Error Analysis

Carrying out error analysis

deeplearning.ai

Error Analysis: چک کردن خطاهایی که ماشین انجام می‌دهد تا اینکه بتوان به مشکل آن پی برد.

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:

→ 5-10 min

"ceiling"

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

→ 5%
5/100

10%
↓
9.5%

→ 50%
50/100

10%
↓
5%

Evaluate multiple ideas in parallel

Ideas for cat detection:

Three proposals

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

#Pictures to be checked manually

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

می‌توان حدس زد که کدام مشکل ماشین را حل کنیم.

deeplearning.ai

Error Analysis

Cleaning up Incorrectly labeled data

Incorrectly labeled examples

x							
y	1	0	1	1	0	$\boxed{1}$	1

Training set

Incorrectly labeled

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

Systematic errors

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%	

Dev Set Error

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.

Correcting incorrect dev/test set examples

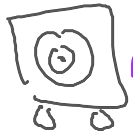
- Apply same process to your dev and test sets to make sure they continue to come from the same distribution
- Consider examining examples your algorithm got 98% right as well as ones it got wrong. 2%
- Train and $\boxed{\text{dev/test}}$ data may now come from slightly different distributions.

deeplearning.ai

Error Analysis

Build your first system quickly, then iterate

Speech recognition example

First  Prototype

- 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.

deeplearning.ai

Mismatched training and dev/test data

Training and testing on different distributions

Cat app example

care about this

Data from webpages

→ ≈200,000

Data from mobile app

→ ≈10,000

210,000

shuffle

× Option 1: train | dev | test

205,000

2,500

2,500

$\frac{200K}{210K}$

2381 - web

119 - mobile app

✓ Option 2: web | dev | test

train

train: 205,000

app

app 2500

app 2500

When distribution of train differs from test/dev:

Idea:
- For test/dev: use only the second source
- For Train: use combination of first and second source

Speech recognition example

Speech activated rearview mirror

<u>Training</u>

Purchased data x, y

Smart speaker control

Voice keyboard

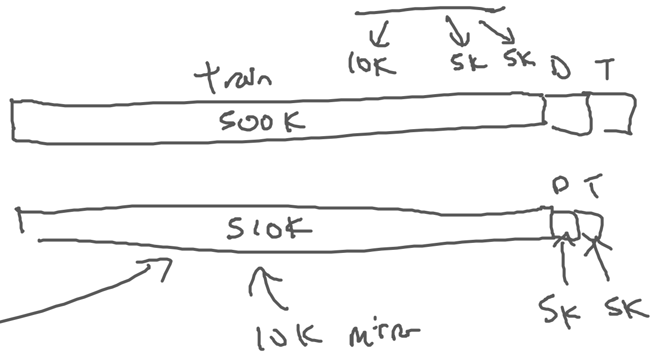
…

500,000 utterances

<u>Dev/test</u>

Speech activated rearview mirror

→ 20,000



deeplearning.ai

Mismatched training and dev/test data

Bias and Variance with mismatched data distributions

Cat classifier example

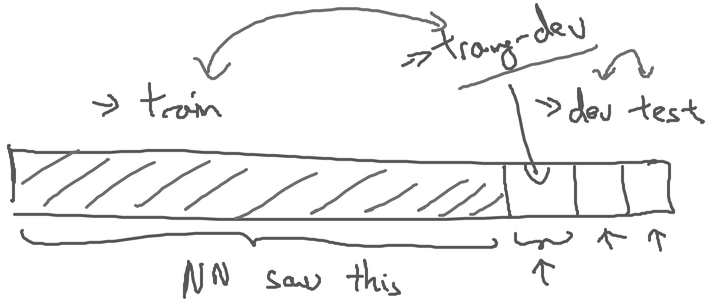
Assume humans get ≈ 0% error.

Training error 1%

Dev error 10%

↓ 9%

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



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

Human error ...	0%	Avoidable bias		Avoidable bias
Training error	10%		10%	variance
Train-dev error	11%		11%	Data mismatch
Dev error	12%		20%	
	Bias		Bias + Data mismatch	

آنالیز Bias vs. Variance وقتی که توزیع train متفاوت از dev/test باشد، فرق می‌کند. علاوه بر امکان Bias و Variance امکان مشکل data mismatch هم داریم. ایده تعریف یک مجموعه جدید به نام train_dev و محاسبه error روی آن.

Data Mismatch: Train_dev Error >> Dev/Test Error

Train_Dev توزیع مشابه به Train دارد ولی الگوریتم آن را نمی‌بیند.

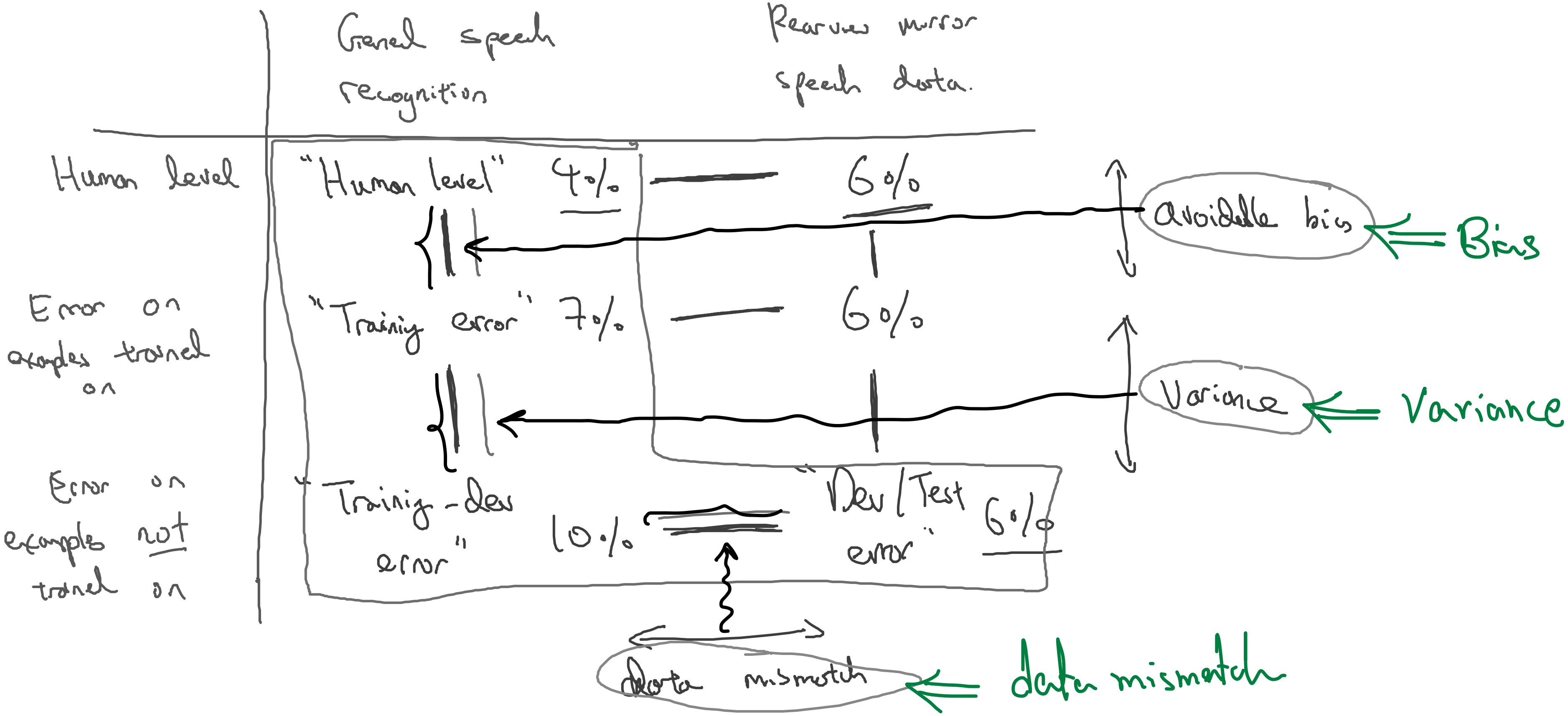
Bias/variance on mismatched training and dev/test sets

Parameters to be checked

1 →	Human level	4%		4%
			avoidable bias	
2 →	Training set error	7%		7%
			variance	
3 →	Training-dev set error	10%		10%
			data mismatch	
4 →	Dev error	12%		6%
			degree of overfitting to dev set.	
5 →	Test error	12%		6%

More general formulation

Rearview mirror



deeplearning.ai

Mismatched training and dev/test data

Addressing data mismatch

Addressing data mismatch

- 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

+ =

“The quick brown fox jumps over the lazy dog.”

10,000 hours

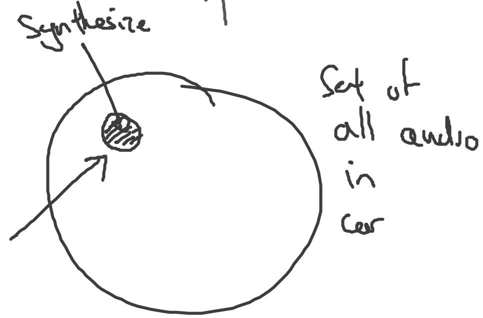
Car noise

1 hour of car noise

Overfit to 1 hour of car noise

10,000 hours

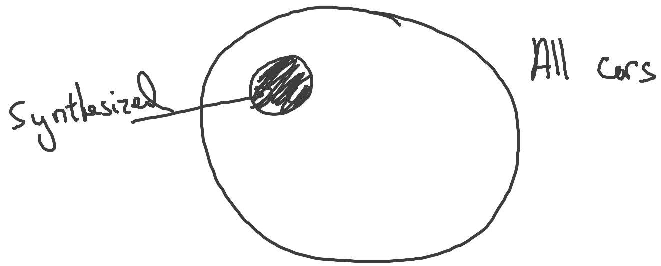
Synthesized in-car audio



Artificial data synthesis

Car recognition:

≈20 cars

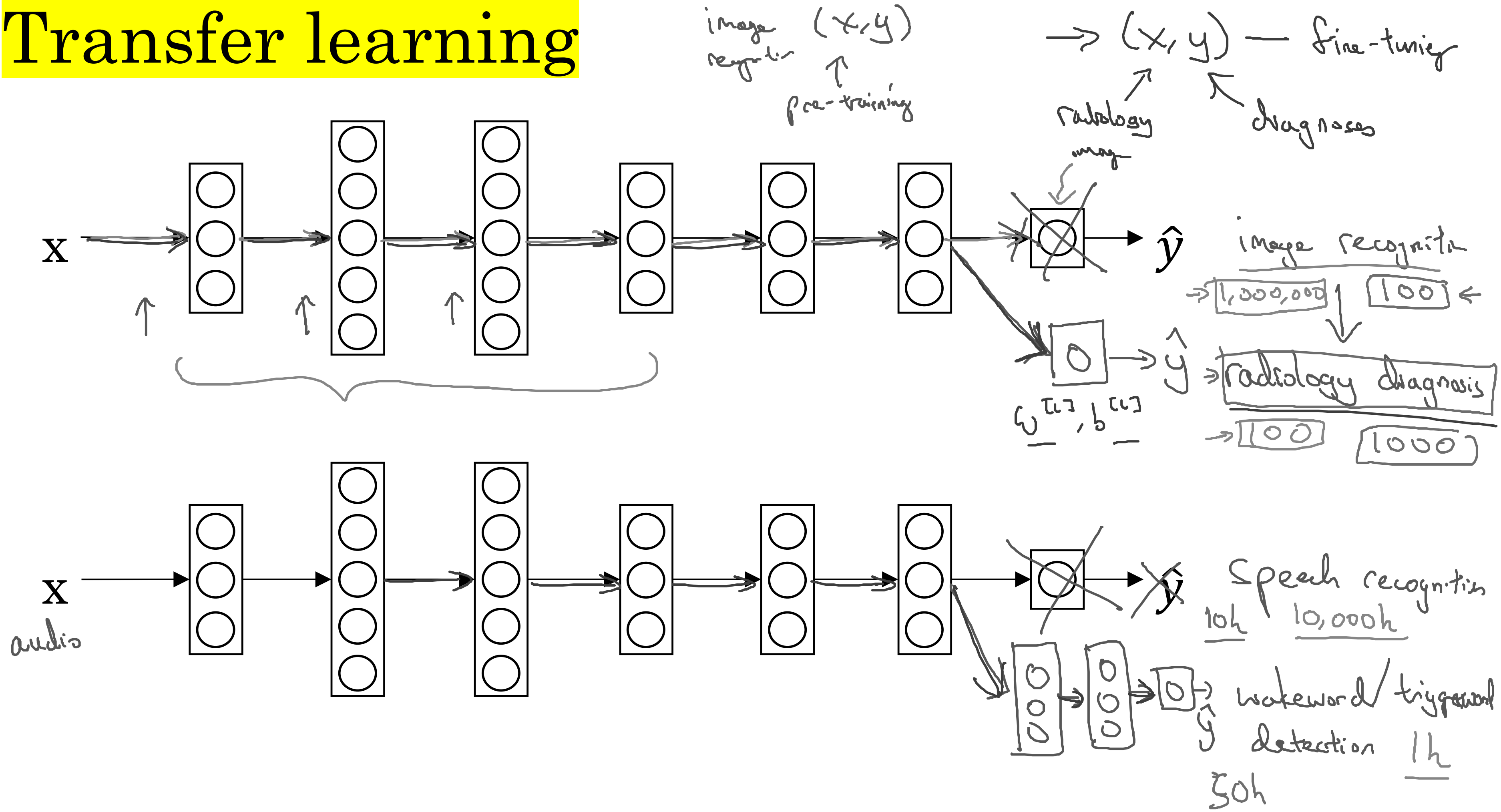


deeplearning.ai

Learning from multiple tasks

Transfer learning

Transfer learning



می‌توان layer آخر را تعویض کرد یا حتی لایه‌های جدید اضافه کرد و فقط چندتا لایه آخر را دوباره train کرد. اصطلاحا Pre-training و
Fine-tuning در این context بکار می‌رود.

When transfer learning makes sense

From To
A → B

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

Simplified autonomous driving example

$X^{(i)}$

$$\begin{array}{ll} & y^{(i)} \\ \text{Pedestrians} & 0 \\ \text{Cars} & 1 \\ \text{Stop signs} & 1 \\ \text{Traffic lights} & 0 \\ \vdots & \vdots \end{array}$$

$(4, 1)$

$$\boxed{Y = \begin{bmatrix} | & | & | & & | \\ y^{(1)} & y^{(2)} & y^{(3)}, & \cdots, & y^{(m)} \\ | & | & | & & | \end{bmatrix} \quad (4, m)}$$

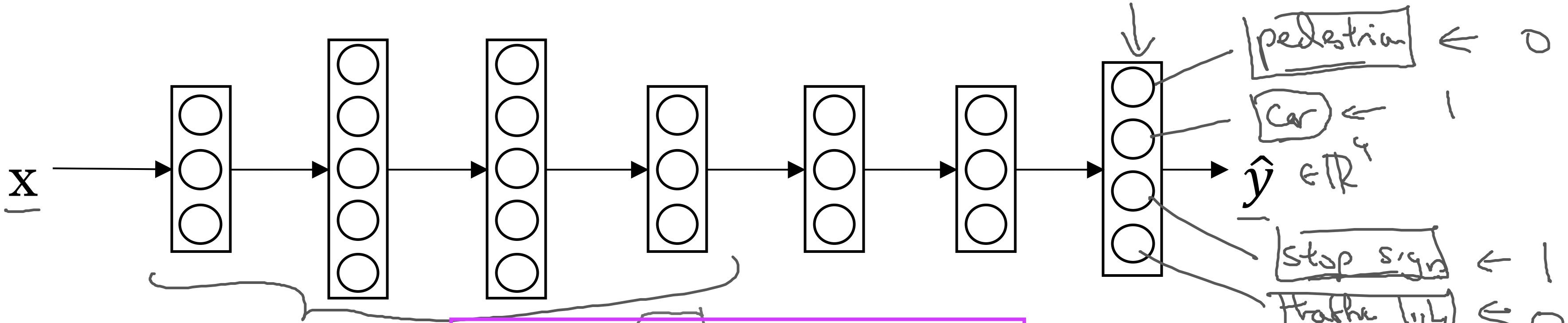
different nodes for different objects/ animals

Multi-Task Learning

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

$(4, m)$

Neural network architecture



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

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

sum only over value of j with 0/1 label.

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

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

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.

A 1,000,000

$\downarrow$ $\downarrow$

B 1,000

A_1 1,000

A_2 1,000

$\vdots$ $\vdots$

$\boxed{A_{100}}$ 1,000

99,000

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

deeplearning.ai

End-to-end deep learning

What is end-to-end deep learning

When a data processing system requires multiple stages of processing, end-to-end deep learning can take all the multiple stages, and replace it usually with just a single neural network.

What is end-to-end learning?

Speech recognition example

"cat"

x → audio $\xrightarrow{\text{MFCC}}$ features $\xrightarrow{\text{ML}}$ Phonemes → words → transcript y

audio → phonemes → ... → transcript

→ audio → transcript

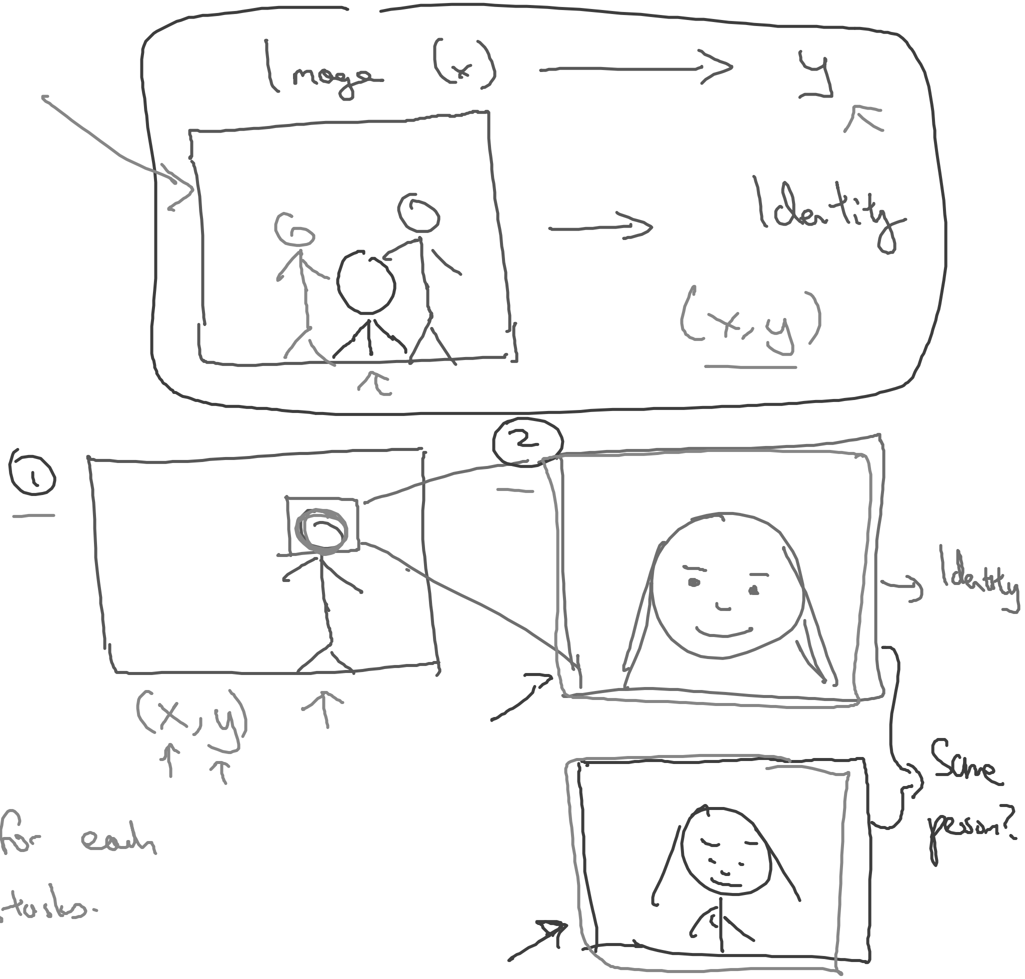
unifying all the stages

3,000h

10,000h
⋮
100,000h

Face recognition

[Image courtesy of Baidu]



Have data for each of 2 subtasks.

More examples

Machine translation

(x, y)

English French

English $\longrightarrow$ text analysis $\longrightarrow$ ... $\longrightarrow$ French

English $\longrightarrow$ French

Estimating child's age:

Image $\xrightarrow{1}$ bones $\xrightarrow{2}$ age

Image $\longrightarrow$ age $\longleftarrow$

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 $x \longrightarrow y$
- Less hand-designing of components needed

$\boxed{\rightarrow \text{“phonemes”} \quad \text{c a t}}$

− **Cons:** $x - - \quad - \quad - \rightarrow y$

input end output end

- May need large amount of data $x \longrightarrow y$ (x, y)
- Excludes potentially useful hand-designed components

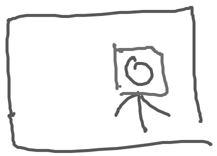
Data. Hand-design.

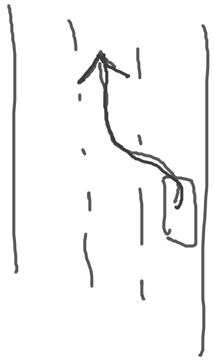
Applying end-to-end deep learning

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

$X \to y$

$\to$ age





self_Driving: {image, radar, lidar} $\xrightarrow{DL}$ cars $\longrightarrow$ route $\xrightarrow{\text{Control}}$ steering

image $\xrightarrow{DL}$ pedestrians $\to$ motion planning $\to$ route

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

$\to$ image $\longrightarrow$ steering

An example of "not" using end_to_end DL