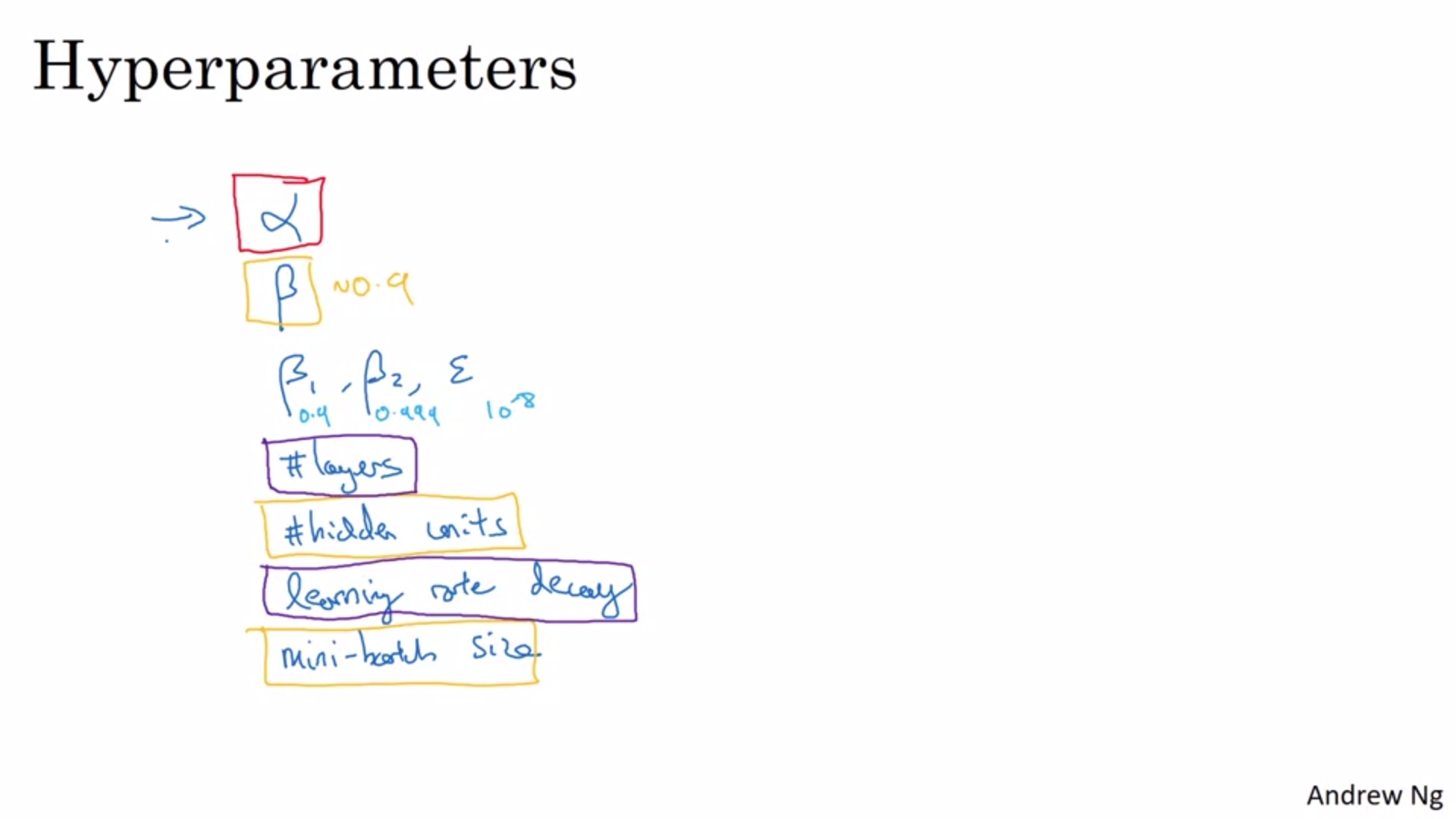
How to find good settings for hyperparameters?

One of the painful things about deep learning is to have to tune so many hyperparameters.

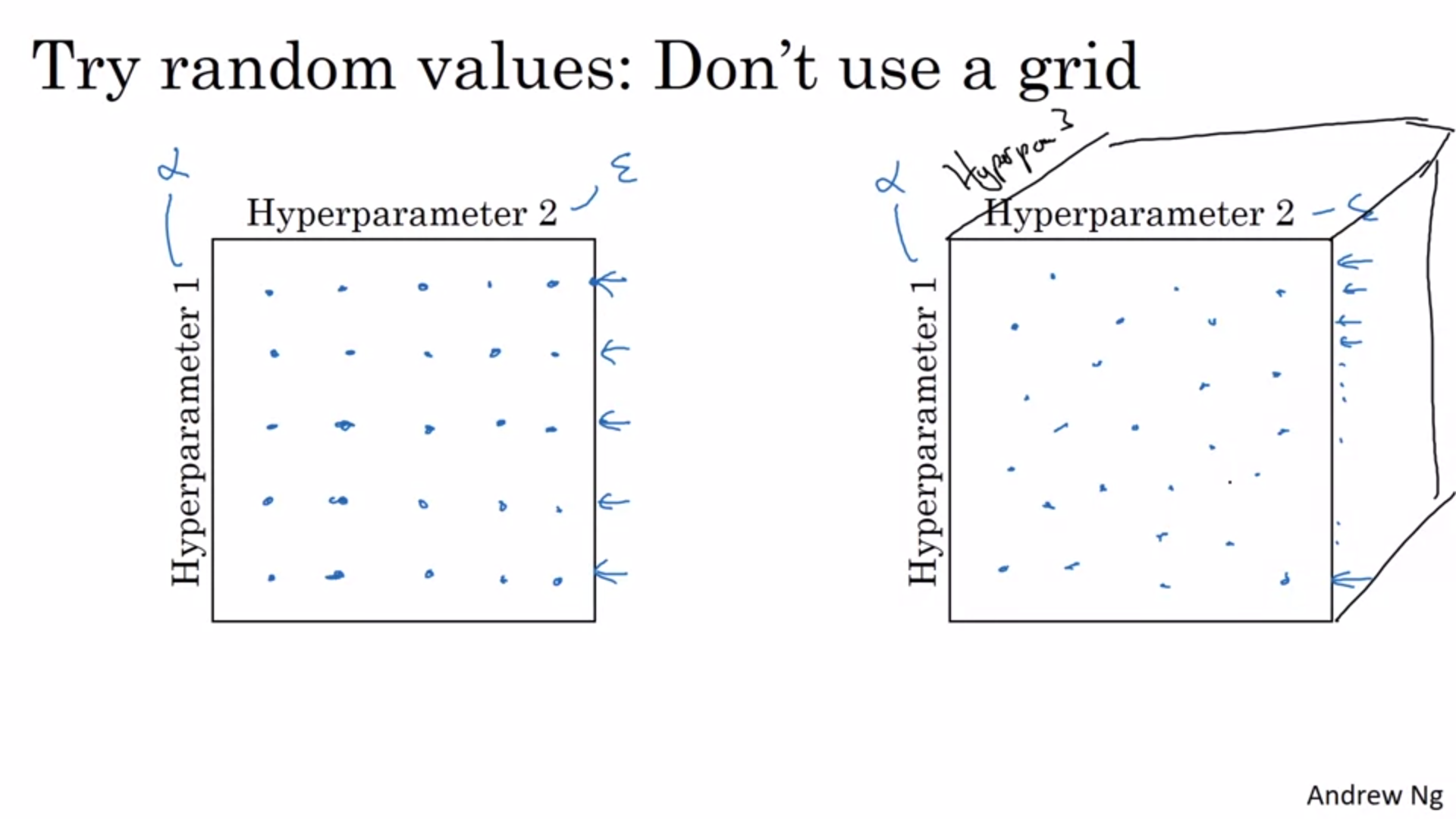


Alpha is the most important hyperparameter to tune.

Orange is the second in importance.

Purple is third in importance.

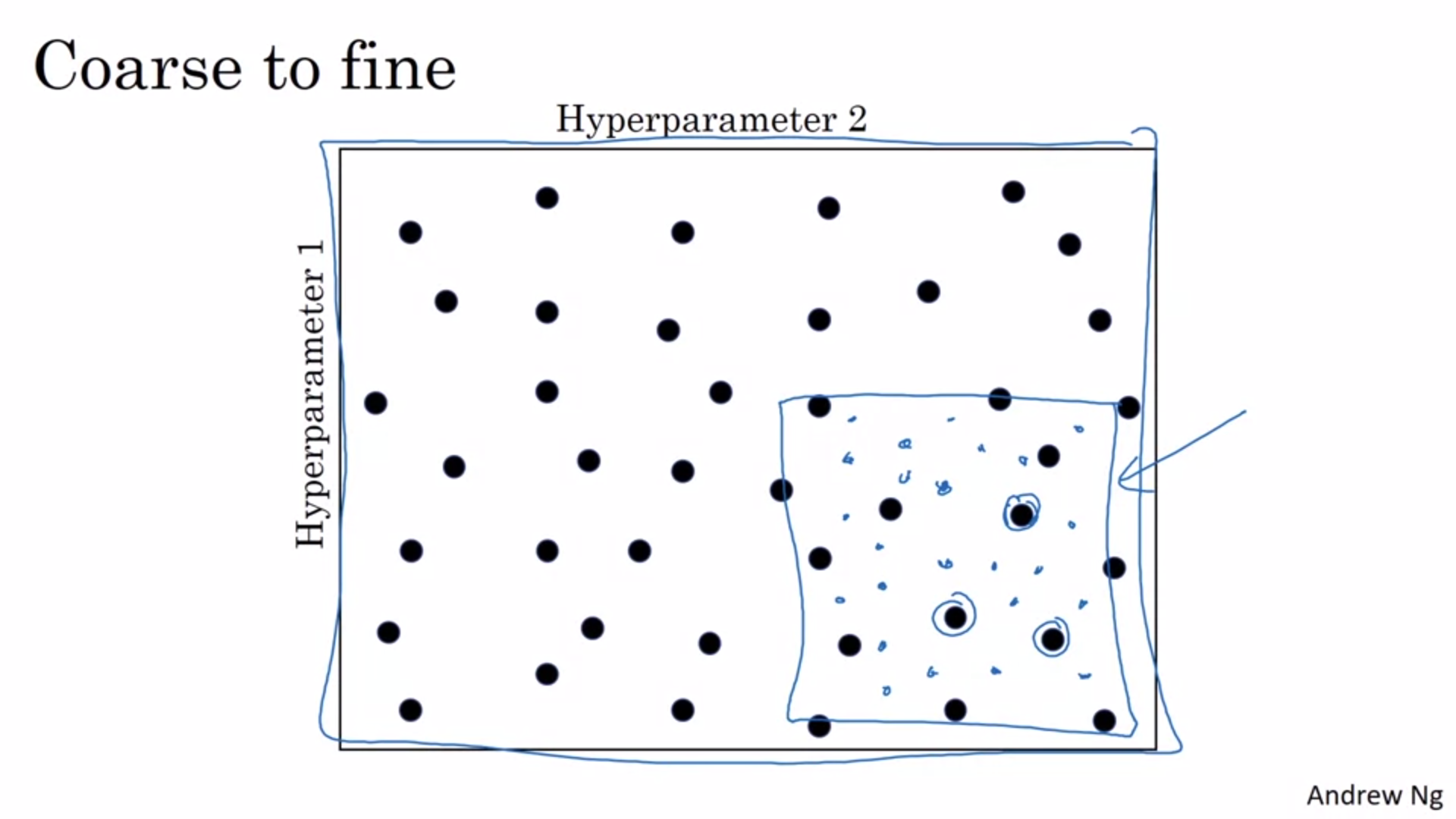
This is not a hard rule. Difference practitioners have different opinions.



Earlier it was common practice to use a grid and try out all points. This only works ok if the number of hyperparameters is quite small. Otherwise you should choose them at random instead!

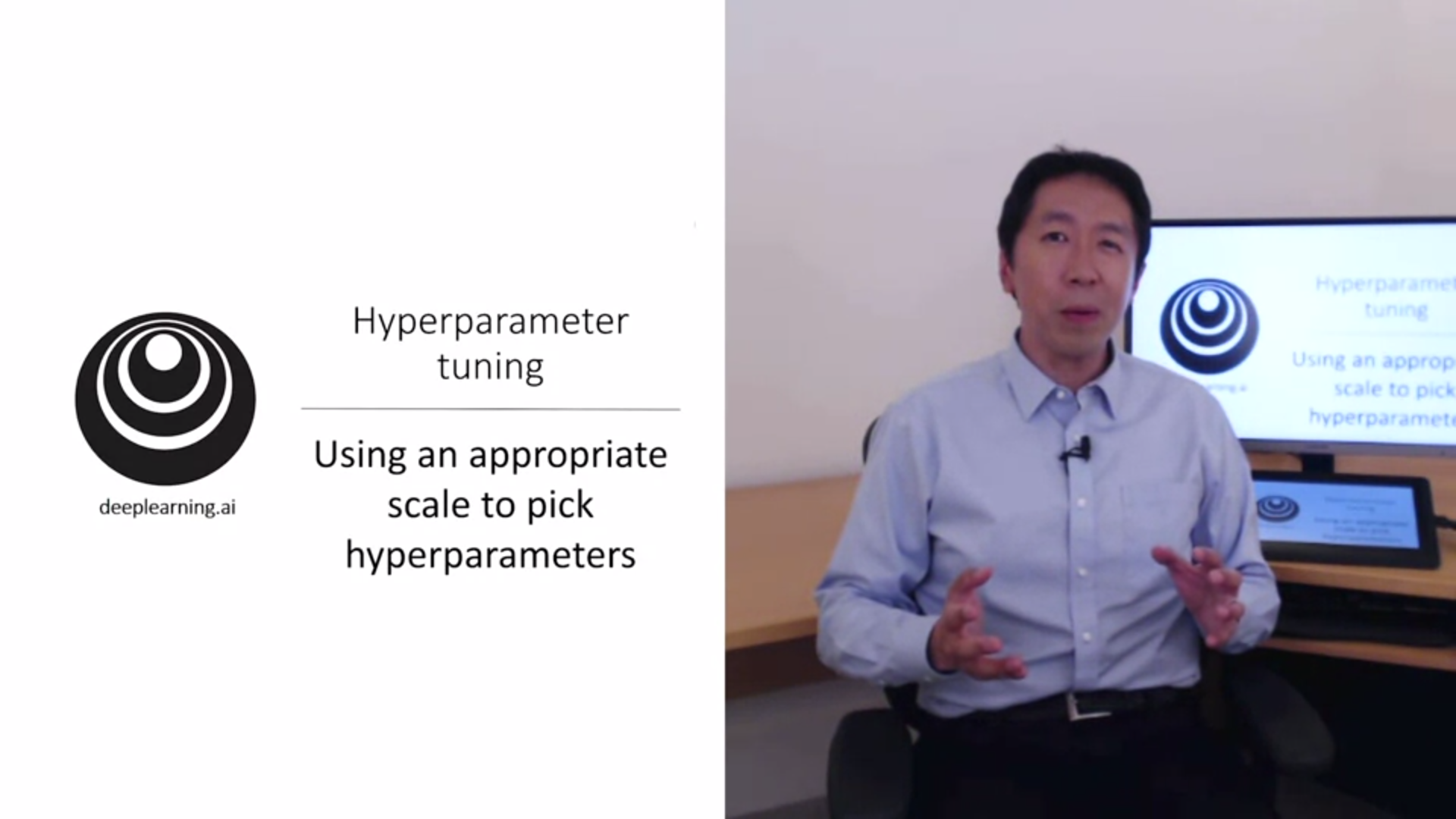
Example: one important hyperparameter (alpha) and one not important (epsilon). You will have only tried out 5 values for alpha at 25 different samples. In higher dimensions this is even more so.

With random sampling you are more richly exploring.

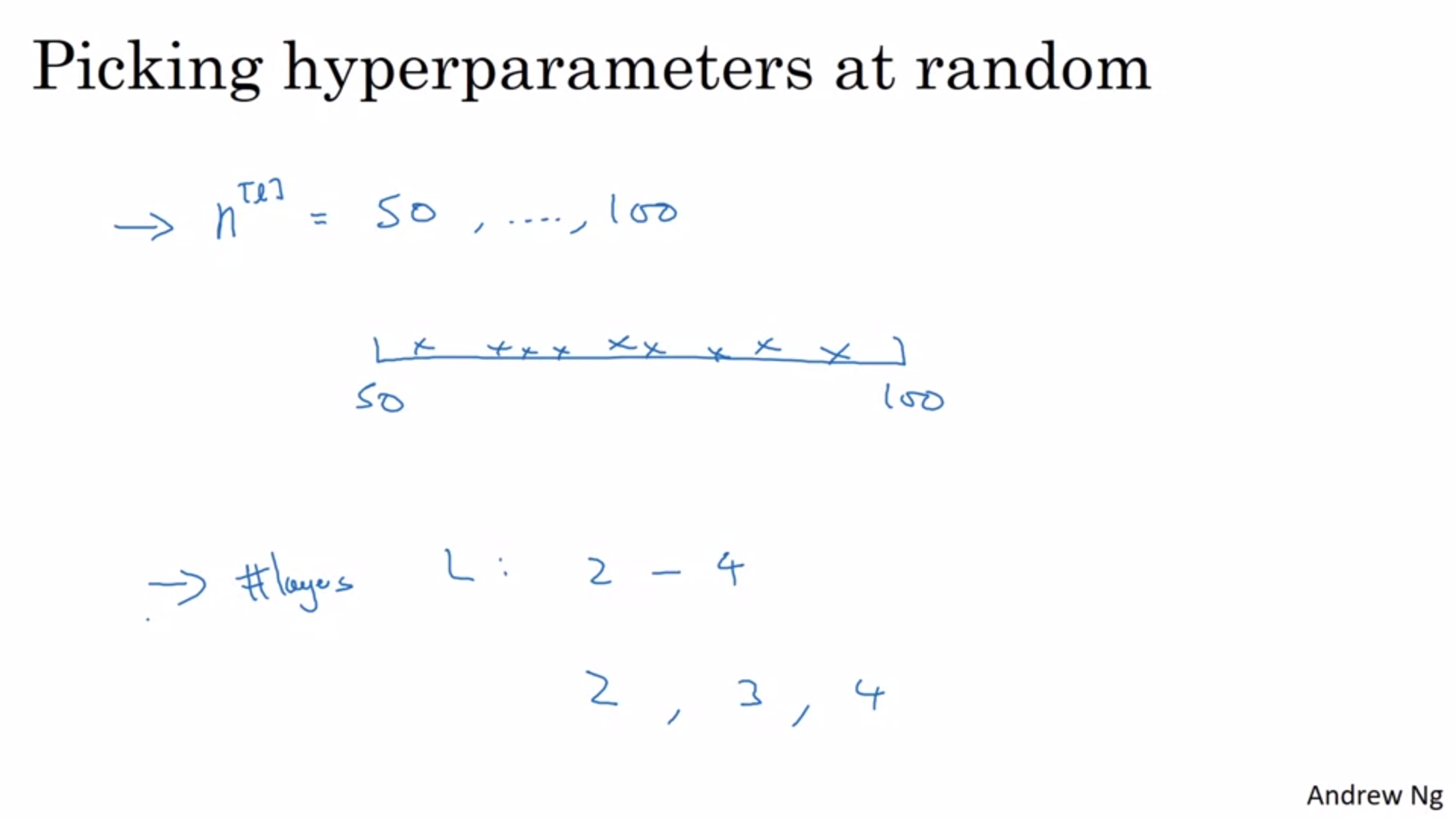


Zoom in to a region that works well and sample more densely in that space.

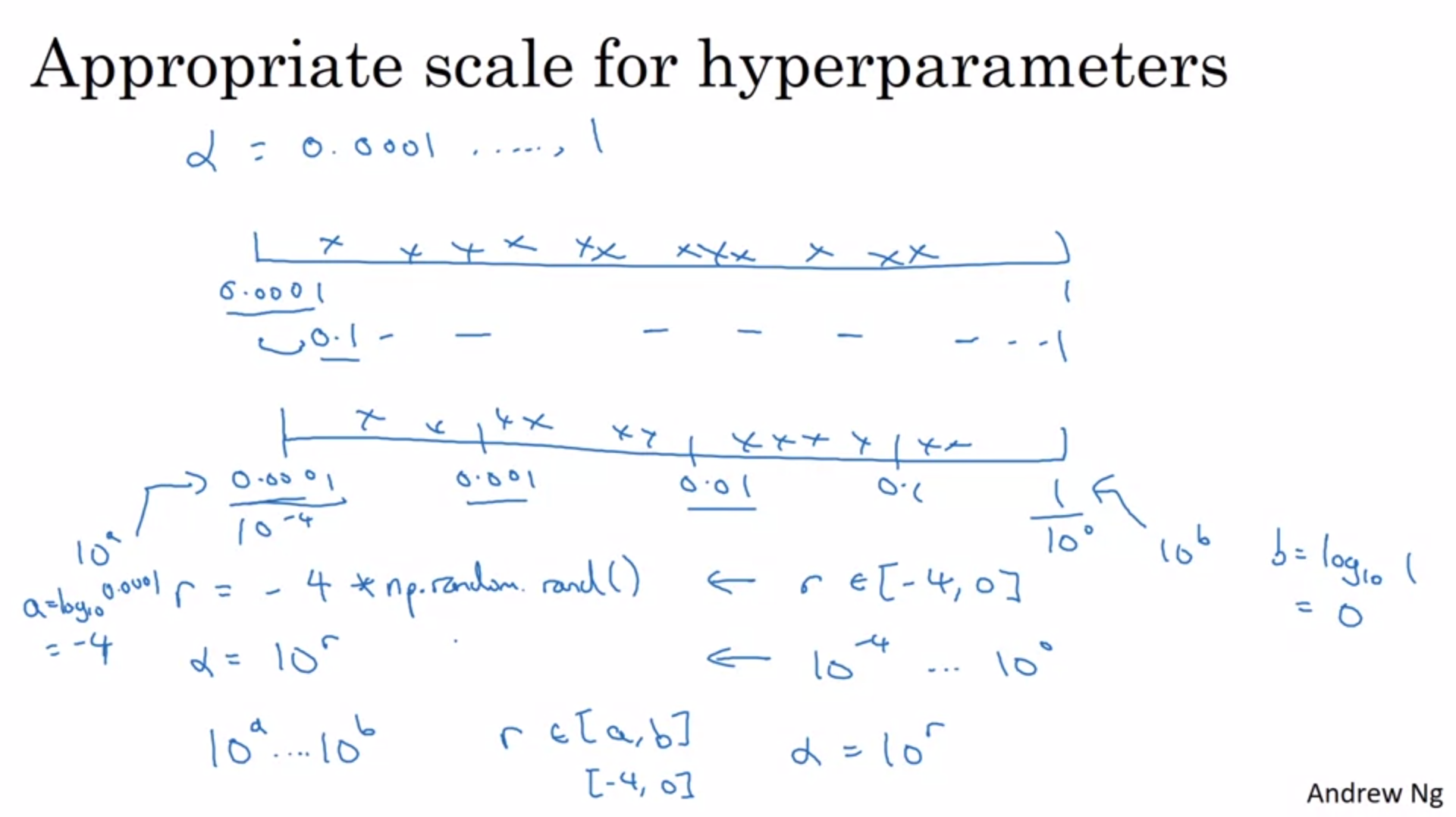
You might look at the dev set error or training cost to optimize hyperparams depending on what you want to achieve.



Sampling at random does not mean sampling uniformly at random over the hyperparameter range.



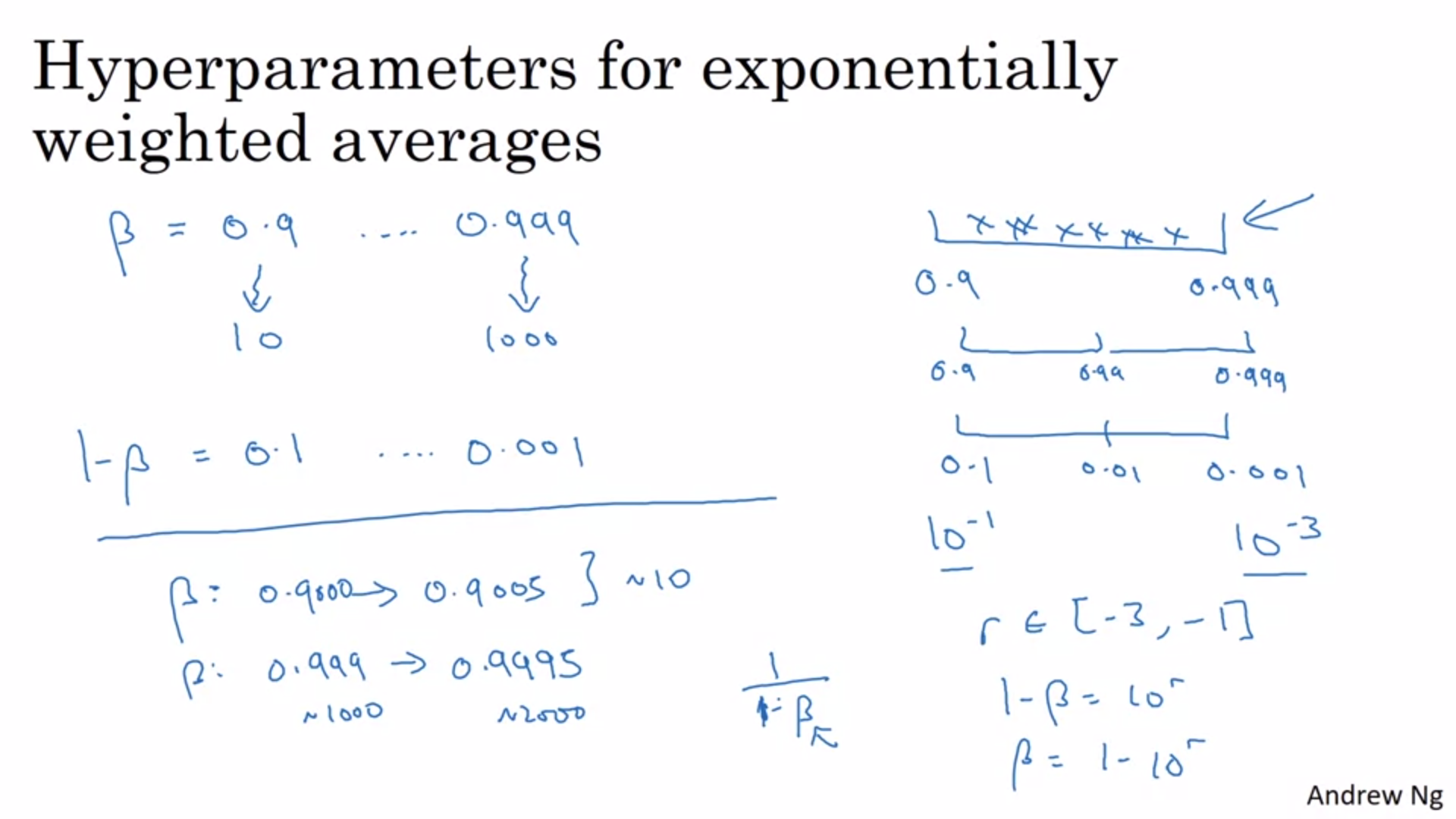
In this case sampling uniformly at random is reasonable. But this is not true for all hyperparameters.



E.g. for the learning rate we do not want to sample uniformly at random. Otherwise we would spend much more resources on the range between 0.1 and 1 rather than between 0.0001 and 0.1.

We want to search for hyperparameters on the log scale!

At the bottom of the slide is how you could implement this in python: first calculate r to be between -4 and 0 and then take 10 to the power of r to retrieve alpha.



Another tricky case is beta.

Remember 0.9 -> averaging over 10 days; 0.999 -> averaging over 1000 days.

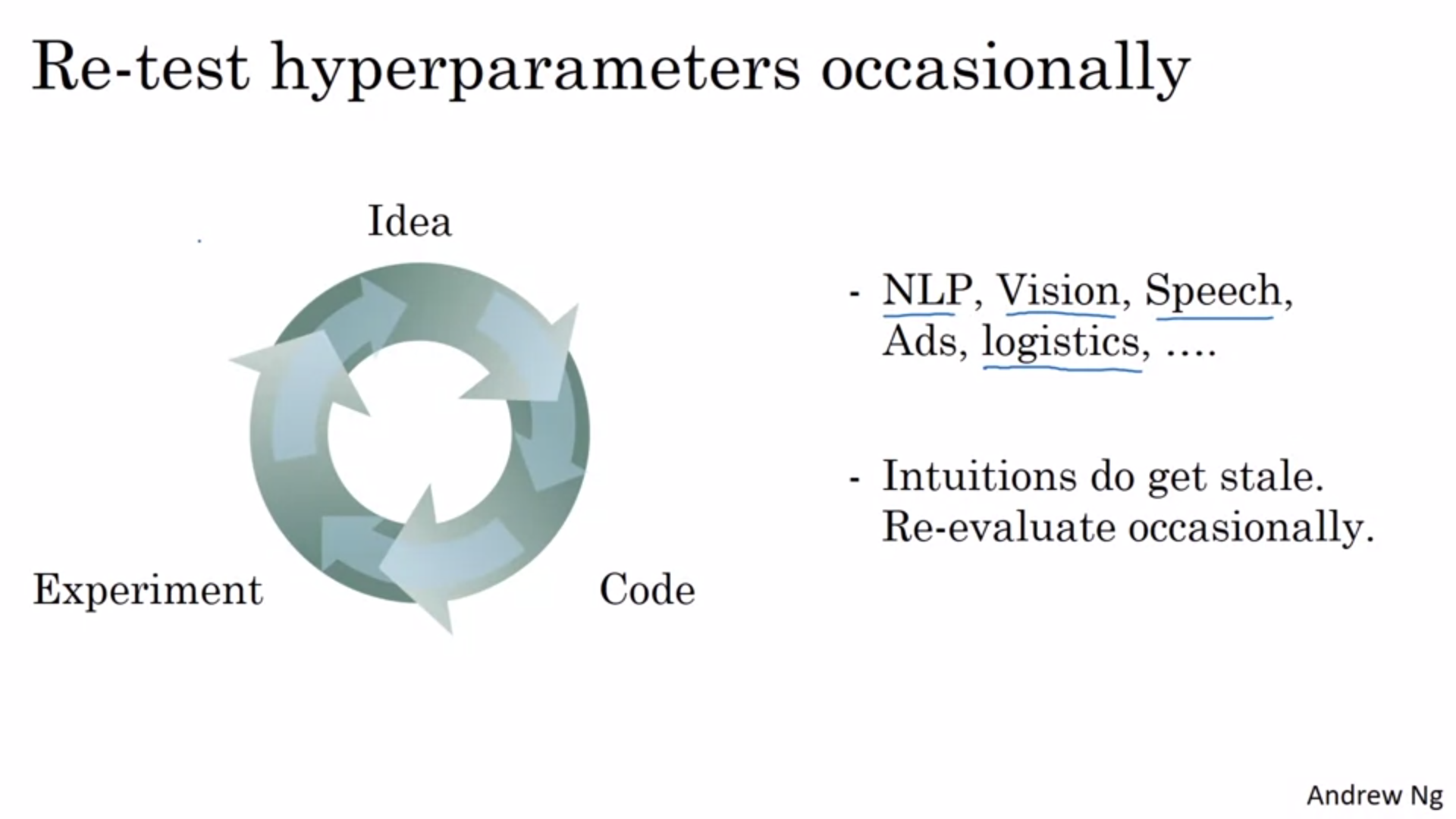
A better way to think about this is to sample 1-beta.

We want to explore betas between 0.9 and 0.999.

The theoretical justification is that the sensitivity of results we get when beta is close to 1 changes even with very small changes to beta (see bottom left of the slide) -> sample more densely when beta is close to 1.

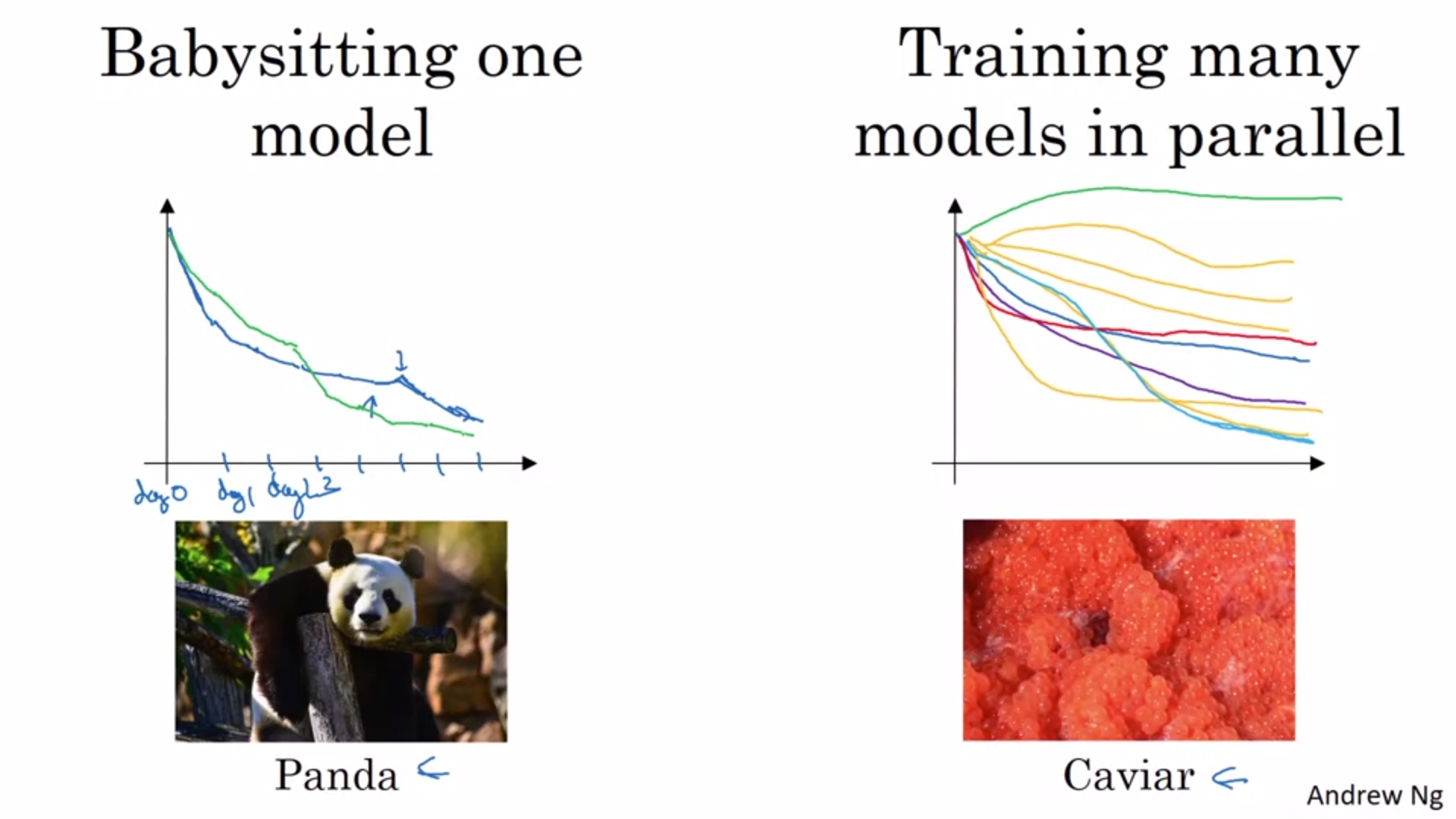
Sampling uniformly at random can also work especially if you use coarse to fine.





Many different application areas for deep learning. There is some crossfertilization but for hyperparameters it is quite limited.

Intuitions do get stale (also when hardware changes for example).



Two schools of thought. Babysitting one model when you can only afford to train one model (e.g. because of computational power).

Panda: Every day you look at it and make a new decision (nudging the learning rate up or down for example).

Caviar: train multiple models at the same time in parallel. Try different hyperparameter settings and at the end pick the one that worked best.

If you can train multiple models choose caviar approach! This is also application dependent.

Even with pandas approach if you notice the model does not work well you can start train another one.