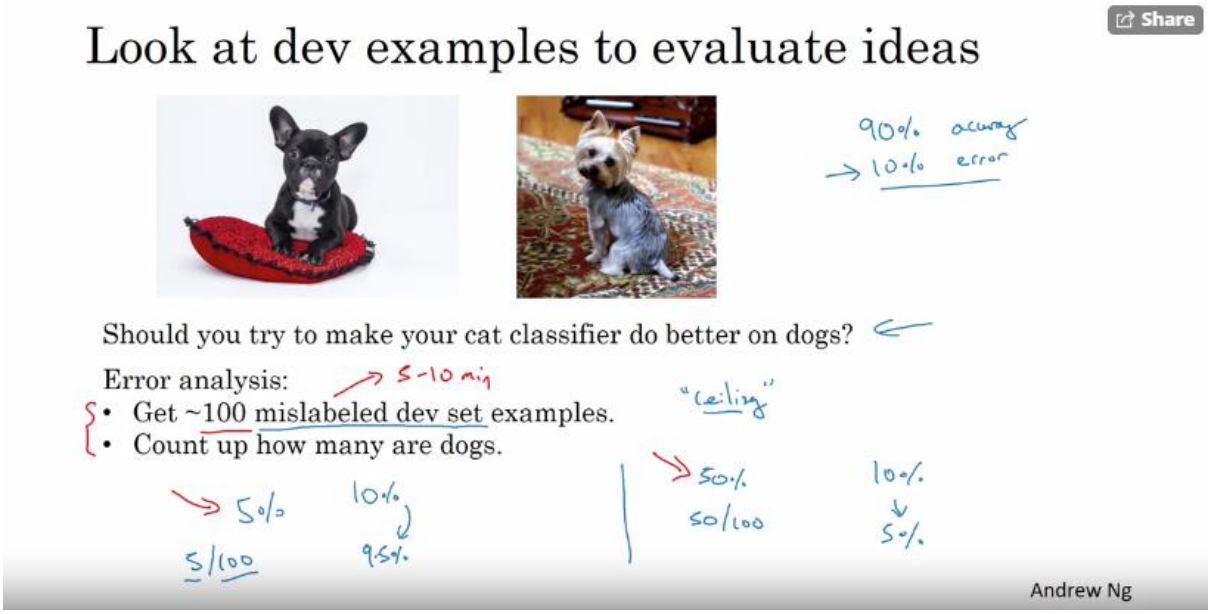
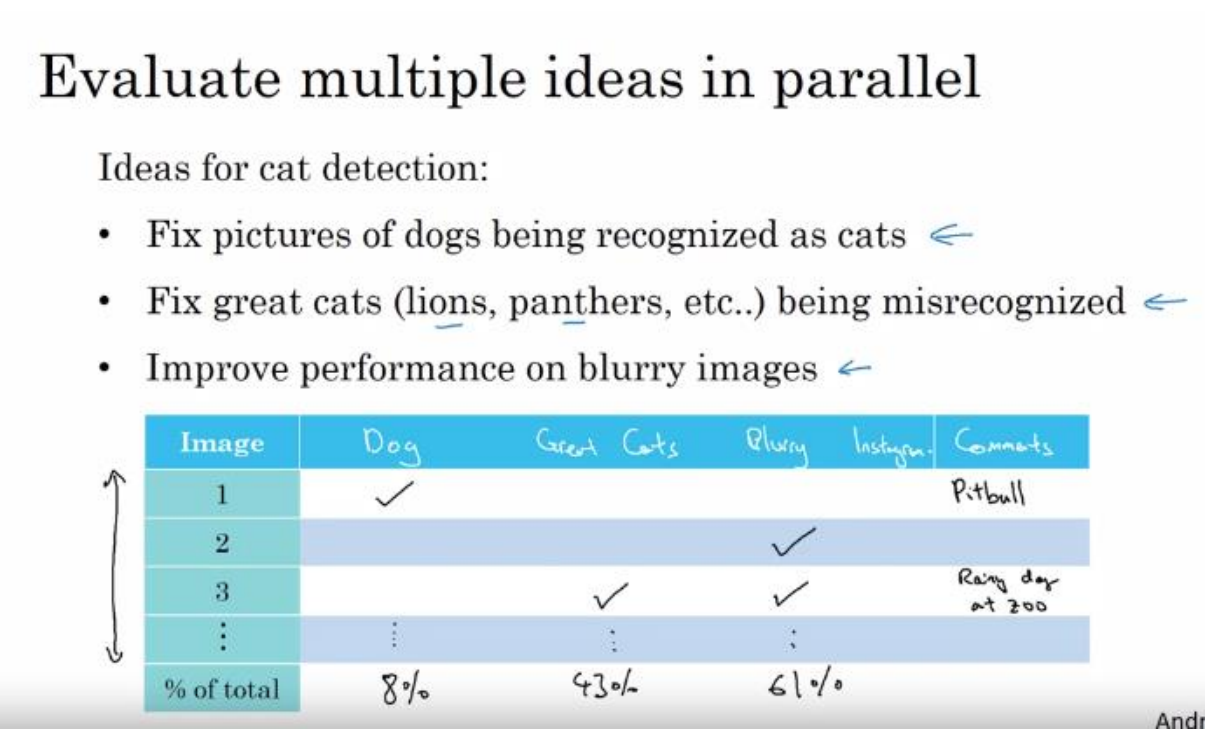
Error analysis





This gives us an estimate of how worthwhile it might be to work on various categories of misclassification.

e.g potential improvement is higher by improving performance on great cats or blurry images

Cleaning up incorrectly labelled data

Incorrectly labeled examples

Image	Dog	Great Cat	Blurry	Incorrectly labeled	Comments
...					
98				✓	Labeler missed cat in background
99		✓			
100				✓	Drawing of a cat; Not a real cat.
% of total	8%	43%	61%	6%	

Overall dev set error ..... 10% | 2%

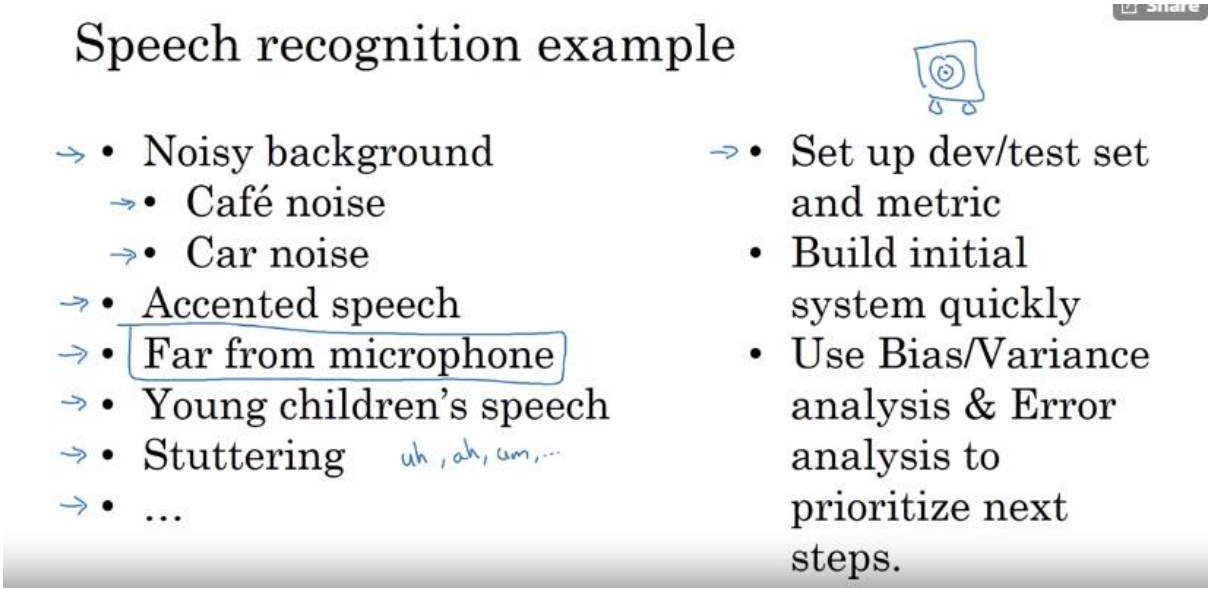
Errors due incorrect labels ..... 0.6% ← | 0.6%

Errors due to other causes ..... 9.4% ← | 1.4%

Correcting incorrect dev/test set examples

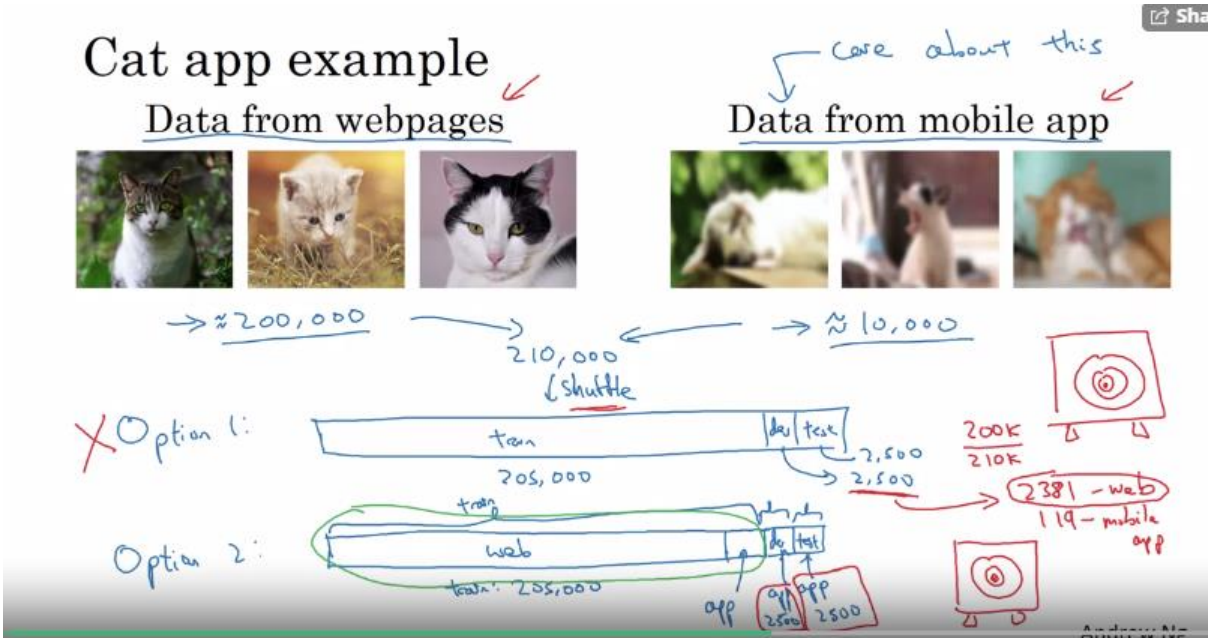
- Apply same process to your dev and test sets to make sure they continue to come from the same distribution
- Consider examining examples your algorithm got right as well as ones it got wrong. (98·1 | 2%)
- Train and dev/test data may now come from slightly different distributions.

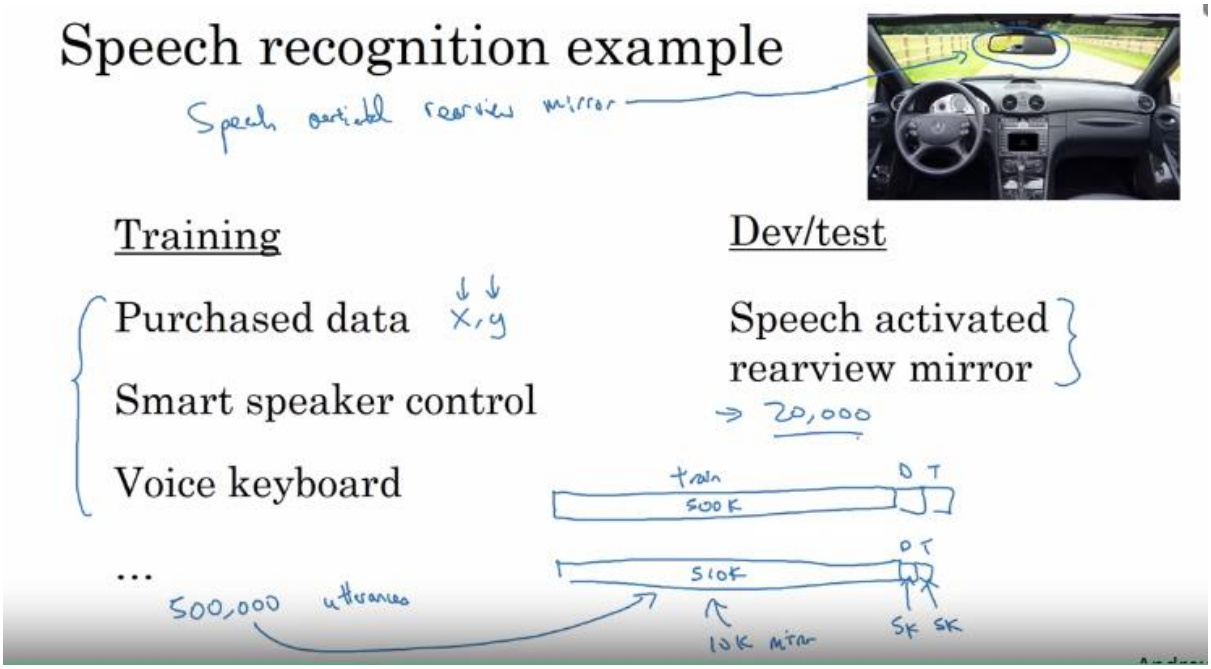
Build your first system quickly, then iterate



Training & testing on different distributions

In dev and test set, keep the daa you really care about, i.e. the oens that come from the distribution that you'll use in practice.



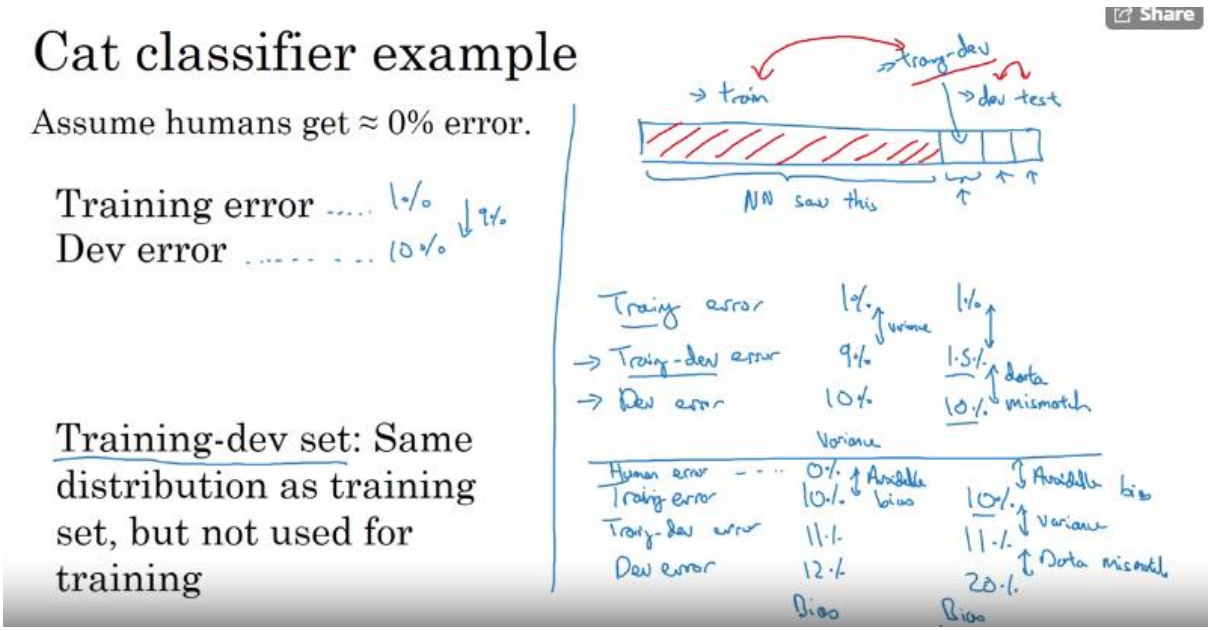


Bias & variance with mismatched data distributions

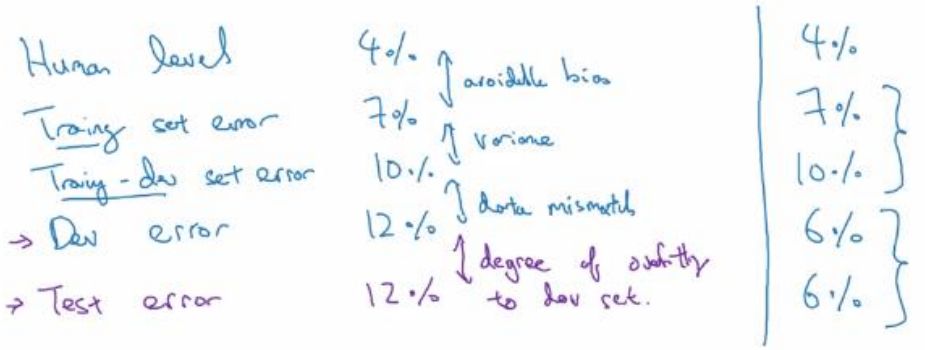
If dev set error is much higher than training error, we could say we have a high variance problem where our model doesn't generalize well on dev set.

But If train & dev sets come from different distributions, we can no longer say this. Maybe the dev set is just much harder to classify, because it comes from a different distribution.

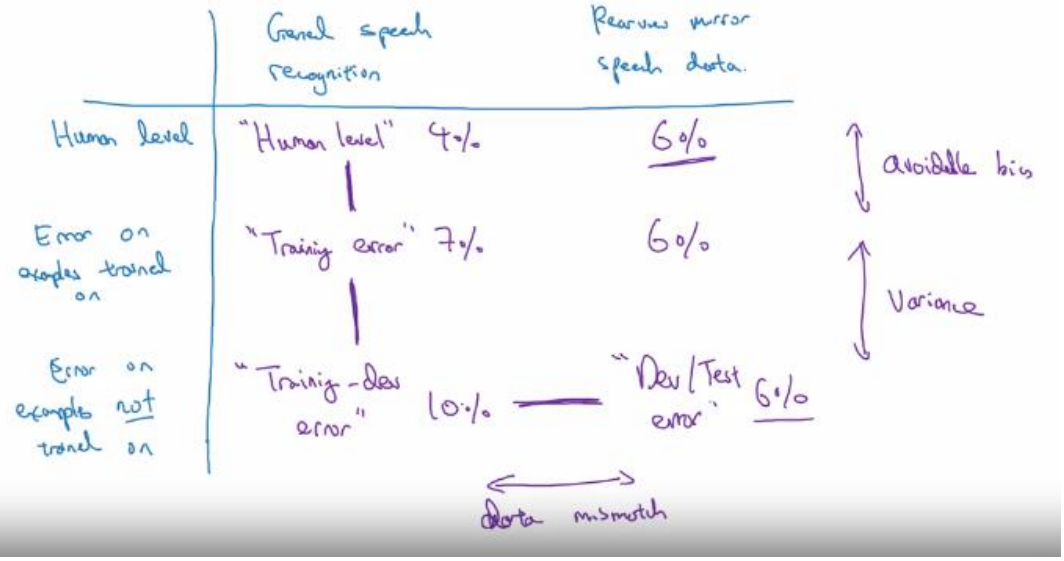
We can use a dev subset out of the train data, which we can be assured comes from same distribution as train data.



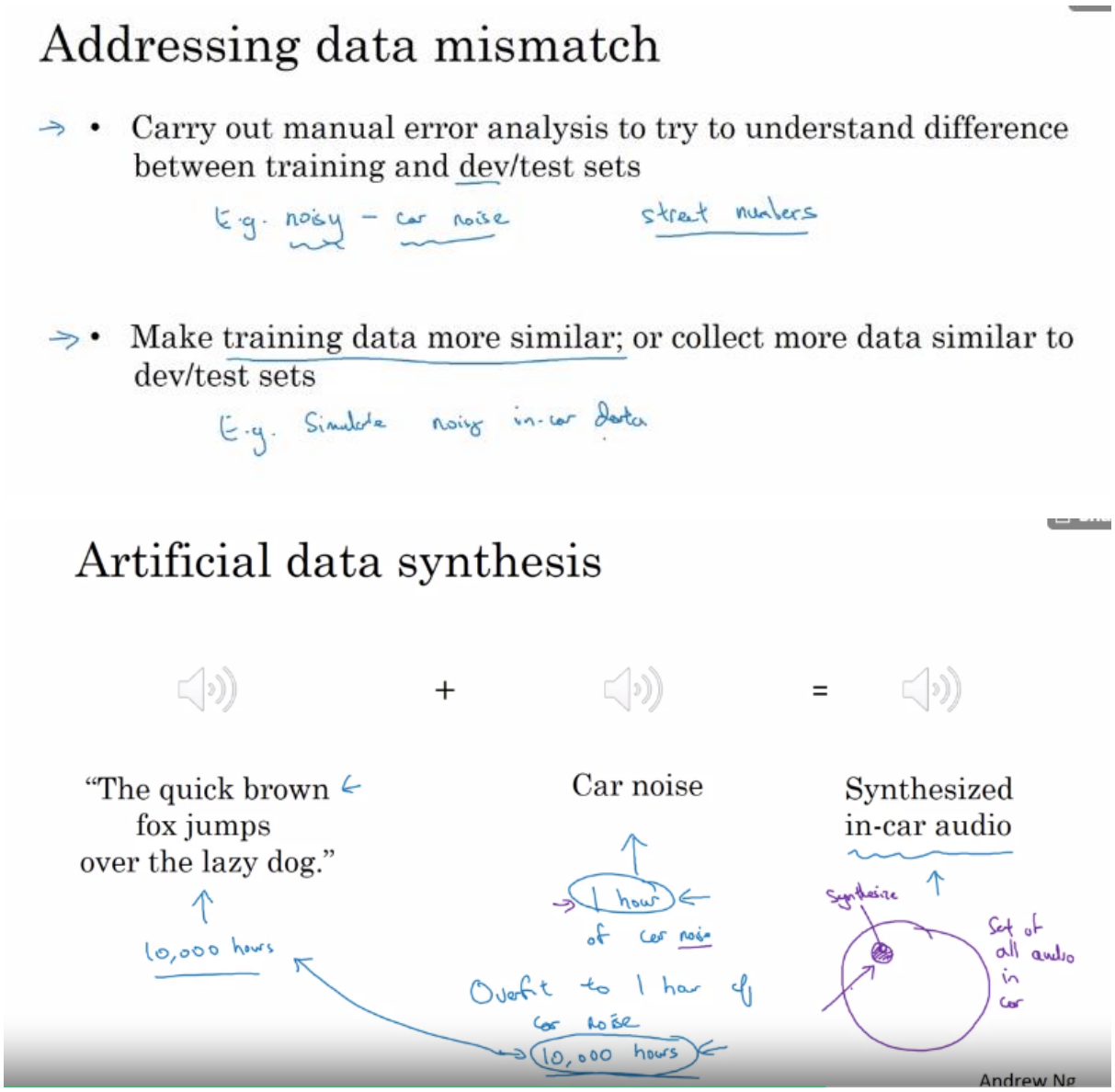
Bias/variance on mismatched training and dev/test sets

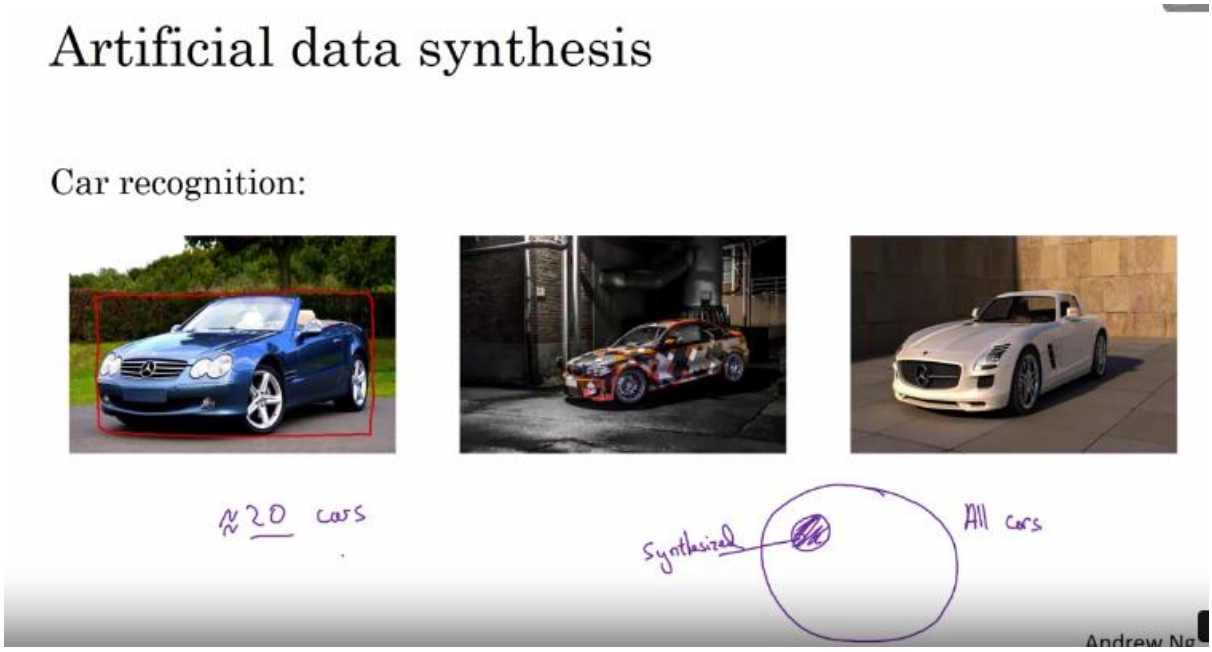


More general formulation



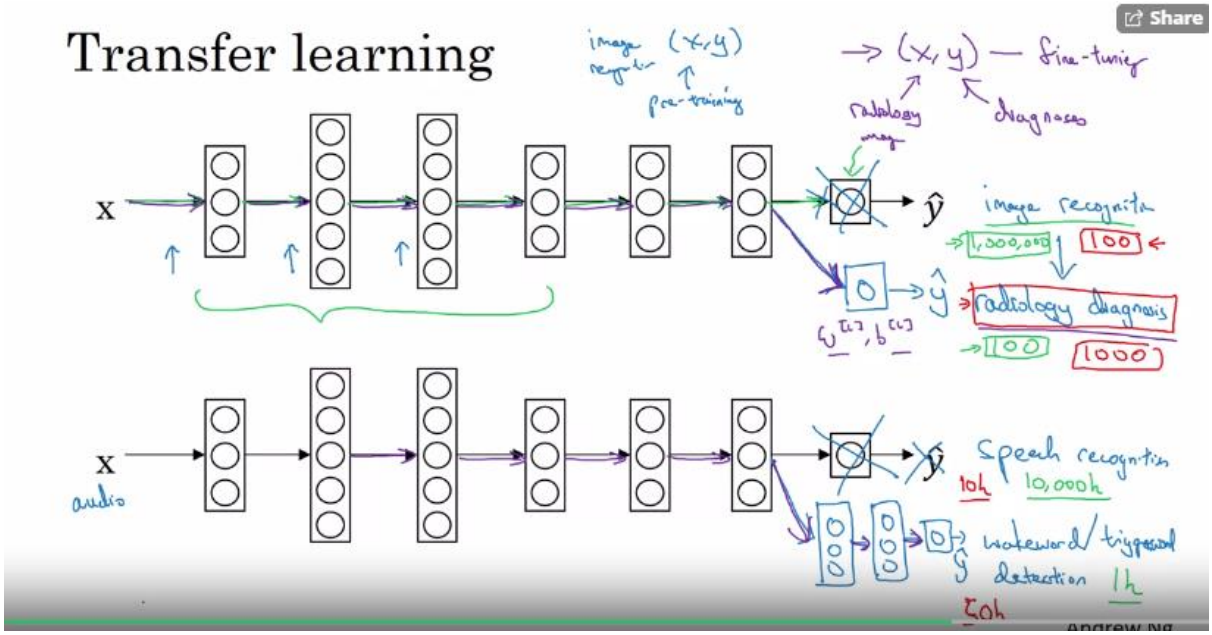
Addressing data mismatch





if you synthesize just a very small subset of these cars, then to the human eye, maybe the synthesized images look fine. But you might overfit to this small subset you're synthesizing

Transfer learning



When transfer learning makes sense

Transfer from A → B

- Task A and B have the same input x.
- You have a lot more data for Task A than Task B.
- Low level features from A could be helpful for learning B.