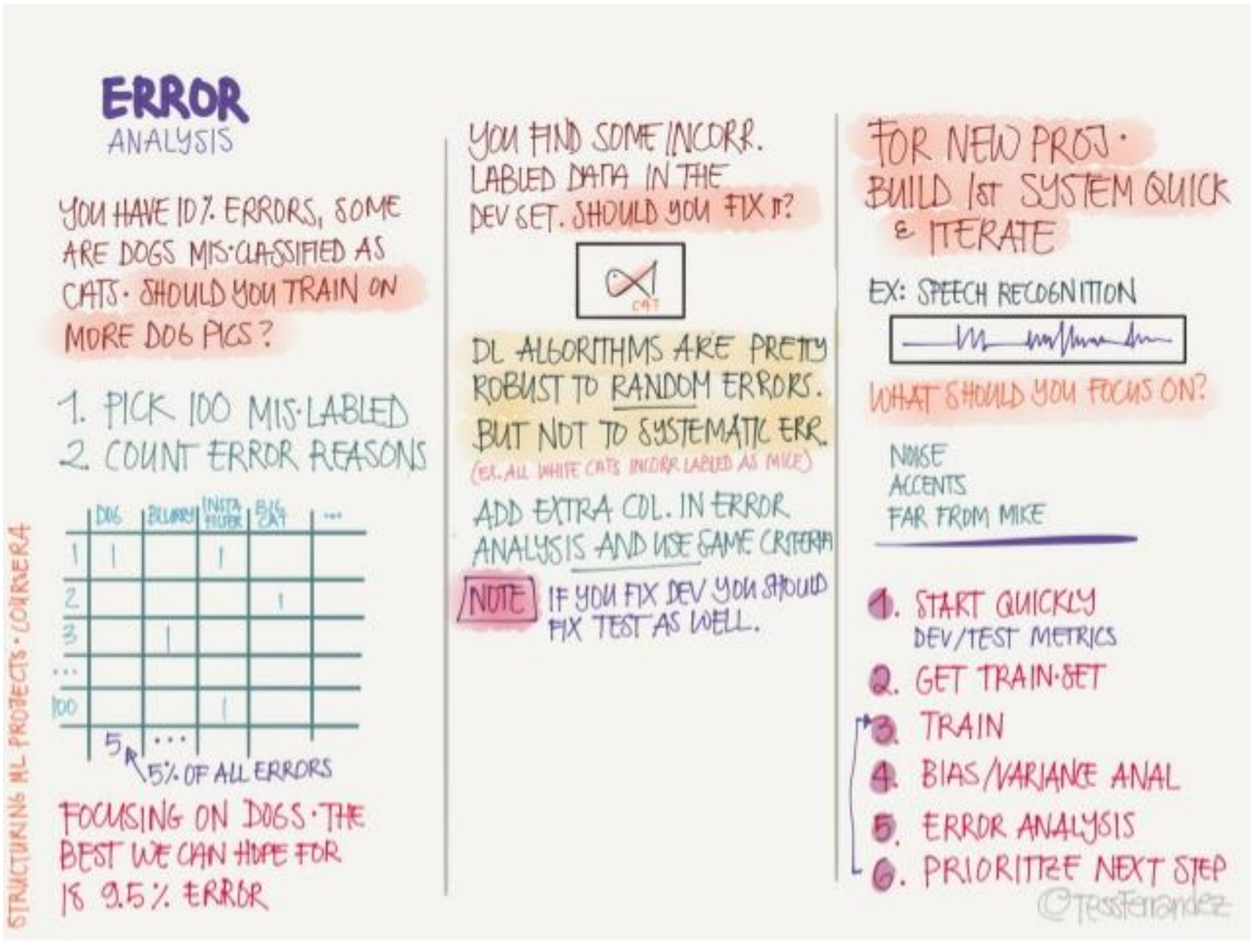


**ERROR**
ANALYSIS

YOU HAVE 10% ERRORS, SOME ARE DOGS MIS·CLASSIFIED AS CATS· SHOULD YOU TRAIN ON MORE DOG PICS?

1. PICK 100 MIS·LABLED
2. COUNT ERROR REASONS

| | DOG | BLURRY | INSTA FILTER | BIG CAT | ... |
|---|---|---|---|---|---|
| 1 | 1 | | 1 | | |
| 2 | | | | 1 | |
| 3 | | 1 | | | |
| ... | | | | | |
| 100 | | | 1 | | |
| | 5 | ... | | | |

5% OF ALL ERRORS

FOCUSING ON DOGS· THE BEST WE CAN HOPE FOR IS 9.5% ERROR

YOU FIND SOME INCORR. LABLED DATA IN THE DEV SET. SHOULD YOU FIX IT?

DL ALGORITHMS ARE PRETTY ROBUST TO RANDOM ERRORS. BUT NOT TO SYSTEMATIC ERR.
(EX. ALL WHITE CATS INCORR LABLED AS MICE)

ADD EXTRA COL. IN ERROR ANALYSIS AND USE SAME CRITERIA

NOTE IF YOU FIX DEV YOU SHOULD FIX TEST AS WELL.

FOR NEW PROJ· BUILD 1ST SYSTEM QUICK & ITERATE

EX: SPEECH RECOGNITION

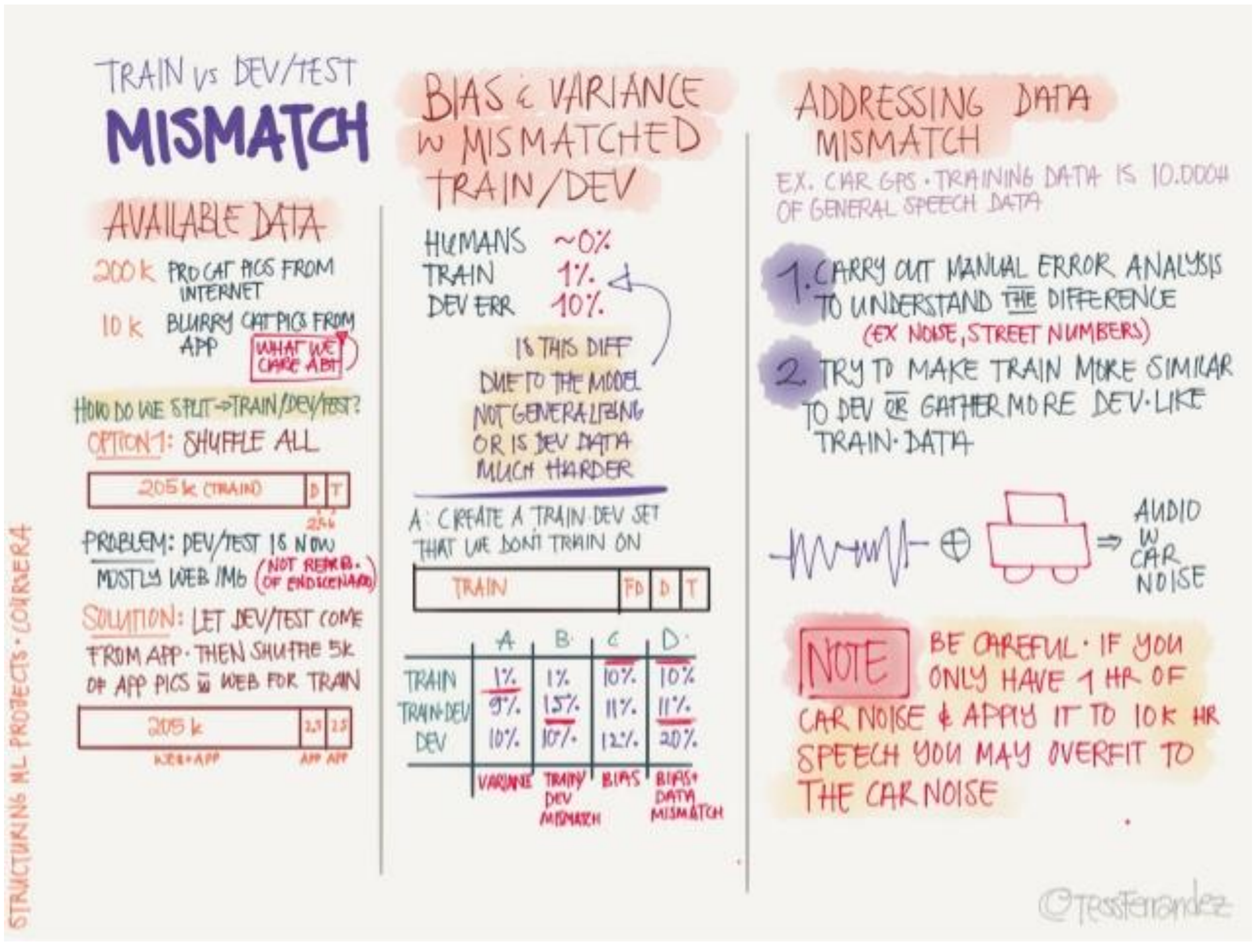
WHAT SHOULD YOU FOCUS ON?

NOISE
ACCENTS
FAR FROM MIKE

1. START QUICKLY
   DEV/TEST METRICS
2. GET TRAIN·SET
3. TRAIN
4. BIAS/VARIANCE ANAL
5. ERROR ANALYSIS
6. PRIORITIZE NEXT STEP

STRUCTURING ML PROJECTS · COURSERA

@TessFerrandez



TRAIN vs DEV/TEST

# MISMATCH

## AVAILABLE DATA

200k PRO CAT PICS FROM INTERNET

10 k BLURRY CAT PICS FROM APP WHAT WE CARE ABT

HOW DO WE SPLIT → TRAIN/DEV/TEST?

OPTION 1: SHUFFLE ALL

| 205 k (TRAIN) | D | T |
|---|---|---|
|  | 2.5 | k |

PROBLEM: DEV/TEST IS NOW MOSTLY WEB IMG (NOT REPRES. OF END SCENARIO)

SOLUTION: LET DEV/TEST COME FROM APP · THEN SHUFFLE 5k OF APP PICS W WEB FOR TRAIN

| 205 k | 2.5 | 2.5 |
|---|---|---|
| WEB+APP | APP | APP |

STRUCTURING ML PROJECTS · COURSERA

## BIAS & VARIANCE W MISMATCHED TRAIN/DEV

| | |
|---|---|
| HUMANS | ~0% |
| TRAIN | 1% |
| DEV ERR | 10% |

IS THIS DIFF DUE TO THE MODEL NOT GENERALIZING OR IS DEV DATA MUCH HARDER

A: CREATE A TRAIN-DEV SET THAT WE DON'T TRAIN ON

| TRAIN | TD | D | T |
|---|---|---|---|

| | A | B | C | D |
|---|---|---|---|---|
| TRAIN | 1% | 1% | 10% | 10% |
| TRAIN-DEV | 9% | 15% | 11% | 11% |
| DEV | 10% | 10% | 12% | 20% |
| | VARIANCE | TRAIN/DEV MISMATCH | BIAS | BIAS + DATA MISMATCH |

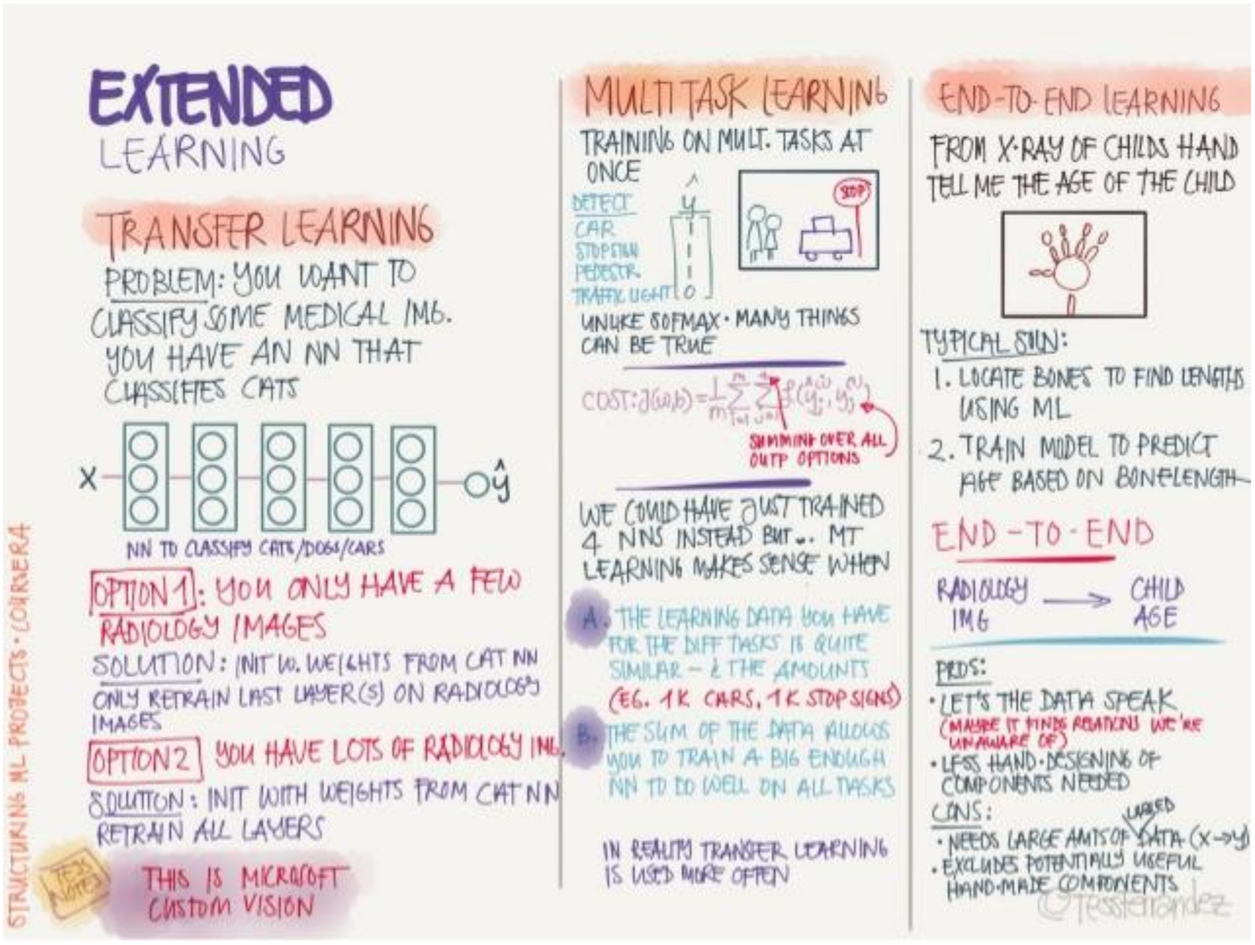
## ADDRESSING DATA MISMATCH

EX. CAR GPS · TRAINING DATA IS 10.000H OF GENERAL SPEECH DATA

1. CARRY OUT MANUAL ERROR ANALYSIS TO UNDERSTAND THE DIFFERENCE (EX NOISE, STREET NUMBERS)
2. TRY TO MAKE TRAIN MORE SIMILAR TO DEV OR GATHER MORE DEV-LIKE TRAIN-DATA

NOTE BE CAREFUL · IF YOU ONLY HAVE 1 HR OF CAR NOISE & APPLY IT TO 10K HR SPEECH YOU MAY OVERFIT TO THE CAR NOISE

@TessFerrandez



EXTENDED LEARNING

TRANSFER LEARNING

PROBLEM: YOU WANT TO CLASSIFY SOME MEDICAL IMG. YOU HAVE AN NN THAT CLASSIFIES CATS

X
$\hat{y}$
NN TO CLASSIFY CATS/DOGS/CARS

OPTION 1: YOU ONLY HAVE A FEW RADIOLOGY IMAGES

SOLUTION: INIT W. WEIGHTS FROM CAT NN ONLY RETRAIN LAST LAYER(S) ON RADIOLOGY IMAGES

OPTION 2 YOU HAVE LOTS OF RADIOLOGY IMG.

SOLUTION: INIT WITH WEIGHTS FROM CAT NN RETRAIN ALL LAYERS

THIS IS MICROSOFT CUSTOM VISION

MULTI TASK LEARNING

TRAINING ON MULT. TASKS AT ONCE

UNLIKE SOFMAX · MANY THINGS CAN BE TRUE

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

SUMMING OVER ALL OUTP OPTIONS

WE COULD HAVE JUST TRAINED 4 NNS INSTEAD BUT.. MT LEARNING MAKES SENSE WHEN

A. THE LEARNING DATA YOU HAVE FOR THE DIFF TASKS IS QUITE SIMILAR – & THE AMOUNTS (EG. 1K CARS, 1K STOP SIGNS)

B. THE SUM OF THE DATA ALLOWS YOU TO TRAIN A BIG ENOUGH NN TO DO WELL ON ALL TASKS

IN REALITY TRANSFER LEARNING IS USED MORE OFTEN

END-TO-END LEARNING

FROM X-RAY OF CHILDS HAND TELL ME THE AGE OF THE CHILD

TYPICAL SOLN:

1. LOCATE BONES TO FIND LENGTHS USING ML
2. TRAIN MODEL TO PREDICT AGE BASED ON BONELENGTH

END-TO-END

RADIOLOGY IMG $\longrightarrow$ CHILD AGE

PROS:

- LET'S THE DATA SPEAK (MAYBE IT FINDS RELATIONS WE'RE UNAWARE OF)
- LESS HAND-DESIGNING OF COMPONENTS NEEDED

CONS:

- NEEDS LARGE AMTS OF LABELED DATA (X→Y)
- EXCLUDES POTENTIALLY USEFUL HAND-MADE COMPONENTS

# ML Strategy 2

## Carrying out error analysis

- Error analysis - process of manually examining mistakes that your algorithm is making. It can give you insights into what to do next. E.g.:
  - In the cat classification example, if you have 10% error on your dev set and you want to decrease the error.
  - You discovered that some of the mislabeled data are dog pictures that look like cats. Should you try to make your cat classifier do better on dogs (this could take some weeks)?
  - Error analysis approach:
    - Get 100 mislabeled dev set examples at random.
    - Count up how many are dogs.
    - if 5 of 100 are dogs then training your classifier to do better on dogs will decrease your error up to 9.5% (called ceiling), which can be too little.
    - if 50 of 100 are dogs then you could decrease your error up to 5%, which is reasonable and you should work on that.

- Based on the last example, error analysis helps you to analyze the error before taking an action that could take lot of time with no need.
- Sometimes, you can evaluate multiple error analysis ideas in parallel and choose the best idea. Create a spreadsheet to do that and decide, e.g.:

| Image | Dog | Great Cats | blurry | Instagram filters | Comments |
|---|---|---|---|---|---|
| 1 | ✓ | | | ✓ | Pitbull |
| 2 | ✓ | | ✓ | ✓ | |
| 3 | | | | | Rainy day at zoo |
| 4 | | ✓ | | | |
| .... | | | | | |
| **% totals** | **8%** | **43%** | **61%** | **12%** | |

- In the last example you will decide to work on great cats or blurry images to improve your performance.
- This quick counting procedure, which you can often do in, at most, small numbers of hours can really help you make much better prioritization decisions, and understand how promising different approaches are to work on.

## Cleaning up incorrectly labeled data

- DL algorithms are quite robust to random errors in the training set but less robust to systematic errors. But it's OK to go and fix these labels if you can.
- If you want to check for mislabeled data in dev/test set, you should also try error analysis with the mislabeled column. Ex:

| Image | Dog | Great Cats | blurry | Mislabeled | Comments |
|---|---|---|---|---|---|
| 1 | ✓ | | | | |
| 2 | ✓ | | ✓ | | |
| 3 | | | | | |
| 4 | | ✓ | | | |
| .... | | | | | |
| **% totals** | **8%** | **43%** | **61%** | **6%** | |

  - Then:
    - If overall dev set error: 10%
      - Then errors due to incorrect data: 0.6%
      - Then errors due to other causes: 9.4%
    - Then you should focus on the 9.4% error rather than the incorrect data.
- Consider these guidelines while correcting the dev/test mislabeled examples:
  - Apply the 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. (Not always done if you reached a good accuracy)
    - Train and (dev/test) data may now come from a slightly different distributions.
    - It's very important to have dev and test sets to come from the same distribution. But it could be OK for a train set to come from slightly other distribution.

## Build your first system quickly, then iterate

- The steps you take to make your deep learning project:
    - Setup dev/test set and metric
    - Build initial system quickly
    - Use Bias/Variance analysis & Error analysis to prioritize next steps.

## Training and testing on different distributions

- A lot of teams are working with deep learning applications that have training sets that are different from the dev/test sets due to the hunger of deep learning to data.
- There are some strategies to follow up when training set distribution differs from dev/test sets distribution.
    - Option one (not recommended): shuffle all the data together and extract randomly training and dev/test sets.
        - Advantages: all the sets now come from the same distribution.
        - Disadvantages: the other (real world) distribution that was in the dev/test sets will occur less in the new dev/test sets and that might be not what you want to achieve.
    - Option two: take some of the dev/test set examples and add them to the training set.
        - Advantages: the distribution you care about is your target now.
        - Disadvantage: the distributions in training and dev/test sets are now different. But you will get a better performance over a long time.

## Bias and Variance with mismatched data distributions

- Bias and Variance analysis changes when training and Dev/test set is from the different distribution.
- Example: the cat classification example. Suppose you've worked in the example and reached this
    - Human error: 0%
    - Train error: 1%
    - Dev error: 10%
    - In this example, you'll think that this is a variance problem, but because the distributions aren't the same you can't tell for sure. Because it could be that train set was easy to train on, but the dev set was more difficult.
- To solve this issue we create a new set called train-dev set as a random subset of the training set (so it has the same distribution) and we get:
    - Human error: 0%

    - Train error: 1%
    - Train-dev error: 9%
    - Dev error: 10%
    - Now we are sure that this is a high variance problem.
- Suppose we have a different situation:
    - Human error: 0%
    - Train error: 1%
    - Train-dev error: 1.5%
    - Dev error: 10%
    - In this case we have something called *Data mismatch* problem.
- Conclusions:
    1. Human-level error (proxy for Bayes error)
    2. Train error
        - Calculate `avoidable bias = training error - human level error`
        - If the difference is big then its **Avoidable bias** problem then you should use a strategy for high **bias**.
    3. Train-dev error
        - Calculate `variance = training-dev error - training error`
        - If the difference is big then its high **variance** problem then you should use a strategy for solving it.
    4. Dev error
        - Calculate `data mismatch = dev error - train-dev error`
        - If difference is much bigger then train-dev error its **Data mismatch** problem.
    5. Test error
        - Calculate `degree of overfitting to dev set = test error - dev error`
        - Is the difference is big (positive) then maybe you need to find a bigger dev set (dev set and test set come from the same distribution, so the only way for there to be a huge gap here, for it to do much better on the dev set than the test set, is if you somehow managed to overfit the dev set).
- Unfortunately, there aren't many systematic ways to deal with data mismatch. There are some things to try about this in the next section.

## Addressing data mismatch

- There aren't completely systematic solutions to this, but there some things you could try.

1. Carry out manual error analysis to try to understand the difference between training and dev/test sets.
2. Make training data more similar, or collect more data similar to dev/test sets.

- If your goal is to make the training data more similar to your dev set one of the techniques you can use **Artificial data synthesis** that can help you make more training data.

  - Combine some of your training data with something that can convert it to the dev/test set distribution.
    - Examples:
      1. Combine normal audio with car noise to get audio with car noise example.
      2. Generate cars using 3D graphics in a car classification example.
  - Be cautious and bear in mind whether or not you might be accidentally simulating data only from a tiny subset of the space of all possible examples because your NN might overfit these generated data (like particular car noise or a particular design of 3D graphics cars).

## Transfer learning

- Apply the knowledge you took in a task A and apply it in another task B.
- For example, you have trained a cat classifier with a lot of data, you can use the part of the trained NN it to solve x-ray classification problem.
- To do transfer learning, delete the last layer of NN and it's weights and:
  1. Option 1: if you have a small data set - keep all the other weights as a fixed weights. Add a new last layer(-s) and initialize the new layer weights and feed the new data to the NN and learn the new weights.
  2. Option 2: if you have enough data you can retrain all the weights.
- Option 1 and 2 are called **fine-tuning** and training on task A called **pretraining**.
- When transfer learning make sense:
  - Task A and B have the same input X (e.g. image, audio).
  - You have a lot of data for the task A you are transferring from and relatively less data for the task B your transferring to.
  - Low level features from task A could be helpful for learning task B.

## Multi-task learning

- Whereas in transfer learning, you have a sequential process where you learn from task A and then transfer that to task B. In multi-task learning, you start off simultaneously, trying to have one neural network do several things at the same time. And then each of these tasks helps hopefully all of the other tasks.
- Example:
  - You want to build an object recognition system that detects pedestrians, cars, stop signs, and traffic lights (image has multiple labels).
  - Then Y shape will be `(4,m)` because we have 4 classes and each one is a binary one.
  - Then
    `Cost = (1/m) * sum(sum(L(y_hat(i)_j, y(i)_j))), i = 1..m, j = 1..4`, where
    `L = - y(i)_j * log(y_hat(i)_j) - (1 - y(i)_j) * log(1 - y_hat(i)_j)`
- In the last example you could have trained 4 neural networks separately but if some of the earlier features in neural network can be shared between these different types of objects, then you find that training one neural network to do four things results in better

performance than training 4 completely separate neural networks to do the four tasks separately.

- Multi-task learning will also work if y isn't complete for some labels. For example:
- `Y = [1 ? 1 ...]`
- `    [0 0 1 ...]`
- `    [? 1 ? ...]`
    - And in this case it will do good with the missing data, just the loss function will be different:
      `Loss = (1/m) * sum(sum(L(y_hat(i)_j, y(i)_j) for all j which y(i)_j != ?))`
- Multi-task learning makes sense:
    1. Training on a set of tasks that could benefit from having shared lower-level features.
    2. Usually, amount of data you have for each task is quite similar.
    3. Can train a big enough network to do well on all the tasks.
- If you can train a big enough NN, the performance of the multi-task learning compared to splitting the tasks is better.
- Today transfer learning is used more often than multi-task learning.

## What is end-to-end deep learning?

- Some systems have multiple stages to implement. An end-to-end deep learning system implements all these stages with a single NN.
- Example 1:
    - Speech recognition system:
    - `Audio ---> Features --> Phonemes --> Words --> Transcript    # non-end-to-end system`
    - `Audio ---------------------------------------> Transcript    # end-to-end deep learning system`
    - End-to-end deep learning gives data more freedom, it might not use phonemes when training!
- To build the end-to-end deep learning system that works well, we need a big dataset (more data then in non end-to-end system). If we have a small dataset the ordinary implementation could work just fine.
- Example 2:
    - Face recognition system:
    - `Image ---------------------> Face recognition    # end-to-end deep learning system`
    - `Image --> Face detection --> Face recognition    # deep learning system - best approach for now`
    - In practice, the best approach is the second one for now.
    - In the second implementation, it's a two steps approach where both parts are implemented using deep learning.
    - Its working well because it's harder to get a lot of pictures with people in front of the camera than getting faces of people and compare them.
    - In the second implementation at the last step, the NN takes two faces as an input and outputs if the two faces are the same person or not.
- Example 3:

    - Machine translation system:
    - `English --> Text analysis --> ... --> French    # non-end-to-end system`
    - `English ---------------------------> French    # end-to-end deep learning system - best approach`
    - Here end-to-end deep leaning system works better because we have enough data to build it.
- Example 4:
    - Estimating child's age from the x-ray picture of a hand:
- `Image --> Bones --> Age    # non-end-to-end system - best approach for now`
- `Image ------------> Age    # end-to-end system`
    - In this example non-end-to-end system works better because we don't have enough data to train end-to-end system.

### Whether to use end-to-end deep learning

- Pros of end-to-end deep learning:
    - Let the data speak. By having a pure machine learning approach, your NN learning input from X to Y may be more able to capture whatever statistics are in the data, rather than being forced to reflect human preconceptions.
    - Less hand-designing of components needed.
- Cons of end-to-end deep learning:
    - May need a large amount of data.
    - Excludes potentially useful hand-design components (it helps more on the smaller dataset).
- 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?
    - Use ML/DL to learn some individual components.
    - When applying supervised learning you should carefully choose what types of X to Y mappings you want to learn depending on what task you can get data for.

# Week 2 Quiz - Autonomous driving (case study)

1. You are just getting started on this project. What is the first thing you do? Assume each of the steps below would take about an equal amount of time (a few days).
    - Spend a few days training a basic model and see what mistakes it makes.

      As discussed in lecture, applied ML is a highly iterative process. If you train a basic model and carry out error analysis (see what mistakes it makes) it will help point you in more promising directions.

2. Your goal is to detect road signs (stop sign, pedestrian crossing sign, construction ahead sign) and traffic signals (red and green lights) in images. The goal is to recognize which

of these objects appear in each image. You plan to use a deep neural network with ReLU units in the hidden layers.

For the output layer, a softmax activation would be a good choice for the output layer because this is a multi-task learning problem. True/False?

- ☐ True
- ☑ False

  Softmax would be a good choice if one and only one of the possibilities (stop sign, speed bump, pedestrian crossing, green light and red light) was present in each image.

3. You are carrying out error analysis and counting up what errors the algorithm makes. Which of these datasets do you think you should manually go through and carefully examine, one image at a time?
   - ☐ 10,000 randomly chosen images
   - ☐ 500 randomly chosen images
   - ☑ 500 images on which the algorithm made a mistake
   - ☐ 10,000 images on which the algorithm made a mistake
4. After working on the data for several weeks, your team ends up with the following data:
   - 100,000 labeled images taken using the front-facing camera of your car.
   - 900,000 labeled images of roads downloaded from the internet.

   Each image's labels precisely indicate the presence of any specific road signs and traffic signals or combinations of them. For example, y(i) = [1 0 0 1 0] means the image contains a stop sign and a red traffic light. Because this is a multi-task learning problem, you need to have all your y(i) vectors fully labeled. If one example is equal to [0 ? 1 1 ?] then the learning algorithm will not be able to use that example. True/False?

   - ☐ True
   - ☑ False

     As seen in the lecture on multi-task learning, you can compute the cost such that it is not influenced by the fact that some entries haven't been labeled.

5. The distribution of data you care about contains images from your car's front-facing camera; which comes from a different distribution than the images you were able to find and download off the internet. How should you split the dataset into train/dev/test sets?
   - ☐ Mix all the 100,000 images with the 900,000 images you found online. Shuffle everything. Split the 1,000,000 images dataset into 600,000 for the training set, 200,000 for the dev set and 200,000 for the test set.

- ☐ Mix all the 100,000 images with the 900,000 images you found online. Shuffle everything. Split the 1,000,000 images dataset into 980,000 for the training set, 10,000 for the dev set and 10,000 for the test set.
- ☑ Choose the training set to be the 900,000 images from the internet along with 80,000 images from your car's front-facing camera. The 20,000 remaining images will be split equally in dev and test sets.
- ☐ Choose the training set to be the 900,000 images from the internet along with 20,000 images from your car's front-facing camera. The 80,000 remaining images will be split equally in dev and test sets.

  As seen in lecture, it is important that your dev and test set have the closest possible distribution to "real"-data. It is also important for the training set to contain enough "real"-data to avoid having a data-mismatch problem.

6. Assume you've finally chosen the following split between of the data:
   - Training 940,000 images randomly picked from (900,000 internet images + 60,000 car's front-facing camera images) 8.8%
   - Training-Dev 20,000 images randomly picked from (900,000 internet images + 60,000 car's front-facing camera images) 9.1%
   - Dev 20,000 images from your car's front-facing camera 14.3%
   - Test 20,000 images from the car's front-facing camera 14.8%

   You also know that human-level error on the road sign and traffic signals classification task is around 0.5%. Which of the following are True? (Check all that apply).

   - You have a large avoidable-bias problem because your training error is quite a bit higher than the human-level error.
   - You have a large data-mismatch problem because your model does a lot better on the training-dev set than on the dev set.
7. Based on table from the previous question, a friend thinks that the training data distribution is much easier than the dev/test distribution. What do you think?
   - There's insufficient information to tell if your friend is right or wrong.

     The algorithm does better on the distribution of data it trained on. But you don't know if it's because it trained on that no distribution or if it really is easier. To get a better sense, measure human-level error separately on both distributions.

8. You decide to focus on the dev set and check by hand what are the errors due to. Here is a table summarizing your discoveries:
   - Overall dev set error 14.3%
   - Errors due to incorrectly labeled data 4.1%
   - Errors due to foggy pictures 8.0%
   - Errors due to rain drops stuck on your car's front-facing camera 2.2%
   - Errors due to other causes 1.0%

In this table, 4.1%, 8.0%, etc.are a fraction of the total dev set (not just examples your algorithm mislabeled). I.e. about 8.0/14.3 = 56% of your errors are due to foggy pictures.

The results from this analysis implies that the team's highest priority should be to bring more foggy pictures into the training set so as to address the 8.0% of errors in that category. True/False?

- ✓ False because this would depend on how easy it is to add this data and how much you think your team thinks it'll help.
- ☐ True because it is the largest category of errors. As discussed in lecture, we should prioritize the largest category of error to avoid wasting the team's time.
- ☐ True because it is greater than the other error categories added together (8.0 > 4.1+2.2+1.0).
- ☐ False because data augmentation (synthesizing foggy images by clean/non-foggy images) is more efficient.

9. You can buy a specially designed windshield wiper that help wipe off some of the raindrops on the front-facing camera. Based on the table from the previous question, which of the following statements do you agree with?
   - 2.2% would be a reasonable estimate of the maximum amount this windshield wiper could improve performance.

     You will probably not improve performance by more than 2.2% by solving the raindrops problem. If your dataset was infinitely big, 2.2% would be a perfect estimate of the improvement you can achieve by purchasing a specially designed windshield wiper that removes the raindrops.

10. You decide to use data augmentation to address foggy images. You find 1,000 pictures of fog off the internet, and "add" them to clean images to synthesize foggy days, like this:

    Which of the following statements do you agree with? (Check all that apply.)

    - So long as the synthesized fog looks realistic to the human eye, you can be confident that the synthesized data is accurately capturing the distribution of real foggy images, since human vision is very accurate for the problem you're solving.

      If the synthesized images look realistic, then the model will just see them as if you had added useful data to identify road signs and traffic signals in a foggy weather. I will very likely help.

11. After working further on the problem, you've decided to correct the incorrectly labeled data on the dev set. Which of these statements do you agree with? (Check all that apply).
    - You should not correct incorrectly labeled data in the training set as well so as to avoid your training set now being even more different from your dev set.

Deep learning algorithms are quite robust to having slightly different train and dev distributions.

- You should also correct the incorrectly labeled data in the test set, so that the dev and test sets continue to come from the same distribution

  Because you want to make sure that your dev and test data come from the same distribution for your algorithm to make your team's iterative development process is efficient.

12. So far your algorithm only recognizes red and green traffic lights. One of your colleagues in the startup is starting to work on recognizing a yellow traffic light. (Some countries call it an orange light rather than a yellow light; we'll use the US convention of calling it yellow.) Images containing yellow lights are quite rare, and she doesn't have enough data to build a good model. She hopes you can help her out using transfer learning.

    What do you tell your colleague?

    - She should try using weights pre-trained on your dataset, and fine-tuning further with the yellow-light dataset.

      You have trained your model on a huge dataset, and she has a small dataset. Although your labels are different, the parameters of your model have been trained to recognize many characteristics of road and traffic images which will be useful for her problem. This is a perfect case for transfer learning, she can start with a model with the same architecture as yours, change what is after the last hidden layer and initialize it with your trained parameters.

13. Another colleague wants to use microphones placed outside the car to better hear if there're other vehicles around you. For example, if there is a police vehicle behind you, you would be able to hear their siren. However, they don't have much to train this audio system. How can you help?
    - Neither transfer learning nor multi-task learning seems promising.

      The problem he is trying to solve is quite different from yours. The different dataset structures make it probably impossible to use transfer learning or multi-task learning.

14. To recognize red and green lights, you have been using this approach:
    - (A) Input an image (x) to a neural network and have it directly learn a mapping to make a prediction as to whether there's a red light and/or green light (y).

    A teammate proposes a different, two-step approach:

    - (B) In this two-step approach, you would first (i) detect the traffic light in the image (if any), then (ii) determine the color of the illuminated lamp in the traffic light. Between these two, Approach B is more of an end-to-end approach because it has distinct steps for the input end and the output end. True/False?

- ☐ True
- ☑ False

(A) is an end-to-end approach as it maps directly the input (x) to the output (y).

15. Approach A (in the question above) tends to be more promising than approach B if you have a ________ (fill in the blank).

- ☑ Large training set
- ☐ Multi-task learning problem.
- ☐ Large bias problem.
- ☐ Problem with a high Bayes error.

In many fields, it has been observed that end-to-end learning works better in practice, but requires a large amount of data.