deeplearning.ai

Error Analysis

Carrying out error analysis

SUMMARY MANUAL ERROR ANALYSIS

- follows Bias/variance analysis.
- Manually analyse random mispredicted samples (from train set if bias problem, from dev if variance problem) to understand the cause of the problem
- Possible causes: noise, mislabeling, data distribution mismatch (e.g. white dogs not represented in training set and mispredicted in dev set).
- "CEILING": Allows estimating expected performance improvements by tackling a certain problem (=> prioritizing the bias/variance reduction techniques).

In tasks where humans still perform better than the model, a useful debugging strategy is ERROR ANALYSIS, that simply means manually analyzing the characteristics of the mispredicted examples.
This helps gaining insights on the sources of bad performance and estimating how much can the performance improve after either of those sources of error are tackled ("CEILING" performance improvement).

Look at dev examples to evaluate ideas

- Normally dev/test samples are manually analyzed
  90% accuracy
  → 10% error

Should you try to make your cat classifier do better on dogs?

EXAMPLE :- I want to improve the performance of a cat classifier

Error analysis:

→ 5-10 min

- Get ~100 mislabeled dev set examples.
- Count up how many are dogs.

"ceiling"

→ 5% → 5/100

10% → 9.5%

→ 50% → 50/100

- A colleague proposes to improve the mis-recognition of dogs
- Before going for that, manually check 100 (randomly selected) misclassified samples and see how many are dogs (many misclassified samples could actually be ginoffes).
  10% → 5%
  - if only 5% are actually dogs then even if I improve the dogs recognition, the CAT classifier performance would improve by 5%, for example from 10% error to 9.5% error
  => NOT WORTH IT!

Andrew Ng

Evaluate multiple ideas in parallel

When analyzing error one could (should) look at different factors at once and estimate the corresponding potential performance improvements. Then, one can tackle the most promising improvement.

Ideas for cat detection:

- Fix pictures of dogs being recognized as cats
- Fix great cats (lions, panthers, etc..) being misrecognized
- Improve performance on blurry images

Image	Dog	Great Cats	Blurry	Instagram	Comments
1	✓			✓	Pitbull
2			✓	✓	
3		✓	✓		Rainy day at zoo
⋮	⋮	⋮	⋮		
% of total	8%	43%	61%	12%	

In this case, we are better off trying to improve the recognition of blurry images.

deeplearning.ai

Error Analysis

Cleaning up Incorrectly labeled data

Incorrectly labeled examples

One possible (although unlikely) source of bias (and therefore error) in the data is mislabeling.

x







y 1 0 1 1 0 1 1

- In general, it's ok if the errors are not too many compared to the size of the dataset and are uniformly distributed ("random errors"). Conversely, systematic errors are problematic (eg all white dogs labeled as cats).

DL algorithms are quite robust to random errors in the training set.

Systematic errors

-> One can numerically estimate the impact of mislabeling during error checking by counting the number of mislabeled samples among the misclassified dev/test samples.

Andrew Ng

Error analysis

- The impact of data mislabeling can be also evaluated with manual error analysis.

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%

2.1% 1.9%

Goal of dev set is to help you select between two classifiers A & B.

Andrew Ng

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.
- Train and dev/test data may now come from slightly different distributions.

deeplearning.ai

Error Analysis

Build your first system quickly, then iterate

Speech recognition example

- Noisy background
  - Café noise
  - Car noise
- Accent
- Far fro
- Young
- Stutter
- …

Guideline:

Build your first system quickly, then iterate

- Set up dev/test set and metric
- Build initial system quickly
- Use Bias/Variance analysis & Error analysis to prioritize next steps.

also on training set?

deeplearning.ai

Mismatched training and dev/test data

Training and testing on different distributions

Summary on data mismatch

- only acceptable between train and dev/test set while it cannot be possible between dev and test set.
- Can be "involuntary" or "introduced" due to necessity
- "introduced" data mismatch:
  - if few data samples for task of interest.
  - introduce "similar" samples in training set.
  - prioritize interesting samples in dev/test sets.
  - expand bias/variance analysis with train/dev set (= portion of training set used as dev).
- Data mismatch can be reduced with synthetic data but synthetic data is often insufficient to model the variability of the actual problem.

- If we don't have much training data for a specific task, one can use the available data for the test/dev set and use data from a different distribution (but similar!) in the train set.
- The training set may not contain the "test distribution data" at all!

> data distrib final task = data distrib test/dev
> potentially ≠ data distrib train set

Cat app example

EXAMPLE: In the task I care about data from APP (blurry). If I only have few data examples from the domain of interest, then I have to prioritize those examples in the test/dev set.

Data from webpages



→ ≈ 200,000

Data from mobile app



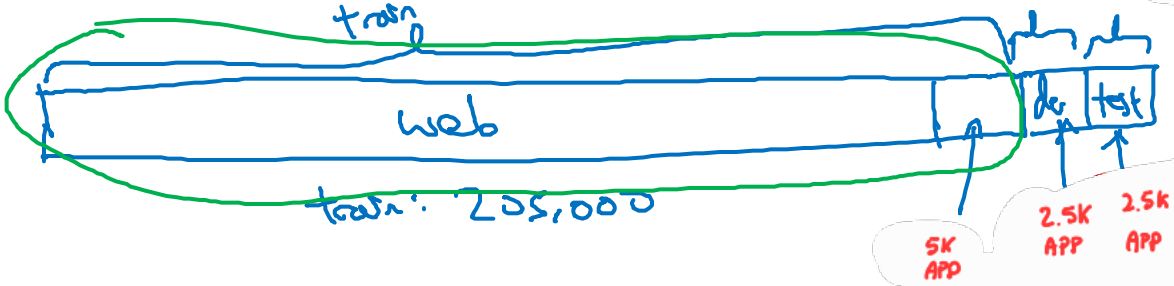
I can then build the training set (even entirely!) with data from a different distribution

→ ≈ 10,000

Option 1:

Both options are valid

Option 2:



Andrew Ng

Speech recognition example

Other example.

Speech activated rearview mirror

Training

- Any speech data, not necessarily for this use case!

Purchased data

Smart speaker control

Voice keyboard

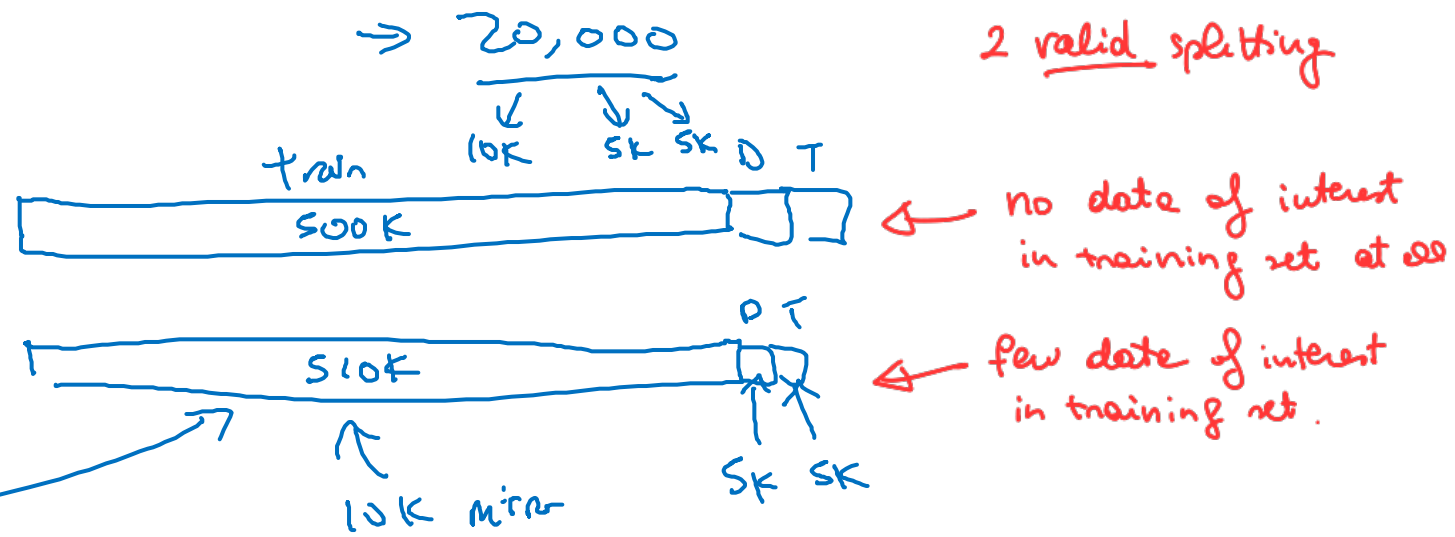
...

500,000 utterances

Dev/test

Speech activated rearview mirror

I CARE ABOUT THIS!



2 valid splitting

Having training and test data from different distributions complicates the bias/variance analysis. Now a gap between training and test error could depend both on the variance (overfitting the training set) and on testing on training/testing on different distribution (data mismatch).

deeplearning.ai

Mismatched training and dev/test data

Bias and Variance with mismatched data distributions

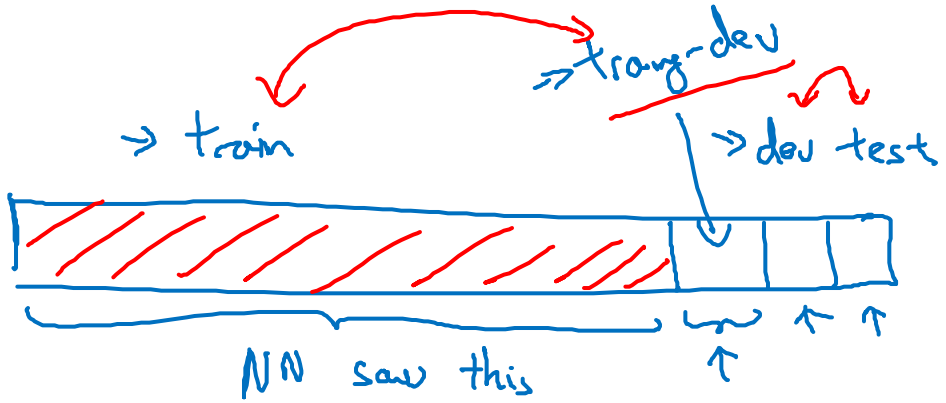
To disentangle and analyze these two aspects separately, one can create a train-dev set, only used for testing (like the dev set). Skip next slide.

Cat classifier example

Assume humans get ≈ 0% error.

Training error 1%
Dev error 10%

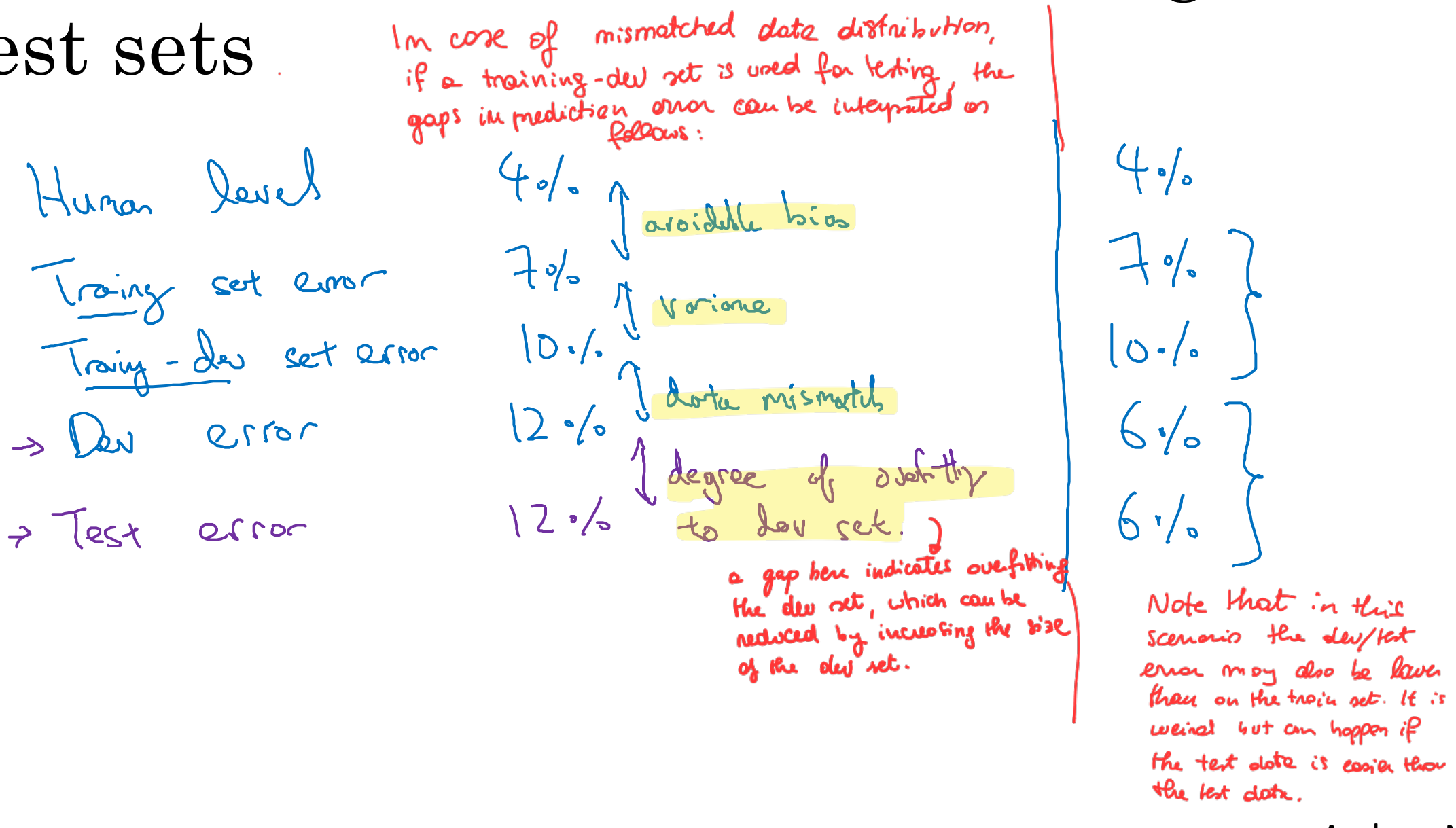
Training-dev set: Same distribution as training set, but not used for training



Training error 1% → 1% (variance)
→ Train-dev error 9% → 1.5% (data mismatch)
→ Dev error 10% → 10%

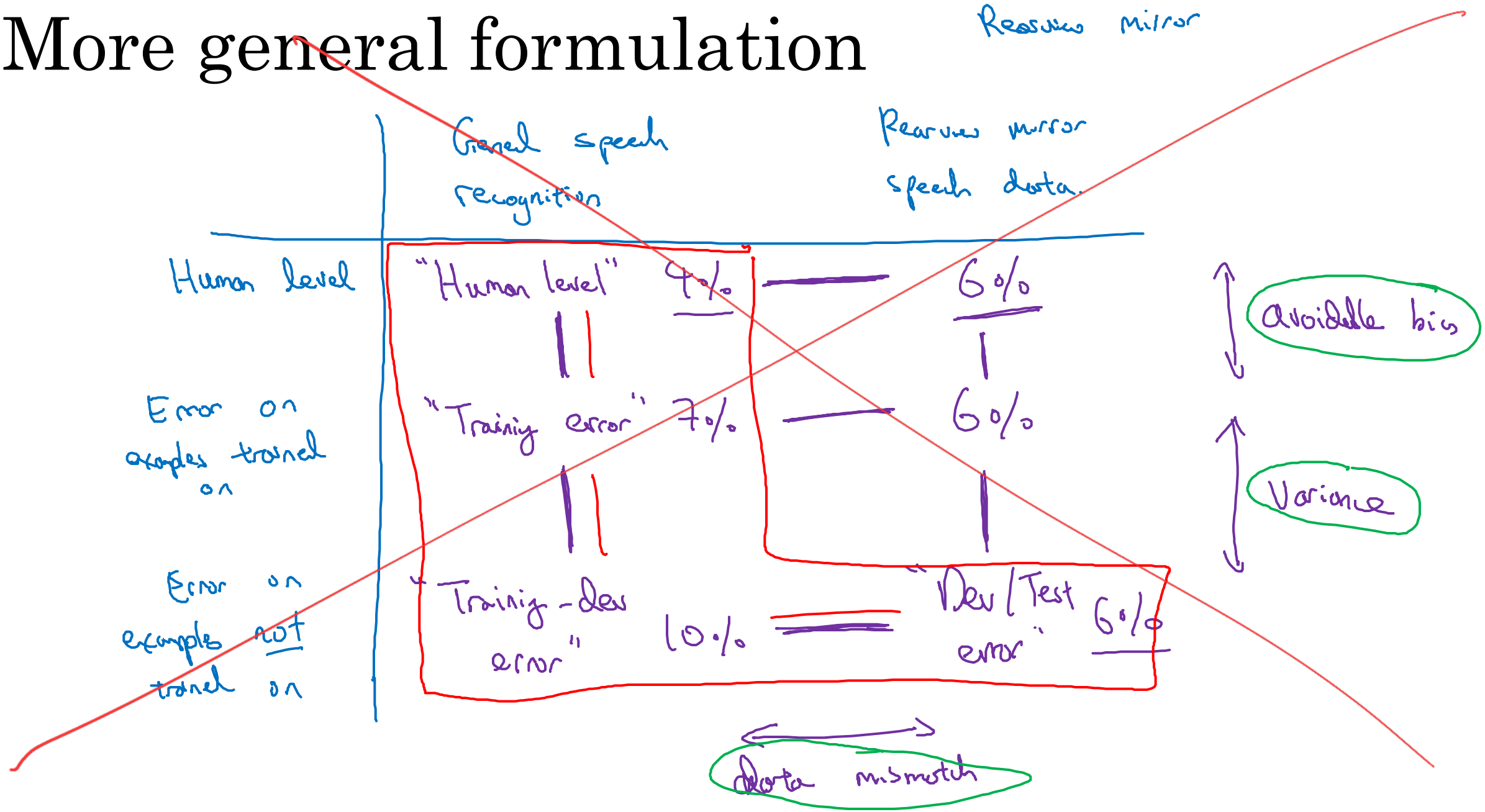
	Variance	
Human error	0%	Avoidable bias
Training error	10%	Avoidable bias / Variance
Train-dev error	11%	Variance / Data mismatch
Dev error	12%	
	Bias	Bias + Data mismatch

Bias/variance on mismatched training and dev/test sets



Andrew Ng

More general formulation



If we suspect a data mismatch between train and dev/test set (or we have introduced it by increasing the size of the training set with out of distribution data because we didn't have enough data to train on) then we might reduce its adverse effects with DATA AUGMENTATION via DATA SYNTHESIS.

deeplearning.ai

Mismatched training and dev/test data

Addressing data mismatch

Addressing data mismatch

The idea is to identify the type of data mismatch and synthesize new training data that makes the training set less different from the dev/test set.

- Carry out manual error analysis to try to understand difference between training and dev/test sets

E.g. noisy – car noise street numbers

- Make training data more similar; or collect more data similar to dev/test sets

E.g. Simulate noisy in-car data

Artificial data synthesis

For example, if I realize that most mispredicted samples are noisy => I suspect a data mismatch due to noise (train is clean, dev/test is noisy) => I can try to synthesize new training data by adding noise to it.

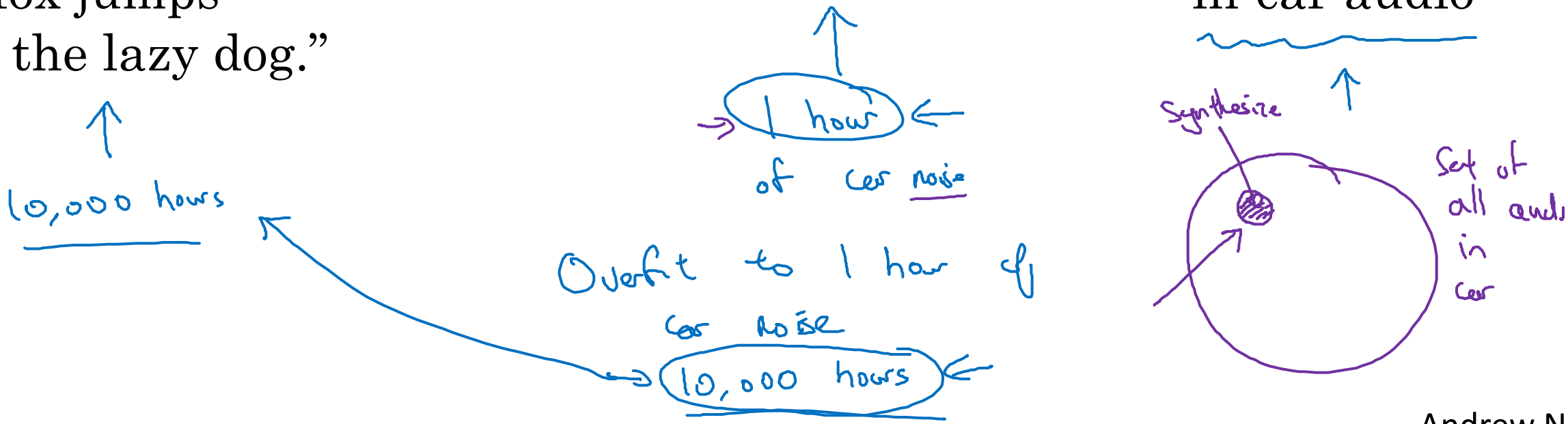
+

=

"The quick brown fox jumps over the lazy dog."

Car noise

Synthesized in-car audio



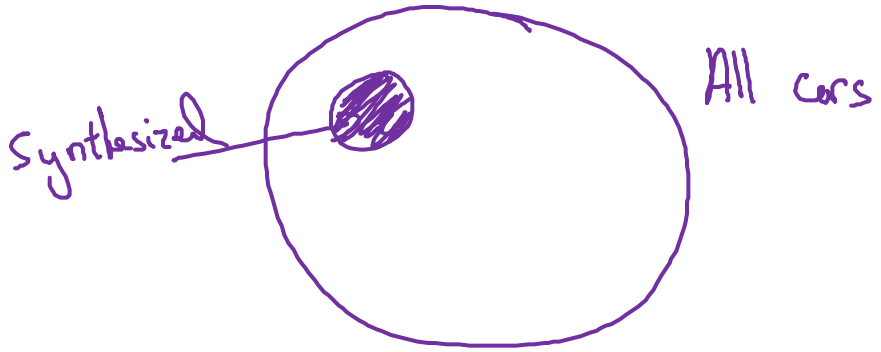
Andrew Ng

Artificial data synthesis

A common problem of data augmentation with synthetic data is that the synthetic data might still not represent the real data...

Car recognition:

≈ 20 cars



Transfer learning Task A -> Task B:

- You want to learn task B but do not have enough data
- Low level features of Task A also make sense for task B

Learning from multiple tasks

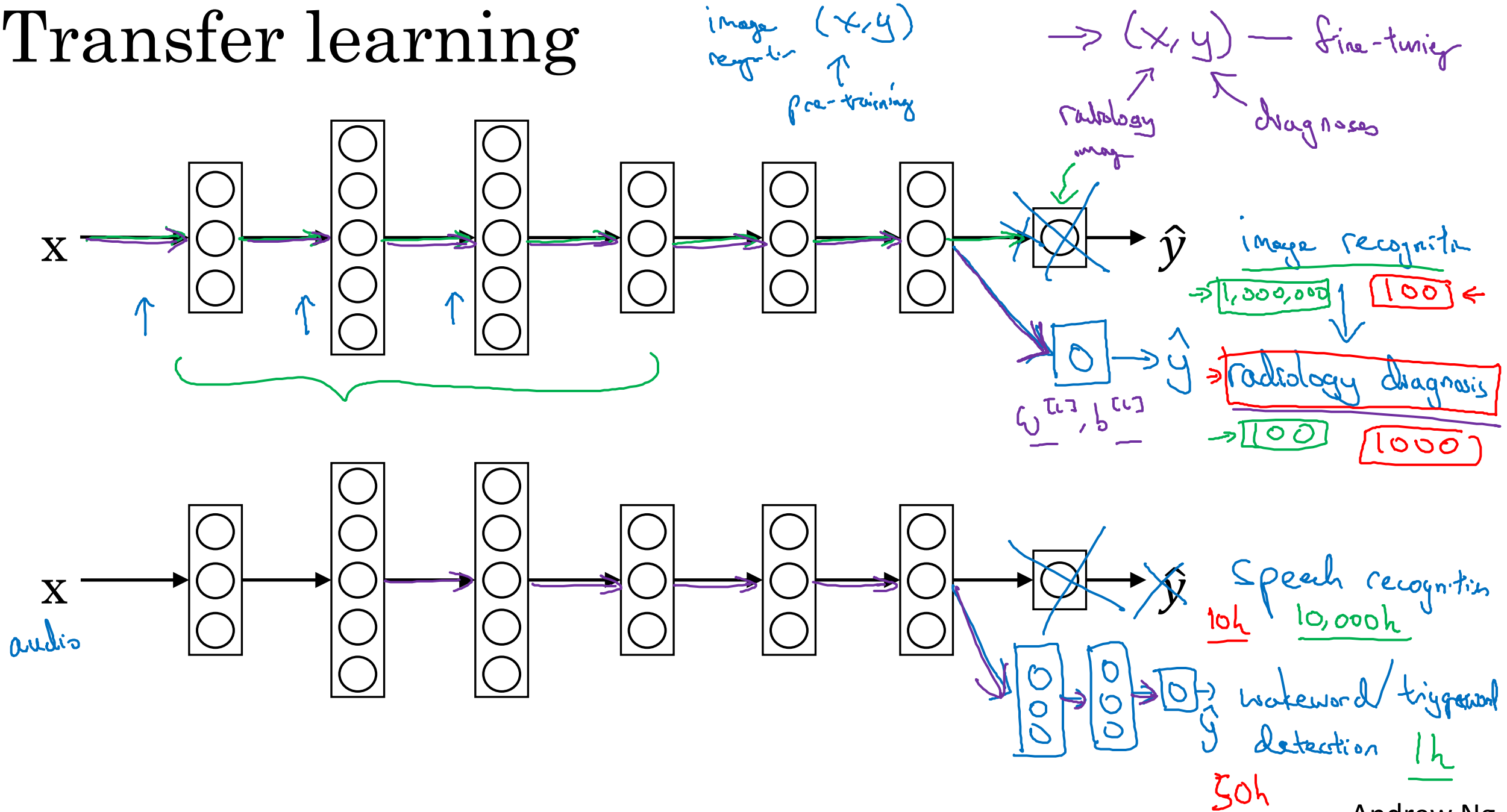
deeplearning.ai

Transfer learning

=> take a NN pretrained on task A and fine-tune the last layer with the data from task B.

- You can add and train more new layers on the head, or fine tune the whole network if you have enough data.

Transfer learning



Andrew Ng

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.

deeplearning.ai

Learning from multiple tasks

Multi-task learning

MULTITASK LEARNING (SUMMARY)

- used to train many tasks that share similar low level features but each task have few samples (comparable amount for each task).
- idea: label each sample with multiple labels and train only one NN.
- similar to transfer learning in that different tasks share the lower level features but different conditions (all tasks have similar no. of samples) and methodology.
- can handle missing labels using specific loss.
- all tasks receive mutual boost exploiting each other's data.
- less used than transfer learning.

- Simultaneously train a NN on different tasks
- useful if the tasks have similar low-level features
- Not all the labels (for each task) may be available for each sample. In this case, tasks can mutually benefit from each other in learning consistent low level features. → virtually, learning each task is boosted by the samples from all the other tasks

- The idea is just to label each sample with multiple labels and have a loss that takes into account all the labels.
- Differs from transfer learning especially because the no. samples for each task should be similar.

Simplified autonomous driving example

$x^{(i)}$

$y^{(i)}$ $(4,1)$

Pedestrians

Cars

Stop signs

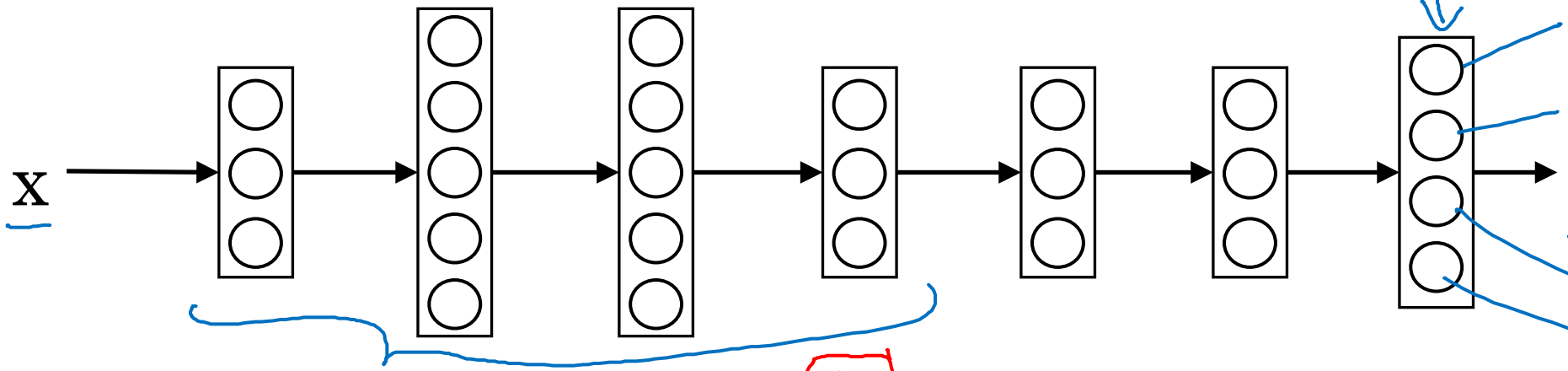
Traffic lights

$$y = \begin{bmatrix} | & | & | & & | \\ y^{(1)} & y^{(2)} & y^{(3)} & \cdots & y^{(m)} \\ | & | & | & & | \end{bmatrix}$$

$(4, m)$

Neural network architecture

- The loss must take into account all the available labels
- The loss must ignore missing labels



Loss: $\hat{y}^{(i)}$ $(4,1)$

$\Rightarrow \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{4} \mathcal{L}(\hat{y}_j^{(i)}, y_j^{(i)})$

Sum only over values of j with 0/1 label.

$\mathcal{L}(\hat{y}_j^{(i)}, y_j^{(i)})$

→ Usual logistic loss

$-y_j^{(i)} \log \hat{y}_j^{(i)} - (1-y_j^{(i)}) \log (1-\hat{y}_j^{(i)})$

Unlike softmax regression:

One image can have multiple labels

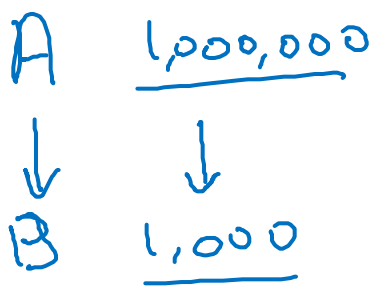
Multi-task learning ←

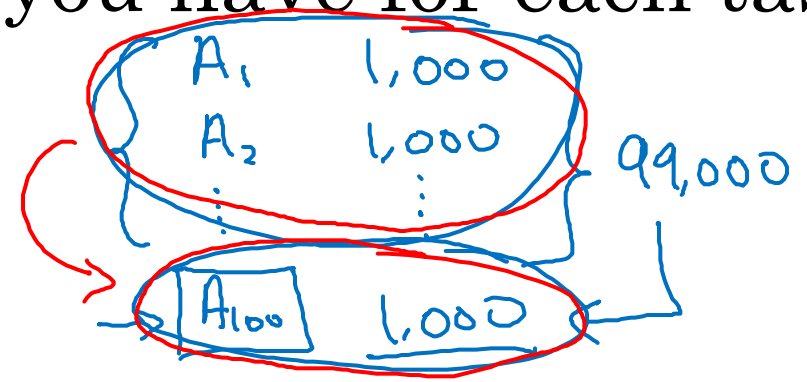
$$Y = \begin{bmatrix} 1 & 1 & \cdots & 0 & \cdots & ? & \cdots \\ 0 & 1 & & 1 & & 1 & \\ ? & ? & & 1 & & ? & \\ ? & ? & & 0 & & ? & \end{bmatrix}$$

Andrew Ng

When multi-task learning makes sense

- Training on a set of tasks that could benefit from having shared lower-level features.
- Usually: Amount of data you have for each task is quite similar.





- Can train a big enough neural network to do well on all the tasks.

- training NN on a complex task as opposed to combining multiple NN trained on intermediate problems.

deeplearning.ai

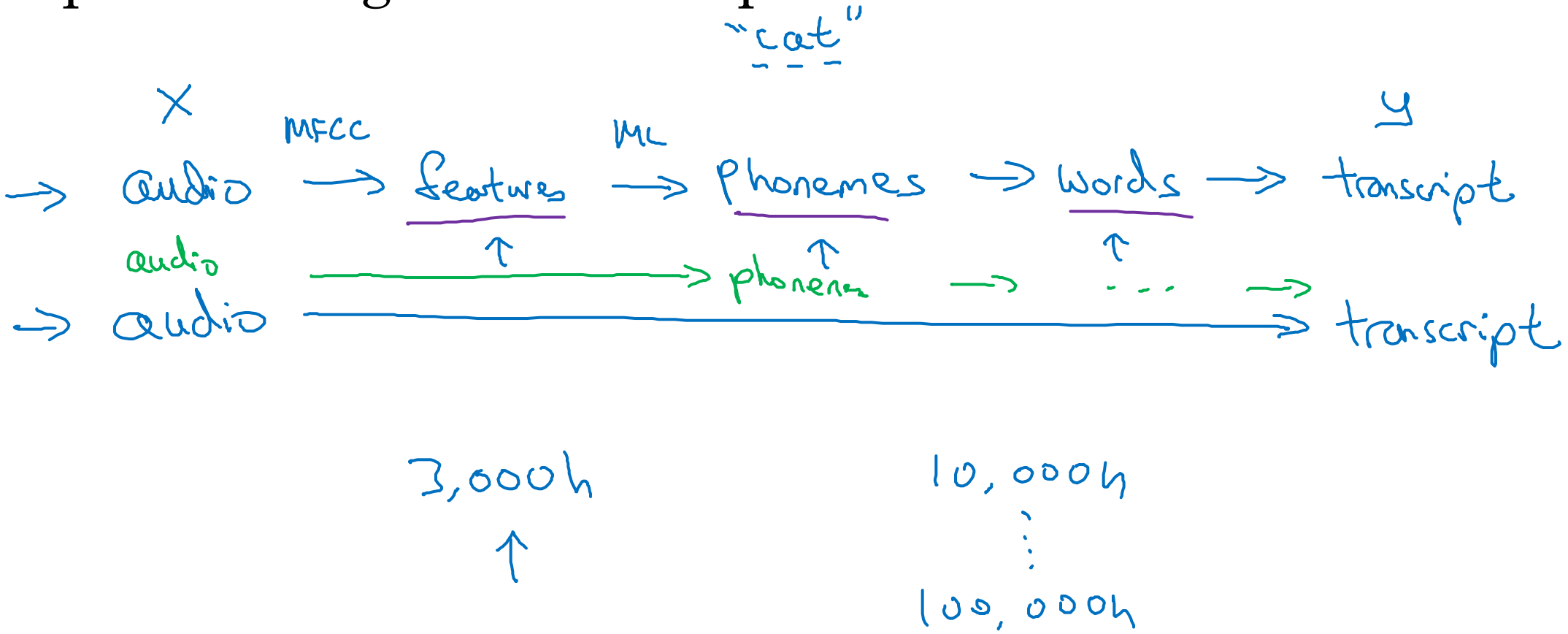
End-to-end deep learning

What is end-to-end deep learning

- solving intermediate problems is referred when more data is available for the intermediate problems or when one wants to have control (eg. for safety reasons) on the intermediate steps.

What is end-to-end learning?

Speech recognition example



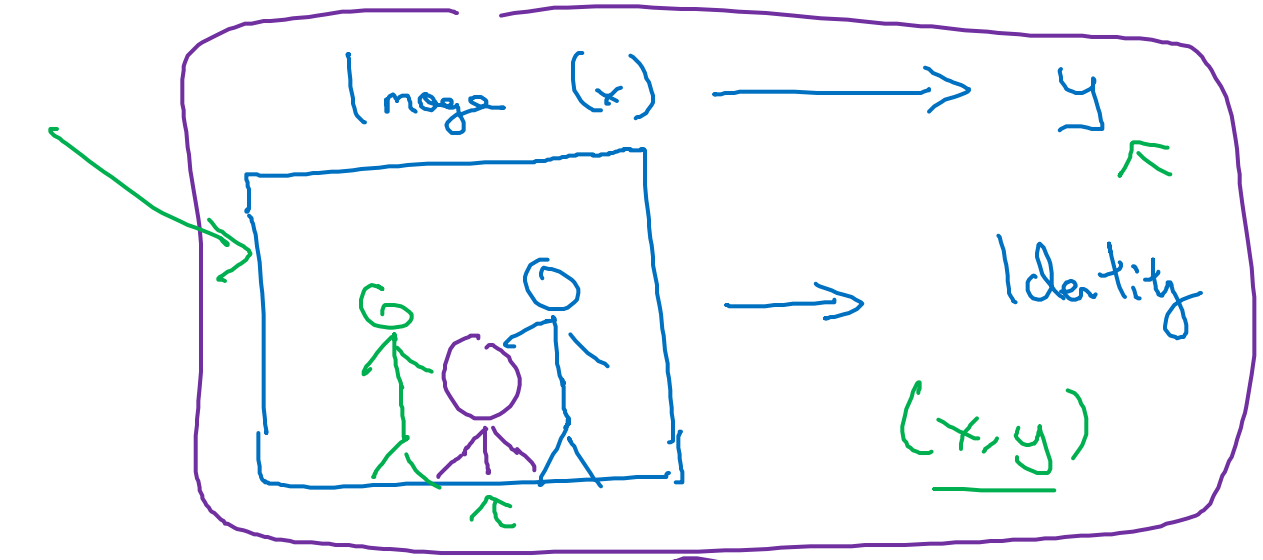
Face recognition

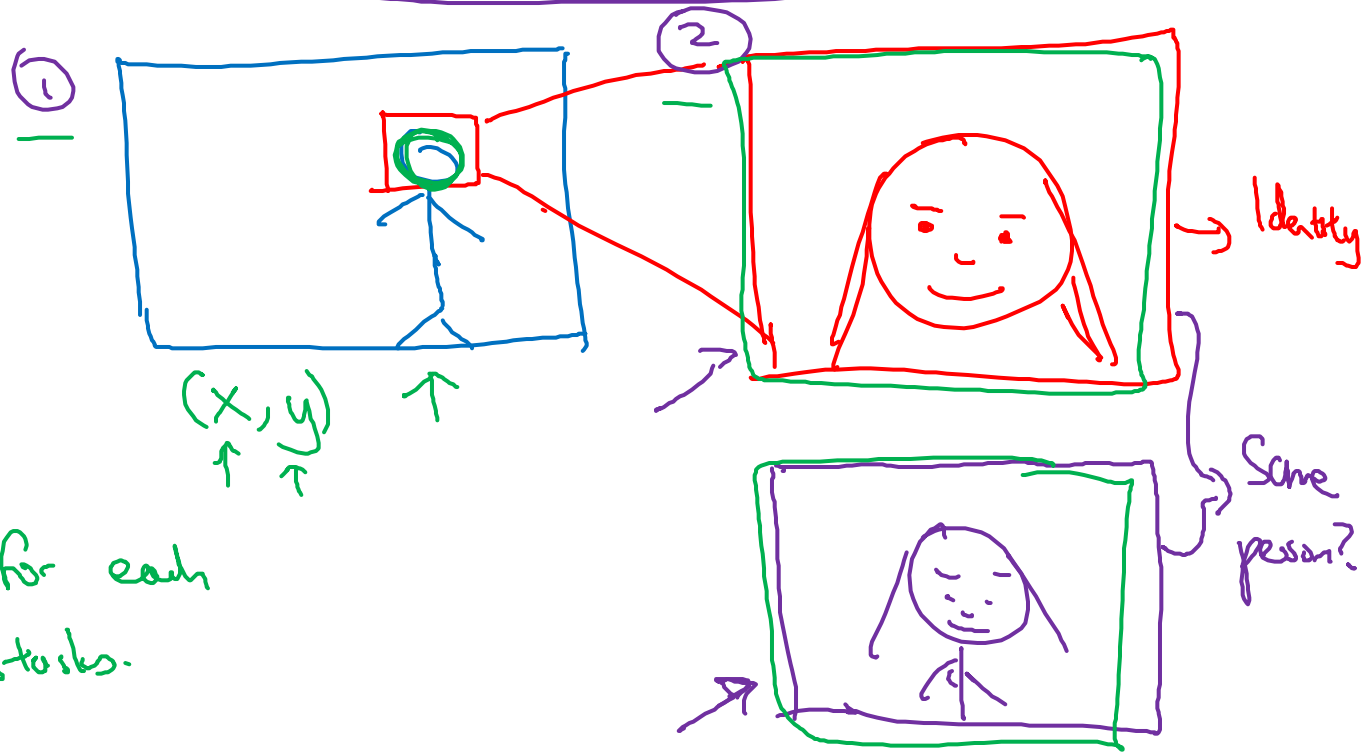
Normally solved sequentially :

1 - face detection

2 - identity checking

[Image courtesy of Baidu]

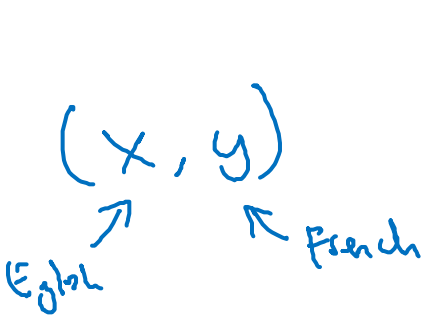


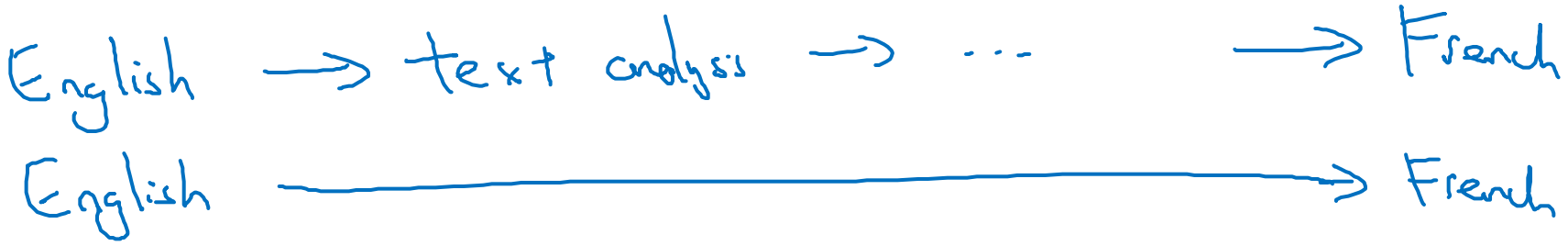


Have data for each of 2 subtasks.

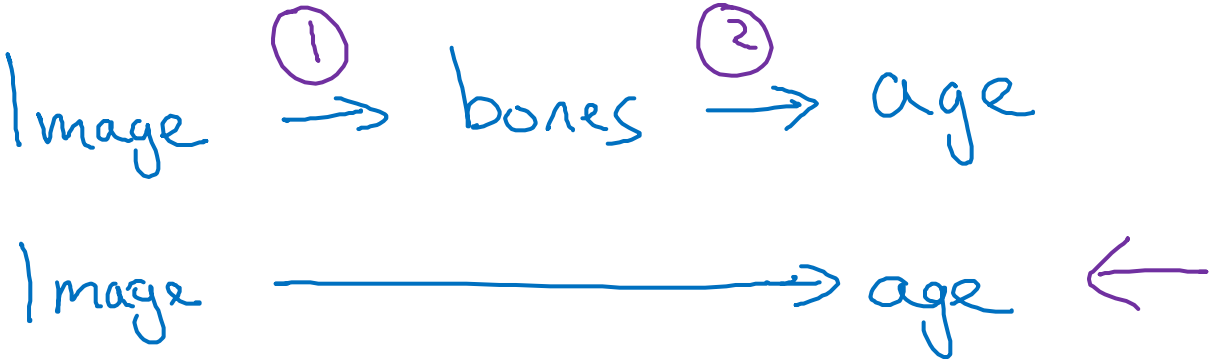
More examples

Machine translation





Estimating child's age:



deeplearning.ai

End-to-end deep learning

Whether to use end-to-end learning

Pros and cons of end-to-end deep learning

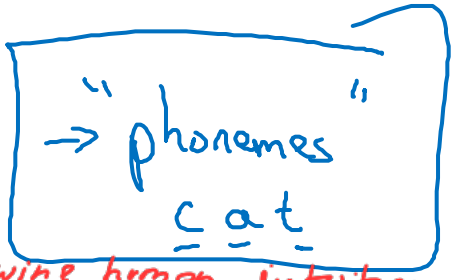
Pros:

- "Let the data speak"
- Less hand-designing of components needed

allows the NN to freely find the best function to express the end-to-end relation, instead of

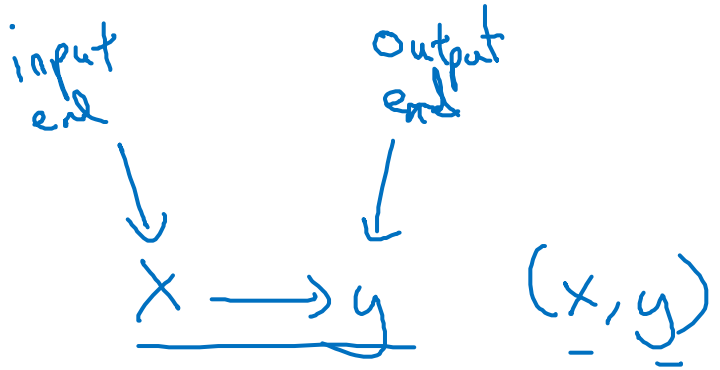
$X \longrightarrow y$

constraining the system to learn subproblems that are created following human intuition (which may not be correct).



Cons:

- May need large amount of data
- Excludes potentially useful hand-designed components



Data.

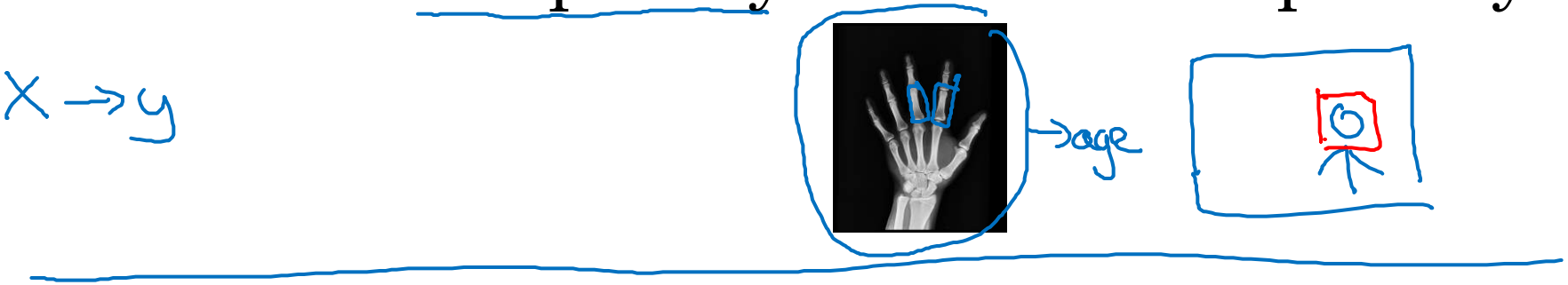
Hand-design.

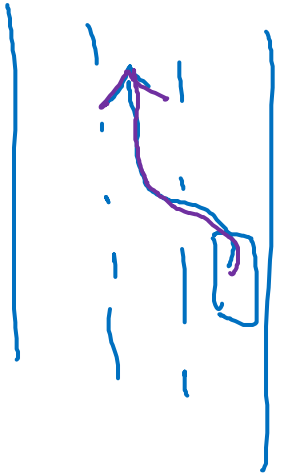
→ it's a way to inject manual knowledge into the problem.

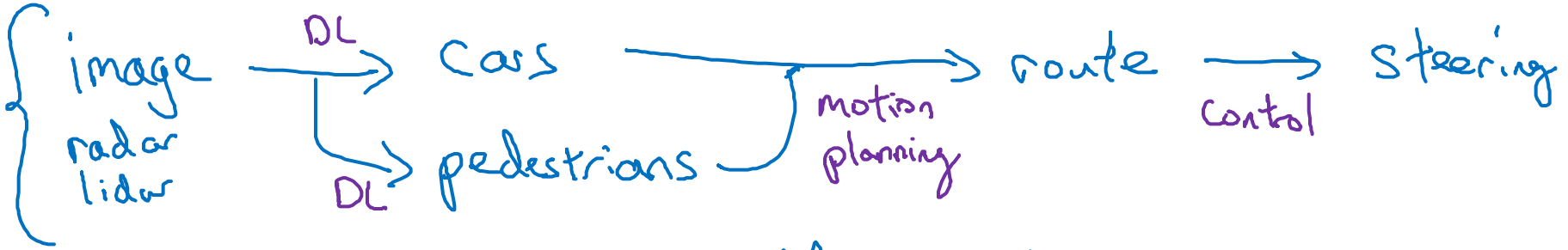
Andrew Ng

Applying end-to-end deep learning

Key question: Do you have sufficient data to learn a function of the complexity needed to map x to y?







- Use DL to learn individual components
- Carefully choose X → Y depending what tasks you can get data for.

Andrew Ng