Error Analysis

Learning Objectives:

- Describe multi-task learning and transfer learning
- Recognize bias, variance and data-mismatch by looking at the performances of your algorithm on train/dev/test sets

Error Analysis

1 Carrying out error analysis

2 Cleaning up Incorrectly labeled data

3 Build your first system quickly, then iterate

4 Training and testing on different distributions

5 Bias and Variance with mismatched data distributions

6 Addressing data mismatch

7 Transfer learning

8 Multi-task learning

9 What is end-to-end deep learning

10 Whether to use end-to-end learning

FPT Education

FPT UNIVERSITY

Error Analysis

Carrying out error analysis

Carrying out error analysis

- After identifying the categories of errors the algorithm is making, you can prioritize which ones to focus on and come up with potential solutions to address them, such as collecting more labeled data, adjusting the model architecture, or changing the optimization algorithm.
- It is also important to track the impact of these changes on the development and test set performance to ensure that they are actually improving the algorithm's performance.

Carrying out error analysis

- By understanding the types of errors the model is making, machine learning engineers can prioritize their efforts and focus on the areas that will have the biggest impact on improving the model's performance.
- Error analysis can reveal new categories of errors that were not previously considered and provide insights into how to improve the algorithm's performance.
- By examining the errors in detail, you can identify patterns and trends that might not have been apparent otherwise. This can help you to develop more effective strategies for improving the algorithm's accuracy and efficiency.

Carrying out error analysis

- Correcting mislabeled examples in development set is essential for accurate error analysis and algorithm training. This can be done manually or with the help of crowd-sourcing or other methods.
- By correcting mislabeled examples, you can improve the accuracy of your error analysis and ensure that your algorithm is being trained on the correct data.

Look at dev examples to evaluate ideas

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

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

Evaluate multiple ideas in parallel

- 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

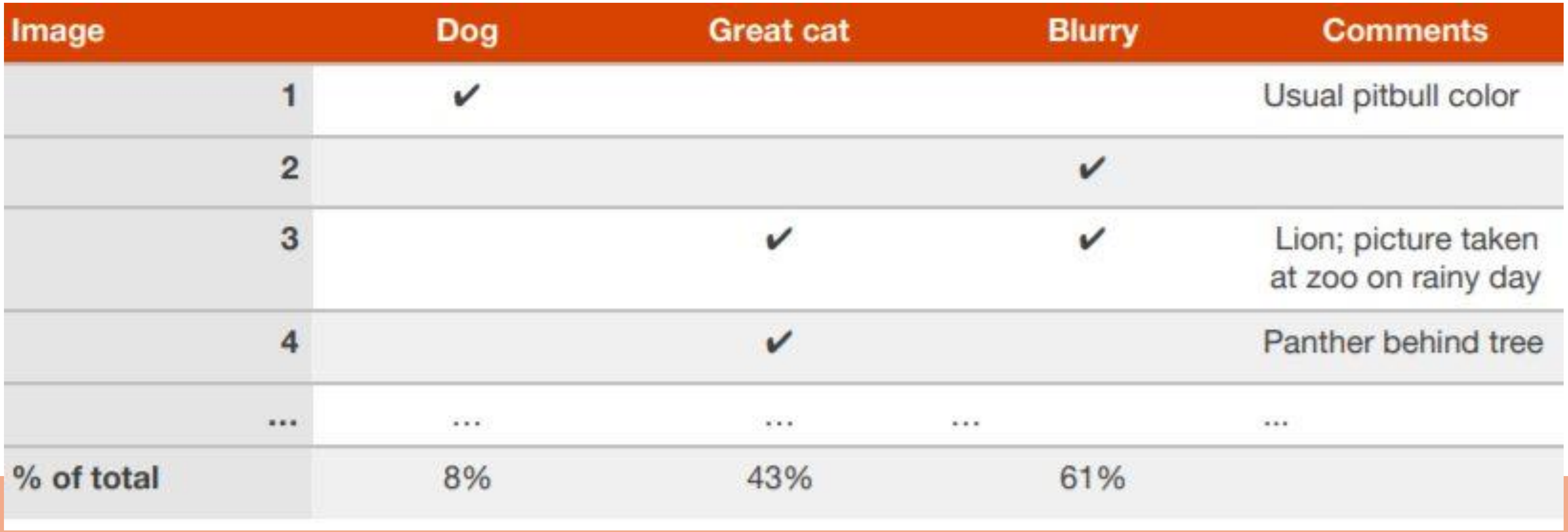


Image	Dog	Great cat	Blurry	Comments
1	✔			Usual pitbull color
2			✔	
3		✔	✔	Lion; picture taken at zoo on rainy day
4		✔		Panther behind tree
...	...	...	...	...
% of total	8%	43%	61%	

FPT Education

FPT UNIVERSITY

Error Analysis

Cleaning up Incorrectly labeled data

Cleaning up Incorrectly labeled data

- The process of supervised learning involves input X and output labels Y, and sometimes you may find that some of the Y labels are incorrect, meaning they were labeled incorrectly by a human labeler. These errors may or may not be worth fixing.
- Deep learning algorithms are usually quite robust to random errors in the training set, as long as the errors are not systematic. So, random or near-random errors can be left as they are, and not much effort needs to be spent on fixing them. However, if the errors are systematic, then they could cause the algorithm to learn incorrectly.

Cleaning up Incorrectly labeled data

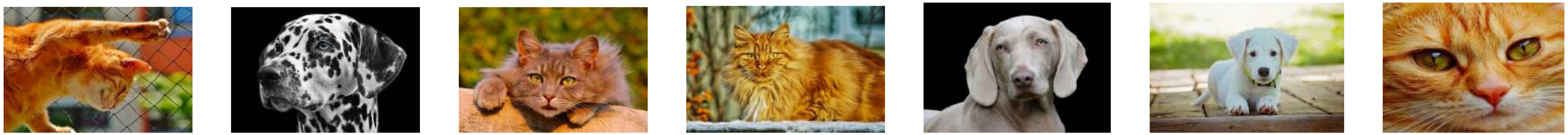
- Additionally, if you decide to fix the incorrect labels in your dev set or test set, it's important to make sure that you correct all instances of the incorrect label, not just the ones that you noticed during error analysis. You can do this by going through the entire dev or test set and checking all labels. Finally, it's important to note that fixing incorrect labels in the dev or test set may require retraining your algorithm, which can be time-consuming and may also require additional labeled data.

Cleaning up Incorrectly labeled data

- Error analysis is indeed a crucial step in improving the accuracy of machine learning algorithms.
- By examining the errors made by the algorithm, you can gain insights into what areas need improvement and prioritize your efforts accordingly.

Incorrectly labeled examples

x



y 1 0 1 1 0 1 1

- Deep learning algorithms are quite robust to random errors in the training set.

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%	

Overall dev set error	10%	2%
Errors due incorrect labels	0.6%	0.6%
Errors due to other causes	9.4%	1.4%

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 right as well as ones it got wrong.

- Train and dev/test data may now come from slightly different distributions.

FPT Education

FPT UNIVERSITY

Error Analysis

Build your first system quickly, then iterate

Build your first system quickly, then iterate

- When building a machine learning application, it can be overwhelming to decide which techniques to prioritize. To tackle this challenge, it is recommended to quickly build an initial system, set up a dev/test set and metric, and evaluate the performance of the system against the set and metric.
- Then, use bias/variance analysis and error analysis to identify areas for improvement and prioritize the next steps accordingly. This iterative process can help to gradually improve the system and achieve better performance.

Build your first system quickly, then iterate

- The initial system in machine learning helps to identify bias and variance, prioritize improvements, and avoid overthinking or making the system too complex.
- This approach is particularly useful for those who are new to the problem or lack expertise in the application area.
- However, for those with significant prior experience or a significant body of literature to draw from, it may be appropriate to build a more complex system from the start.

Build your first system quickly, then iterate

- Overall, the goal of building a machine learning application is to get something that works well. By building your first system quickly and then iterating based on bias/variance analysis and error analysis, you can improve the system and prioritize the areas where the most significant improvements can be made.

Speech recognition example

- Noisy background
  - Café noise
  - Car noise
- Accented speech
- Far from microphone
- Young children’s speech
- Stuttering
- …

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

Guideline: Build your first system quickly, then iterate

FPT Education

FPT UNIVERSITY

Mismatched training and dev/test data

Training and testing on different distributions

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 (see the next slide).

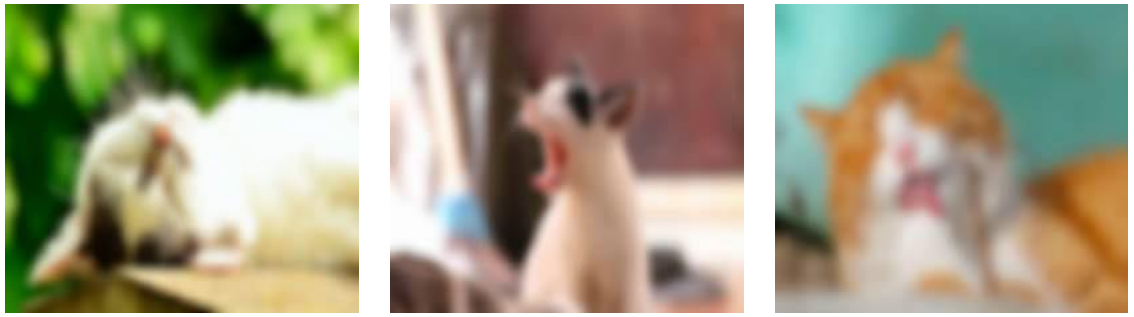
Training and testing on different distributions

- 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.
- 2. 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.

Cat app example

Data from webpages

Data from mobile app



Cat app example

- There are two sources of data used to develop the mobile app.

1. The first data distribution is small, 10 000 pictures uploaded from the mobile application. Since they are from amateur users, the pictures are not professionally shot, not well framed and blurrier.
2. The second source is from the web, you downloaded 200 000 pictures where cat's pictures are professionally framed and in high resolution.
   - The problem is that you have a different distribution:

- Small data set from pictures uploaded by users. This distribution is important for the mobile app.
- Bigger data set from the web.
  - The guideline used is that you have to choose a development set and test set to reflect data you expect to get in the future and consider important to do well.

Cat app example

- **Option 1**: randomly shuffle 210,000 images into a train, dev, and test set

205,000	**2,500**	**2,500**
Train	Dev	Test

- **Advantage**: training, dev and test sets will all come from the same distribution, so that makes it easier to manage.
- **Disadvantage**: in dev set, of these 2,500 examples, a lot of it will come from the web page distribution of images, rather than what you actually care about, which is the mobile app distribution of images.

Cat app example

- **Option 2**: the training set will include 200,000 images from the web and 5,000 from the mobile app. The dev set will be 2,500 images from the mobile app, and the test set will be 2,500 images also from the mobile app

200,000	**205,000**	**2,500**	**2,500**
Train	App	Dev (app)	Test(app)

- **Advantage**: the target is well defined.
- **Disadvantage**: the training distribution is different from the development and test set distributions. However, this way of splitting the data has a better performance in long term..

Speech recognition example

A speech activated rearview mirror for a car

- Training: 500,000 utterences from:
- - Purchased data
- - Smart voice activated speakers
- - Voice activated keyboards
- …

Dev/test: 20,000 utterences from:

A speech activated rearview mirror.

Speech recognition example

- **Option 1**: the training set includes the 500,000 utterances and the dev and test sets 10,000 utterances each.

500,000	**10,000**	**10,000**
Train	Dev	Test

- **Option 2**: the training set includes all 500 from there and 10,000 from the rearview mirror, the dev and test sets include 5,000 utterances each.

500,000 + 10,000	**5,000**	**5,000**
Train	Dev	Test

Training and testing on different distributions

- When allowing your training set data to come from a different distribution than your dev and test set allows you to have much more training data and it will cause your learning algorithm to perform better.

Mismatched training and dev/test data

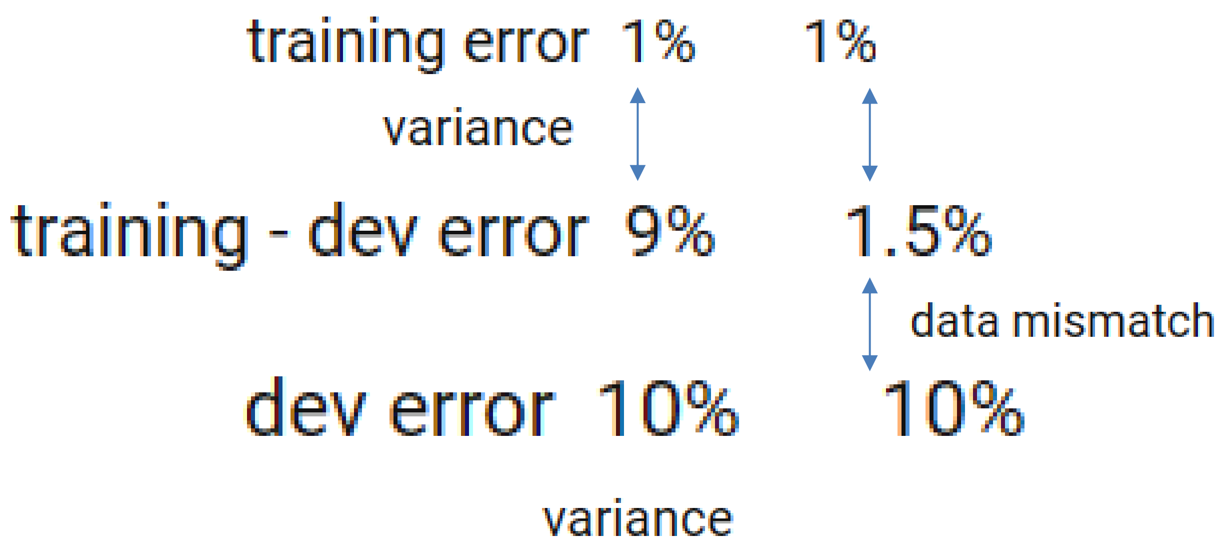
Bias and Variance with mismatched data distributions

Bias and Variance with mismatched data distributions

- To assess bias and variance discrepancies arising from different data distributions, a training-dev set, reflecting the training set's distribution, can be utilized.
- Analyzing the classifier's errors on the training, training-dev, and dev sets aids in identifying avoidable bias, variance, and data mismatch issues. Addressing data mismatch involves monitoring the model's performance and making necessary adjustments to enhance its generalization.

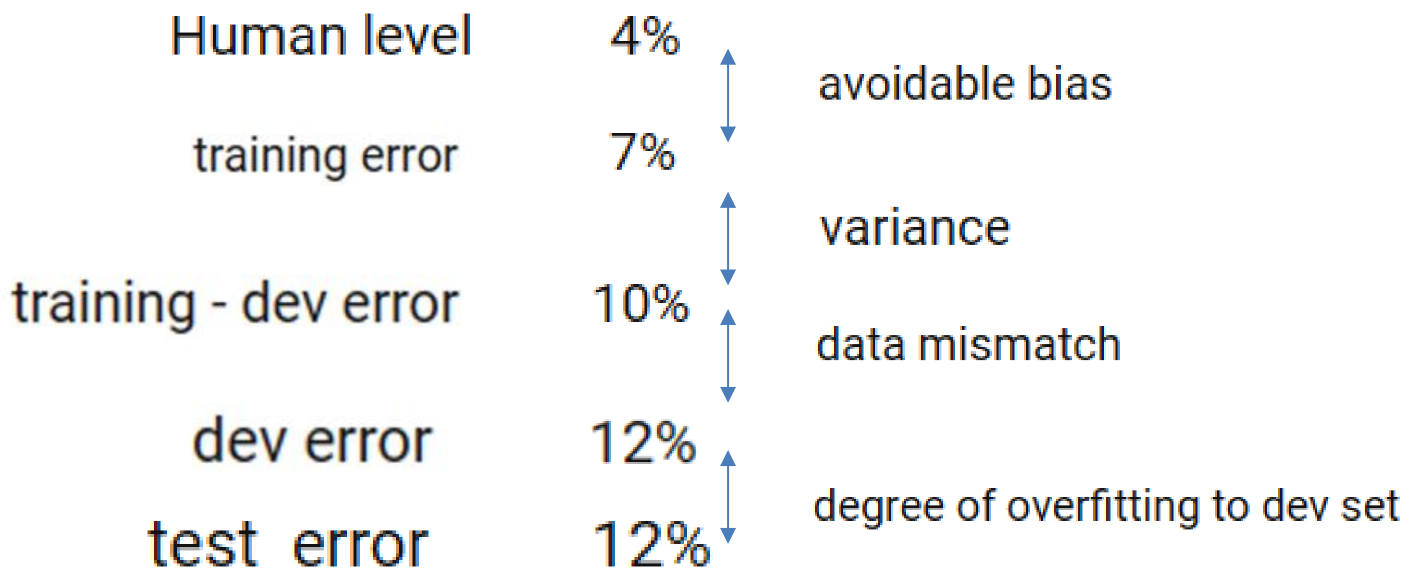
Cat classifier example

- Assume humans get ≈ 0% error.
- Training error 1%
- Dev error 10%



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

Bias/variance on mismatched training and dev/test sets



Human level	4%	
		avoidable bias
training error	7%	
		variance
training - dev error	10%	
		data mismatch
dev error	12%	
		degree of overfitting to dev set
test error	12%	

FPT Education

FPT UNIVERSITY

Mismatched training and dev/test data

Addressing data mismatch

Addressing data mismatch

- There aren't completely systematic solutions to this, but there are 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).

Artificial data synthesis

+ =

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

Car noise

Synthesized in-car audio

- Make training data or collect data similar to development and test sets. To make the training data more similar to your development set, you can use is artificial data synthesis. However, it is possible that if you might be accidentally simulating data only from a tiny subset of the space of all possible examples.

Artificial data synthesis

- Car recognition:

FPT Education

FPT UNIVERSITY

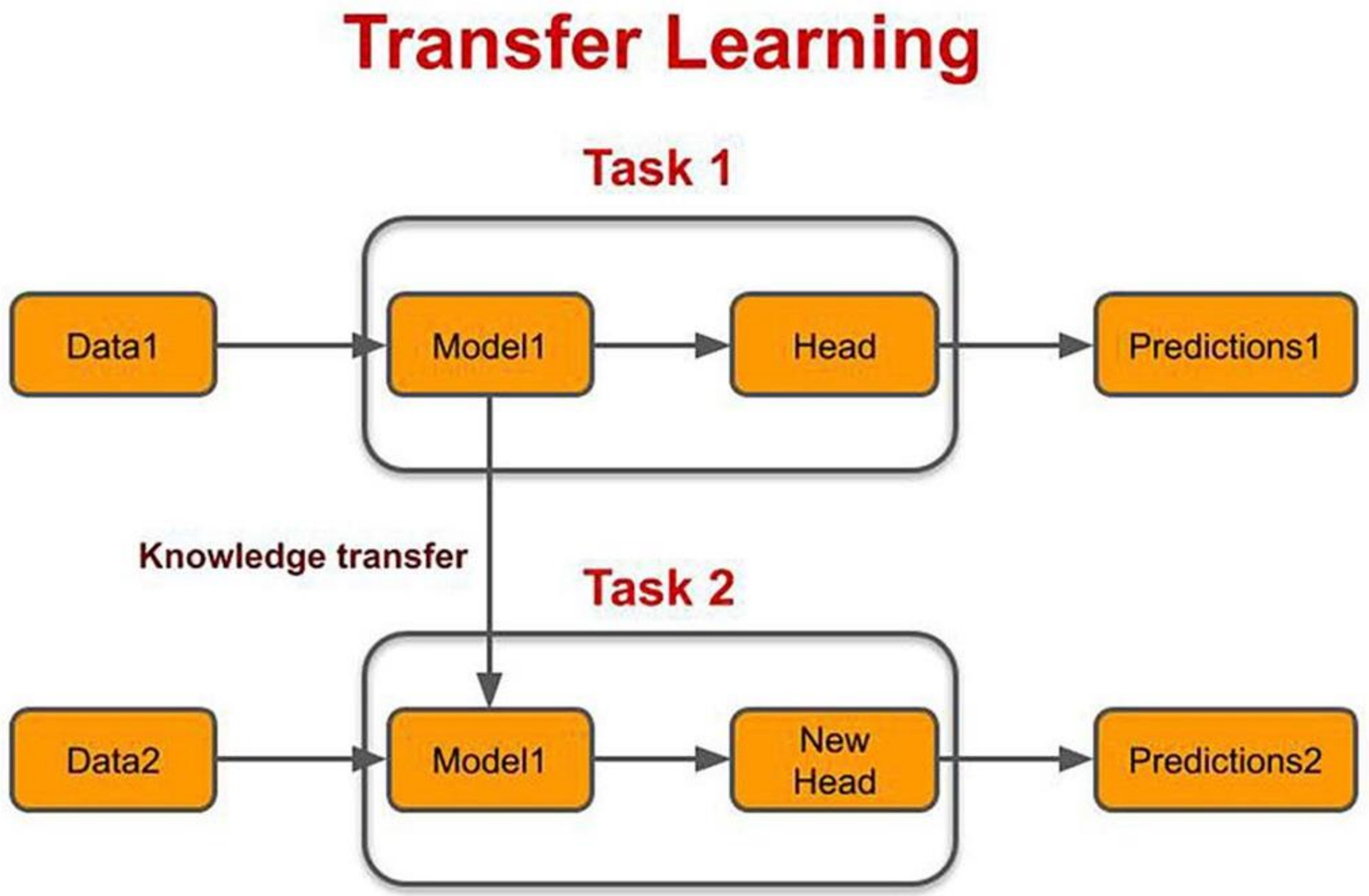
Learning from multiple tasks

Transfer learning

Transfer learning

- Transfer learning is a technique in deep learning that enables one to take knowledge learned from a neural network on one task and apply it to another task. This is particularly useful when you have a significant amount of data for the task you're transferring from but relatively little data for the task you're transferring to.
- To implement transfer learning, you train a neural network on the source task and then remove the last output layer and create a new set of randomly initialized weights for the last layer to output the target task. Depending on the amount of data, you may retrain only the last layer's weights or all the layers' weights.
- Transfer learning is useful when the input for both tasks is the same, and there is more data for the source task than for the target task. It is also helpful when low-level features from the source task could be beneficial for learning the target task.

Transfer learning



When transfer learning makes sense

- 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.

Transfer learning

- In summary, transfer learning has been most useful when you're trying to do well on a problem with relatively little data, such as radiology diagnosis. It helps the performance of the learning task significantly when the conditions for transfer learning are met.
- However, transfer learning may not be helpful if the source task has less data than the target task.
- Multitask learning is another version of learning from multiple tasks simultaneously, and it will be discussed in the next section.

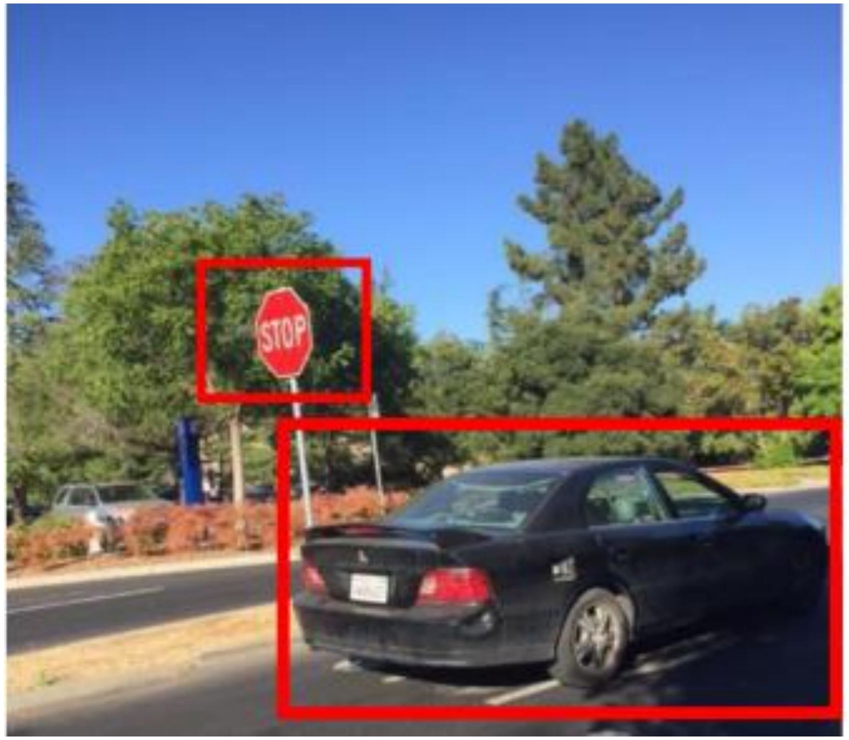
Learning from multiple tasks

Multi-task learning

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).

Simplified autonomous driving example



$$y^{(i)} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \begin{array}{l} \text{Pedestrians} \\ \text{Cars} \\ \text{Road signs - Stop} \\ \text{Traffic lights} \end{array}$$

$$\downarrow$$

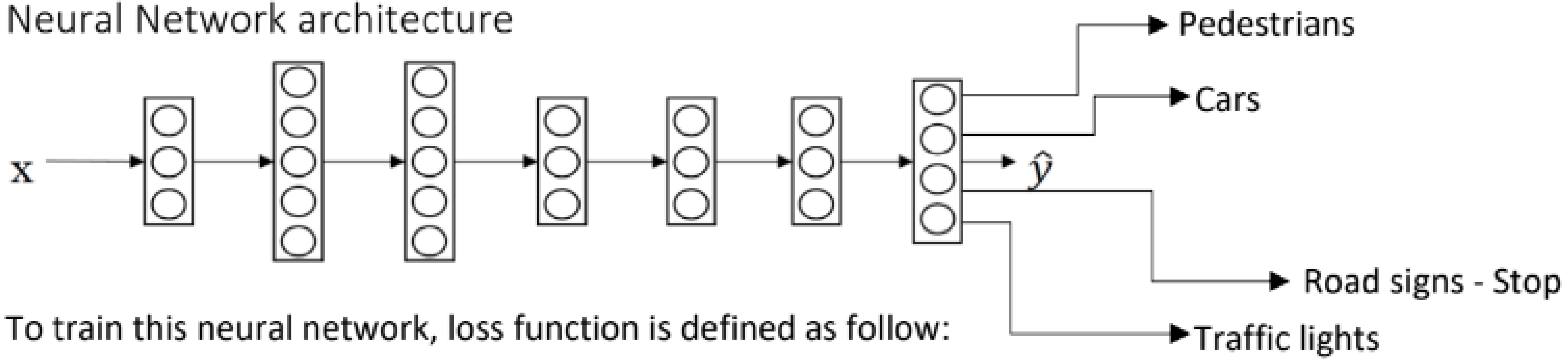
$$Y = \begin{bmatrix} | & | & | & | \\ y^{(1)} & y^{(2)} & y^{(3)} & y^{(4)} \\ | & | & | & | \end{bmatrix}$$

$$Y = (4, m)$$

$$Y = (4,1)$$

Neural network architecture

Neural Network architecture



To train this neural network, loss function is defined as follow:

$$-\frac{1}{m}\sum_{i=1}^{m}\boxed{\sum_{j=1}^{4}}\left(y_j^{(i)}\log\left(\hat{y}_j^{(i)}\right)+\left(1-y_j^{(i)}\right)\log\left(1-\hat{y}_j^{(i)}\right)\right)$$

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

Multi-task learning

- 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.

End-to-end deep learning

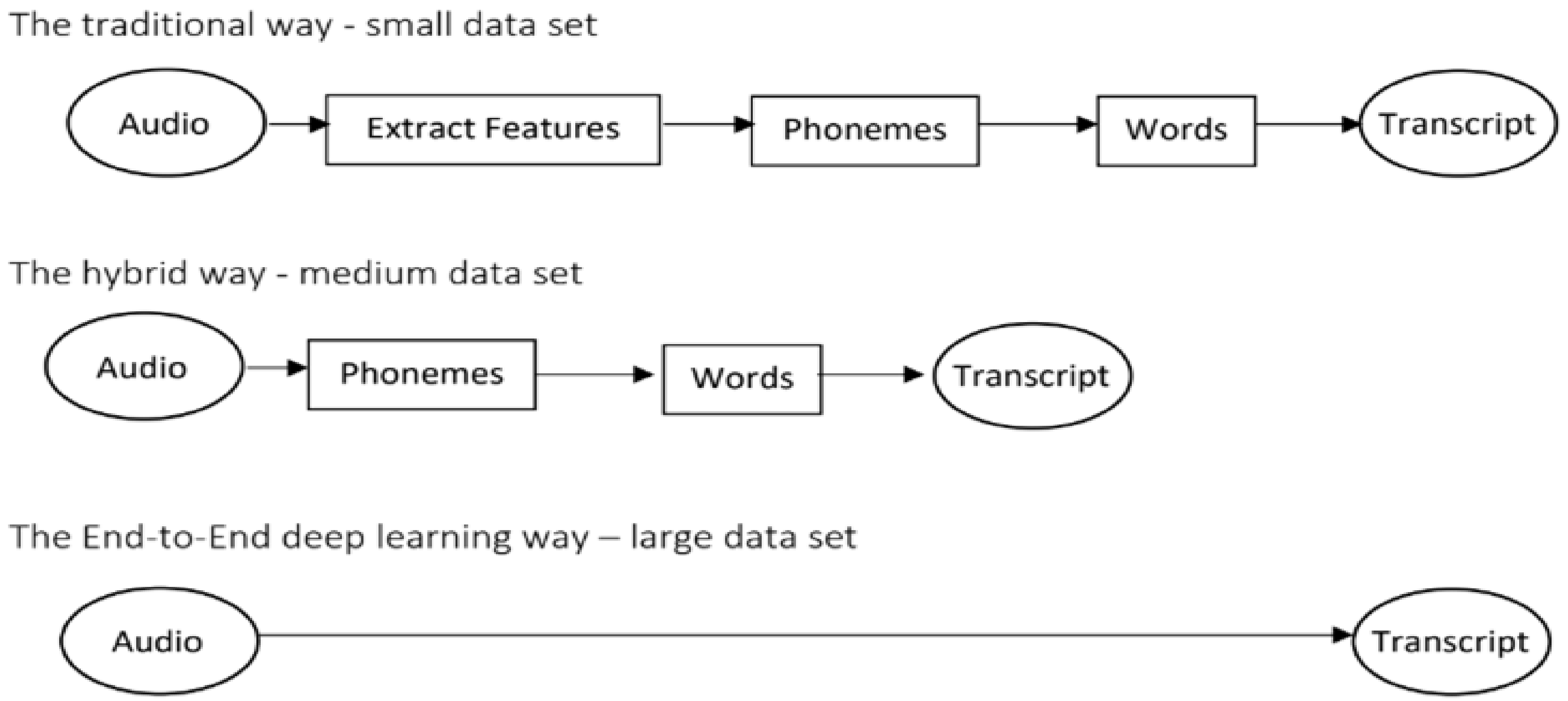
What is end-to-end deep learning

What is end-to-end deep learning

- End-to-end deep learning is a technique that aims to replace multiple stages of processing with a single neural network, making it simpler and more efficient. While traditional data processing systems require multiple stages to achieve a goal, end-to-end learning trains a single neural network to input the data and output the final prediction directly.
- However, end-to-end learning requires a large amount of data to work well, and may not work for all tasks. In some cases, intermediate approaches that break down the problem into multiple steps may be more effective. Therefore, researchers should consider the nature of the task and the availability of data when deciding whether to use end-to-end deep learning or other approaches.

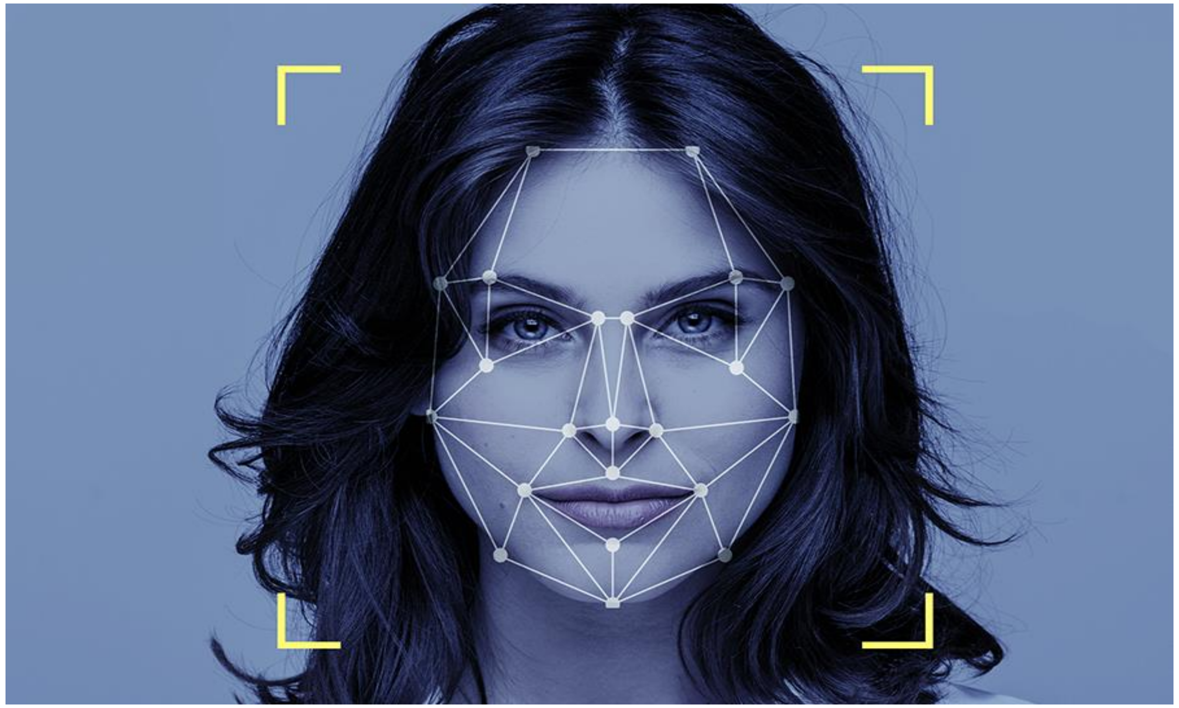
What is end-to-end learning?

- Speech recognition example


The traditional way - small data set
Audio
Extract Features
Phonemes
Words
Transcript
The hybrid way - medium data set
Audio
Phonemes
Words
Transcript
The End-to-End deep learning way – large data set
Audio
Transcript

- End-to-end deep learning cannot be used for every problem since it needs a lot of labeled data. It is used mainly in audio transcripts, image captures, image synthesis, machine translation, steering in self-driving cars, etc.

Face recognition



More examples

- Machine translation

English >>> text analysis >>> ... >>> French

$$\begin{tikzcd} & (X , Y) \arrow[dl] \arrow[dr] & \\ \text{English} & & \text{French} \end{tikzcd}$$

- Estimating child's age:

img >>> bones >>> age

img >>> age <<<

FPT Education

FPT UNIVERSITY

End-to-end deep learning

Whether to use end-to-end learning

Whether to use end-to-end learning

- In machine learning, an end-to-end approach involves feeding input data directly into the algorithm and getting the output as the final result, without any intermediate steps or manual feature engineering. This approach can be beneficial as it allows the algorithm to capture complex statistics present in the data without being limited by human preconceptions.
- However, it may require a large amount of data to learn the direct mapping from input to output, and hand-designed components may be useful for learning from less data. Therefore, it is essential to consider the complexity of the problem and the availability of data before deciding whether to use end-to-end deep learning or a combination of machine learning and hand-designed components.
- While end-to-end deep learning can work well for certain tasks, it is important to be mindful of where it is applied to create more effective and efficient systems.

Pros and cons of end-to-end deep learning

- Pros:
  - Let the data speak
  - Less hand-designing of components needed

- Cons:
  - May need large amount of data
  - Excludes potentially useful hand-designed components

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?

$$\begin{tikzcd}
\text{image , radar} \arrow[r] \arrow[dr] & \text{cars} \arrow[r] & \text{route} \arrow[r] & \text{steering} \\
& \text{pedestrians} \arrow[ur, "\text{motion planning}"'] & &
\end{tikzcd}$$

Summarization

- Error analysis identifies patterns, prioritizes improvements, and refines training.
- Correcting inaccuracies in labeled data involves careful examination and validation.
- Building an initial model quickly establishes a baseline for iterative improvements.
- Consistent data distribution between training and testing ensures accurate evaluation.
- Mismatched data distributions lead to bias and variance issues, addressed through strategies like manual labeling and domain adaptation.
- Transfer learning uses pre-trained models for related tasks, saving time and resources.
- Multi-task learning trains on multiple tasks simultaneously to enhance overall performance.
- End-to-end deep learning automates the entire pipeline, minimizing human involvement. The decision to use end-to-end learning depends on factors like data availability, resources, and task complexity.

Questions

1. What is the first step you should take when starting the project?
2. Is softmax a good choice for the output layer in a multi-task learning problem like detecting road signs and traffic signals?
3. When performing error analysis, which set of images should you manually examine one at a time?
4. In multi-task learning, does the learning algorithm require all y(i) label vectors to be fully labeled?
5. Given that the real data comes from your car's front-facing camera, how should you split the training/dev/test datasets to reduce data mismatch?
6. Given the performance metrics and human-level error of 0.5%, what problems can be identified from the training and dev/test results?
7. Based on the data split and error rates, how would you respond to a friend who says the training data distribution is much easier than the dev/test distribution?
8. If you synthesize foggy images using clean images and realistic fog overlays, can this be a good way to address fog-related errors?
9. When correcting mislabeled data in the dev set, should you also correct labels in the training and test sets?
10. What advice would you give a teammate who wants to build a yellow light detector using a small dataset?
11. A colleague wants to use audio from external microphones to detect nearby vehicles. How can you help given they have limited training data?