Lec1: Training and testing on different distributions:

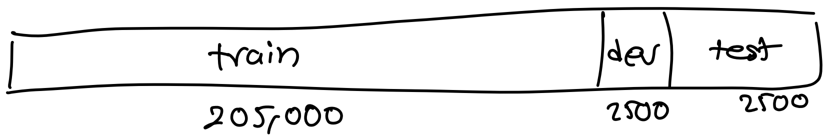
Cat App example:-

training data
- data from mobile app (blurry, low res) ≃ 10,000
- data from web (high quality, high res) ≃ 200,000

test data: data from mobile app.

Will data from web help: ?

✗ option1: combine $m = 210,000$, random shuffle (bad dev set)



adv: train, dev, test have same distribution

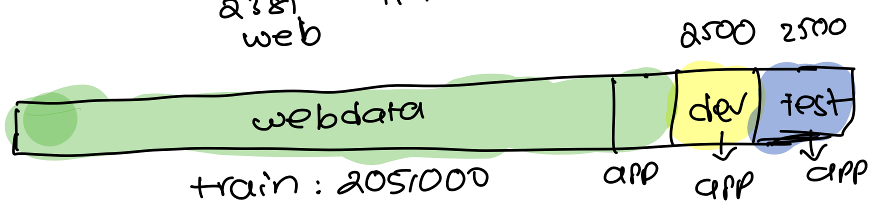
disadv: dev set major data will come from web page data.

dev $m = 2500$

2381 web

119 mobile

✓ Option 2:



train set : 200,000 + 5000

↓ web ↓ app

dev : 2500 app

test : 2500 app.

adv: dev set is accurate to test set

dis: training dist is different from dev, test set

speech recognition example:

- speech activated rear view mirror in car.

Training:

purchased data (x, y)

smart speaker control

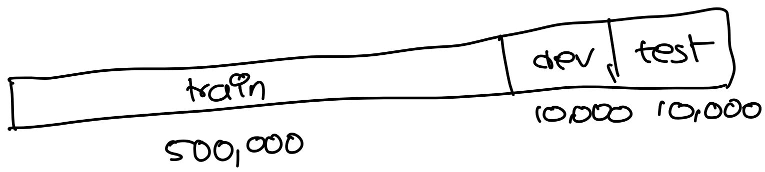
voice keyboard

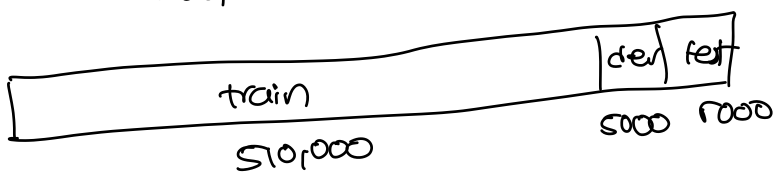
500,000

Dev/test

speech activated rearview mirror

20,000





train: 500,000 from others
10,000 from rearview

dev: 5000 from rearview

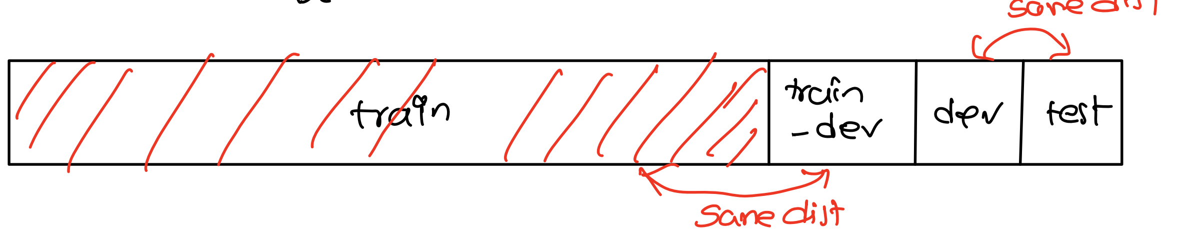
test: 5000 from rear view.

Lec2: Bais & variance with Mismatched Data distribution:-

Assume human error on cat classification = 0%

Training error = 1%

Dev error = 10%



if Train dist = Dev dist

problem is variance, more data, $\alpha 2$ reg, dropout.

if Train dist $\neq$ Dev dist

- → maybe it is doing fine on dev set.
- → training set may be easy (high res, high quality)
- → dev set may be hard (low res, low quality)

Human error	0%	0%	0%	0%	avoidable bias
Training error =	1%	1%	10%	10%	variance
Training dev set =	9%	1.5%	11%	11%	data mismatch
Dev set =	10%	10%	12%	20%	degree of overfit to dev set. (find bigger dev set)
Test set =				x%	

This is variance problem → not generalising well on training-dev set which comes from same train dist	low variance → data mismatch problem. → get more train data dist to dev, test	→ high bias	→ high bias → Data mismatch problem. → low variance

Human level error :	4%	avoidable bias	4%
Training set error :	7%	variance	7%
Training dev set error :	10%	data mismatch	10%
Dev set error :	12%	degree to overfit dev set	6%
Test set error :	12%		6%

training data is harder than dev, test set.

More General Formulation:-

	General speech recognition	rear view mirror speech data
Human level	human level: 4%	human level: 6%
Error on examples NN trained on	Training error: 7%	Training error: 6%
Error on examples NN has not trained	Training-dev error: 10%	Dev/test error: 6%

avoidable bias

variance

data mismatch

Lec3: Data Mismatch Problem:-

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

  Eg: 1. rear view number - car noise in dev set

  2. rear view number - misrecognizing street numbers in dev set.

sol: collect more data similar to dev/ test set.

→ add car noise to original audio

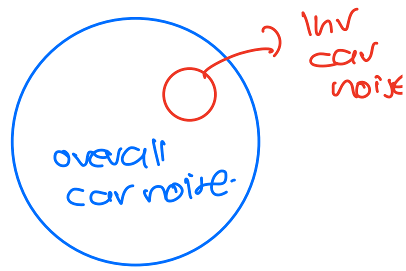
→ add more numbers audio in train set.

Artifical data synthesis:-

normal audio + "car noise" = synthesized in car audio
10,000hr   1hr

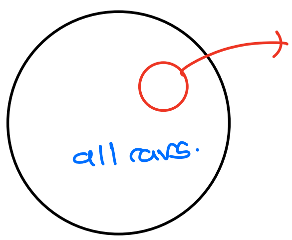
disadv : overfit data 1hr car noise

10,000hr unique car noise can be even better performance.



Car recognition:-

computer graphic generation of car image.



synthesized ≈ 20

model might overfit training synthesized data.

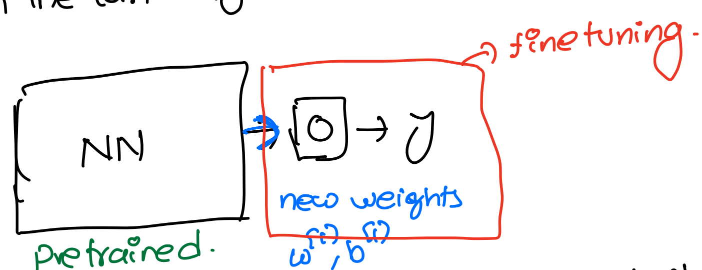
Lec4: Transfer Learning:

trained NN : (x, y) $x \to$ image
$y \to$ cat, dog, bird

Transfer learning $x \to$ image (radiology diagnosis)

→ remove last layer weights & output radiology output.

→ retrain the last layer.



① if you have small data, then retrain last layer weights.

② if you have more data, then retrain complete NN.

Use case:-

→ If you have alot of data, for the problem you are transfering from $(m = 1{,}000{,}000)$

→ and less data, for the problem, you are transfering to. $(m \geq 100 \leq 1000)$

Transfer from $A \to B$

① Task A, B have same input X

② You have lot more data Task A than Task B.

③ low level features from A could be helpful for learning B.

Lec 5: Multi Task Learning:

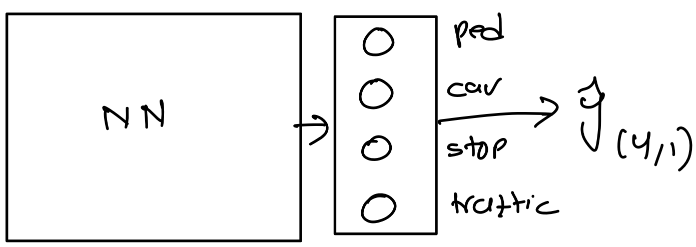
Autonomous vehicle:-

detect:
- pedestrians
- other cars
- stop signs
- traffic lights

input

$x^{(i)}$ (image) $\longrightarrow$ $y^{(i)}$ [ped, car, stop, traffic lights] $(4,1)$

$$Y = [y^{(1)}\ y^{(2)} \dots y^{(m)}]\ (4,m)$$



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

$$\text{Loss} = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{m} \mathcal{L}(\hat{y}_j^{(i)}, \hat{y}_j^{(i)})$$ (usual logistic loss)

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

$\rightarrow$ This is multi task learning.

unlike softmax regression:

one image has multiple lables.

$$Y = \begin{bmatrix} 1 & 1 & 0 & ? \\ 0 & 1 & 0 & 1 \\ ? & ? & 1 & ? \\ ? & ? & 1 & ? \end{bmatrix}$$ $\rightarrow$ still works for multi task learning.

(sum only over labelled clais).

When Multitask learning works:-

① training on set of tasks that could benefit from learning shared lower level features.

② Usually: amount data you have for each task is quite similar.

Transfer learning

A (1,000,000)

↓

B (1,000)

multi task leaning

A_1 1000

A_2 1000

⋮

A_{100} 1000

→ A_{100} now has 99,000 more data point which can be used for low level feature extraction.

③ can train a big enough NN to do well on all tasks.

deco: End to End Deep learning:-

Speech recognition example:

traditional:-

$m = 3000h$

X audio $\xrightarrow{MFCC}$ features $\xrightarrow{ML}$ phonemes → words → transcripts y

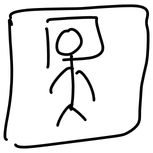
End to End:-

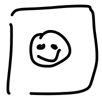
$m = 10{,}000h - 100{,}000h$

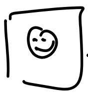
X audio ⟶ y transcripts

→ need a lot of data for end to end aproach.

Face recognition :-

 X (image) → y identity

① detect face and crop it  } works much better.

②  → identity.

 avery data have.

→ 2 seperate easy problems.

→ Task A, Task B have individual more data.

Machine translation :-

before: English → text analysis . . . - . → french

today: English (x) → French (y) work very well.

Estimate child's age :-

before

X (xray image of hand of child) → $\underbrace{\text{bones}}_{①}$ → $\underbrace{\text{age}}_{②}$ y can work better.

today X → age (y) (doesn't work very well because of less data).

Lec7: Whether to use End to End Deep learning :-

Pros:

- Lets the data speak (hopefully NN learns best $f(x) = y$)
- Less hand-designing of components needed.

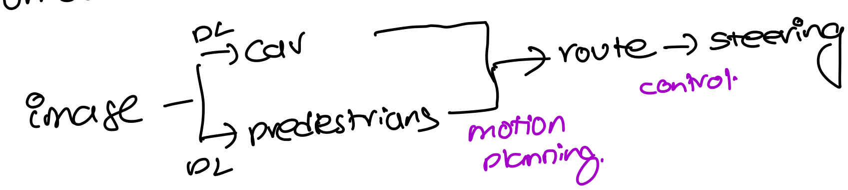
Cons:

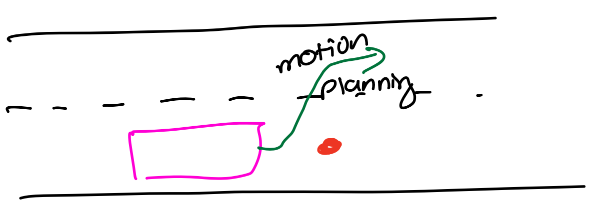
- May need large amount data which is hard to collect.
- Excludes pontentially usefully hand designed components. which can be very helpful and can be cheap.

Key question:

Do you have sufficient data to learn a function of the complexity needed to map $X \to y$?

Autonomous car:-





- Use DL to learn individual components
- carefully choose $X \to Y$ depending on what you can data for.

image $\longrightarrow$ steering (?) (work in progress).