# Strategies for Structuring Machine Learning Project

Anthony Faustine

PhD researcher machine learning
(IDLab, imec, research group at Ghent University)

Saturday 1st September, 2018

GHENT UNIVERSITY

IDLab
INTERNET & DATA LAB

# Learning goal

- Understand why ML strategies is important.
- Understand how to define optimizing and satisfying evaluation metrics.
- Understand how to define human level performance.
- Learn how to do define key priorities in ML projects.

# Outline

# Introduction

Consider a classification problem

- Suppose you train ML model for such problem and achieve 90% accuracy.
- This is not good performance

Question: What should we do to improve performance?

# Introduction: Why ML Strategy ?

Question: What should we do to improve performance?

## Several options to try

- Collect more data.
- Increase number of iterations with SGD or try different optimization algorithms (Adam etc).
- Increase model complexity.
- Use regularization (dropout, L2, or L1)
- Change network architecture (hidden units, activation function)

# Introduction: Why ML Strategy ?

Challenge: How to select best and effective options to pursue?

- Poor selection → end up spending more time in direction that wont improve performance at the end.
- Best selection → quickly and efficiently get your machine learning systems working.
- Need ML strategy to perform best selection.

Machine learning strategy is useful to iterate through ideas quickly and to efficiently reach the project outcome.

- It offer ways to analyse ML problem and guide in the direction of the most promising options to try.

# Introduction: Why ML Strategy ?

Challenge: How to select best and effective options to pursue?

- Poor selection → end up spending more time in direction that wont improve performance at the end.
- Best selection → quickly and efficiently get your machine learning systems working.
- Need ML strategy to perform best selection.

Machine learning strategy is useful to iterate through ideas quickly and to efficiently reach the project outcome.

- It offer ways to analyse ML problem and guide in the direction of the most promising options to try.

# Orthogalization

The challenges with building machine learning systems $\Rightarrow$ so many things to try or change (hyperparameters etc).

- It is very important to be specific on what to tune in order to try achieving one effect.

## Orthogonalization

Refers to the concept of picking parameters (knobs) to tune which only adjust one outcome of the machine learning model.

- It is a system design property that insure that modifying parameter of algorithm will not create or propagate side effects to other component of the system.
- It make easier to verify the algorithms independently from one another.
- It reduce testing and development time.

# Orthogalization

The challenges with building machine learning systems $\Rightarrow$ so many things to try or change (hyperparameters etc).

- It is very important to be specific on what to tune in order to try achieving one effect.

## Orthogonalization

Refers to the concept of picking parameters (knobs) to tune which only adjust one outcome of the machine learning model.

- It is a system design property that insure that modifying parameter of algorithm will not create or propagate side effects to other component of the system.
- It make easier to verify the algorithms independently from one another.
- It reduce testing and development time.

# Orthogalization

- For Supervised ML system to work well you have to achieve
  1. Best performance in training set
  2. Best performance in validation/dev set
  3. Best performance in test set
  4. Perform well in real world.
- Use different knobs (parameters) to improve performance of each part.

# Orthogalization

1. To improve performance in training set
   - use bigger neural network or switch to a better optimization algorithms (adam etc)
2. To improve performance in validation/dev set
   - Apply regularization or use bigger training set
3. To improve performance in test set
   - Increase size of dev set.
4. Poor performance in real world.
   - Change development set.
   - Change the loss function

# Outline

# Evaluation metric

Consider week one dropout challenge:

| Model | Precision | Recall |
|---|---|---|
| A | 95% | 90% |
| B | 98% | 85% |

**Precision**

- how precise/accurate your model is out of those predicted positive, how many of them are actual positive.
- good measure to determine, when the costs of False Positive is high

**Recall**

- how many of the actual positives the model capture through labelling it as positive
- good measure to determine, when the costs of False Negative is high

# Evaluation metric: use single evaluation metric

The problem of using two or more evaluation metrics → difficult to make decision

- Use one evaluation metric e.g (F-1 score which is harmonic average of precision and recall)

| **Model** | **Precision** | **Recall** | **F-score** |
|---|---|---|---|
| A | 95% | 90% | 92.4% |
| B | 98% | 85% | 91.0% |

- Having single number evaluation metric ⇒ improve efficiency in decision making.

# Evaluation metric: Satisfying and optimizing metric

What if you want to combine more than one metrics?

- Suppose you are interested in both F-score and running time.

| **Model** | **F-score** | **Running time** |
|---|---|---|
| A | 92.4% | $80ms$ |
| B | 91.0% | $35ms$ |
| C | 95.0% | $100s$ |

- Choose one metric as optimizing metric and others as satisfying metrics.
- For example: maximize F-score (*optimizing metric*) and minimize running time (*satisfying metric*) such as it is less than $t$

# Evaluation metric: Satisfying and optimizing metric

If you have $N$ metrics choose one metric as optimizing metrics and $N-1$ metrics as satisfying metrics

- Consider fraud detection system
  - How likely to detect fraud transaction
  - Minimize False Negative.
- Optimize Accuracy subject to minimizing False Negative

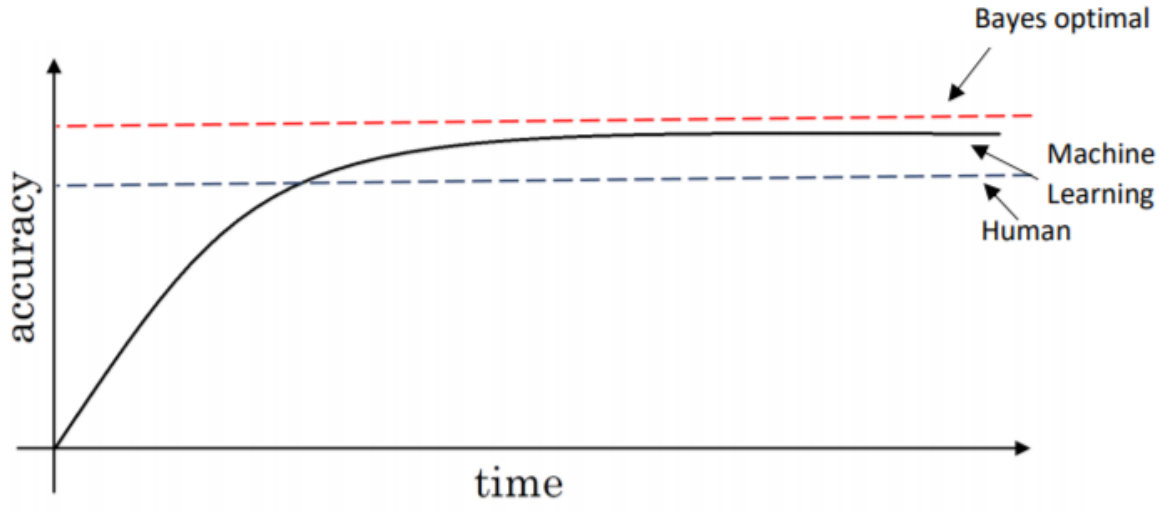
# Data setup: Train/dev/test set

The way you set up/ divide your data into train/dev/test impact the progress of your project.

- Make sure the data have same distribution in each partition $\rightarrow$ randomly shaffle data before split etc.
- Choose a dev set and test to reflect data you expect in future
- If you have enough dataset use 98/1/1 ratio instead of the traditional 60/20/20 $\rightarrow$ use more data for training and less dev and test set
- The development set should be big enough to evaluate different ideas.

# Outline

# Comparing to human level performance




- Bayes optimal performance: best possible performance $\rightarrow$ best theoretical function $f_\theta(x : y)$
- Human level performance is not much different to bayes optimal performance.

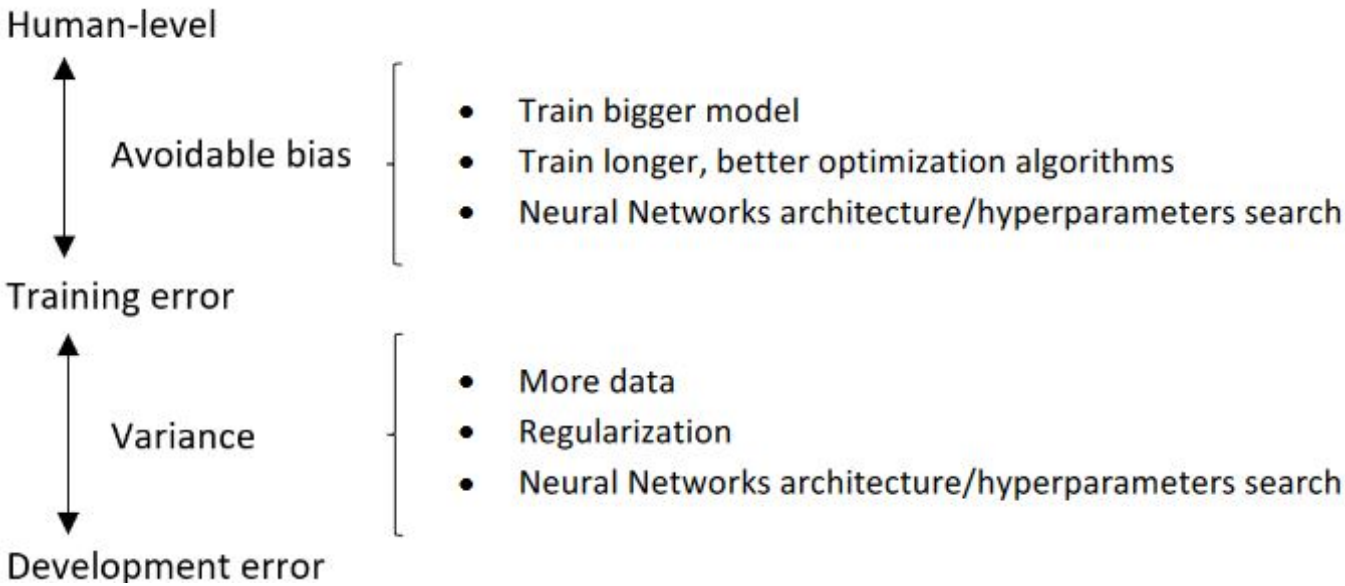
# Comparing to human level performance

## Why compare with human level performance

Human are quite good at lot of tasks $\rightarrow$ comparing your poorly performing ML to human level performance can help

- Get labelled data from humans
- Gain insight from manual error analysis $\rightarrow$ why did a person get it right?
- Better analysis of bias and variance.

# Bias Variance Analysis: Avoidable bias




- If avoidable bias $>$ variance focus on reducing bias.
- If avoidable bias $<$ variance focus on reducing variance.

# Bias Variance Analysis: Avoidable bias

Consider image classification problem with the following performance (classification error).

| | **scenario A** | **scenario B** |
|---|---|---|
| Human | 1% | 7.5% |
| Training performance | 8% | 8% |
| Dev performance | 10% | 10% |

What technique should we use to improve performance in scenario A and B ?

# Quantify human level performance

Consider x-ray image classification: suppose

a. Typical human achieve 3%
b. Typical doctor achieve 1%
c. Experienced doctor achieve 0.7%
d. Team of experienced doctors achieve 0.5%

What is human level error ?

# Quantify human level performance

Human level performance $\{1, 0.7, 0.5\}$

| | **scenario A** | **scenario B** | **scenario C** |
|---|---|---|---|
| Training performance | 5% | 1% | 0.7% |
| Dev performance | 6% | 5% | 0.8% |

# Quantify human level performance

Scenario A:

- Avoidable bias is between $4 - 4.5\%$ and the variance is $1\%$ $\rightarrow$ focus on bias reduction techniques
- Choice of human-level performance doesn't have an impact.

Scenario B:

- Avoidable bias is between $0 - 0.5\%$ and the variance is $4\%$ $\rightarrow$ focus on variance reduction techniques
- Choice of human-level performance doesn't have an impact.

Scenario C:

- The estimate for bayes error has to be $0.5\%$ Avoidable bias is between $0.2\%$ and the variance is $0.1\%$ $\rightarrow$ focus on bias reduction techniques.

# Outline

# Error Analysis

If the performance of your ML algorithm is still poor compared to human level performance $\Rightarrow$ perform error analysis

- Manually examine mistakes that your ML algorithm is making $\rightarrow$ gain insight of what to do next.

**Error analysis**

1. First get about 100 mislabelled dev set samples.
2. Manually examine the samples for false negatives and false positives.
3. Count up the number of error that fall into various different categories.

This will help you prioritize or give you inspiriton for new direction to go in.

# Error Analysis

Consider a cat vs dog classification problem:

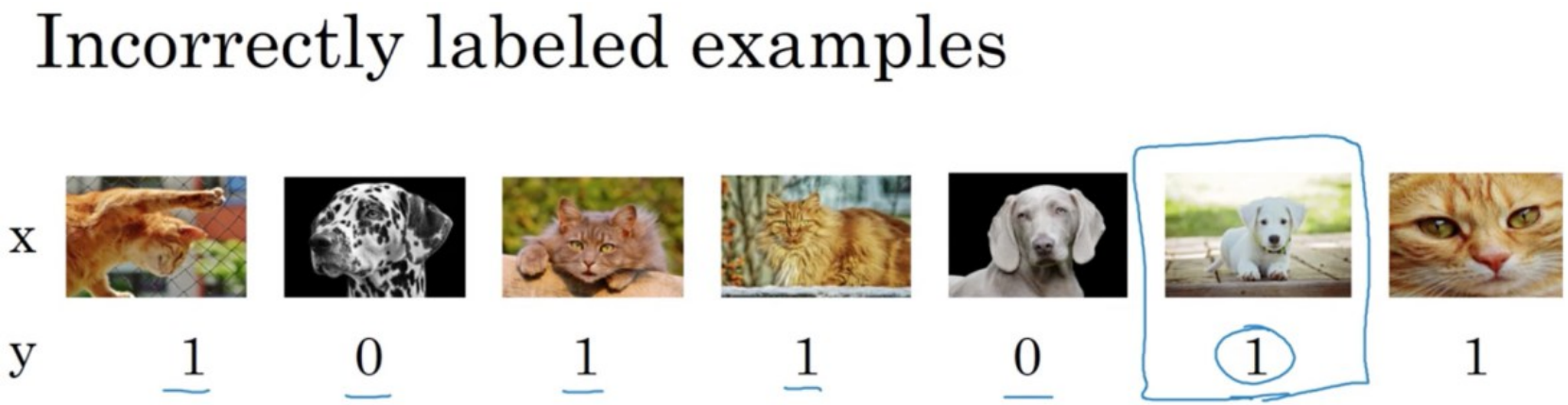
- Your team achieve 90% accuracy

| Image | Dog | Great Cats | Blurry | Instagram | Comments |
|---|---|---|---|---|---|
| 1 | ✓ | | | ✓ | Pitbull |
| 2 | | | ✓ | ✓ | |
| 3 | | ✓ | ✓ | | Rainy day at zoo |
| ⋮ | ⋮ | ⋮ | ⋮ | | |
| % of total | 8% | 43% | 61% | 12% | |

# Cleaning incorrectly labelled data in training set

In supervised ML the data comprises input $X$ and label $Y$

- What if you going through the data and find some of the labels are incorrect.
- What should you do?



# Cleaning incorrectly labelled data in training set

If the errors (incorrectly labelled example) are random $\rightarrow$ leave the errors as they are not spend time correcting them

- This is because deep learning are robust to random errors.

However, deep learning are less robust to systematic errors $\Rightarrow$ constantly labels white dogs as cats

# Cleaning incorrectly labelled data in dev/test set

To address the impact of incorrectly label in dev/test set:

- Add extra column for incorrectly label during error analysis
- If your dev set error is 10% and you have 0.5% error here due to mislabeled dev set $\Rightarrow$ probably not a very good use of your time to try to fix them.
- But instead, if you have 2% dev set error and 0.5% error here is due to mislabeled dev set, $\Rightarrow$ it is wise to fix them because it amounts for 25% of your total error.

| 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% | |

# Outline

# Training and testing on different distribution

Suppose you are building an app that classifies cats from the images uploaded by users. The images are taken from users cell phone. Suppose you have data from two sources

1. 200,000 high resolution images from the web and
2. 10,000 unprofessional/blurry images on the app, uploaded by users.

Question: What is the best approach to distribute these data into train/dev/test set?

# Training and testing on different distribution

Question: What is the best approach to distribute these data into train/dev/test set?

- One approach you can use: combine the dataset and randomly shuffle them into train/dev/test set.
- Advantage: Your data will come from the same distribution
- Disadvantage: Most of your dev and test data will come from the web page distribution rather than the actual mobile phone distribution which you care about.

# Training and testing on different distribution

Question: What is the best approach to distribute these data into train/dev/test set?

- Best approach: have all images in dev/test set come from mobile users and put the remaining images from mobile users in the train set along with the web images.
- For example: Train set 205000 (web plus 5000 mobile data), dev set 5000 (mobile data) and test set 5000 (mobile data)
- This will cause inconsistent distribution in train and dev/test set but it will let you hit where you intend to in the long run.

Take away

- use large training set, even if distribution is different from dev/test set.
- dev/test data should reflect what to expect from the system.

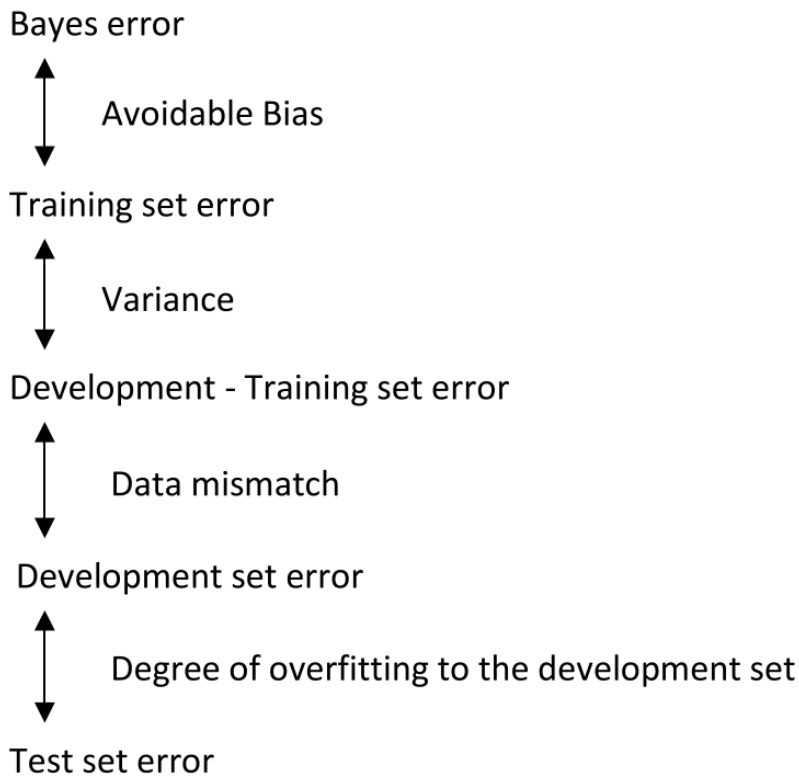
# Bias and Variance with mismatched training and dev/test set

Analysing bias and variance change when your training set come from different distribution than the your dev/test set.

- You can no longer call the error between train and dev set as variance $\Rightarrow$ they are already coming from different distributions.
- To analyse actual variance define a new Training / Dev set which will have same distribution as training set but will not be used for training.

# Bias and Variance with mismatched training and dev/test set

You can then analyse your model as shown in the figure below.



# Bias and Variance with mismatched training and dev/test set

| **Scenario** | **A** | **B** | **C** | **D** | **E** | **F** |
|---|---|---|---|---|---|---|
| Human performance | 0% | 0% | 0% | 0% | 0% | 4% |
| Training performance | 1% | 1% | 1% | 10% | 10% | 7% |
| Training-dev performance | – | 9% | 1.5% | 11% | 11% | 10% |
| Dev performance | 10% | 10% | 10% | 12% | 20% | 6% |
| Test performance | – | – | – | – | – | 6% |

# Addressing data mismatch

- Perform manual error analysis to understand the error differences between training/dev/test set.
- Collect more training data similar to dev/test set $\Rightarrow$ you can use synthetic data.

# Build your system quickly

# Outline

# Transfer learning

Transfer learning: ML method where a model developed for a task ( task) is reused as the starting point for a model on a second task (target task).

- Define source and target domain.
- Learn on source domain.
- Generalize on target domain $\Rightarrow$ Learned knowledge from source domain applied to a target domain.
- why it work: some low-level features can be shared for different tasks.

# Transfer learning

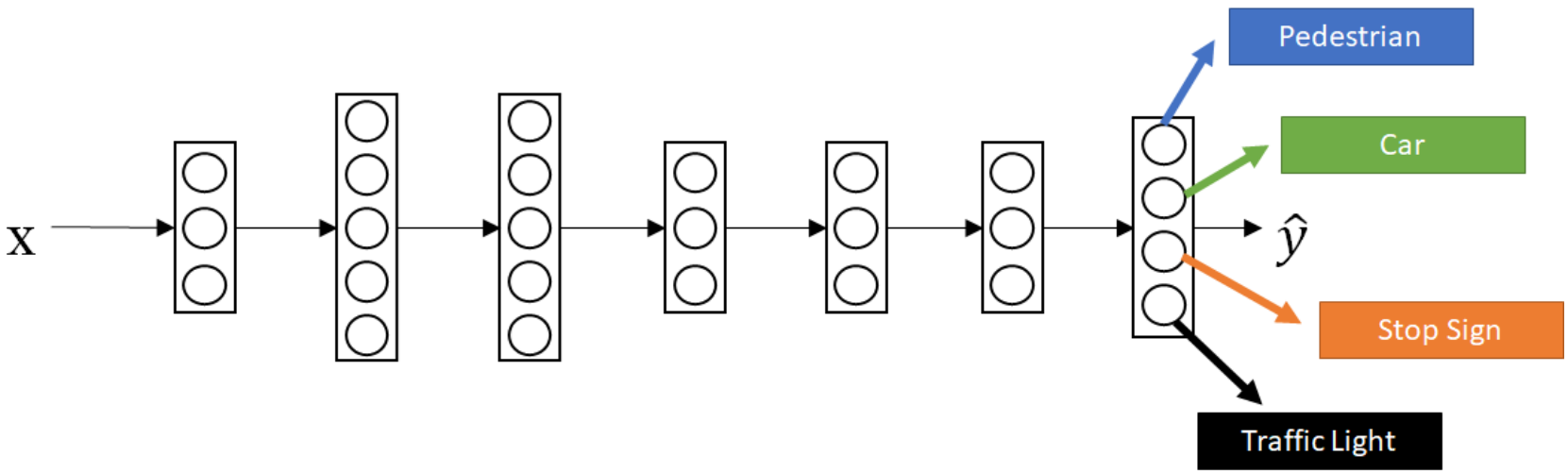
When to use transfer learning

- source task and target task have the same input.
- There are lot of data for source task and relatively small amount of data for target task.
- Low level feature of source task could be helpful for target task.

# Multi-task learning

Multi-task learning: Use a single neural network to do simultaneously several tasks.

- Suppose you want to build a self-driving car and a part of the problem is to classify objects on the street.




More details on multi-task learning here

# Multi-task learning

When to use multi-task learning:

- Lower-level features can be shared.
- Similar amount of data for each task $\rightarrow$ data for other tasks could help learning of main task.
- Can train a big enough NN to do well on all tasks.

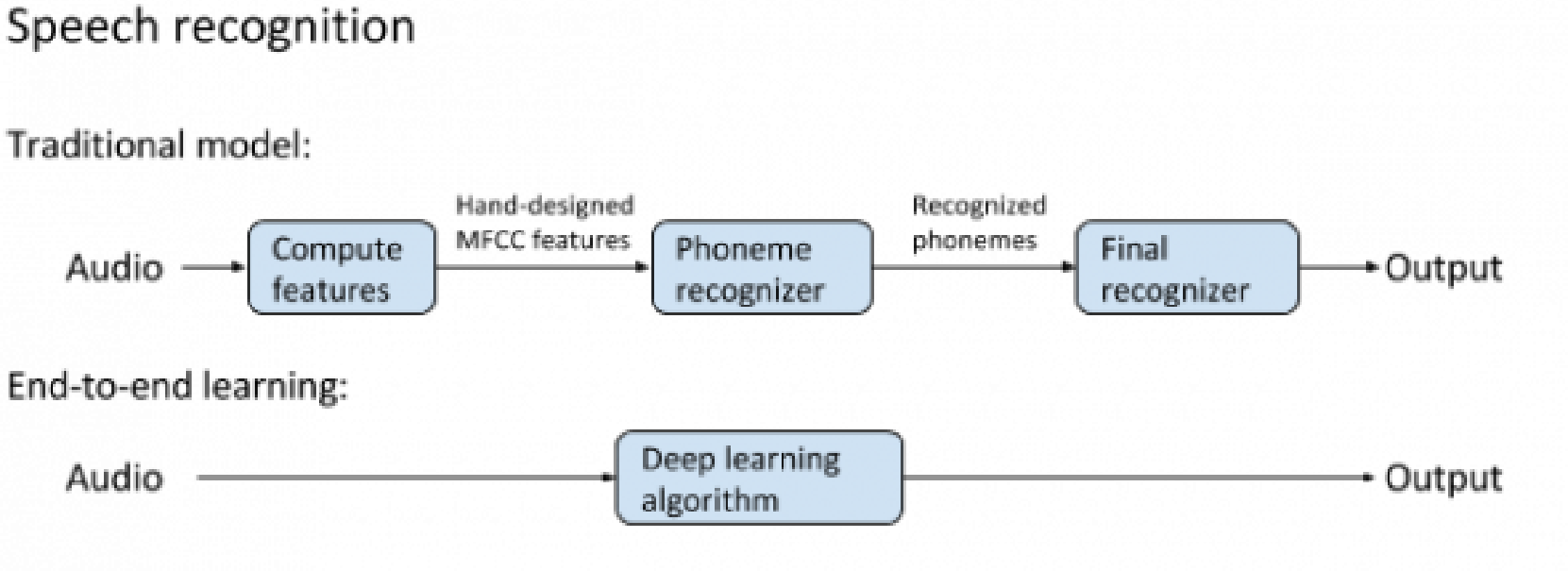
# Outline

# What is end to end deep learning

A simplification of processing or learning systems into one neural network.

- Instead of using many different steps and manual feature engineering to generate a prediction $\rightarrow$ use one neural network to figure out the underlying pattern
- This omit multiple stages in pipeline by a single NN.
- It work well only when have really large dataset.

# What is end to end deep learning



# Whether to use end-to end deep learning

Consider the following two problems:

1. Face recognition from camera
2. Machine translation.

# Whether to use end-to end deep learning

## Advantages

- Let the data speak $\rightarrow$ the neural network will find which statistics are in the data rather than being forced to reflect human preconceptions.
- Less hand-designing of components needed $\rightarrow$ it simplifies the design work flow.

## Disadvantages

- Require large amount of labelled data $\rightarrow$ can not used for every problem.
- Excludes potentially usefully hand designed component.
  - Data and any hand-design's components or features are the main two sources of knowledge for learning algorithm.
  - If the data set is small than hand-design system is a way to give manual knowledge into the algorithm.

# Important advice

- ML is not plug and play.
- Learn both theory and practical implementation.
- Practice, Practice, Practice: compete in Kaggle competitions and read associated blog posts and forum discussions.
- Do the Dirty Work: read a lot of papers and try to replicate the results. Soon enough, you'll get your own ideas and build your own models