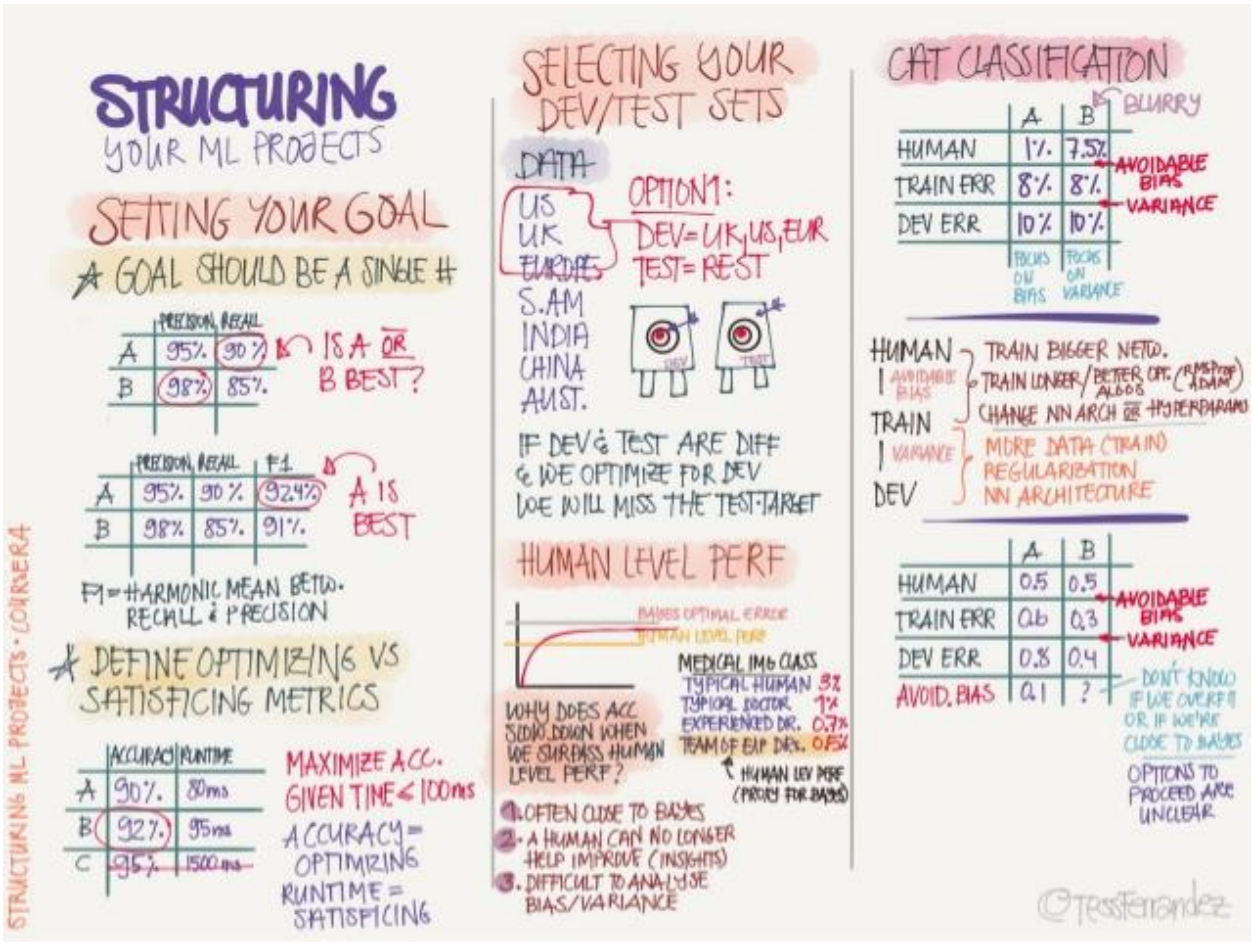
[ML Strategy 1]

- Why ML Strategy
- Orthogonalization
- Single number evaluation metric
- Satisfying and Optimizing metric
- Train/dev/test distributions
- Size of the dev and test sets
- When to change dev/test sets and metrics
- Why human-level performance?
- Avoidable bias
- Understanding human-level performance
- Surpassing human-level performance
- Improving your model performance



ML Strategy 1

Why ML Strategy

- You have a lot of ideas for how to improve the accuracy of your deep learning system:
  - Collect more data.
  - Collect more diverse training set.
  - Train algorithm longer with gradient descent.
  - Try different optimization algorithm (e.g. Adam).
  - Try bigger network.
  - Try smaller network.
  - Try dropout.
  - Add L2 regularization.
  - Change network architecture (activation functions, # of hidden units, etc.)
- This course will give you some strategies to help analyze your problem to go in a direction that will help you get better results.

Orthogonalization

- Some deep learning developers know exactly what hyperparameter to tune in order to try to achieve one effect. This is a process we call orthogonalization.
- In orthogonalization, you have some controls, but each control does a specific task and doesn't affect other controls.
- For a supervised learning system to do well, you usually need to tune the knobs of your system to make sure that four things hold true - chain of assumptions in machine learning:
  1. You'll have to fit training set well on cost function (near human level performance if possible).
     - If it's not achieved you could try bigger network, another optimization algorithm (like Adam)...
  2. Fit dev set well on cost function.
     - If its not achieved you could try regularization, bigger training set...
  3. Fit test set well on cost function.
     - If its not achieved you could try bigger dev. set...
  4. Performs well in real world.
     - If its not achieved you could try change dev. set, change cost function...

Single number evaluation metric

- Its better and faster to set a single number evaluation metric for your project before you start it.
- Difference between precision and recall (in cat classification example):
  - Suppose we run the classifier on 10 images which are 5 cats and 5 non-cats. The classifier identifies that there are 4 cats, but it identified 1 wrong cat.
  - Confusion matrix:

	Predicted cat	Predicted non-cat
Actual cat	3	2
Actual non-cat	1	4

  - **Precision**: percentage of true cats in the recognized result: P = 3/(3 + 1)
  - **Recall**: percentage of true recognition cat of the all cat predictions: R = 3/(3 + 2)
  - **Accuracy**: (3+4)/10
- Using a precision/recall for evaluation is good in a lot of cases, but separately they don't tell you which algothims is better. Ex:

Classifier	Precision	Recall
A	95%	90%
B	98%	85%

- A better thing is to combine precision and recall in one single (real) number evaluation metric. There a metric called `F1` score, which combines them
  - You can think of `F1` score as an average of precision and recall `F1 = 2 / ((1/P) + (1/R))`

Satisfying and Optimizing metric

- Its hard sometimes to get a single number evaluation metric. Ex:

Classifier	F1	Running time
A	90%	80 ms
B	92%	95 ms
C	92%	1,500 ms

- So we can solve that by choosing a single optimizing metric and decide that other metrics are satisfying. Ex:
- `Maximize F1                    # optimizing metric`
- `subject to running time < 100ms # satisficing metric`
- So as a general rule:
- `Maximize 1     # optimizing metric (one optimizing metric)`
- `subject to N-1 # satisficing metric (N-1 satisficing metrics)`

Train/dev/test distributions

- Dev and test sets have to come from the same distribution.
- Choose dev set and test set to reflect data you expect to get in the future and consider important to do well on.
- Setting up the dev set, as well as the validation metric is really defining what target you want to aim at.

Size of the dev and test sets

- An old way of splitting the data was 70% training, 30% test or 60% training, 20% dev, 20% test.
- The old way was valid for a number of examples ~ <100000
- In the modern deep learning if you have a million or more examples a reasonable split would be 98% training, 1% dev, 1% test.

When to change dev/test sets and metrics

- Let's take an example. In a cat classification example we have these metric results:

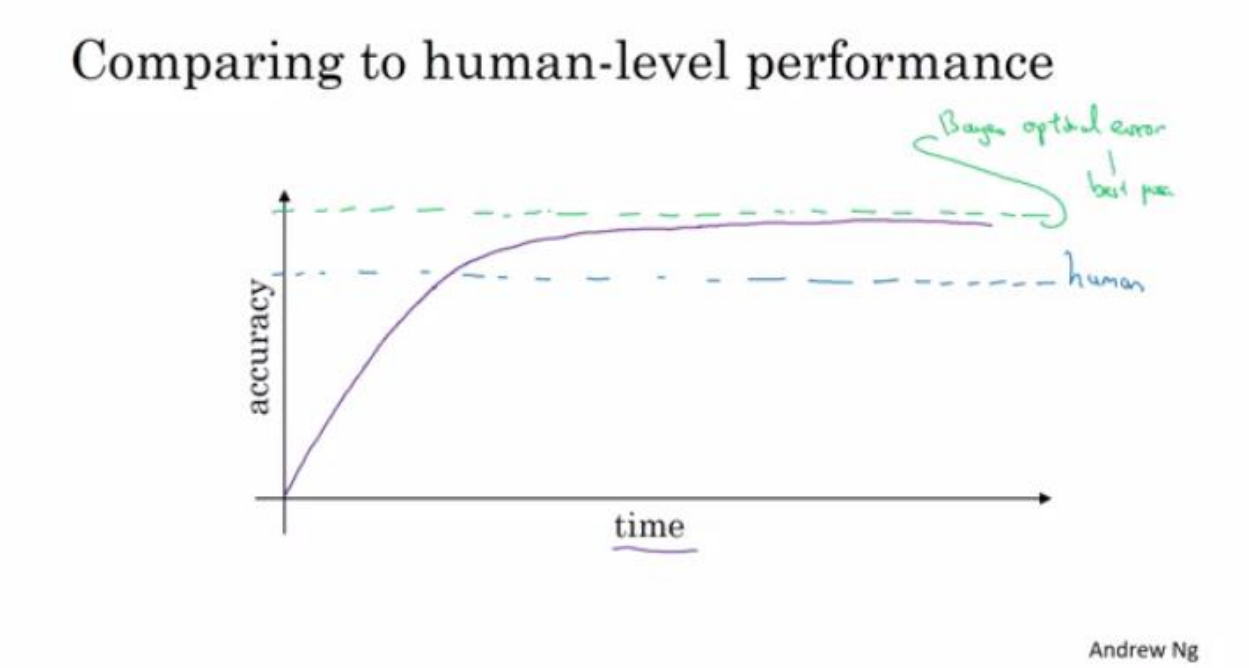
Metric	Classification error
Algorithm A	3% error (But a lot of porn images are treated as cat images here)
Algorithm B	5% error

    - In the last example if we choose the best algorithm by metric it would be "A", but if the users decide it will be "B"
    - Thus in this case, we want and need to change our metric.
    - `OldMetric = (1/m) * sum(y_pred[i] != y[i] ,m)`
        - Where m is the number of Dev set items.
    - `NewMetric = (1/sum(w[i])) * sum(w[i] * (y_pred[i] != y[i]) ,m)`
        - where:
            - `w[i] = 1 if x[i] is not porn`
            - `w[i] = 10 if x[i] is porn`
- This is actually an example of an orthogonalization where you should take a machine learning problem and break it into distinct steps:
    1. Figure out how to define a metric that captures what you want to do - place the target.
    2. Worry about how to actually do well on this metric - how to aim/shoot accurately at the target.
- Conclusion: if doing well on your metric + dev/test set doesn't correspond to doing well in your application, change your metric and/or dev/test set.

Why human-level performance?

- We compare to human-level performance because of two main reasons:
    1. Because of advances in deep learning, machine learning algorithms are suddenly working much better and so it has become much more feasible in a lot of application areas for machine learning algorithms to actually become competitive with human-level performance.
    2. It turns out that the workflow of designing and building a machine learning system is much more efficient when you're trying to do something that humans can also do.

- After an algorithm reaches the human level performance the progress and accuracy slow down.



- You won't surpass an error that's called "Bayes optimal error".
- There isn't much error range between human-level error and Bayes optimal error.
- Humans are quite good at a lot of tasks. So as long as Machine learning is worse than humans, you can:
  - Get labeled data from humans.
  - Gain insight from manual error analysis: why did a person get it right?
  - Better analysis of bias/variance.

Avoidable bias

- Suppose that the cat classification algorithm gives these results:

Humans	**1%**	**7.5%**
Training error	8%	8%
Dev Error	10%	10%

  - In the left example, because the human level error is 1% then we have to focus on the **bias**.
  - In the right example, because the human level error is 7.5% then we have to focus on the **variance**.
  - The human-level error as a proxy (estimate) for Bayes optimal error. Bayes optimal error is always less (better), but human-level in most cases is not far from it.
  - You can't do better than Bayes error unless you are overfitting.
  - `Avoidable bias = Training error - Human (Bayes) error`

    - `Variance = Dev error - Training error`

Understanding human-level performance

- When choosing human-level performance, it has to be chosen in the terms of what you want to achieve with the system.
- You might have multiple human-level performances based on the human experience. Then you choose the human-level performance (proxy for Bayes error) that is more suitable for the system you're trying to build.
- Improving deep learning algorithms is harder once you reach a human-level performance.
- Summary of bias/variance with human-level performance:
    1. human-level error (proxy for Bayes error)
        - Calculate `avoidable bias = training error - human-level error`
        - If **avoidable bias** difference is the bigger, then it's *bias* problem and you should use a strategy for **bias** resolving.
    2. training error
        - Calculate `variance = dev error - training error`
        - If **variance** difference is bigger, then you should use a strategy for **variance** resolving.
    3. Dev error
- So having an estimate of human-level performance gives you an estimate of Bayes error. And this allows you to more quickly make decisions as to whether you should focus on trying to reduce a bias or trying to reduce the variance of your algorithm.
- These techniques will tend to work well until you surpass human-level performance, whereupon you might no longer have a good estimate of Bayes error that still helps you make this decision really clearly.

Surpassing human-level performance

- In some problems, deep learning has surpassed human-level performance. Like:
    - Online advertising.
    - Product recommendation.
    - Loan approval.
- The last examples are not natural perception task, rather learning on structural data. Humans are far better in natural perception tasks like computer vision and speech recognition.
- It's harder for machines to surpass human-level performance in natural perception task. But there are already some systems that achieved it.

Improving your model performance

- The two fundamental asssumptions of supervised learning:
    1. You can fit the training set pretty well. This is roughly saying that you can achieve low **avoidable bias**.
    2. The training set performance generalizes pretty well to the dev/test set. This is roughly saying that **variance** is not too bad.

- To improve your deep learning supervised system follow these guidelines:
    1. Look at the difference between human level error and the training error - **avoidable bias**.
    2. Look at the difference between the dev/test set and training set error - **Variance**.
    3. If **avoidable bias** is large you have these options:
        - Train bigger model.
        - Train longer/better optimization algorithm (like Momentum, RMSprop, Adam).
        - Find better NN architecture/hyperparameters search.
    4. If **variance** is large you have these options:
        - Get more training data.
        - Regularization (L2, Dropout, data augmentation).
        - Find better NN architecture/hyperparameters search.

Week 1 Quiz - Bird recognition in the city of Peacetopia (case study)

You are a famous researcher in the City of Peacetopia. The people of Peacetopia have a common characteristic: they are afraid of birds. To save them, you have **to build an algorithm that will detect any bird flying over Peacetopia** and alert the population.

The City Council gives you a dataset of 10,000,000 images of the sky above Peacetopia, taken from the city's security cameras. They are labelled:

- y = 0: There is no bird on the image
- y = 1: There is a bird on the image

Your goal is to build an algorithm able to classify new images taken by security cameras from Peacetopia.

There are a lot of decisions to make:

- What is the evaluation metric?
- How do you structure your data into train/dev/test sets?

Metric of success

The City Council tells you that they want an algorithm that

1. Has high accuracy
2. Runs quickly and takes only a short time to classify a new image.

3. Can fit in a small amount of memory, so that it can run in a small processor that the city will attach to many different security cameras.

1. Having three evaluation metrics makes it harder for you to quickly choose between two different algorithms, and will slow down the speed with which your team can iterate. True/False?
   - ☑ True
   - ☐ False
2. If you had the three following models, which one would you choose?

If you had the three following models, which one would you choose?

Test Accuracy	Runtime	Memory size
97%	1 sec	3MB

Test Accuracy	Runtime	Memory size
99%	13 sec	9MB

Test Accuracy	Runtime	Memory size
97%	3 sec	2MB

Test Accuracy	Runtime	Memory size
98%	9 sec	9MB

   - Test Accuracy 98%
   - Runtime 9 sec
   - Memory size 9MB
3. Based on the city's requests, which of the following would you say is true?
   - ☑ Accuracy is an optimizing metric; running time and memory size are a satisficing metrics.
   - ☐ Accuracy is a satisficing metric; running time and memory size are an optimizing metric.
   - ☐ Accuracy, running time and memory size are all optimizing metrics because you want to do well on all three.
   - ☐ Accuracy, running time and memory size are all satisficing metrics because you have to do sufficiently well on all three for your system to be acceptable.
4. Before implementing your algorithm, you need to split your data into train/dev/test sets. Which of these do you think is the best choice?

Train	Dev	Test
3,333,334	3,333,333	3,333,333

Train	Dev	Test
9,500,000	250,000	250,000

Train	Dev	Test
6,000,000	3,000,000	1,000,000

Train	Dev	Test
6,000,000	1,000,000	3,000,000

- Train 9,500,000
- Dev 250,000
- Test 250,000

5. After setting up your train/dev/test sets, the City Council comes across another 1,000,000 images, called the "citizens' data". Apparently the citizens of Peacetopia are so scared of birds that they volunteered to take pictures of the sky and label them, thus contributing these additional 1,000,000 images. These images are different from the distribution of images the City Council had originally given you, but you think it could help your algorithm.

   You should not add the citizens' data to the training set, because this will cause the training and dev/test set distributions to become different, thus hurting dev and test set performance. True/False?

   - ☐ True
   - ☑ False

6. One member of the City Council knows a little about machine learning, and thinks you should add the 1,000,000 citizens' data images to the test set. You object because:
   - The test set no longer reflects the distribution of data (security cameras) you most care about.
   - This would cause the dev and test set distributions to become different. This is a bad idea because you're not aiming where you want to hit.
7. You train a system, and its errors are as follows (error = 100%-Accuracy):
   - Training set error 4.0%
   - Dev set error 4.5%

   This suggests that one good avenue for improving performance is to train a bigger network so as to drive down the 4.0% training error. Do you agree?

- No, because there is insufficient information to tell.

8. You ask a few people to label the dataset so as to find out what is human-level performance. You find the following levels of accuracy:
   - Bird watching expert #1 0.3% error
   - Bird watching expert #2 0.5% error
   - Normal person #1 (not a bird watching expert) 1.0% error
   - Normal person #2 (not a bird watching expert) 1.2% error

   If your goal is to have "human-level performance" be a proxy (or estimate) for Bayes error, how would you define "human-level performance"?

   - 0.3% (accuracy of expert #1)
9. Which of the following statements do you agree with?
   - A learning algorithm's performance can be better human-level performance but it can never be better than Bayes error.
10. You find that a team of ornithologists debating and discussing an image gets an even better 0.1% performance, so you define that as "human-level performance." After working further on your algorithm, you end up with the following:
    - Human-level performance 0.1%
    - Training set error 2.0%
    - Dev set error 2.1%

    Based on the evidence you have, which two of the following four options seem the most promising to try? (Check two options.)

    - Try decreasing regularization.
    - Train a bigger model to try to do better on the training set.
11. You also evaluate your model on the test set, and find the following:
    - Human-level performance 0.1%
    - Training set error 2.0%
    - Dev set error 2.1%
    - Test set error 7.0%

    What does this mean? (Check the two best options.)

    - You should try to get a bigger dev set.
    - You have overfit to the dev set.
12. After working on this project for a year, you finally achieve:
    - Human-level performance 0.10%
    - Training set error 0.05%
    - Dev set error 0.05%

    What can you conclude? (Check all that apply.)

    - It is now harder to measure avoidable bias, thus progress will be slower going forward.

        - If the test set is big enough for the 0,05% error estimate to be accurate, this implies Bayes error is ≤ 0.05

13. It turns out Peacetopia has hired one of your competitors to build a system as well. Your system and your competitor both deliver systems with about the same running time and memory size. However, your system has higher accuracy! However, when Peacetopia tries out your and your competitor's systems, they conclude they actually like your competitor's system better, because even though you have higher overall accuracy, you have more false negatives (failing to raise an alarm when a bird is in the air). What should you do?
    - Rethink the appropriate metric for this task, and ask your team to tune to the new metric.
14. You've handily beaten your competitor, and your system is now deployed in Peacetopia and is protecting the citizens from birds! But over the last few months, a new species of bird has been slowly migrating into the area, so the performance of your system slowly degrades because your data is being tested on a new type of data.
    - Use the data you have to define a new evaluation metric (using a new dev/test set) taking into account the new species, and use that to drive further progress for your team.
15. The City Council thinks that having more Cats in the city would help scare off birds. They are so happy with your work on the Bird detector that they also hire you to build a Cat detector. (Wow Cat detectors are just incredibly useful aren't they.) Because of years of working on Cat detectors, you have such a huge dataset of 100,000,000 cat images that training on this data takes about two weeks. Which of the statements do you agree with? (Check all that agree.)
    - If 100,000,000 examples is enough to build a good enough Cat detector, you might be better of training with just 10,000,000 examples to gain a $\approx$10x improvement in how quickly you can run experiments, even if each model performs a bit worse because it's trained on less data.
    - Buying faster computers could speed up your teams' iteration speed and thus your team's productivity.
    - Needing two weeks to train will limit the speed at which you can iterate.