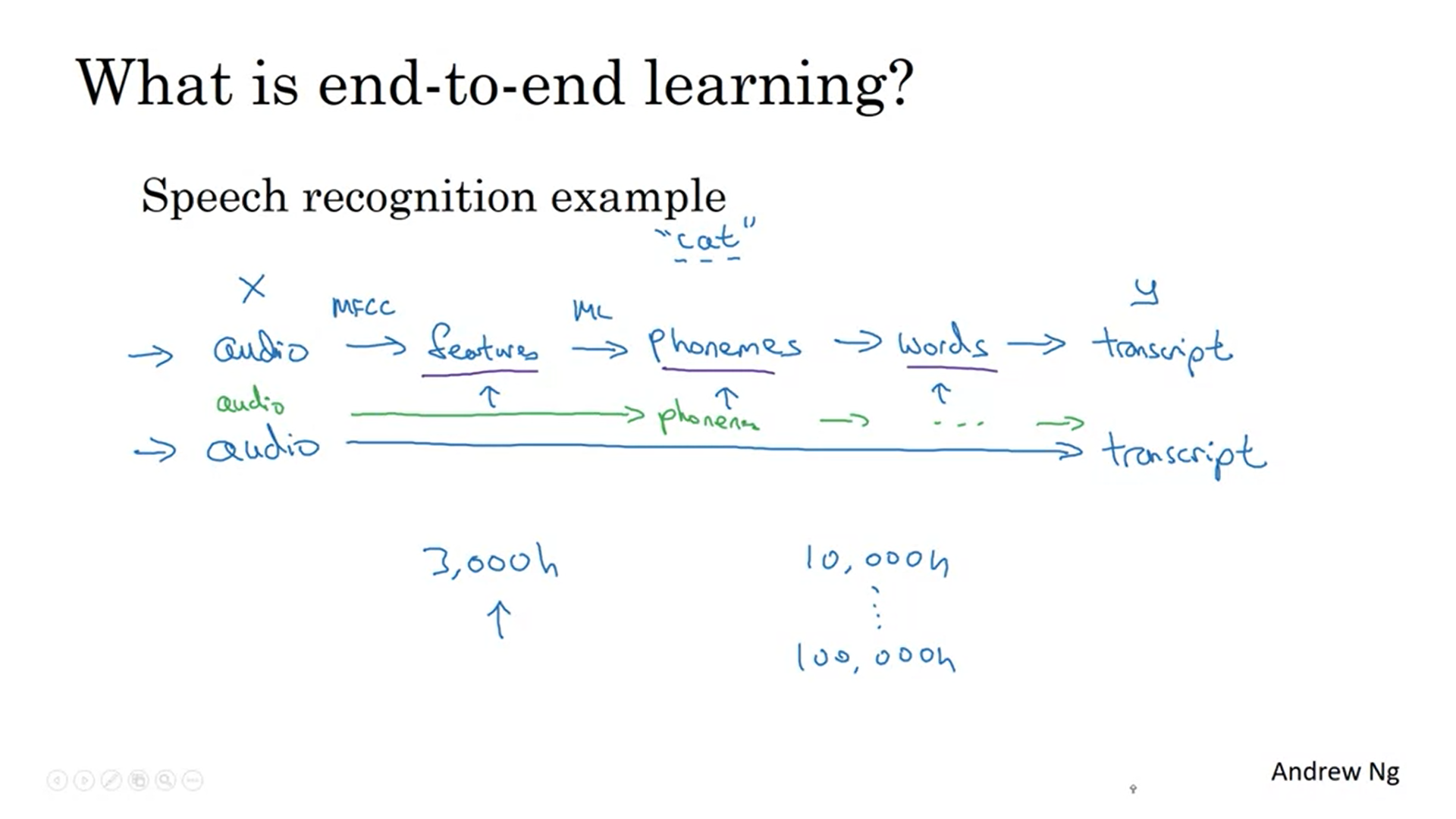


End-to-end deep learning has been one of the most exciting developments in deep learning.

There have been some data processing / learning systems that require multiple stages in the process. End-to-end deep learning will take all those stages and replace them with a single neural network.

Let’s look at some examples.

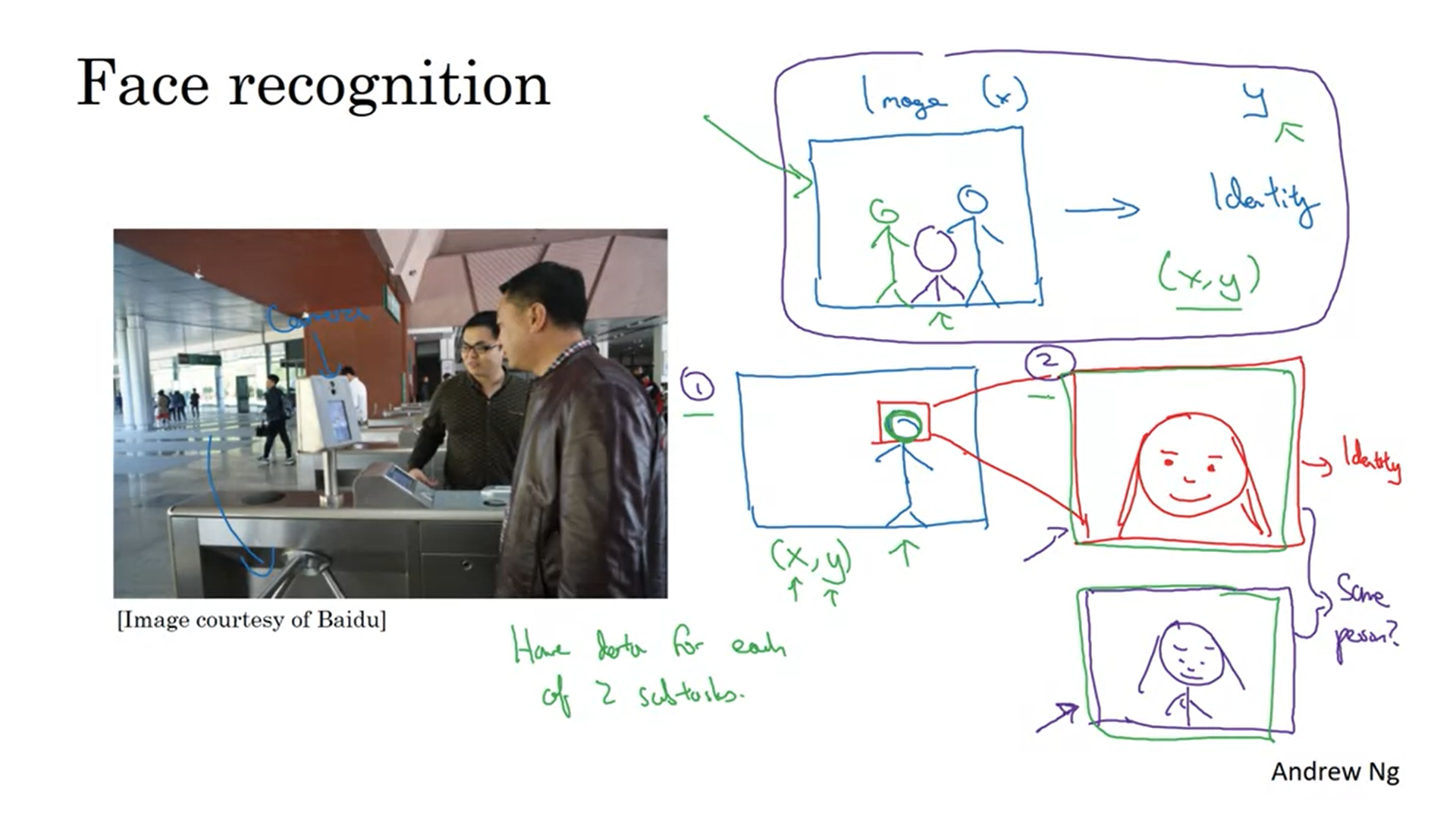


MFCC: algorithm to extract features for audio.

The pipeline with lots of different stages is not needed anymore. Some research got obsolete (this was challenging to accept).

One of the challenges of end-to-end deep learning is that you might need a lot of data before it works well.

3000h data -> traditional pipeline works well.

10000h – 100000h -> end-to-end approach works better.

Face recognition at turnstyle instead of batch.

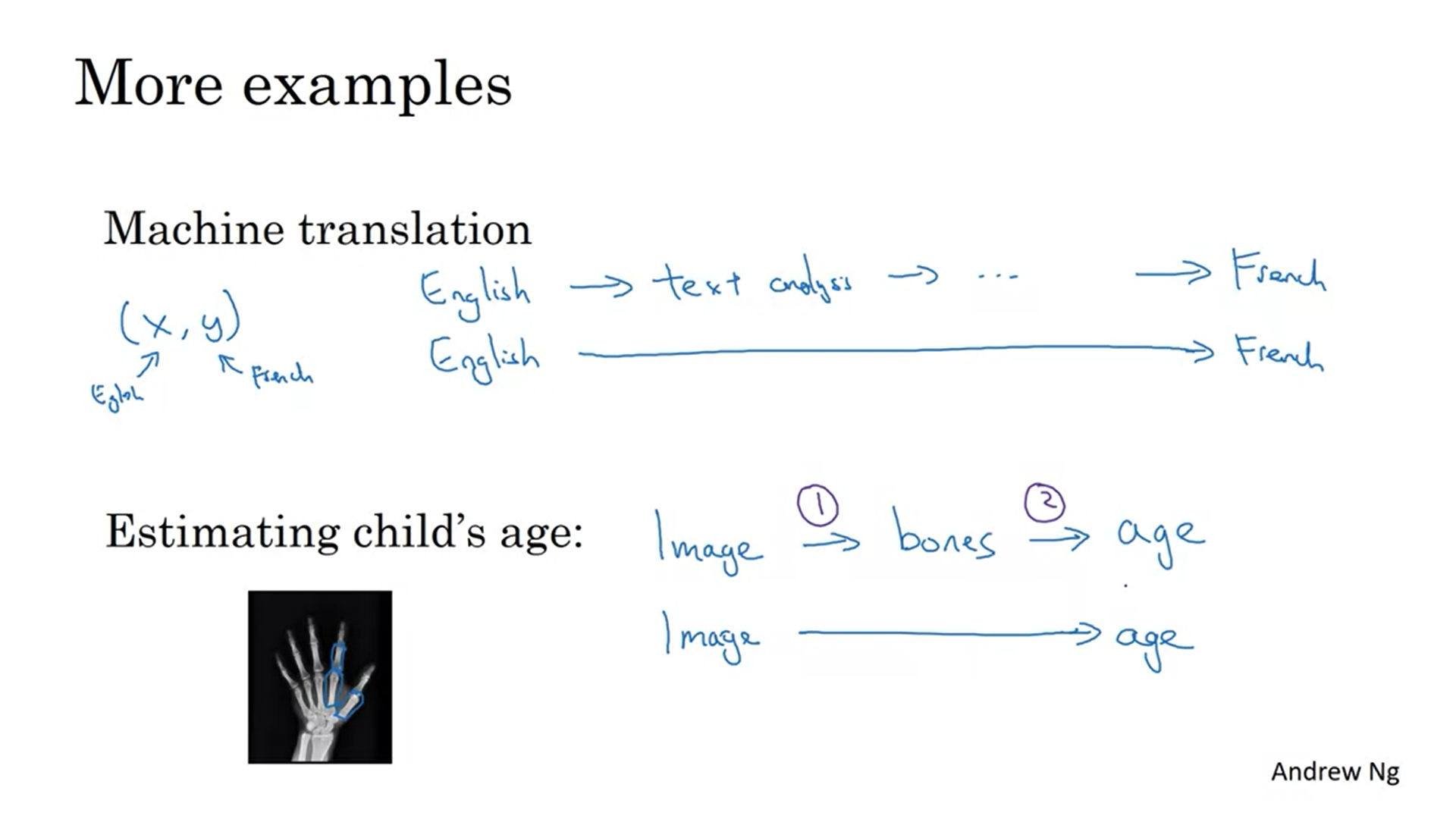
Mapping from the image to the identity is not the best approach because the person could approach the turnstyle from any position.

The best approach today seems to be a multistep approach. First a face detector to detect the person’s face, then zoom in to that part of the image, then estimate the person’s identity.

These are two simpler tasks and result in better overall performance.

Actually the network will take as input 2 images and tell you whether these are the same person or not. If you have 10k employees the image will be compared to these 10k employees.

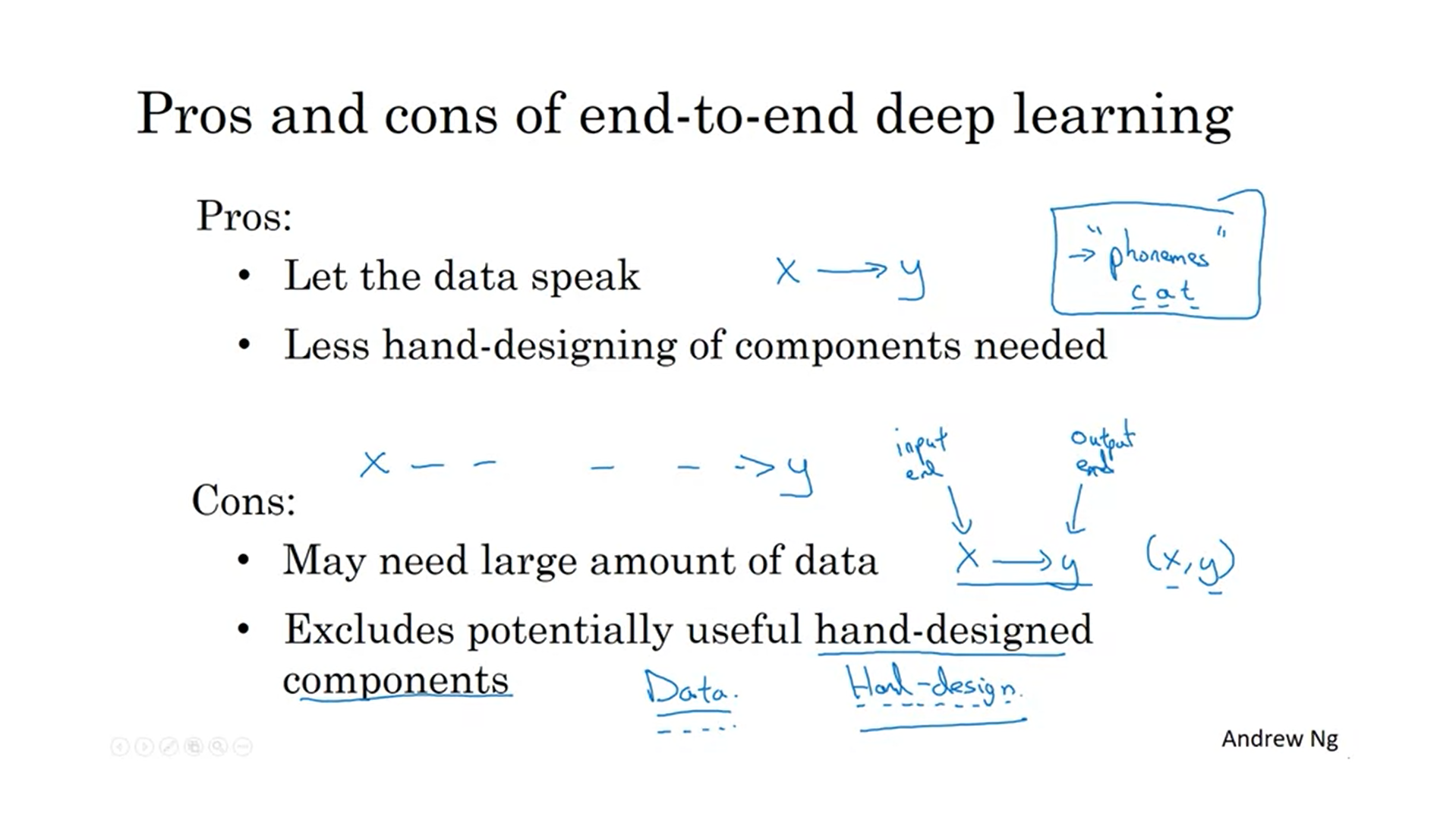
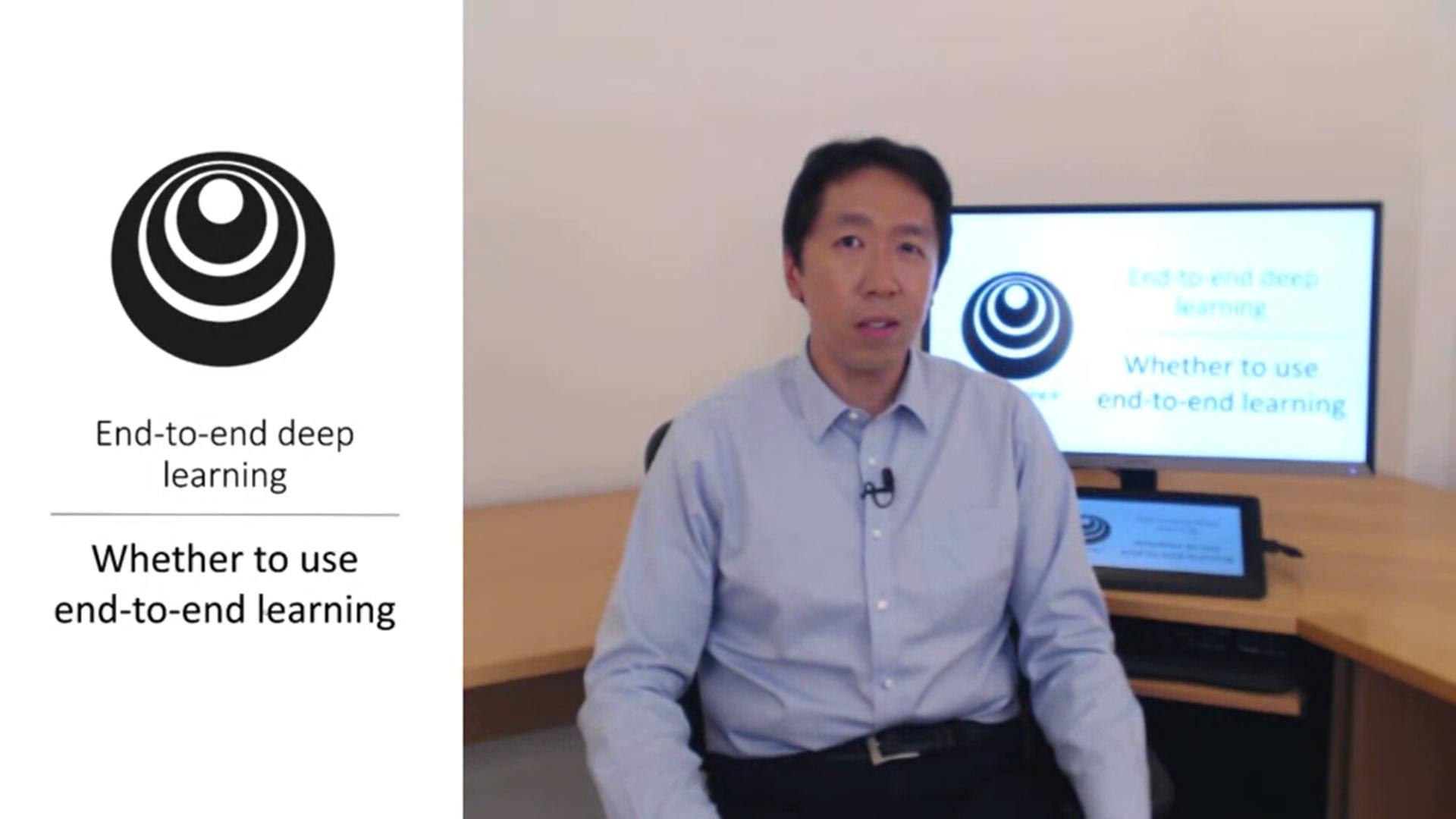
You have a lot of data for both of the subtasks. In contrast, if you try learning everything at the same time there is much less data. That is why in practice breaking down the task works better.



End-to-end deep learning works well for machine translation because it is possible to get a very large dataset.

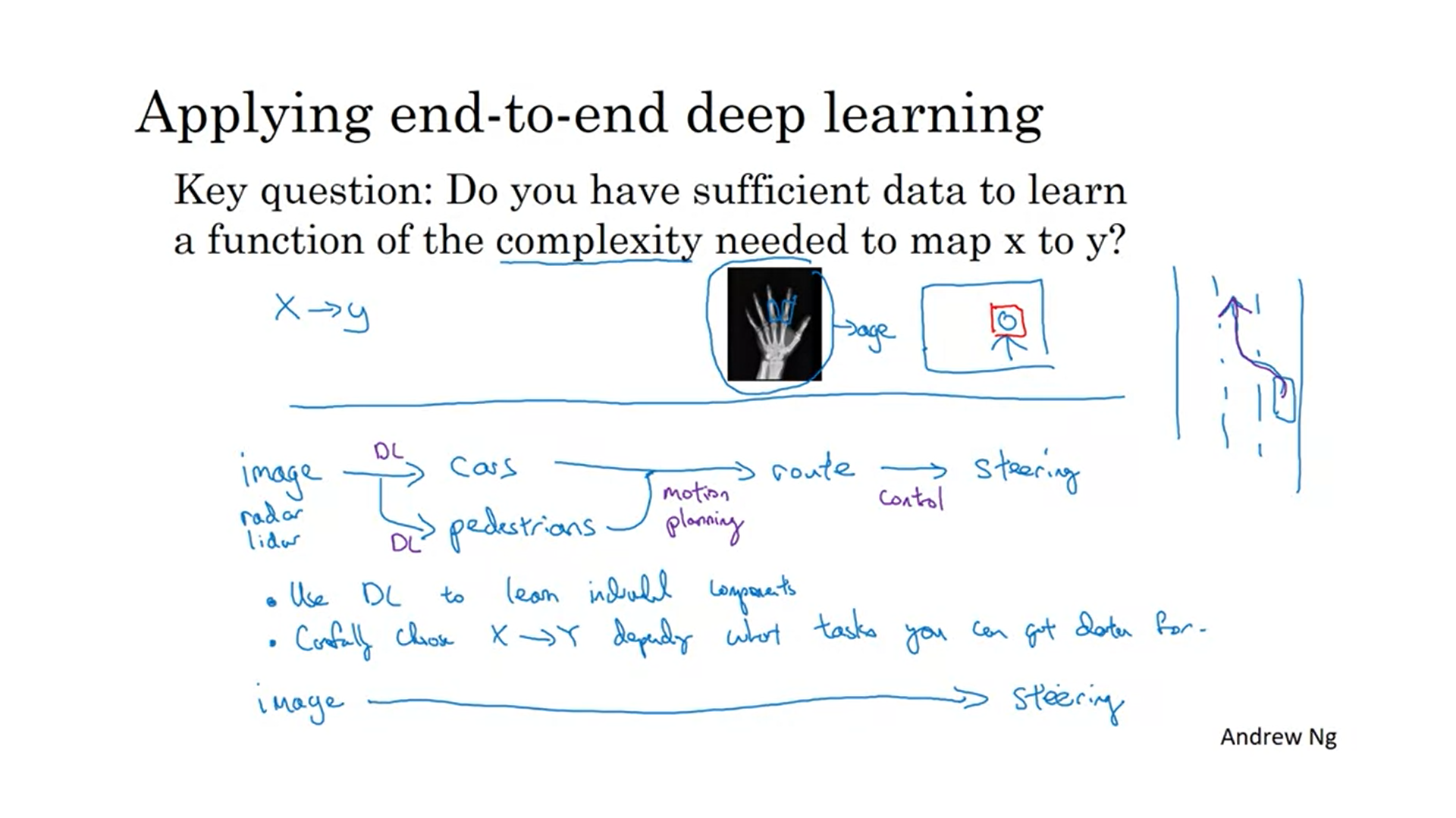
End-to-end deep learning does not work so well for estimating a child’s age. There is not too much data. By separating the tasks you need less data.

In the next video Andrew will share a more systematic approach to when end-to-end deep learning should be used and when it should not be used.



If the neural network is big enough it should figure out the most appropriate function from the data rather than reflecting human preperceptions (like phonemes). You do not want to force your algorithm to think in phonemes as representation.

Hand-designing a component can be a way to manually inject knowledge into the algorithm. That is not always a bad thing. Especially if you do not have much data.



Finding the bones or face in an image does not seem too hard of a problem. But mapping directly to the age of the child from the image of a hand is a much more complex problem.

Let’s look at a more complex example: autonomous driving.

One thing you can do (not end-to-end approach): take pictures what is in front of your car to find other cars, pedestrians etc. -> then plan your own route -> then generate steering commands. Usually only first part is done with DL, other parts with motion planning and control.

Conclusion: use dl for some individual components. Carefully choose what you want to learn depending on what you can get data for.