ML STRATEGY

[Chain of assumptions in ML]

→ Fit training set well on cost function — bigger network, Adam

→ Fit dev set well on cost function — Reg., Bigger training set

→ Fit test set well on cost function — Bigger dev set

→ Performs well in real world — Change dev set or cost function

1) Set 1 single evaluation metric
   i.e instead of looking at precision and recall, you could use the F1 score
2) Optimizing vs Satisficing metrics
3) Dev and Test sets need to come from the same distributions
4)

[Transfer Learning]

- Take knowledge the NN has learned from 1 task and apply that to a separate task.
- You train your network on img recognition first. Then let's say you want to do radiology diagnosis. With transfer learning, you get rid of the last layer and you train that with with the radiology diag. dataset.
  This is useful in case you have a small dataset.
- Pre-training + fine tuning: train the NN on img rec. and after update all the weights training on the radiology diag. dataset
- Transfer learning makes sense when:
  - → Task A and B have the same type of input X
  - → Dataset for Task A is bigger than B
  - → Low level features from A could be helpful for B

[Multitask Learning]

- One NN to do several things at the same time (i.e: autonomous vehicle driving)
- We are solving multiple problems at the same time, i.e: 1 img will have multiple labels assigned
- When multitask learning makes sense:
  - → Training on a set of tasks that could benefit from having shared low-level features
  - → Amount of data for each task is quite similar
  - → Can train a big enough NN to do well on all the tasks

[End-to-end Deep Learning]

- Merge multiple stages of data processing into a single NN
- Small dataset → Normal pipeline
- Breaking up problems into sub-problems and train multiple NN