A running example

Training Data

Food Image Classifier

Food Image Classifier
Basic steps to build an ML system
The steps overview

• Step 1: collect data
• Step 2: look at your data
• Step 3: Create train/dev/test splits
• Step 4: build model
• Step 5: Evaluate your model
• Step 6: Diagnose error and repeat
Acquire and annotate data
Data should be **diverse**

(Annotation can be expensive)
Data should be **realistic**

Ideal data sampled from the distribution your product will be run on.

Real photo taken by users

Professional ads photo
Look at your data.
Look at your data.

- You have some food images, take a closer look at them!
- Food from Europe different than from Africa? from Asia?
- Any potential bias in your data?
- Have the right people look at your data.
- Do this at every stage!
Expertise sometimes can be required

• Biomedical imaging annotation can be expensive
• Professionally trained radiologists
• Domain knowledge
Train/Dev/Test Split
Partitioning Data: Train, Test, and Validation

(1) Fit model to the training dataset
(2) Fit hyperparameters to the validation (or development) dataset
(3) Test model performance on the test set

Slides credit: Chris Ré, Stanford CS229
What makes a good split?

- **Ideal**: Train, test, & dev randomly sampled
  - Allows us to say train quality is approximately test quality

- Test is a **proxy** for the real world!
  - We’ll talk more about this later...

- **Challenge**: Leakage.
  - (Nearly) same example in train and dev.
  - Causes performance to be overstated!
    - Eg., same senders in train and test?

Slides credit: Chris Ré, Stanford CS229
Build your model.
Build your model.

• A bag of learning algorithms learned from class.

• Simple model vs. deep models
Underfitting
Overfitting
Model Capacity

• The ability to fit variety of functions
• Low capacity models struggles to fit training set
  • Underfitting
• High capacity models can memorize the training set
  • Overfitting
### Underfitting and Overfitting

<table>
<thead>
<tr>
<th>Model capacity</th>
<th>Data complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Normal</td>
</tr>
<tr>
<td></td>
<td>Underfitting</td>
</tr>
<tr>
<td>High</td>
<td>Overfitting</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
</tr>
</tbody>
</table>

- **Simple** data complexity:
  - Low model capacity: Normal
  - High model capacity: Overfitting
- **Complex** data complexity:
  - Low model capacity: Underfitting
Data Complexity

- Multiple factors matters
  - # of examples
  - # of features in each example
  - time/space structure
  - # of labels
Ablation studies.

- You’ve built up a model, it has many different components.
  - Which matter?
  - which are stable?
- Remove one feature at a time!
  - Adding features + baseline could overestimate overlap. How?
- Measure performance.
  - Critical for research!
A running example

**Closed-world:** Training and testing distributions match

**Open-world:** Training and testing distributions differ
How to build machine learning models that function reliably in an open world?
Out-of-distribution Uncertainty

This is "out of distribution"!
Out-of-distribution Uncertainty
For safety critical applications
High confidence in classifying traffic signs.
Cross-Entropy Loss

\[ L_{CE} = \sum_{i} - Y_i \log(S_i) \]

\[ = - \log(0.8) \]

Goal: push \( S \) and \( Y \) to be identical
Training examples: traffic signs

\[ p_i(x) = \frac{\exp(f_i(x))}{\sum_{j=1}^{N} \exp(f_j(x))}, \]

confidence: \( \max_i p_i \)

High confidence in classifying traffic signs.
Test time: **out-of-distribution** example

Ideally: *Low confidence in predicting as traffic sign*
Neural networks can be over-confident to *out-of-distribution (OOD)* examples.

[Nguyen et al. 2015]
Confidence Score Distribution

Score distribution

0.99 0.98 0.94 0.97

0.85 0.89 0.92 0.82

In-distribution

Out-distribution

Confidence $\max_i p_i$
How can we distinguish out-of-distribution examples from in-distribution data?
ODIN: Out-of-distribution Image Detector

[Liang et al. ICLR 2018]

Shiyu Liang  Sharon Y. Li  R. Srikant
ODIN: Out-of-distribution Image Detector

\[ p_i(\mathbf{x}; T) = \frac{\exp \left( \frac{f_i(\mathbf{x})}{T} \right)}{\sum_{j=1}^{N} \exp \left( \frac{f_j(\mathbf{x})}{T} \right)} , \]

- Green: In-distribution
- Gray: Out-distribution

Score distribution

1/N

Confidence \( \max_i p_i \)

1
Training Task

In-distribution data: CIFAR-10
Detection Task

Out-of-distribution data
Results

ROC curve of detecting in- and out-of-distribution images.

- FPR reduced from 34.7% to 4.3%
- TPR on in-distribution images (CIFAR-10)
- FPR on out-of-distribution images (TinyImageNet (crop))
The steps overview

• Step 1: collect data
• Step 2: look at your data
• Step 3: Create train/dev/test splits
• Step 4: build model
• Step 5: Evaluate your model
• Step 6: Diagnose error and repeat
Industry-scale Machine Learning
Model Complexity Keeps Increasing

LeNet (Lecun et al. 1998)

ResNet (He et al. 2016)
Challenge: Limited labeled data

ImageNet, 1M images ~thousand annotation hours  x 1000  1B images ~million annotation hours

[Deng et al. 2009]
Levels of Supervision

- Fully Supervised
  - CAT, DOG
  - FLOOR

- Weakly Supervised
  - A CUTE CAT COUPLE
  - Un-supervised

- Un-supervised
  - ???

Training at Scale

- Instagram/Flickr
- ImageNet
- Crawled web images
Noisy Data

TRAINING AT SCALE

Non-Visual Labels

#LOVE #CAT #DOG #HUSKY

Incorrect Labels

Missing Labels
Can we use images with noisy labels for training?

[Mahajan et al. 2018]
Largest Weakly Supervised Training

3.5B PUBLIC INSTAGRAM IMAGES
LARGE CAPACITY MODEL (RESNEXT101-32X48) 17K UNIQUE LABELS
DISTRIBUTED TRAINING (350 GPUS)

[Mahajan et al. 2018]
Self-supervised Learning
(no label)
“Pure” Reinforcement Learning (cherry)
- The machine predicts a scalar reward given once in a while.

A few bits for some samples

 Supervised Learning (icing)
- The machine predicts a category or a few numbers for each input
- Predicting human-supplied data
- 10→10,000 bits per sample

Self-Supervised Learning (cake génoise)
- The machine predicts any part of its input for any observed part.

Source: Yann LeCun’s talk
What if we can get labels for free for unlabelled data and train unsupervised dataset in a supervised manner?
Pretext Tasks

- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.
- Predict the occluded from the visible
- Pretend there is a part of the input you don’t know and predict that.
Rotation

$g(X, y=0)$
- Rotate 0 degrees

$g(X, y=1)$
- Rotate 90 degrees

$g(X, y=2)$
- Rotate 180 degrees

$g(X, y=3)$
- Rotate 270 degrees

[Gidaris et al. 2018]
**Rotation**

Image $X$

- $g(X, y=0)$: Rotate 0 degrees
  - Rotated image: $X^0$
  - ConvNet model $F(.)$
    - Maximize prob. $F^0(X^0)$
    - Predict 0 degrees rotation ($y=0$)

- $g(X, y=1)$: Rotate 90 degrees
  - Rotated image: $X^1$
  - ConvNet model $F(.)$
    - Maximize prob. $F^1(X^1)$
    - Predict 90 degrees rotation ($y=1$)

- $g(X, y=2)$: Rotate 180 degrees
  - Rotated image: $X^2$
  - ConvNet model $F(.)$
    - Maximize prob. $F^2(X^2)$
    - Predict 180 degrees rotation ($y=2$)

- $g(X, y=3)$: Rotate 270 degrees
  - Rotated image: $X^3$
  - ConvNet model $F(.)$
    - Maximize prob. $F^3(X^3)$
    - Predict 270 degrees rotation ($y=3$)

Gidaris et al. 2018
Doersch et al., 2015

Patches

Example:

Question 1:

Question 2:

[X = (X, Y); Y = 3]
Summary

• Basic steps to build an ML system
• Open-world machine learning
• Industry-scale machine learning
Thank you!