Week-2 Error Analysis

In case if the cat classifier identifies some dogs as cat. Should you teach the classifier some features of dog? Check if it may work, by error analysis.

(accuracy = 90%)

Error analysis:

1) Get all mislabeled dev set examples
2) count up how many are dogs

For e.g. if out of 100 mislabeled set, only 5 are dogs, it will be 5% which will decrease error from 10% to 9.5%.

Evaluate multiple ideas in parallel

Image	Dog	Cats	Blurry	Instagram	Comments
1	✓			✓	
2		✓	✓		
⋮					
% of total	8%	43%	61%	12%	

Tick the ones which give error.
The one with highest error should be fixed first to improve accuracy more.

Incorrectly labeled Examples in Training set

→ If the mislabelled data is random, then one should leave it as it is, as DL algos are quite robust to random errors in the training set

→ But it is not robust to systematic errors; if white dogs are always classified as cats, the model will learn it

→ There may also be incorrectly labelled data in dev set too.

→ In such a case draw a table as 64 (prev. page last) add "incorretly labeled" column. Calculate error %.

→ If the error makes significant difference to evaluat^n of algorithm, only then should it be fixed.

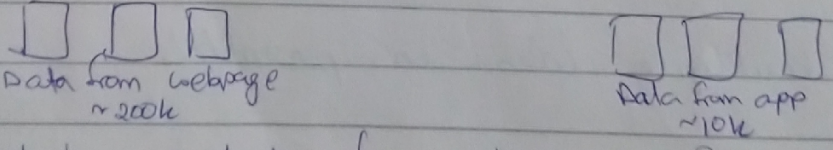
Correcting incorrect dev/test set examples

→ Apply same process to dev & test sets, as, they need to be from same distribution.

→ Consider examining examples that your algo got right, as well as wrong. It may happen that the accuracy is 98% in such a case, it is easier to focus on wrongs (2%). Thus, this trick isn't used always

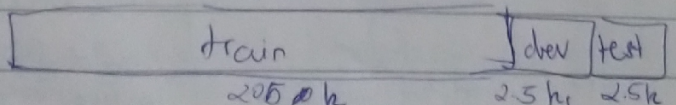
→ Train & dev/test data may now come from slightly different distributions. That is, its okay if the same process is not appled to training set.

Training and Testing on different distributions



Need to classify cat images from mobile app

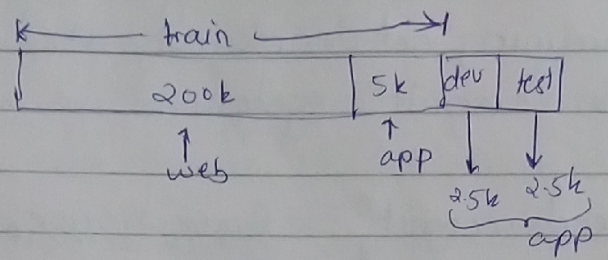
Option-1 Shuffle all 210k and divide



→ may contain majority of webpage images

So it may not give good accuracy

Option 2



Bias And Variance when training and dev/test come from different distributions

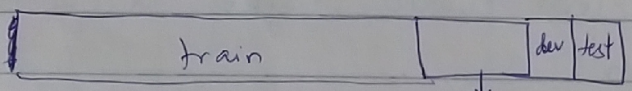
Assume humans ≈ 0% error

Train. error	1%
Dev error	10%

This dev error cannot be said as not fitted properly. Because may be training set was too easy to train Or dev set has some data, that was not recognized before.

In such a case,

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

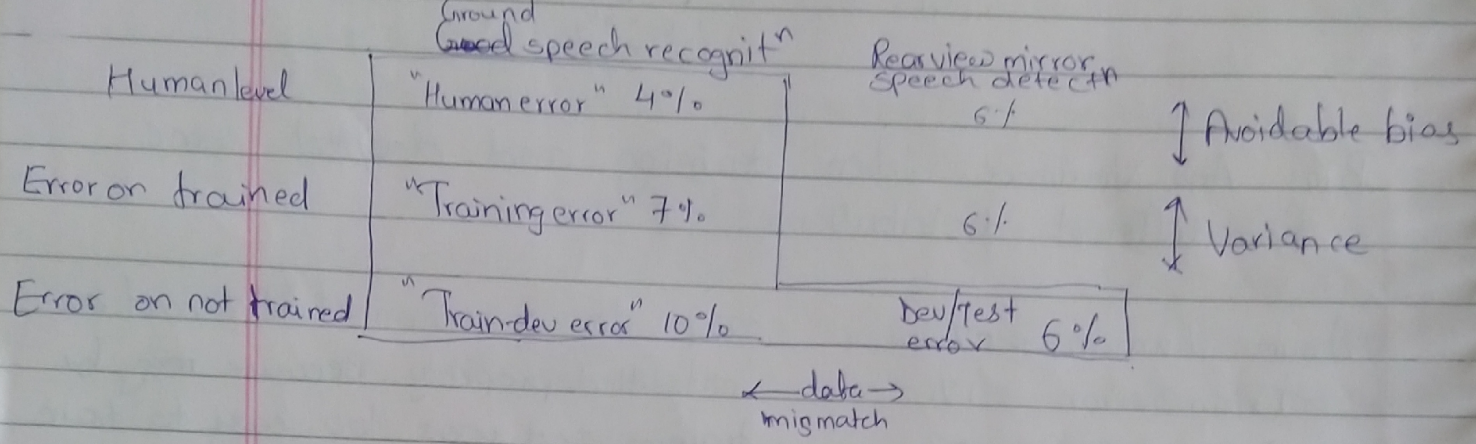


			Human error ≈ 0%	
Training error	1%	↕ variance	1%	10% ↕ Avoidable bias
Train-dev error	9%		1.5%	11%
Dev error	10%		10% ↕ data mismatch	12%

Finally

Human error	4%	↕ Avoidable bias
Training set error	7%	↕ variance
Training-dev "	10%	↕ data mismatch
Dev error	12%	↕ degree of overfitting to dev set
Test error	12%	

A more general formulat^n can be done.
A rear view mirror speech recognit^n system needs to be made. The training set contains other data of other ground speech recognit^n systems.



Solving Data mismatch problem

1. Do error analysis. Find which type of data has error.
2. One way to generate data, is artificial data synthesis.
   For eg. if your model makes error due to car noise, then, take a clear audio, take a car noise audio and combine them
3. It may be possible that clear audio is of ~10k hrs, while car noise is of 1 hr. One may repeat the car noise audio and add to background. But in such a case, the model will overfit for that 1hr car noise audio.

Transfer Learning

→ Taking a network already trained on some task, and then using it for another task ⇒ Transfer Learning.
→ One may remove the last layer and retrain it.
→ If one has more data, all the layers can also be trained.

In case if last layer (or last 2) is trained, it is called fine-tuning.

and the model is called pre-trained.

→ You may add more than one layers, if you want.

→ When should you use it?

1. When the data you are transfering from is high and the data you are trasfering to is less
   For eg. millions of egs. of Image Recognit" are available, but less for radiology diagnosis
2. Transfer from A ⟶ B, both A & B have same i/p x. (image, audio, -)
3. Low level features of A could be helpful for learning B

What happens, when you have multi-task?
For e.g. Auto. driving. One has to check for other cars, stop signs, pedestrians, traffic lights, etc
We use multi-task learning.

$$Y = \begin{bmatrix} \text{cars} & 1 & 1 & \cdots & 0 & \cdots \\ \text{stopsign} & 0 & 1 & \cdots & 0 & \cdots \\ \text{pedestrian} & ? & ? & & 1 & \\ \text{trafficlight} & ? & ? & & ? & \end{bmatrix}$$

↑ single image, multiple labels

When is multi-task helpful?

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

   eg. A_1 1000 examples
   A_i 1000 "
   A_{100} 1000 "

   here, 1000 egs for a particulare task is not going to hamper learning, as the rest of tasks egs will be there to help
3. One can train a big enough NN to do well on all tasks

When one designs a NN for each task, it hurts performance, but if you train a big NN, it rarely hurts performance

End-to-end learning

Speech recognitn eg

x → (MFCC) features → (ML) Phonemes → words → Transcript → transcript (y)

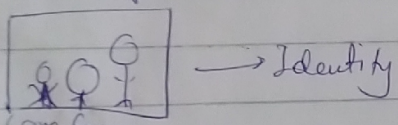
end to end: audio → transcript

End to end learning needs a lot of data. If you have small dataset, use the 1st approach If you have medium dataset, use this.

audio → Phonemes → Transcript

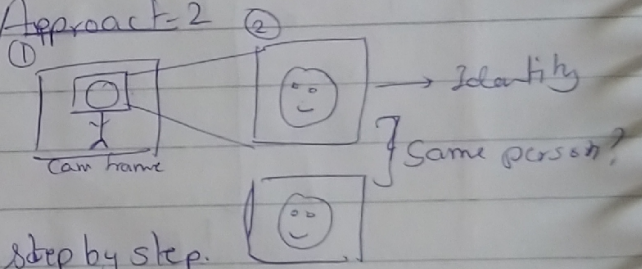
Face Recognitn (at company gate)

~~to~~ Approach-1



end-to-end

Approach-2



step by step.

It turns out that approach-2 was better & simpler. Because in approach-1, we need to have a large dataset. Data of each employee in different positions of frame which cannot be possible ~~to~~ in practice today.

Use cases of end to end learning: machine translatn

Pros and cons of end to end ~~to~~ deep learning

Pros:

1. Let ~~It~~ big n/w figure out x → y
2. Less hand designing of components needed.

Cons:

May need large amount of data

Excludes, potentially useful hand-designed components

When apply 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?