

Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



deeplearning.ai

학습 알고리즘을 개선하는데 있어 학습에 대한 이해가 있으면,
수학적으로 알고리즘의 error를 개선하는 데 도움이 될 수 있다.
이런 프로세스를 Error Analysis라고 한다.

Error Analysis

Carrying out error analysis

Look at dev examples to evaluate ideas



90% accuracy

→ 10% error

(예제): 대략 100개의 mis-labeled 된 dev set sample 를 찾았을 때입니다.
그리고 100개의 잘못된 dev set example 중에 5%가 끝내 선택되어
가격지였습니다. 즉, 잘못된 100개의 dev set example 중에 5개인 5%만이
이었습니다. 다른 95%는 선택을 하면서 인식되었던 수준의 헷갈림은 100개
개인으로 분류되는 의미입니다. 그래서 5%인 5개는 선택되었지만, dev set error는
10%인 9.5%가 되면 충분히 의미는 의미입니다. 헷갈림으로 사용되었을 때,
가격지 선택을 해도 인식률은 향상되는 편입니다.
하지만 문제되는 것은,

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

Error analysis: $\rightarrow 5-10 \text{ min}$

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

$$\begin{array}{l} \rightarrow 50\% \\ 5/100 \\ \downarrow \\ 10\% \\ \downarrow \\ 9.5\% \end{array}$$

"Ceiling"
(예제): 이 예제는 100개의 잘못된 example 중 50개가 끝내 선택되어
가격지였습니다. 50%의 인식률은 선택을 해도 그저 그저 헷갈임.
느끼길 수 있습니다. 이런 좋은 경우에는 만약 경계선을 향상시킨다면,
dev error는 10%인 5%이며 경계선을 수 있는 경우입니다. 이런 경우
헷갈임이 훨씬 적어질 것입니다.
 10%
선택률이 저조한 경우는 헷갈임을 통해 경계선을 향상시킬 수 있는 경우입니다.
 50%

Evaluate multiple ideas in parallel

Error Analysis of the Parallelization

이 결과를 살펴보면, 어떤 종의 죄수에게 가해될 위험 역시 가장 높은 경계치를 갖는다.
우리 종은 영국에는 브리튼 아일랜드에서 있는 Engle가 암, Great Cat or cat Cat은
제일 많았다. 불과 100년 동안 브리튼 아일랜드에서 가장 많았던 종은 아파. 이들은 절대로
집에 들어오지 않는 종이다. 그리고 그들 전통적으로는 무기력한 가이드라인을 제시하는 것이다.
영국의 일부 문제를 살펴보면 8%가 배터리 가방에 걸렸거나, Great Cat 인식 문제를
제기하는 43% 정도가 개인이나 가족에게는 그들이 걸렸을 때마다 이들은 각 문제에 대해서
개인적인 보통에 있어서 통증으로부터 벗어나고 있다.

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 ←

→ 이렇게 수능점으로 물어볼 것을 시간이 허용하기 어렵거나 선택되는가, 그때에도 물어보기 우려
해야하는 것을 통해 원인을 찾는다. 이런 문제를 보면, 같은 문제를 두고, 같은 방법에 대한 흐름은
수능처럼 충분히 출제할 수 있다.

한국어로 된 책입니다. 저작권은 저에게 있습니다.

Image	Dog	Cat	Blurry	Instagram	Comments
1	✓			✓	Pitbull
2			✓	✓	
3		✓	✓		Rainy day at zoo
:	:	⋮	⋮	⋮	
% of total	<u>8%</u>	<u>43%</u>	<u>61%</u>	<u>12%</u>	



deeplearning.ai

Error Analysis

Cleaning up
Incorrectly labeled
data

Incorrectly labeled examples

X							
y	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	1

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

Systematic errors

Error analysis

< 만약 dev set 을 test set의 같은 데이터로 example을 헷갈렸다면? >
 : 같은 이미지를 examples %를 찾는다. 이때를 가정해 dev set 을 알고리즘에 적용해보면 개별적인 예제를 찾을 수 있는지 확인할 수 있다. 그 예제가 실제 레이블과 맞는지 확인해보면, 그 예제가 실제 레이블과 맞는지 알 수 있다. 전반적인 dev set error 를 살펴보면 예제를 찾을 수 있는지 알 수 있다. 이런 경우 10%의 dev error 를 가지고 있고, 같은 레이블인 이미지를 찾았을 때 개별적인 10% error는 6%, 즉 0.6%를 개선할 수 있음을 의미, 이는 매우 비중이 차이로 볼 수 있다. = '잘못된 label 수' 사용으로 비중이 차이다.

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	<u>8%</u>	<u>43%</u>	<u>61%</u>	<u>6%</u>	

Overall dev set error 100%

Errors due to incorrect labels 0.6% ←

Errors due to other causes 9.4% ←

위 이미지와 같이 예제를 살펴보면, 만약 예제는 2%이고, dev error는 같은 레이블을 찾은 경우 0.6%이기 때문에 앞의 예제는 2.1%로, 만약 예제는 10%의 dev error라면 0.6%의 같은 레이블을 찾은 경우 0.6%이기 때문에 앞의 예제는 1.4%로 볼 수 있다.

2.1%
1.4%
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

처음에는 어떤 방식을 사용해 dev set or test set의 annotation을 같은 분포로 맞춰야 한다.
refit, dev set이나 test set을 망가뜨리면 안된다. test set으로 튜닝해 개선해야 한다.

- 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.

각각에 따른 예상 예상과 실제 분포한 example 보면 아니라,
제대로 분포한 example들을 선택해야 한다. 7%의 차이를 두거나,
증가 2%인 경우에 있고, 98%를 선택할 때 이정도는 선택해도 된다
(증가에 대해서는 차이가 있을 수 있음)

선택한 train data와 dev/test data에 같은 방식을 적용해 두면,
개선이 가능해 진다. 이전에 대해서는 train data를 적용해 같은 평균이고,
평균 노드를 적용해 같은 평균을 갖다. train과 dev/test set의 평균과 평
균은 같다.



deeplearning.ai

Error Analysis

Build your first system
quickly, then iterate

Speech recognition example

- • Noisy background
 - • Café noise
 - • Car noise
- • Accent
- • Far from microphone
- • Young
- • Stuttering
- • ...

Guideline:
Build your first
system quickly,
then iterate

이전 단계에서 학습한 모델을 app을 적용하면서, 학습하는 동안 우선 시제품을
제작해보기 좋다. 보통 데스크 템플릿을 활용하는 경우이다.

요약 해설, 초기 시제품으로 우선 학습을 진행하고, 학습 성과를 시제품을 통해 bias/variance
을 조사해, error analysis를 통해 error를 분석하고 이를 개선하는 과정을
반복하는 것을 의미하는 단계를 정교화하는 단계이다.



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

답변은 train set 이 충분할 때 가능할 확률이다. 이런 이유로 많은 개발자들이 단지 모델의 데잍를 확장한 뿐만 아니라 training set 으로 사용하는 결과를 최적화한다. 결과로는 이전에 사용하는 데이터 중에 일부, 혹은 많은 데이터의 dev / test set과 같은 분포를 가지지 않을 수 있다.



deeplearning.ai

Mismatched training and dev/test data

Training and testing on different distributions

Cat app example

Data from webpages



Care about this

Data from mobile app



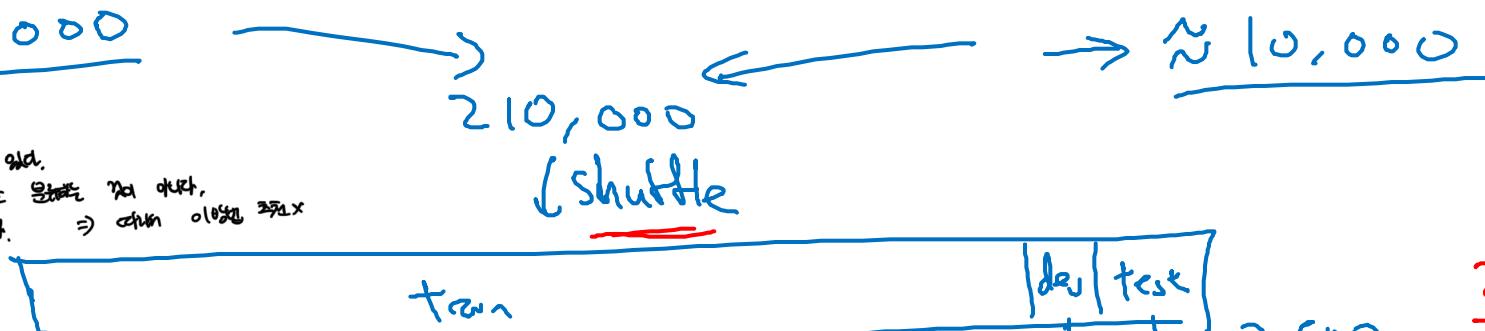
$\rightarrow \approx 200,000$
75%: train / dev / test set
25% 불필요한 훈련 할 수 있다.

당뇨: 학습률이 높아 모드를 만들 수 있다.
즉, 학습률이 높을 때는 학습률을 낮추는 게 좋다.
학습률을 낮춰 학습률이 더 감속화된다. \Rightarrow 학습률이 높아지면 더 감속화된다.

X Option 1:

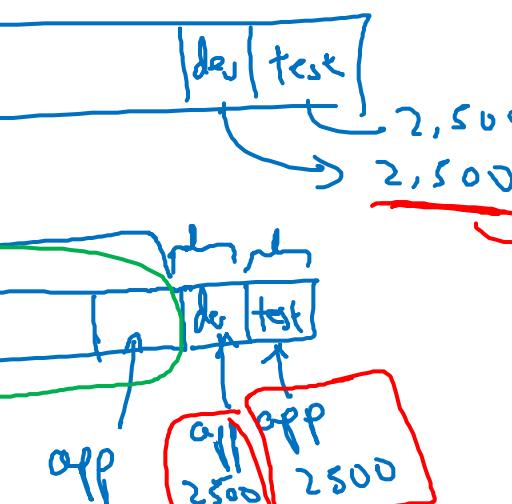
각각 200,000개이상 이를 통해 학습률이 10,000 개 일때,
training set은 웹 200,000개를 쓰고 앱은 웹 5,000개를
사용했습니다. 224x224는 dev / test set은 5000. 5000 개는 dev / test set은
2500개로 정해져 있어 이를 활용할 수 있는 이유인 것 같습니다.
그리고 training data와 dev / test set은 같은 모드입니다. 224x224는 5000
그리고 2500개가 정해져 있어 이를 활용하는 것입니다.

Option 2:

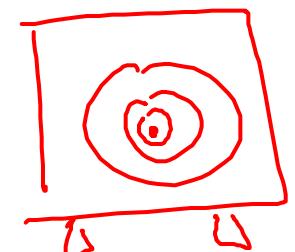


train: 205,000

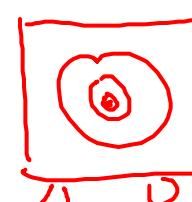
web



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



2381 - web
119 - mobile app



Speech recognition example

Speech activated rearview mirror



Training

Purchased data $\downarrow \downarrow$
 x, y

Smart speaker control

Voice keyboard

...

500,000 utterances

Dev/test

Speech activated
rearview mirror

$\rightarrow 20,000$

\downarrow
10K SK SK D T
train 500K

510K O T
10K mirror SK SK

우리 목표는 원하는 데이터를 학습하는
방법을 찾는 것입니다. 우리는 이 목표를 dev/test
셋으로 나누어 두었습니다.

우리 목표는 원하는 데이터를 학습하는
방법을 찾는 것입니다. 우리는 이 목표를 dev/test
셋으로 나누어 두었습니다.
이는 원래 SK, SK는 dev/test set을 사용하는
방법입니다.

학습 알고리즘의 bias or variance를 추적하는 것은 다음 스텝으로 어떤 영향을
수행해야하는지에 대한 유연성을 확장하는데 큰 도움을 준다. 예전에, 학습
알고리즘의 training set과 dev/test set의 차이 때문을 제거할 때면
bias/ variance를 통해 쉽게 알 수 있다.



deeplearning.ai

Mismatched training and dev/test data

Bias and Variance with mismatched data distributions

Cat classifier example

Assume humans get $\approx 0\%$ error.

Training error 1% \downarrow 9%

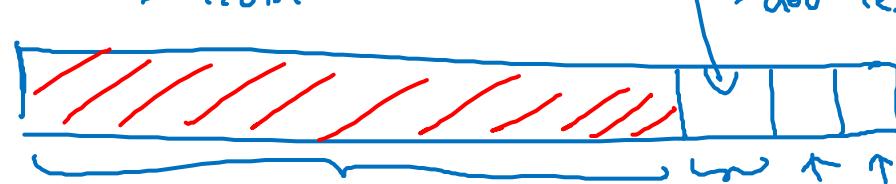
Dev error 10%

위 그림 training error가 1%이고, dev error가 10%라고 했어. 그래서 이 정도로는 훈련
잘되었던 variance 를 통해 학습 알고리즘이 학습된다는 걸 알 수 있다. 하지만, training set과
dev set의 분포가 MZ 다른 때로, variance 를 통해 이를 알 수 있다. training set과 dev set
데이터 자체로 이를 쉽게 알 수가 가능하지만, dev set을 훈련 데이터로 훈련 후 이를 voir.
이걸 Error Analysis 부분으로서 training error와 dev error를 비교할 때, training set과 dev set
의 분포가 MZ 다르게 되었을 때, error는 그대로 dev set의 분포를 훈련하지 못
하는 경우, 즉 variance 같은 것들이 확률에 헷갈리게 된다.

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

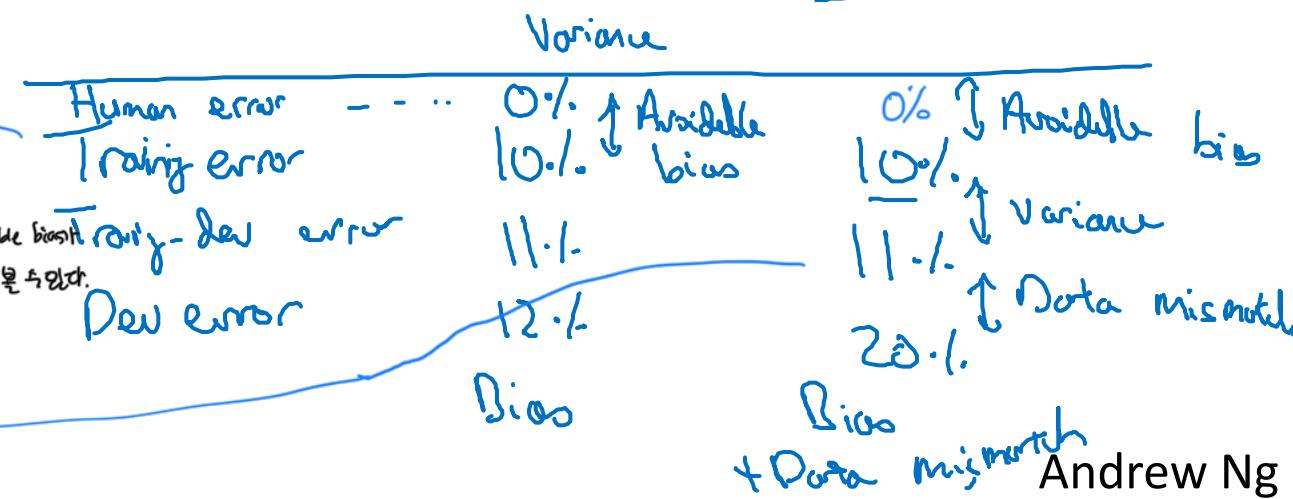
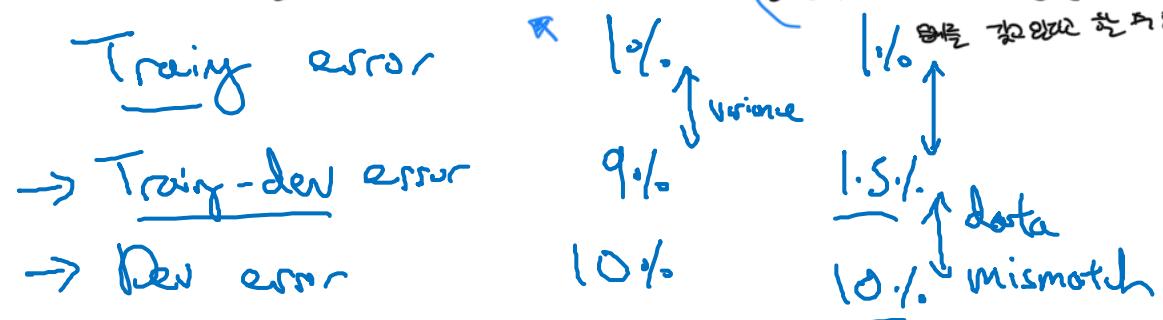
Human error 0%
Training error 10%
Training-dev error 11.1%
Dev error 12%
Bias 20%
+ Data mismatch

기본적으로 데이터를 분류하기 위해, train/dev는 MZ 같은 분포를 갖고, dev/test가 MZ 같은
분포를 갖다. 물론 train/train-dev와 dev/test의 MZ는 다르다. \rightarrow train-dev
 \rightarrow train
 \rightarrow dev/test



인공지능의 MZ training & training-dev 차이가
제일 크게 일어나고, low variance를 가진다고 할 수 있어,
dev error는 많이 증가한다. 이런 경우 전형적인
data mismatch 문제는 발생해 충격을 입는다.

인공지능은 training과 training-dev에서
의 차이가 제일 크게 일어나고, low variance를
가진다고 할 수 있어, dev error는 많이
증가한다. 이런 경우 전형적인 data mismatch
문제는 발생해 충격을 입는다.

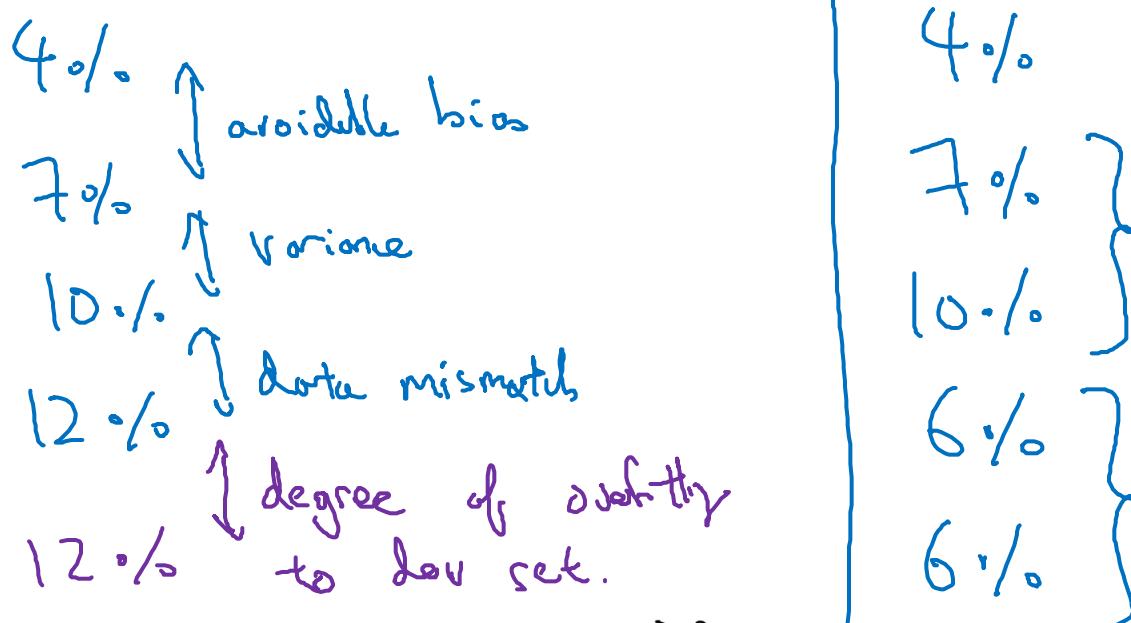


Bias/variance on mismatched training and dev/test sets

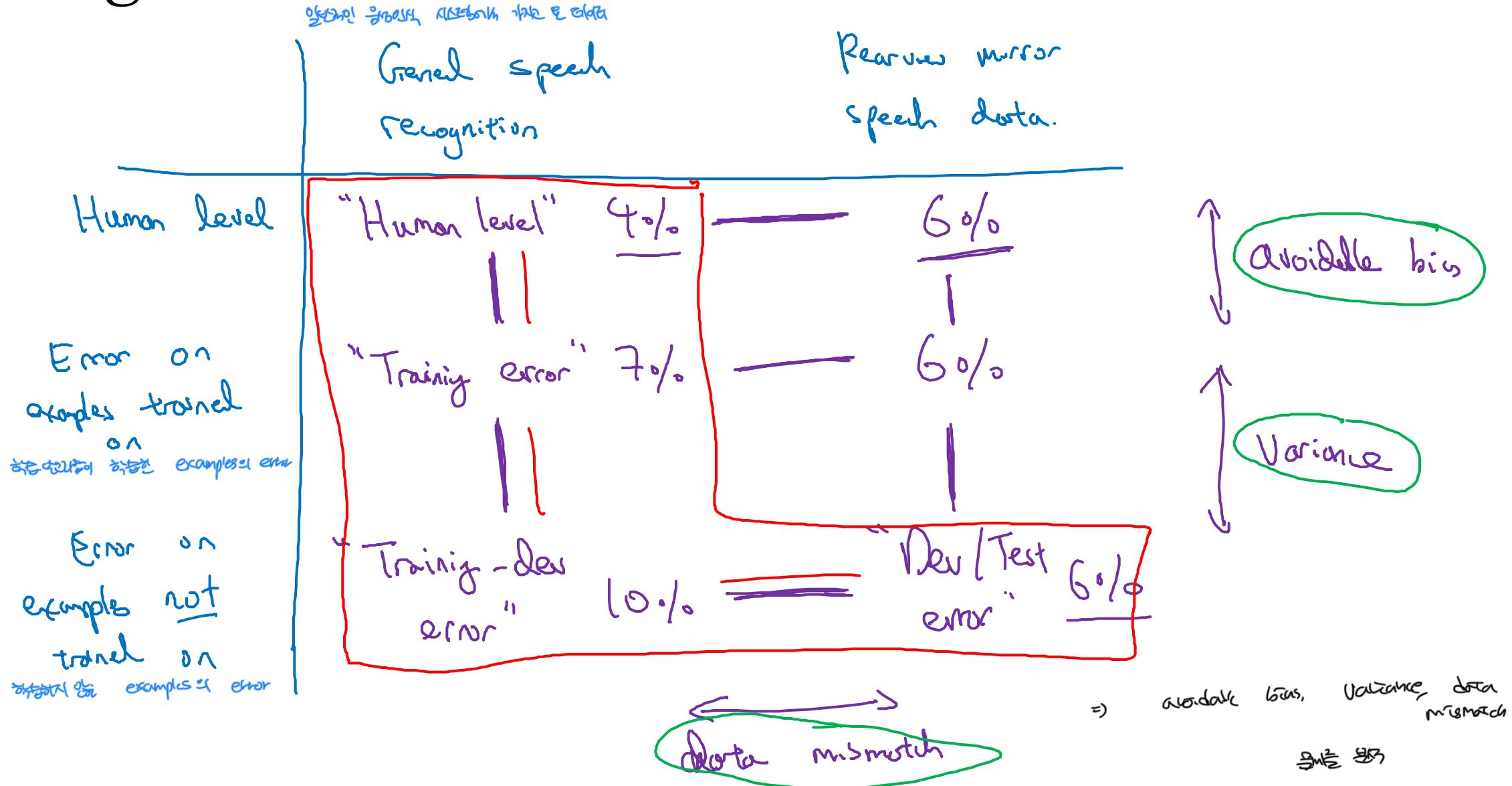
Human level

- Training set error
- Training - dev set error
- Dev error
- Test error

각각, test error는 train set이 dev set에 overfitting을 하면서 생긴 오류.
negative가 있다. 그때 training error가 dev/test error보다 잘 맞는다.
이유는 dev/test set이 training set의 확장성이 있는 경우 때문이다.
High bias를 한다.



More general formulation





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

처음에는 training set과 dev/test set이 같은 차이가 있는지 수작으로 error analysis를 수행해 보자. 예를 들어, 어떤 지역의 주소를 개별로 살펴보면, training set과 dev set은 비교해서 dev set이 훨씬 더 정교하고, 개별 주소를 많다는 것을 발견할 수 있다. 이처럼 training set이 dev set과 함께 다른 파악할 수 있다면, 이를 training data는 dev set이 더 유용히 활용할 수 있다.

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

E.g. Simulate noisy in-car data

두 번째 방법은 dev/test set과 유사한 데이터를 만들어내는 것이다.
가장 좋은 경우는 확인하고 발견하는데, 가장 좋은 경우가 발견되는 경우다.

Artificial data synthesis

+

까먹은 데이터를 많이 놓쳤을 때 기준하고, 차수를 더해줄 때 기준은 인대연, 이 늦게 학습하는 학습률에서 시그모이드 캐릭터를 막는 것과 같은 효과를 낼 수 있다. 그래서 학습률 초기값을 높으면서 업데이트 영역을 넓지 않을 필요 없어, 인공적인 데이터 학습을 통해 더 빨리 학습할 수 있다. 하지만, 이 학습률을 사용할 때 한 가지 고려할 점이 있다. 바로 우리나라 10,000 시간 동안 소음에 까먹은 데이터 있고, 저수준 소음 데이터가 (시그모이드 인대연, 저수준 소음 데이터를 0,000번 반복시켜 학습할 수 있다. 실제 저수준에는 데이터 노동이 있지만, 이 경우에는 우리가 시그모이드에 대해서만 학습을 진행하면서 학습률, 한 시그모이드 단위의 overfitting

2nd.

1

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

10,000 hours

A hand-drawn diagram consisting of a blue oval containing the word "how". An arrow points upwards from the top center of the oval, and another arrow points to the right from the bottom right corner of the oval.

Overfit to 1 hour fly

Car noise
10,000 hours

Synthesizer

↓

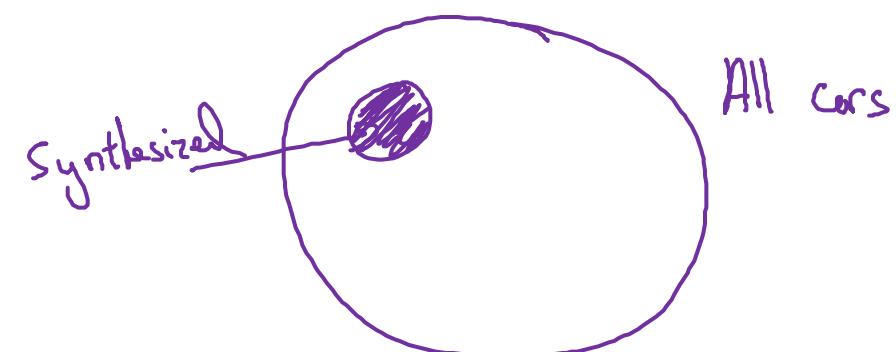
Set of
all audio
in
car

Artificial data synthesis

Car recognition:



$N \approx 20$ cars





deeplearning.ai

Learning from multiple tasks

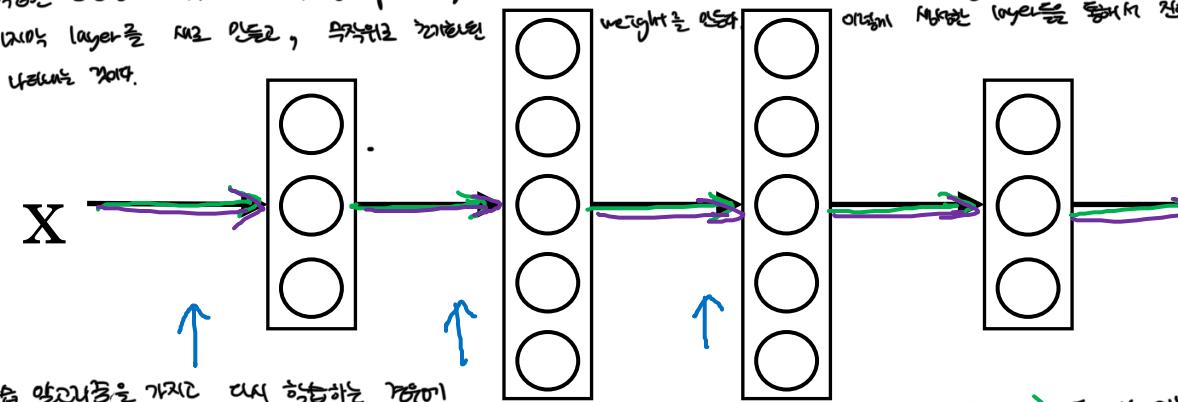
Transfer learning

전이학습(Transfer learning) : 어떤 학습을 이전에 위해 학습한 모델을 다른 주제에 이용
전이학습을 이용하는 이유 : 같은 데이터를 학습해 학습하는 다른 학습들이 있다.
① 학습이 번거롭거나 더러울 수 있다. 이미 양적되는 데이터에 대해 루프를 훈련해 추출하기 어렵다.
학습을 데이터에 대해 루프를 추출하여 이를 학습해 블록으로 훈련해 복잡도를 줄기 때문이다.
② 같은 데이터에서 다른 학습을 위해 예상할 수 있다. 같은 데이터로 루프를 추출하기 위한
학습은 한계 때문, 데이터 수가 적어 모델의 가중치가 많을 수 있기 때문에 학습에 드는
학습률이 높아지며, 학습의 속도가 느려 가중치가 많을 수 있는 대신 학습에 드는
학습률이 낮을 수 있다. 전이학습을 이용해 이전의 학습으로 학습하게 된다면, 학습의 가중치 수가
줄어 고친 학습의 어려움을 줄 수 있다.

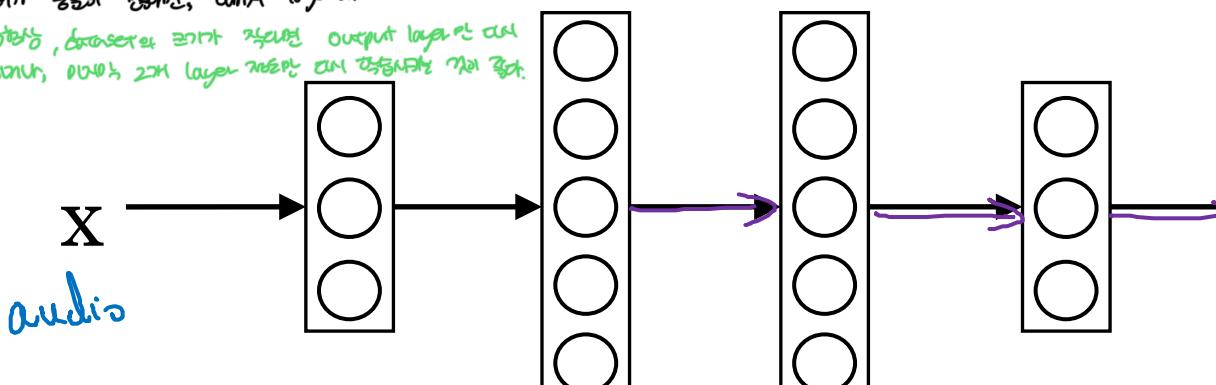
Transfer learning

x : 이미지, y : 인식 결과

우선, 학습한 신경망의 output layer를 사용해, 다른 신경망의 input layer로, weight(과 bias)도 가져온다.
그리고, 마지막 layer를 새로 만들고, 목적함수로 loss function
경과율을 계산하는 것이다.



<기존 학습 알고리즘을 가지고 다시 학습하는 경우>
→ θ 를 data set의 (x_i, y_i) 에 대해 초기화 한다.
▷ 만약, 학습된 이미지 dataset이 많지 않다면, 마지막 신경망의
학습률이 높아지면, 다른 신경망의 학습률은 저하될 수 있다. 반면,
학습률이 낮다면, 다른 신경망은 학습률을 높여 fine-tuning 할 수 있다.
+ 특히, dataset의 크기가 작으면 output layer만 대
학습된다면, 다른 layer들은 대 학습하지 않고 있다.



이전에 이미지와 같은 특징을 가진 이미지를 인식하고 전달하는 것을 $task$ (transfer) 한다.
이미지 같은 것은 edges를 강조하며, curve를 찾고 positive objects를 찾는
low level features ($edge$) 때문이다.
이미지 인식을 위한 데이터셋은 이미지의 URL, 이미지가 어떤지 설명하는 텍스트, 이미지의 특징들을
인식하도록 학습한 것이다. 그래서 이미지 인식을 구현할 수 있다.

image
recognition
(x, y)

pre-training

$\rightarrow (x, y)$ — fine-tuning

Radiology
model

Diagnoses

image
recognition

$\rightarrow [1,000,000] \rightarrow [100]$

Radiology diagnosis

$\rightarrow [100] \rightarrow [1000]$

Speech
recognition
10h 10,000h

wakeword / triggered
detection 1h
50h

When transfer learning makes sense

Transfer from A \rightarrow B

앞의 예제들로부터 우리는 Transfer Learning은 언제 적용해야 하는지 표시하는 규칙을 찾았다.
우리가 transfer 학습하는 것은 data + 모델, transfer 학습하고 하는 것은 dataset + 모델이다.
음성인식 가능성이 10,000 시간의 데이터로 학습 했지만, trigger word 같은 특성을 주면
데이터는 (시킬 범위 내에서) 학습 안해도 가능하게 되어버려 데이터 특성이 trigger word 같은 특성을
학습하는데 도움을 줄 수 있다.

- Task A and B have the same input x.

구현하기 어렵지만, Transfer learning은 Task A와 Task B가 같은 입력 (이미지나 음성같은)으로 구현되어 있고,

- You have a lot more data for Task A than Task B.

Task A의 데이터가 Task B의 데이터보다 훨씬 더 많은 양으로 가족이다.



- Low level features from A could be helpful for learning B.

기억 더 치기 어렵지만, Task A에서 학습하는 low level 특성을 Task B를 학습하는데
쓸 수 있다면 학습하는 데 도움이 될 것이다.

Multi-task Learning 이론 (MTL)

- 여러 task를 포함한 유사한 정보를 학습하는 task의 전략으로
기반으로 학습시켜는 것을 기본적인 학습방법이다.

MTL 종류

- MTL의 중요한 종류는 데이터 혼합과 문제를 선택하는 것이다
- 각 task의 경우 더 자세한 학습률을 얻기 위해 Data Augmentation 기법으로
다른 task에 대해 데이터를 얻는다. 단일 task 학습과 비슷한 다른 task에
도 적용할 수 있다.



deeplearning.ai

MTL vs 전이학습

둘의 차이점은?

- MTL에서는 각각 다른 task의 구조와 규칙을 학습하는 대상이다.
전이학습은 각각의 task의 구조와 규칙을 학습하는 대상이다.
- 전이학습은 source task의 정보를 target task의
학습에 활용하는 경우 source task의 정보를 활용하는 경우
전이학습이라고 한다.

Learning from multiple tasks

Multi-task learning

Simplified autonomous driving example



$x^{(i)}$

Pedestrians

Cars

Stop signs

Traffic lights

⋮

$y^{(i)}$

0

1

1

0

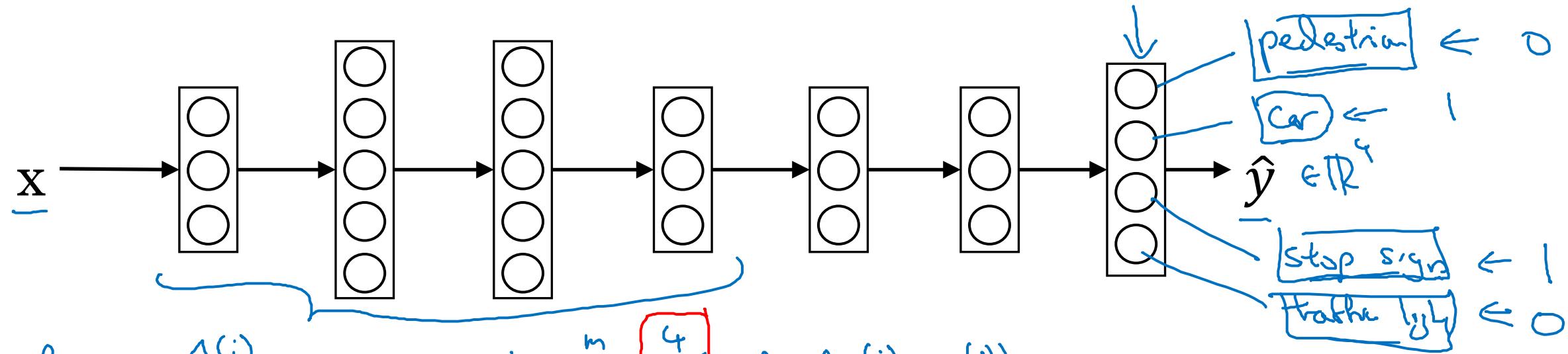
⋮

(4, 1)

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

(4, m)

Neural network architecture



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

$$\rightarrow \frac{1}{m} \sum_{i=1}^m \sum_{j=1}^4$$

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

Unlike softmax regression:

One image can have multiple labels

여기서, 차량, 신호등, 물체 표지판 등이 같은 NN을 갖는다. 각각은 다른 종류의 예측을 한다. 예전에는 각각의 신호등이나 차량이나 물체를 예측하는 별도의 신호망을 갖는다. 하지만 최근에는 같은 신호망을 갖는다. 한 번에 신호망을 갖는다. 그리고 각각의 신호망은 다른 신호망과 상호작용하지 않는다.

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

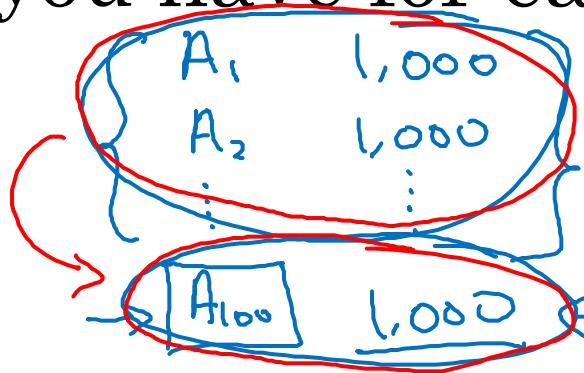
Multi-task learning

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

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.

$$\begin{array}{ll} A & \underline{1,000,000} \\ \downarrow & \downarrow \\ B & \underline{1,000} \end{array}$$



각각의 task는 풀수 규모는 대고, 허깅 둘째로 많지만
각각의 task의 dataset 양이 유사하게 가능하다.
99,000 데이터에 적용되는 것은 데이터만,
multi-task learning을 했거나 학습되었을 때
각각 다른 task별 data 양이 유사한 task이다.
회전 방향이 같다.

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

마지막으로, 충분히 큰 neural network로in 학습시기는 경우에 잘 동작한다. Rich Caruana 연구원은 multi-task learning
각각의 NN으로 학습하는 것과 다른 방법으로 학습하는 경우에, NN이 충분히 크지 못하는 경우에는 이를 발견했다.

예를 들어, multi-task learning은 transfer learning과 흡사한 디자인이다. transfer learning은 학습하는 데이터를 재사용하고,
특정한 학습을 위해 사용되는 경우를 가로 암시 한다.
예전에 computer vision object detection이라는 transfer learning을 multi-task learning으로 디자인해, 같은 디자인을
학습하는 디자인을 찾았습니다.

< End-to-end learning >

: 입력 (input)과 출력 (output) 모두 학습하는 네트워크입니다.

설명: 흐름과 차이

파이프라인 네트워크인 전체 네트워크를 이루는 하위 네트워크 (sub-network)입니다.

⇒ 예전에는 process를 거치는 것을 통해 NN으로 변환하는 것



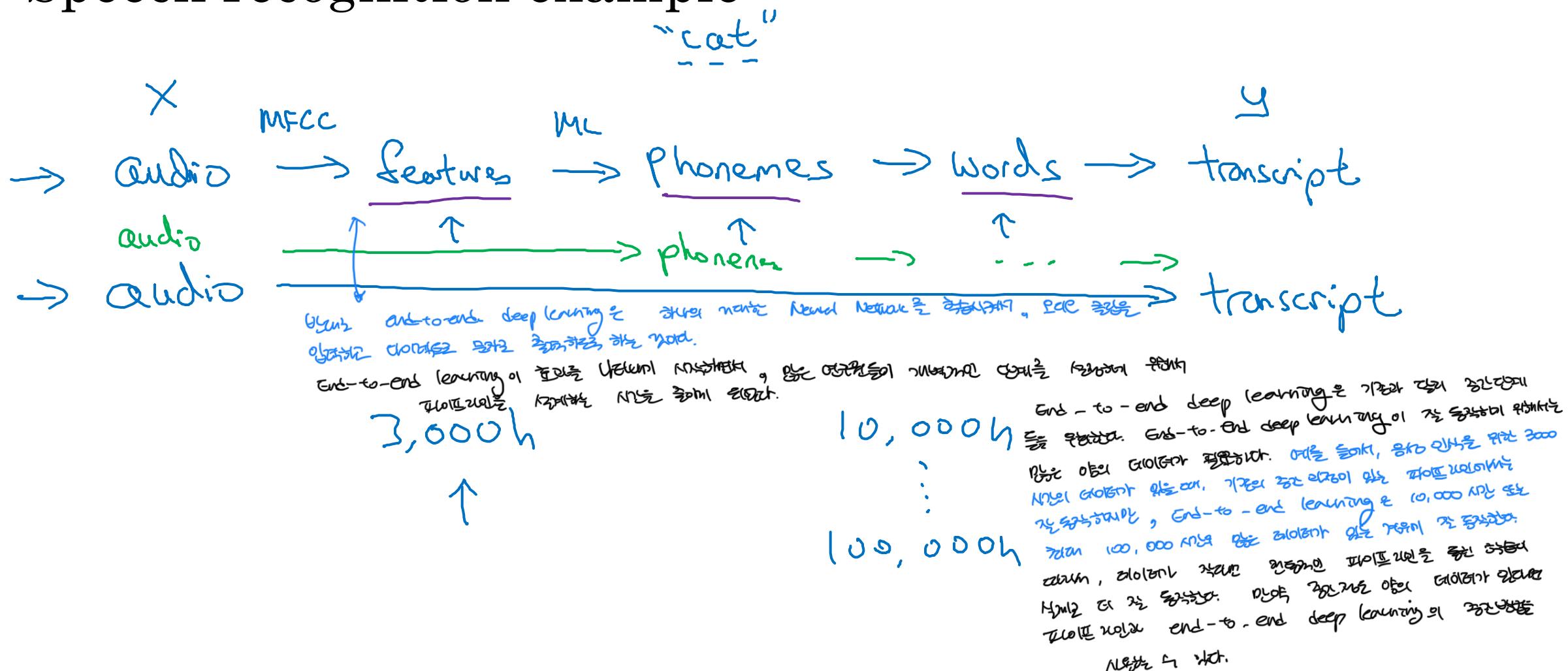
deeplearning.ai

End-to-end deep learning

What is
end-to-end
deep learning

What is end-to-end learning?

Speech recognition example



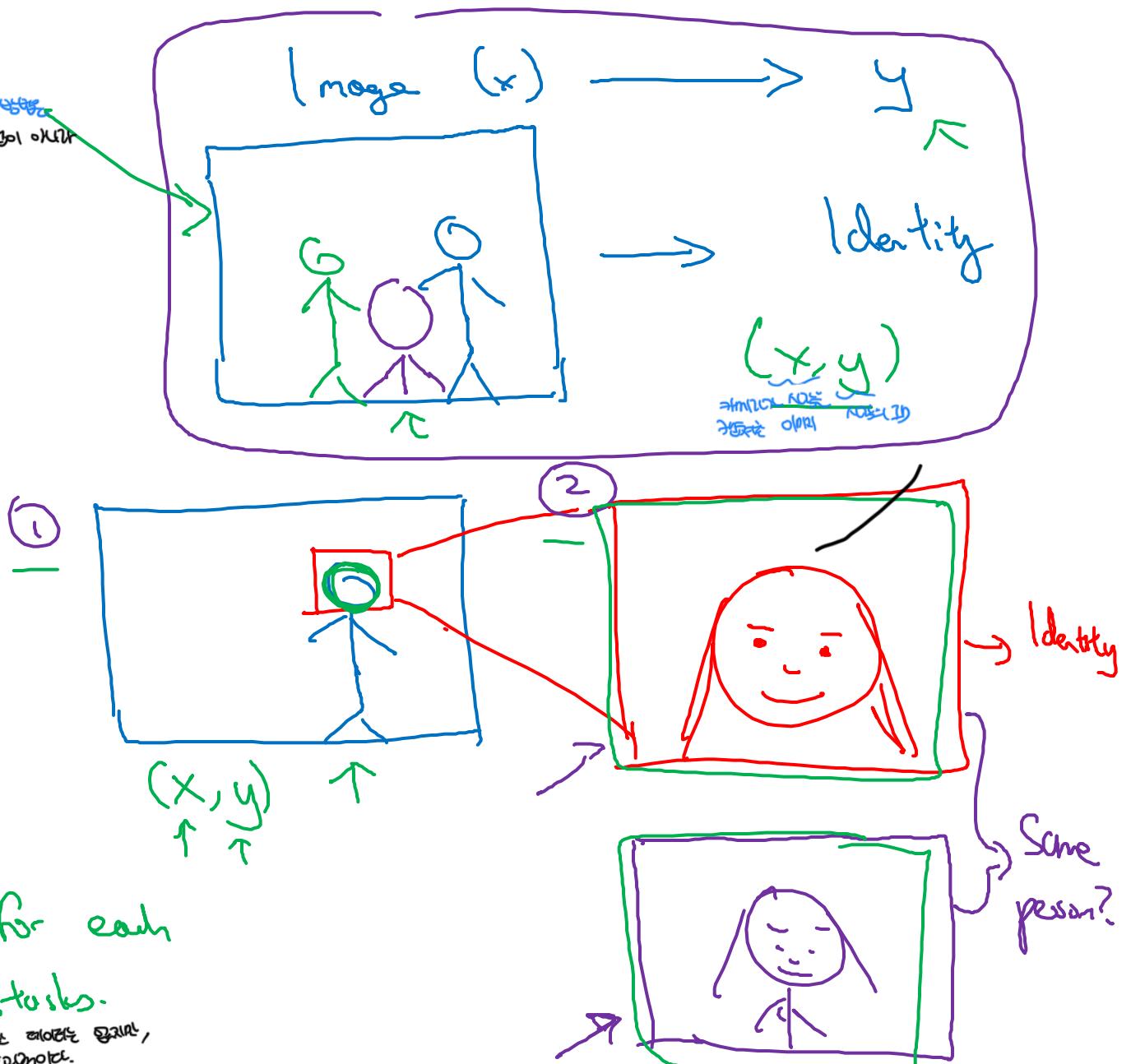
Face recognition

End-to-end deep learning 방법으로 raw image를 NN으로 넣어 ID를 알아내는 방식이다.
여러 단계로 나누는 대신 하나다.



[Image courtesy of Baidu]

< 왜 두개의 단계로 디코딩할까? >
우선 각각의 알고리즘이 학습하고자 하는 문제를 만들기에는 맞지 않아서다.
우선 각각의 알고리즘은 학습을 위한 데이터가 충분히 많아야 한다. 예전 face detection을
기반의 알고리즘 학습을 위한 데이터는 충분히 많았지만, 이제 얼굴만 추출하는 데이터는 턱 없어졌다.
위한 데이터는 매우 많다. 그리고 동일인을 인지하는 문제는 얼굴만 추출하는 데이터로 턱 없어졌다.
그리고 End-to-end deep learning 방법으로 모든 것을 동시에 학습해보기 어렵기 때문이다.
그러나 End-to-end deep learning 방법으로 학습하기 위해서 충분한 데이터는 있지만,
 $x \rightarrow y$ 로 데이터는 데이터가 매우 적을 것이다. 그래서 End-to-end deep learning 방법으로 학습하기 위해서 충분한 데이터는 있지만,
실제로 문제를 해결하기 위해 데이터는 충분하지 않다. 문제를 해결하기 위해서 디코딩해야 한다.
물론, 디코딩 데이터가 충분하다면, 아래와 같은 단계로 디코딩이 더 훨씬 쉬울 것이다.

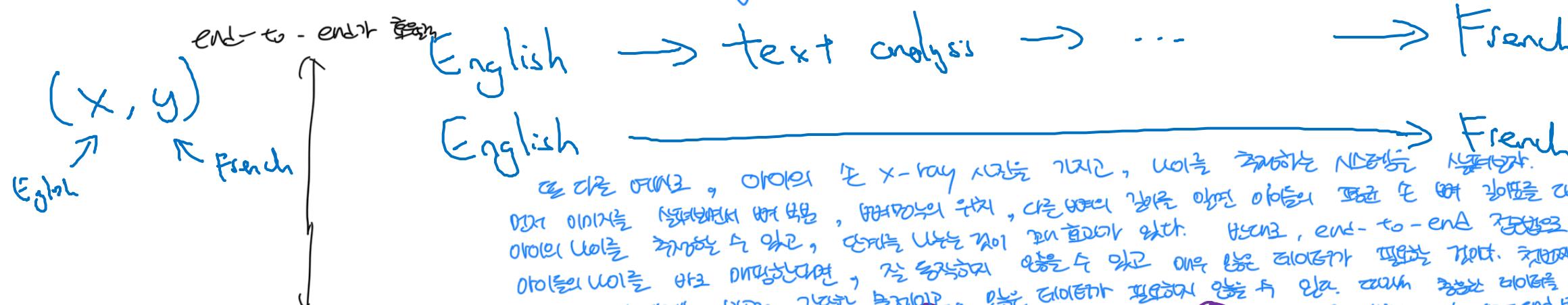


Andrew Ng

More examples

기계번역을 예로 들면, 전통적으로 기계번역, 시스템은 복잡한 프로파일을 구현하기 위해,
영어 문장을 가지고 여러 단계를 거쳐서, 프로그램에 있다. 오늘날에는 까만
(영어, 프랑스어) 대장 데이터가 있기 때문에, 이런 경우는 end-to-end
deep learning or 기계번역까지 매우 흐르며 할 수 있다.

Machine translation



Estimating child's age:





deeplearning.ai

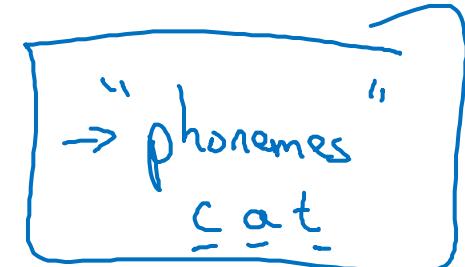
End-to-end deep
learning

Whether to use
end-to-end learning

Pros and cons of end-to-end deep learning

Pros: 데이터 학습을 통해 데이터 만들하는 것 가능할 수 있다.

- Let the data speak $X \rightarrow y$



- Less hand-designing of components needed

기계로이 가능하다. 이 경우에는 우�플로우가 단순화되고, 중간과정들을 설계하는데 많은 시간을 투여하지 않아도 된다.

Cons:

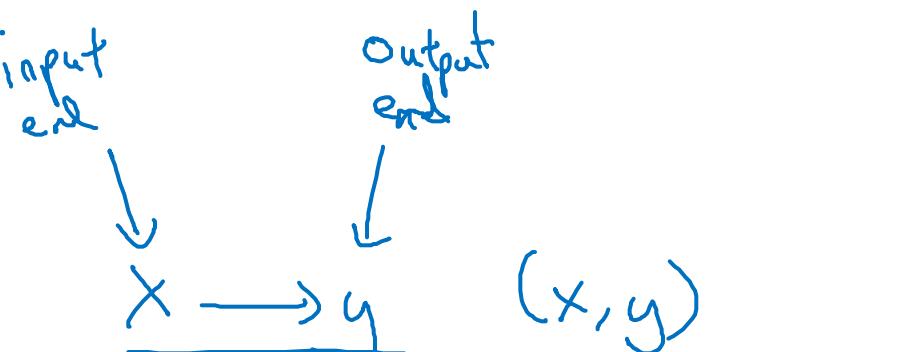
- May need large amount of data

데이터가 필요하다. $x \rightarrow y$ 매핑을 얻기 위한 데이터가 필요하다.

- Excludes potentially useful hand-designed components

Data
.....

Hand-designn.
.....



유용하게 작동하는 핸드디자인 컴포넌트는 제거되는 경우, 데이터가 필요하다.
train set은 있을 수 있는 경우, 각 핸드디자인 hand-designed component를 사용하는 경우 더 좋은 성능을 보여줄 것이다.

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?

