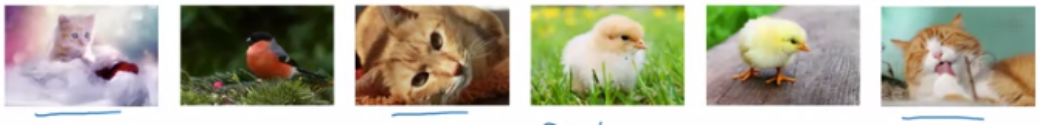
Why ML Strategy

Motivating example



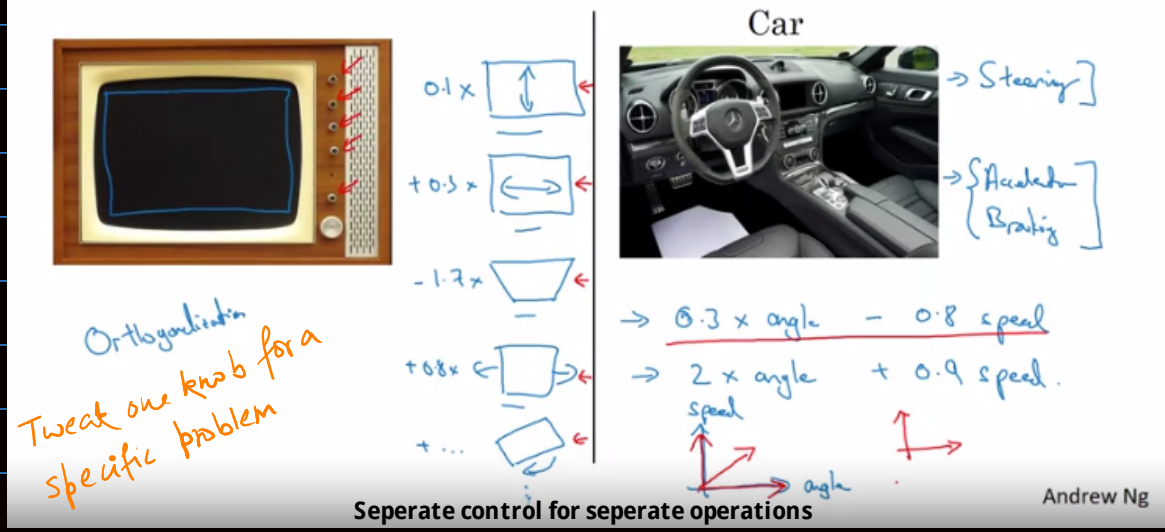
90%

Ideas:

- Collect more data
- Collect more diverse training set
- Train algorithm longer with gradient descent
- Try Adam instead of gradient descent
- Try bigger network
- Try smaller network
- Try dropout
- Add L_2 regularization
- Network architecture
  - Activation functions
  - # hidden units
  - ...

Andrew Ng

Orthogonalization



Seperate control for seperate operations

Andrew Ng

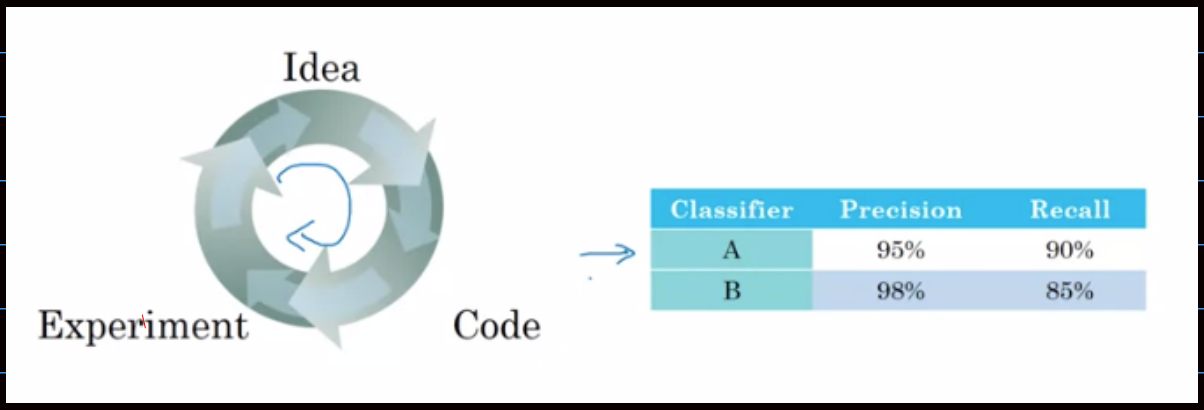
Chain of assumptions in ML

1 Fit training set well on cost fun^n [human level performance]

2. Fit dev set well on cost fun's.
3. Fit test set well on cost fun's
4. Performs well in real world

Single number evaluation metric.

Cat - No Cat Classifier



Classifier	Precision	Recall
A	95%	90%
B	98%	85%

Precision : A = When classifier A says "it is a cat" — there is 95% chance that that sample really is a cat.

Recall : B = What % of actual cats are correctly recognized

Evaluation Metric.
Precision or Recall?

Rather use a combination of Precision & Recall → F1 Score

F1 Score = "Average of Precision (P) & Recall (R)" {Informal}

$$= \frac{2}{\left(\frac{1}{P} + \frac{1}{R}\right)}$$ {Formal} → Harmonic Mean

Well defined Dev Set + Single Evaluation Metric ≈ Helps to identify which classifier is better

Example:

Algorithm	US	China	India	Other	Average (say error)
A	3%	7%	5%	9%	6%
B	5%	6%	5%	10%	6.5%
C	2%	3%	4%	5%	3.5% → lowest error.
D	5%	8%	7%	2%	5.25%
E	4%	5%	2%	4%	3.75%
F	7%	11%	8%	12%	9.5%

chosen single value evaluation metric

Satisficing & Optimizing Metric.

Classifier	Accuracy (optimizing)	Running time (satisficing)
A	90%	80ms
B	92%	95ms
C	95%	1,500ms

cost = accuracy - 0.5 running time

or

Maximize accuracy
Subject to running time $\leq$ 100ms
{i.e as long as it is < 100ms it is like the same}

N metrics : 1 optimizing

N-1 satisficing

Train/Dev/Test Sets

* The dev & test set should always come from the same distribution

Example:

Remedy

Regions: → Classes

- US
- UK
- Other Europe
- South America
- India
- China
- Other Asia
- Australia

dev

test

✗ a bad idea to have dev & test set from different example

→ Shuffle the data & then split into test & dev sets
Both sets then stays in the same distribution

Size of the dev & test set

Typical Splits = 70% | 30% , 60% | 20% | 20%

If dataset size = 10,00,000
then we can use split as small as 98% | 1% | 1%

Size of test set : Should be just big enough to give high confidence in the overall performance of your system

* Some times only train & dev sets are used

When to change dev/test sets & metrics

Metric: classification error

Algorithm A: 3% error → pornographic

✓ Algorithm B: 5% error

Huge Metric + Dev : Prefer A

You | users : Prefer B

If we are not satisfied with our evaluation metric Go for a new design metric. for our task

$$\text{Error} : \frac{1}{\sum w^{(i)}/m} \sum_{i=1}^{m_{dev}} w^{(i)} \mathbb{1}\{y_{pred} \neq y^{(i)}\}$$

↑ predicted value (0|1)

$$w^{(i)} = \begin{cases} 1 & \text{if } x^{(i)} \text{ is not porn} \\ 10 & \text{if } x^{(i)} \text{ is porn} \end{cases}$$

Orthogonalization for cat pictures: anti-porn

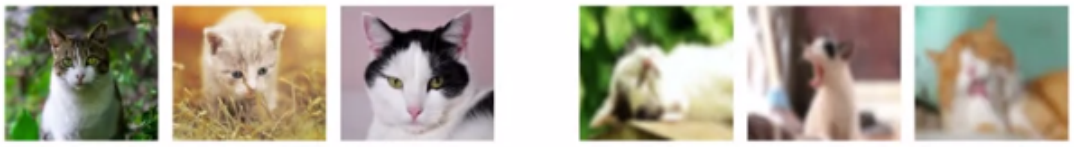
1. So far we've only discussed how to define a metric to evaluate classifiers.
2. Worry separately about how to do well on this metric.

Another example

Algorithm A: 3% error

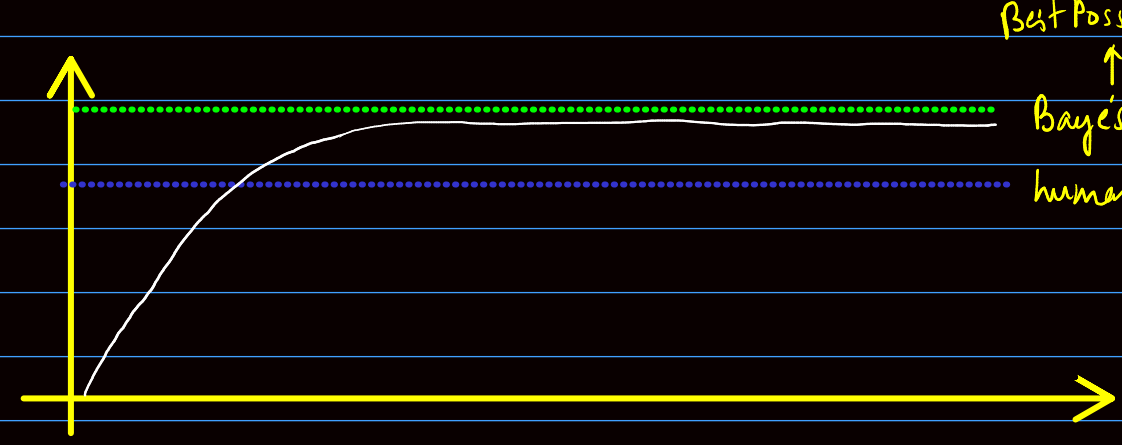
✓ Algorithm B: 5% error ←

→ Dev/test → User images



If doing well on your metric + dev/test set does not correspond to doing well on your application, change your metric and/or dev/test set.

Why human level performance?



$X \longrightarrow Y$

- audio $\longrightarrow$ transcript
- image $\longrightarrow$ cat (0/1)

Why compare to human-level performance

Humans are quite good at a lot of tasks. So long as ML is worse than humans, you can:

- Get labeled data from humans. (x, y)
- Gain insight from manual error analysis: Why did a person get this right?
- Better analysis of bias/variance.

Avoidable bias:

Cat classification

Bayes Error	≈ Humans	1%
	Training Error	8%
	Dev Error	10%

↕ May be we can do better on the training set

Focus on bias

Diff b/t Bayes Error & Training Error = Avoidable Bias

Diff = Variance

Humans 7.5%

Training Error 8%

Dev Error 10%

↑ Reduce ↓ variance

Human level error as a proxy for Baye's Error

Understanding Human level performance:

Human-level error as a proxy for Bayes error

Medical image classification example:

Suppose:

(a) Typical human ..... 3 % error

(b) Typical doctor ..... 1 % error

(c) Experienced doctor ..... 0.7 % error

→ (d) Team of experienced doctors .. 0.5 % error

What is "human-level" error?

Bayes error $\leq$ 0.5%

∴ Human level. Error = 0.5

Error Analysis Example

Human level error / Proxy for Bayes Error 1% / 0.7%

↕ 4.5%

Avoidable Bias

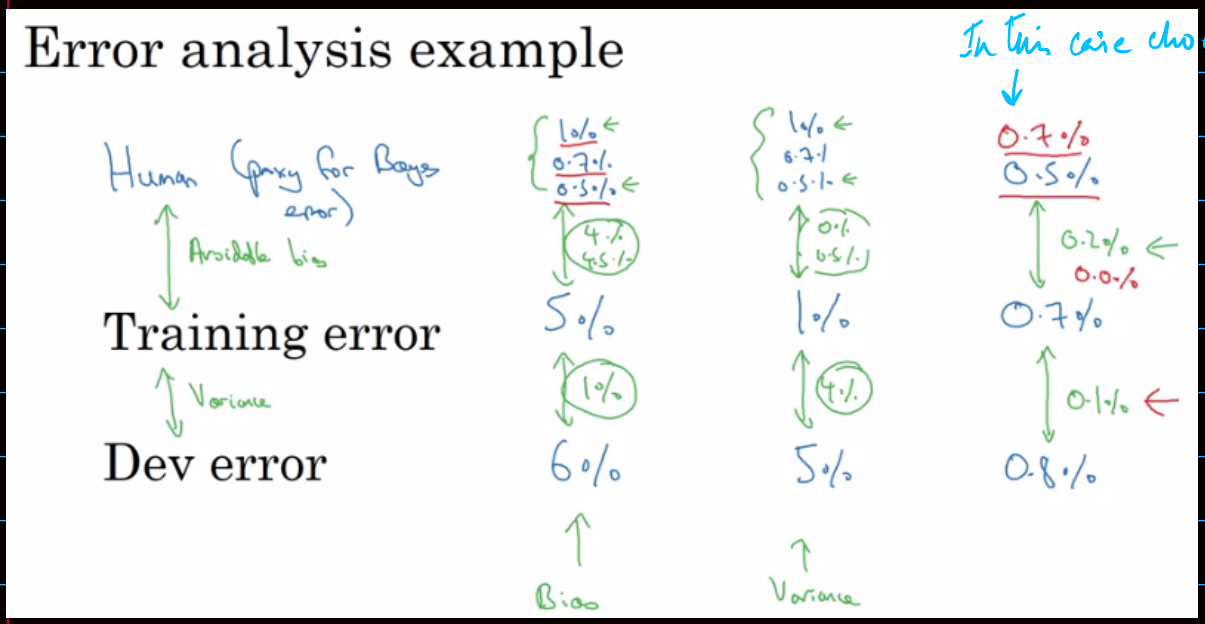
Training Error 5%

↕ 1%

Variance.

Dev Error 6%

Focus on Bias Reduction (∵ 4.5 > 1%)



In this case choosing human level error is imp. as if we choose 0.7 we might miss something

* Bias, Variance analysis is very important & tricky.
*** Baye's errors / Human error is not always 0%.

Surpassing Human-level performance.

Team of Humans	0.5%	0.5%	
One Human	1%	1%	0.1
Training Error	0.6%	0.3%	
Dev Error	0.8%	0.4%	0.2

Reduce Variance

?? Here either the data is overfitted or a team of humans are not able to do well as the computer

But does that mean that the computer is wrong? or the human?

Problems where ML significantly surpasses human-level performance

- Online advertising
- Product recommendations
- Logistics (predicting transit time)
- Loan approvals

Reasons:
→ Standard data available.
→ Lots of available data.
→ Not natural perception

The two fundamental assumptions of supervised learning

1. You can fit the training set pretty well.

   Low avoidable bias

2. The training set performance generalizes pretty well to the dev/test set.

   Low Variance

To reduce ① Avoidable Bias —
1. Train bigger model
2. NN Architecture / Hyperparameter Search (CNN, RNN)
3. Train Longer / Better Optimization Algorithms (Momentum, RMS Prop, Adam)

② Variance

1. More data
2. Regularization (L2, Dropout, Data Augmentation)
3. NN Architecture / Hyperparameters Search

Week 2

Carrying out error analysis:

Look at dev examples to evaluate ideas

90% accuracy
10% error

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

Error analysis:

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

let 5% of mislabelled examples are dog. i.e 5/100 of them are dog

∴ Even if I solve the dog problem, still there are a lot mislabelled examples left.

If 50% of the mislabelled examples are dog then we can spend time on correcting the dog error.

Evaluate multiple ideas in parallel

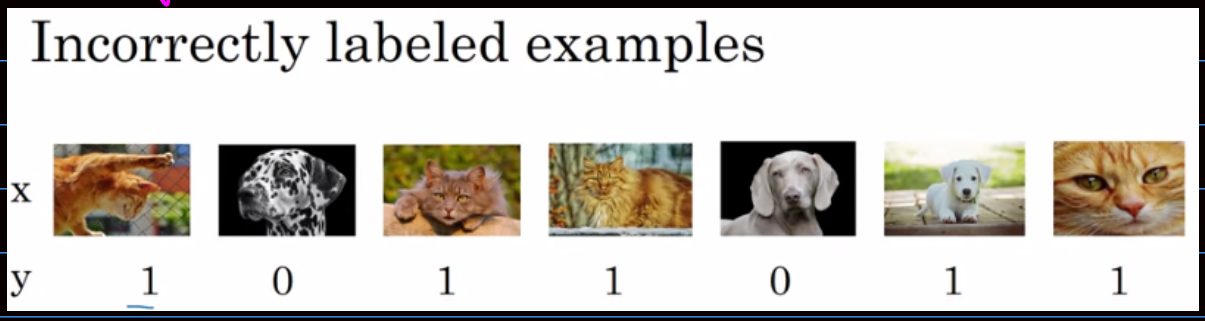
Ideas for cat detection:

- 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%	

Such a table helps us evaluate which problems to focus on more to reduce error. Observing the percentages of category based mislabelled data.

Cleaning up incorrectly labelled data.



Incorrect labels

DL algorithms are quite robust to random errors in the training

For training set

Near random errors → not much problem

But if systematic errors → like all white dogs are labelled cats – we have to address that.

For test/dev set,

add this

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 to incorrect labels :	0.6%	0.6%
Errors due to other causes :	9.4%	1.4%
Incorrect labelling :	Can be ignored	Can't be ignored

Dev set is important as it helps us to select bet^n 2 classifiers A & B

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. (98.6) (2%)
- Train and dev/test data may now come from slightly different distributions.

* Build your system quickly & then iterate?

Speech recognition example

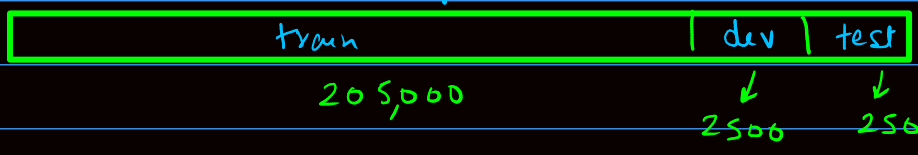
- Noisy background
  - Café noise
  - Car noise
- Accented speech
- Far from microphone
- Young children's speech
- Stuttering
- ...

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

Training & Testing on different distributions

Distribution ≈ 200,000 ≈ 10,000

Not good ✗ Option 1: Take all images from all distributions and shuffle them before splitting them into train, dev & test set

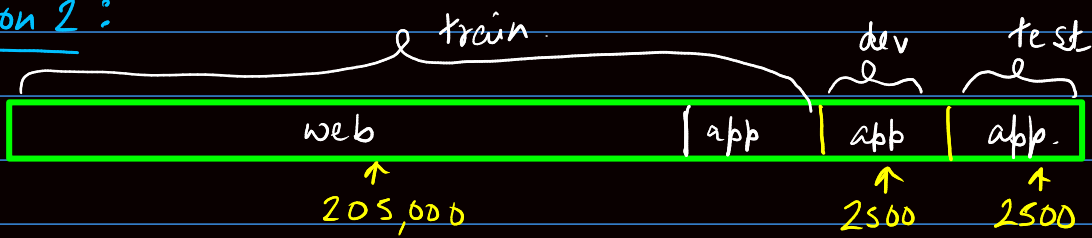


mobile app images might not be generalized

On avg → 2381 → web
← 119 → mobile app

Again a bad distribution

Better than previous → Option 2:



App images will now be classified well, although the train & dev/test distributions are different.

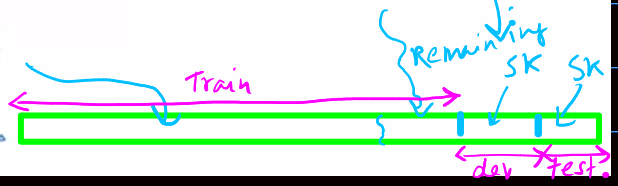
Speech Recognition Example

Speech activated rearview mirror

Training	Dev/test
Purchased data (x, y)	Speech activated rearview mirror
Smart speaker control	
Voice keyboard	
...	

500,000 utterances

20,000



Bias & Variance with mismatched data distributions

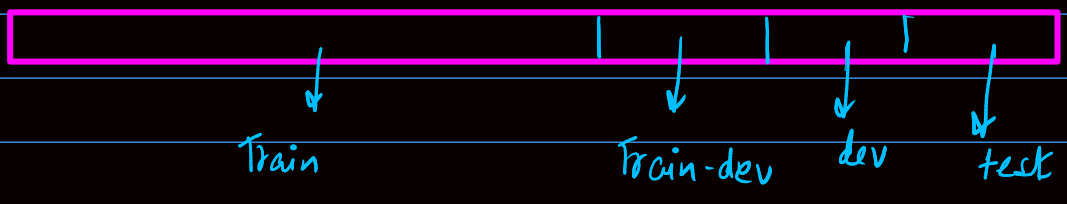
Cat Classifier Example

Let human error ≈ 0%.

Training Error → 1%.

Dev Error → 10%.

} This gap either shows variance problem or maybe the training & the dev sets have entirely different distributions



Training Dev Set : Same distribution as the training set, but not used for training.

Let for the previous example,

		↗ Bayes Error,		
		0%	↕ high bias	0%
Training Error	1%	1%	10%	problem 10%
Training-Dev Set (Variance Issue ↕)	9%	1.5% (↕ X problem)	11%	11%
		↕ data mismatch problem →		
Dev Set	10%	10%	12%	20%
Test Set (degree of overfitting to dev set ↕)	10%			

'%'s are assumed *

Distributions	General Speech Recognition	Rear view Mirror Speech Data
Human level	"Human level" 4%	6%
	↕ Avoidable Bais	↕
Error on examples trained on	'Training error' 7%	6%
	↕ Variance	↕
Error on examples NOT trained on	'Training-dev' 10%	Dev-Test 6%
	↔ Data Mismatch	error

Addressing data mismatch

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

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

* Solution:

Artificial data synthesis

Human Speech + Car Noise = ✓ more data created

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

Car noise

Synthesized in-car audio

let, 10,000 hrs of clean noiseless data

only 1 hour of car noise

Combination might sometimes lead to overfitting to 1 hr of car noise.

We are actually synthesizing data from a very small subset.

* Example 2:

Car recognition:

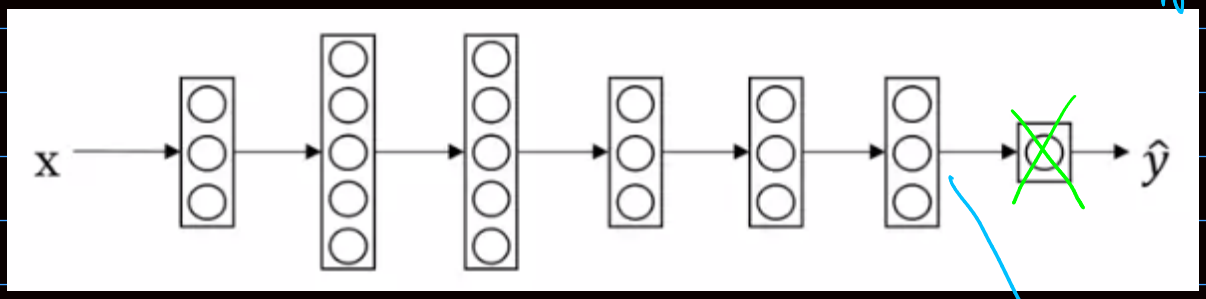
≈ 20 cars

Transfer Learning

In Transfer Learning we can take knowledge the neural network has learned from one task and apply that knowledge to a separate task. For example we have the neural network learn to recognize objects like cats and then use that knowledge or use part of that knowledge to help you do a better job reading x-ray scans.

Image recognition

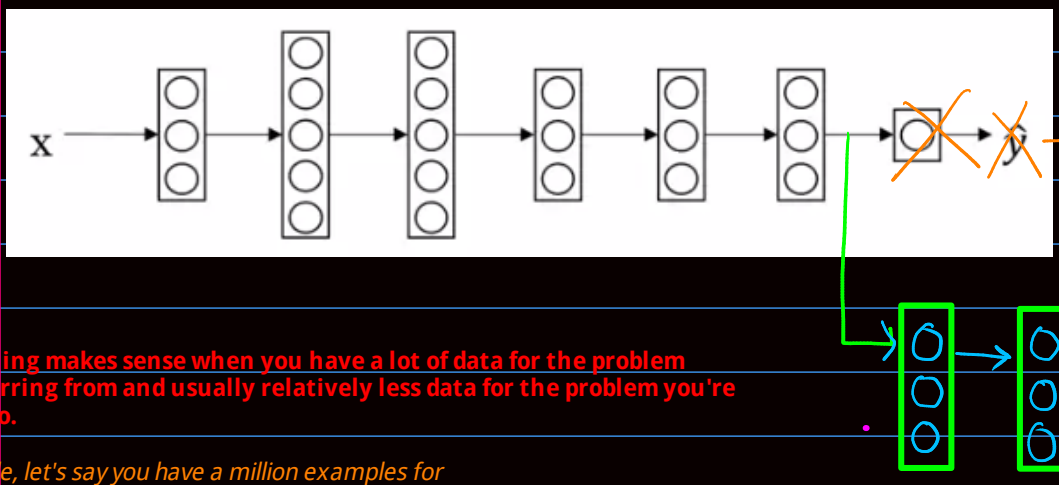
(x, y)



Retrain. either only the last layer which is changed or if we have more data retrain the entire model. (all parameters)

* Initial Phase of training on the basis of image recognition initialization is called → Pre-training.
* After updating all the weights & training on radiology data is called fine tuning

Example 2



Transfer learning makes sense when you have a lot of data for the problem you're transferring from and usually relatively less data for the problem you're transferring to.

So for example, let's say you have a million examples for ~~image recognition task. So that's a lot of data to learn a lot of low level features~~ or to learn a lot of useful features in the earlier layers in neural network. But for the radiology task, maybe you have only a hundred examples. So you have very low data for the radiology diagnosis problem, maybe only 100 x-ray scans. So a lot of knowledge you learn from English recognition can be transferred and can really help you get going with radiology recognition even if you don't have all the data for radiology.

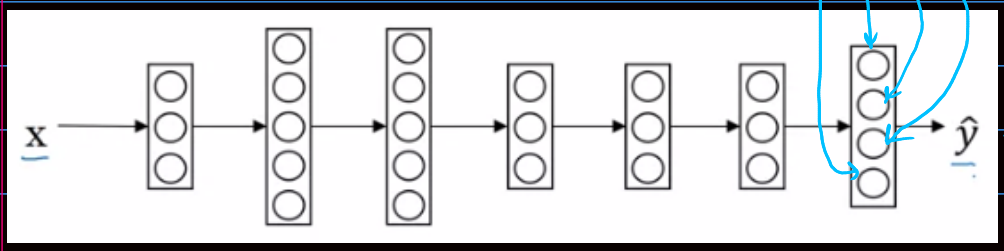
Transfer Learning makes sense

→ Task A & B have same i/p x.
→ We have a lot more data for Task A than Task B
→ Low level features from A could be helpful for learning B

Multi Task Learning

$y^{(i)}$

$y =$ Pedestrians 0
Cars 1
Stop Signs 1
Traffic Lights 0



$$\text{Loss} = \hat{y}^{(i)} \quad (4,1)$$

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

↳ Usual Logistic Loss

Unlike Softmax regression → here one image can have multiple labels
(Multilabel classification)
✗ NOT Multi class classification

This is called Multi task Learning

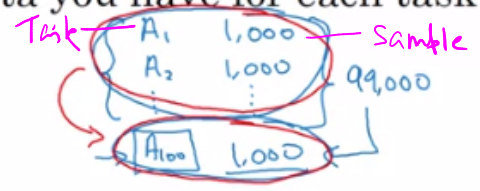
$$Y = \begin{bmatrix} 1 & 1 & 0 & \cdots \\ 0 & ? & ? & \cdots \\ ? & 1 & ? & \cdots \\ 0 & ? & 0 & \end{bmatrix}$$

If even works when certain labels are unmarked or absent. { Summation sums only over values of $j = 0$ or 1 }

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.

Transfer Learning → A 1,000,000 ↓ B 1,000



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

* Alternate to Multitask learning is to train as many neural networks as the no. of tasks.

* Used much rare.
Object Detection use Multi task Learning

End to End Deep Learning

What is end-to-end learning?

Speech recognition example

"cat"

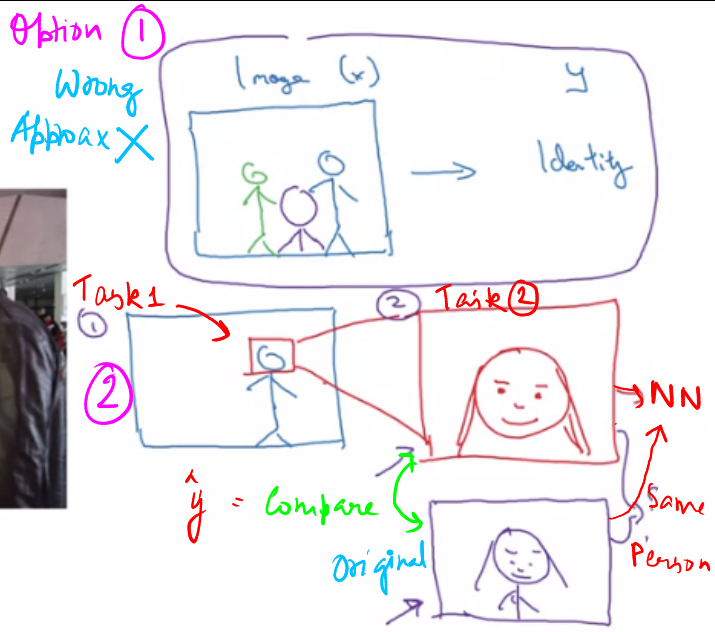
x → MFCC → features → ML → Phonemes → words → y transcript

End to end DL → audio → transcript

For smaller dataset the traditional multi stage learning does well
Eg. 3000hrs of data.
For big datasets, the end to end approach works well Eg. 10000 to 1000000 hrs of data.

Face recognition

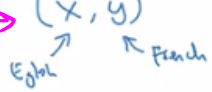

[Image courtesy of Baidu]



Task1 → to crop out the face from the image
Task2 → Compare with the original face.

* End to end approax (Option(1)) will only work when we have a lot of data. Otherwise stage wise learning is better

Machine translation



English → text analysis → ... → French
English → French

Here end to end DL works well

Estimating child's age:

Image → bones → age
Image → age

Here only multi stage approax works

Whether to use end to end DL

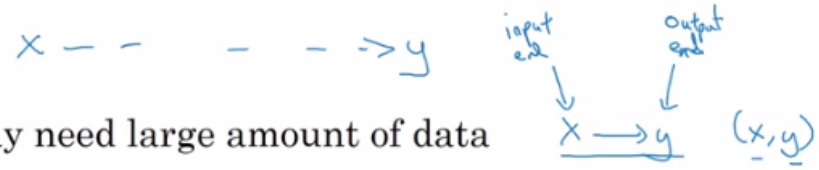
Pros and cons of end-to-end deep learning

Pros:

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

Cons:

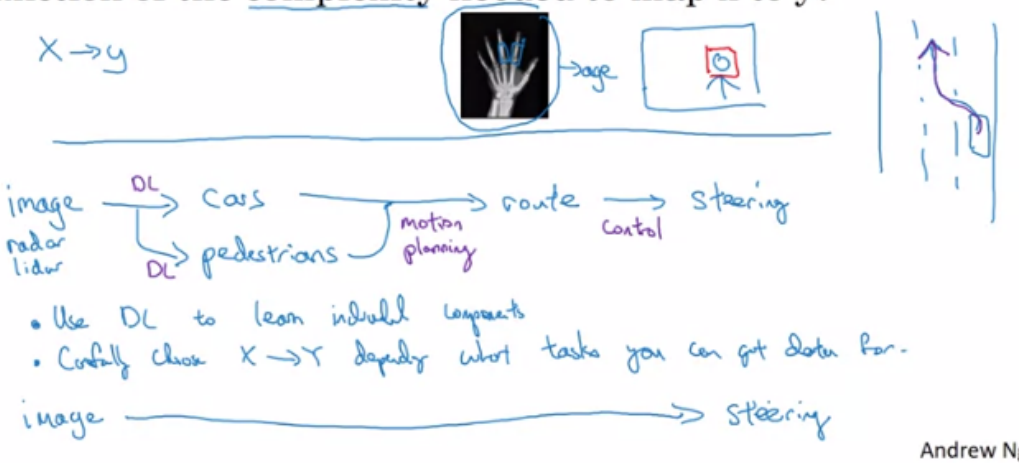
- May need large amount of data
- Excludes potentially useful hand-designed components



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?



Andrew Ng