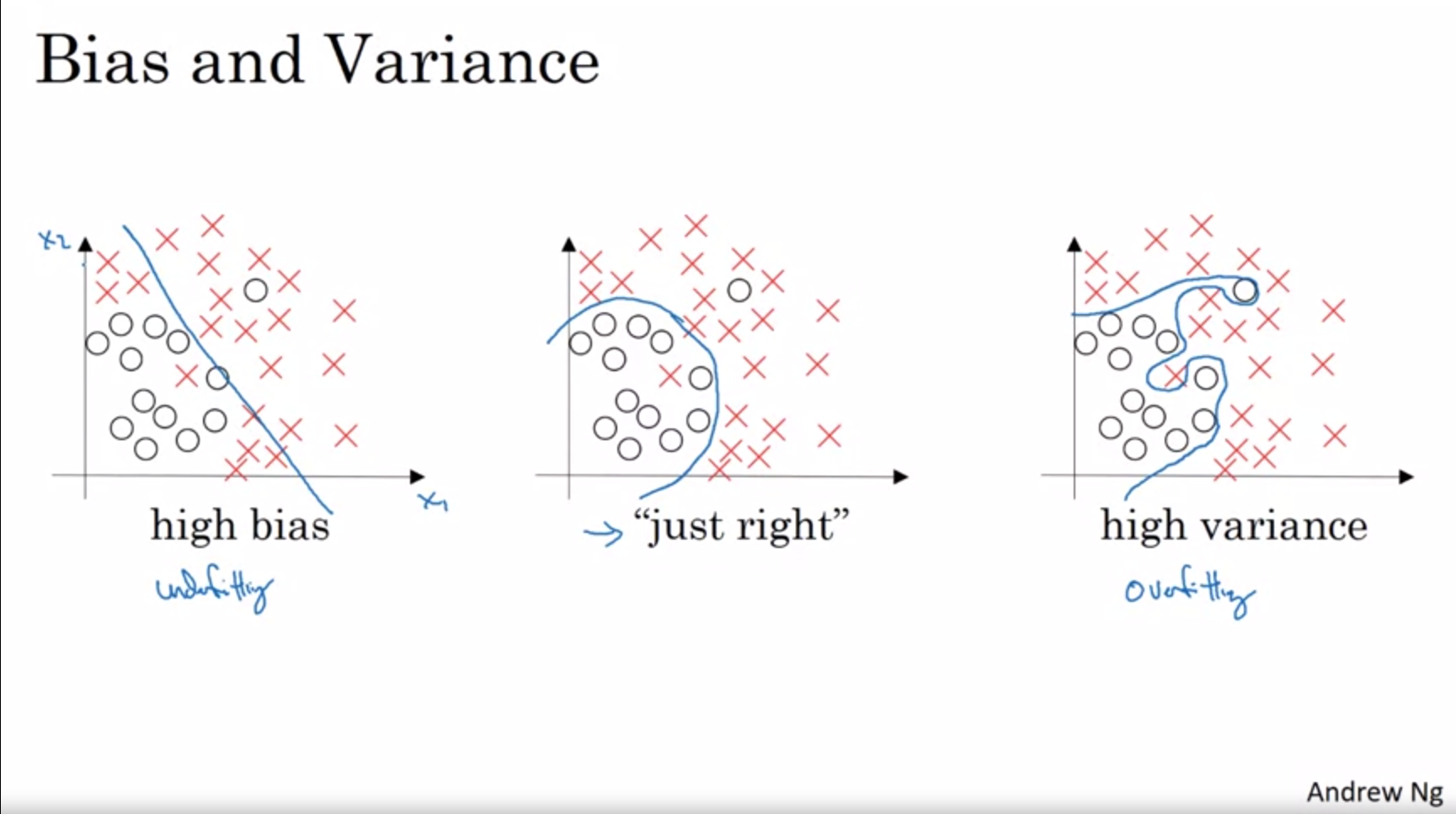
The rule of thumb is to keep at least dev and test from the same distribution (the one you are interested in).

Some people misuse the terminology here.

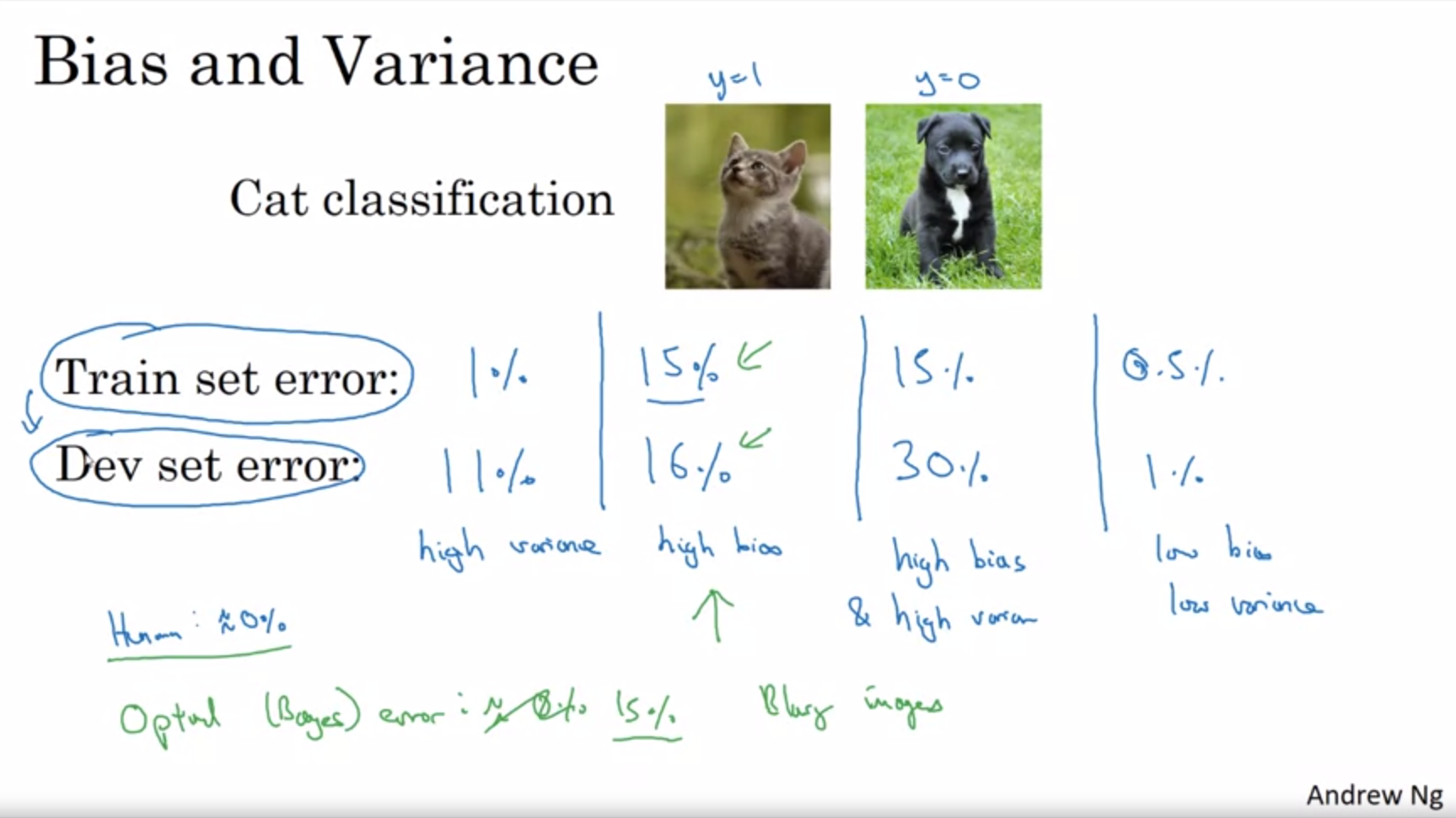


Bias / Variance is easy to learn but difficult to master!

In the deep learning era there is less of a tradeoff between the two. Previously this was a tradeoff mostly.

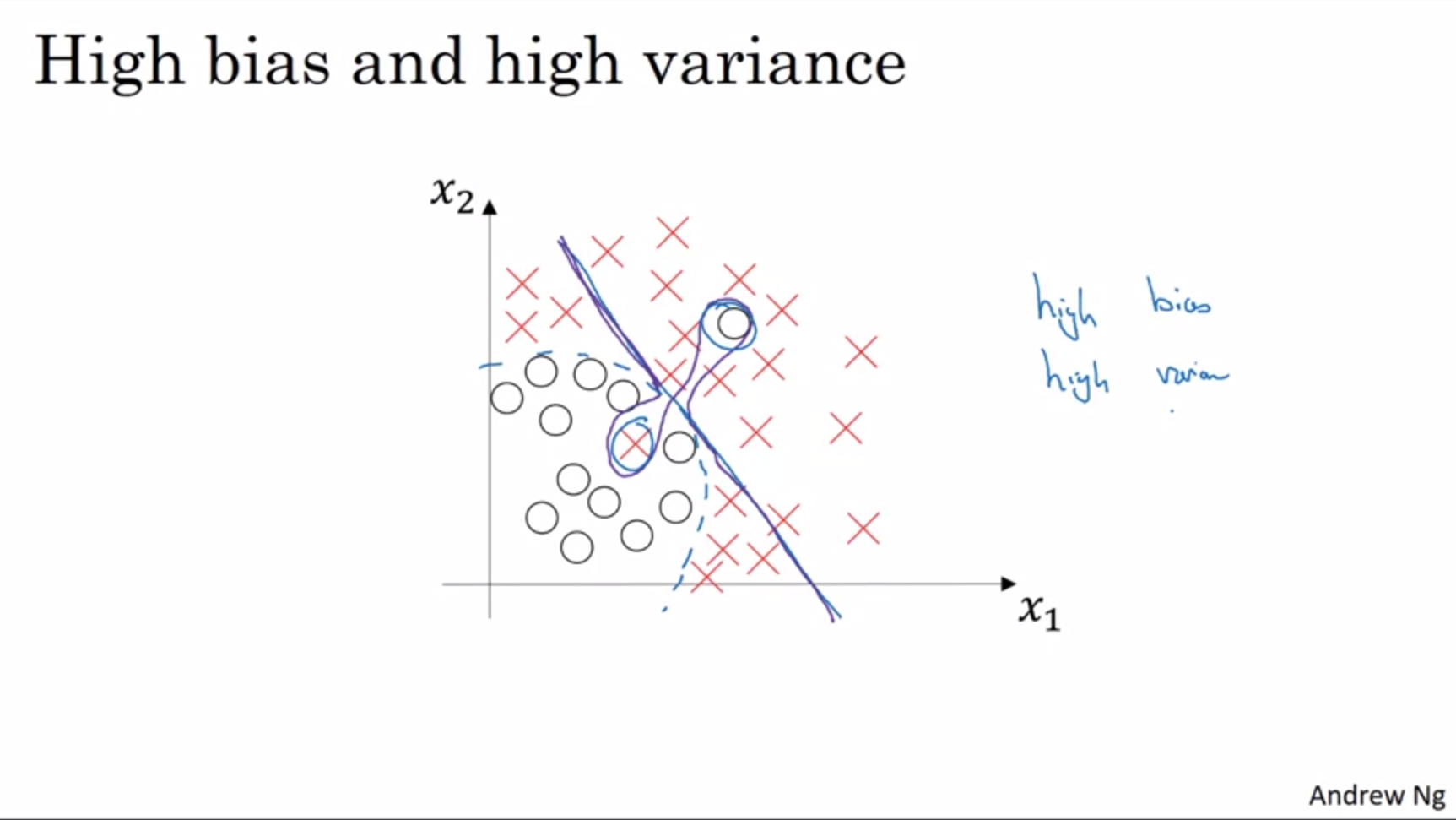


In high-dimensional problems you cannot just plot the data so we have to look at some metrics (relation between train set error and dev set error) to detect bias and variance.



This analysis is predicated on the assumption that human-level performance (or optimal (bayes) error) gets to nearly 0% error. If the bayes error would be ~15% the second case would not have high bias. For example, if the images would be blurry this could be the case.

We will talk about a more sophisticated analysis later.

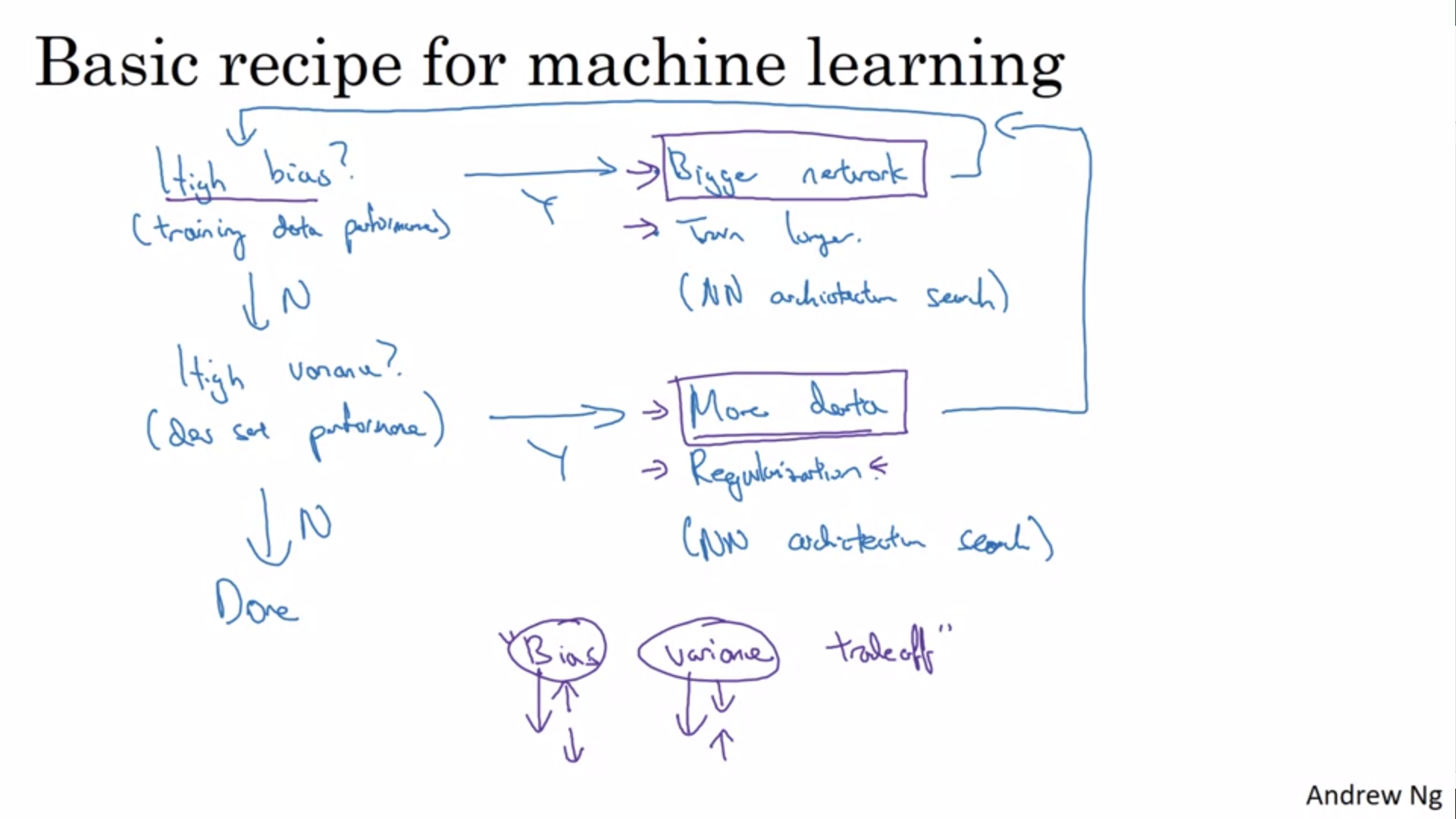


This example is a bit contrived with 2 dimensions but with very high dimensional inputs this can actually happen.

Depending on whether the algorithm suffers from high bias or high variance there are different things that you can try. Andrew will present us with some kind of basic recipe in the next video.



Go systematically about improving the algorithm!



First question: high bias? Bigger network and train longer often helps. Continue until the training set is fit well! (at least if a human can do the task and bayes error is not too high)

Only once bias is reduced to an acceptable amount look at variance! Best way to solve high variance is get more data! But regularization is also useful!

Depending on high bias or high variance the things you should try are very different. Look at the training and dev set errors!

For traditional ML techniques to reduce bias would often increase variance. No tools without hurting the other one. But in the modern deep learning era, at least while you can create a bigger network and collect more data, there is not really a tradeoff (as long as you regularize properly).

The two purple marked boxes are tools to only drive down bias or variance without hurting the other one.

Training a bigger network almost never hurts except for computational costs.

Next we will look at regularization.