

Introduction: Deep Learning for Computer Vision

Beginnings (sorta)

How it works

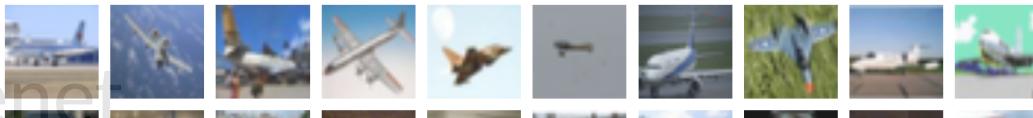
Some uses

Simple Example using Fastai

Imagenet

14 million images in 1000 categories

airplane



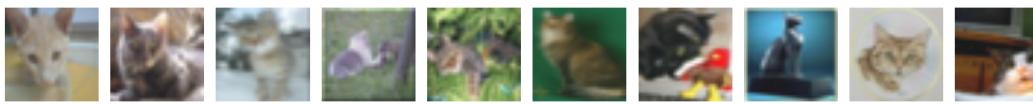
automobile



bird



cat



deer



dog



frog



horse



ship



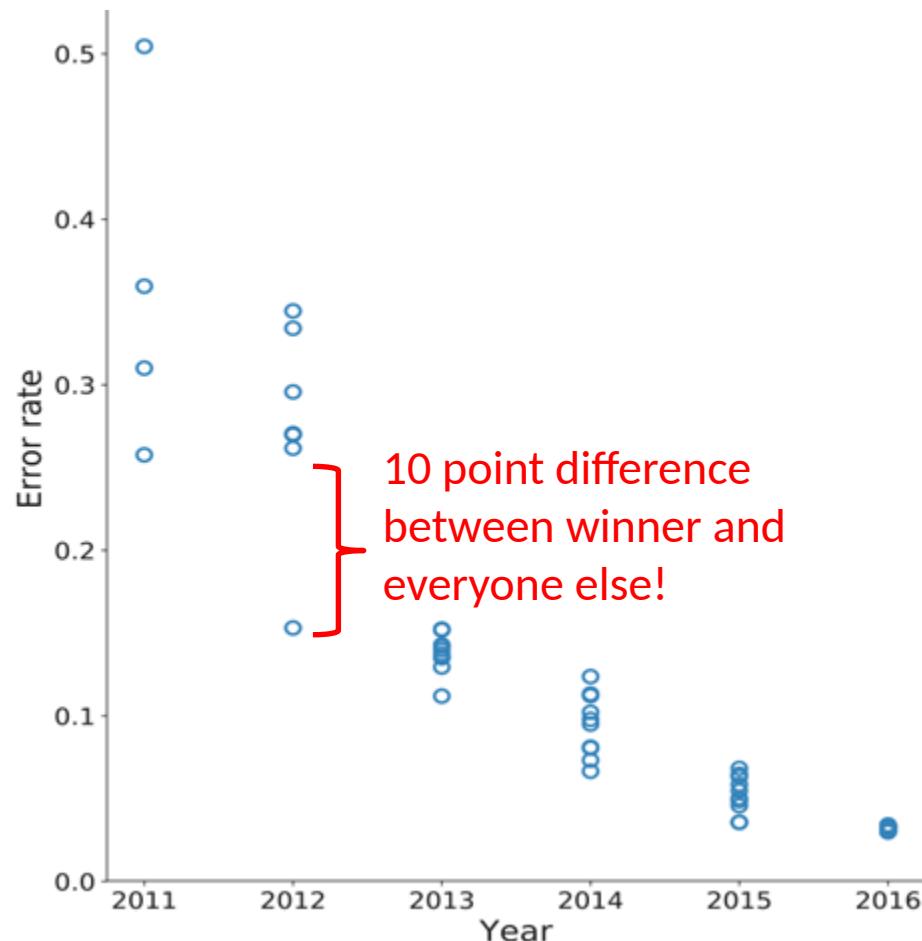
truck



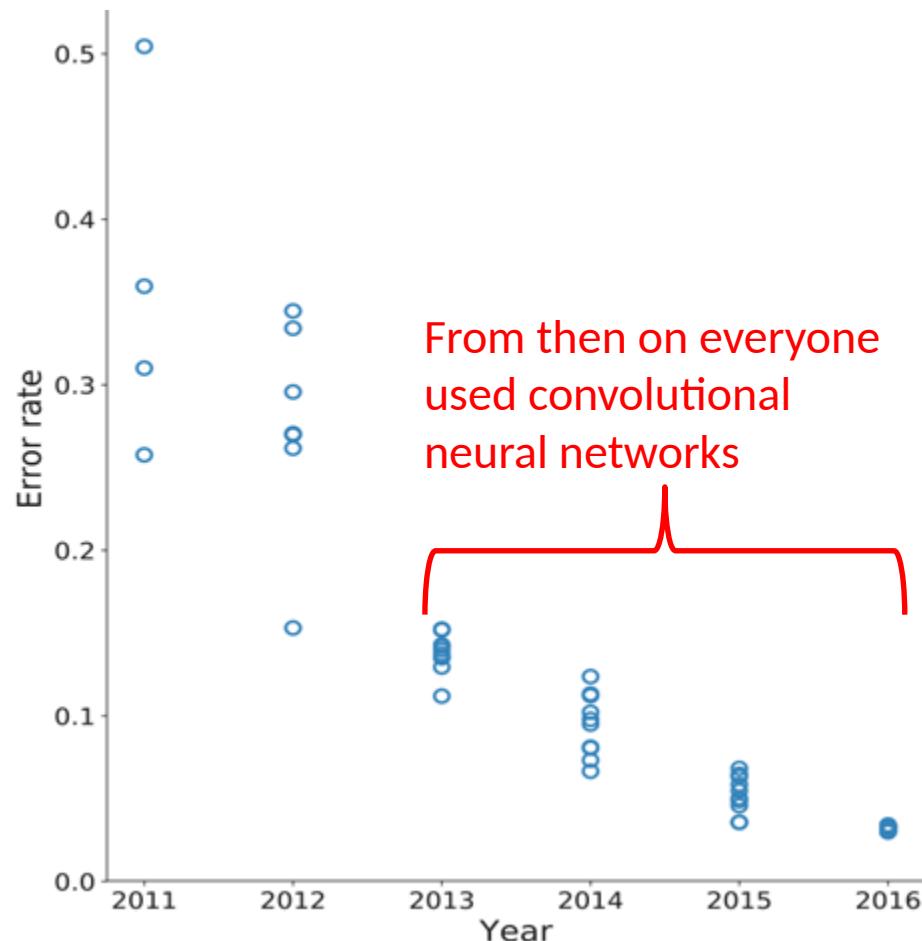
1 million also
Have bounding
box data which
frames object



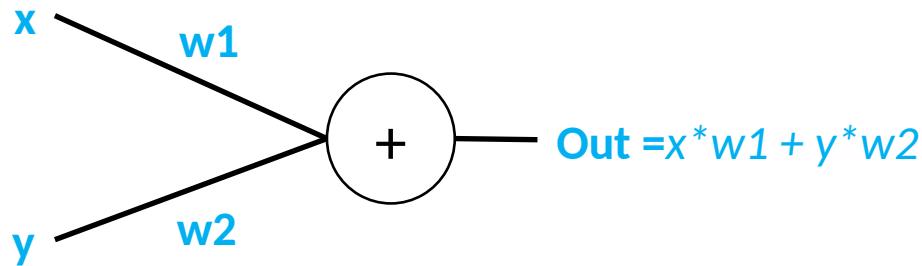
ImageNet Large Scale Visual Recognition Challenge



ImageNet Large Scale Visual Recognition Challenge



How it works – single neuron



$$\text{Error} = \frac{1}{2}(\text{CA}-\text{Out})^{**2}$$

Out = current answer

CA = correct answer

$$\text{Error} = \frac{1}{2}(x * w1 + y * w2 - CA)^{**2}$$

Chain rule -> the derivative of $f(g(x)) = f'(g(z))g'(z)$

Error is made up of $f(g) = \frac{1}{2}(g)^{**2}$ and $g(z) = CA - x * w1 - y * w2$

$f'(g(x)) = df/dg = g$ (by the power rule)

$g'(w1) = dg/dw1 = -x$

$g'(w2) = dg/dw2 = -y$

Chain rule

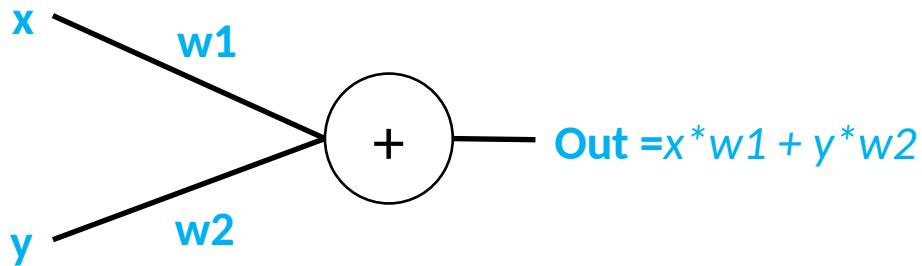
$$df/dg * dg/dw1 = (CA - x * w1 - y * w2)^{*} - x$$

$$df/dg * dg/dw2 = (CA - x * w1 - y * w2)^{*} - y$$

direction of greatest increase in error! Reverse sign get greatest decrease in error.

Use this to make Error go to 0 by adding a fraction (Learning rate) of these values to $w1$ and $w2$.

How it works – single neuron



$$\text{Error} = \frac{1}{2}(\text{CA}-\text{Out})^{**2}$$

Out = current answer

CA = correct answer

$$\text{Error} = \frac{1}{2}(x*w1 + y*w2 - CA)^{**2}$$

Chain rule -> the derivative of $f(g(x)) = f'(g(z))g'(z)$

Error is made up of $f(g) = \frac{1}{2}(g)^{**2}$ and $g(z) = CA - x*w1 - y*w2$

$f'(g(x)) = df/dg = g$ (by the power rule)

$g'(w1) = dg/dw1 = -x$

$g'(w2) = dg/dw2 = -y$

Run backpropagation.py to demo

Chain rule

$$df/dg * dg/dw1 = (CA - x*w1 - y*w2)^* - x$$

$$df/dg * dg/dw2 = (CA - x*w1 - y*w2)^* - y$$

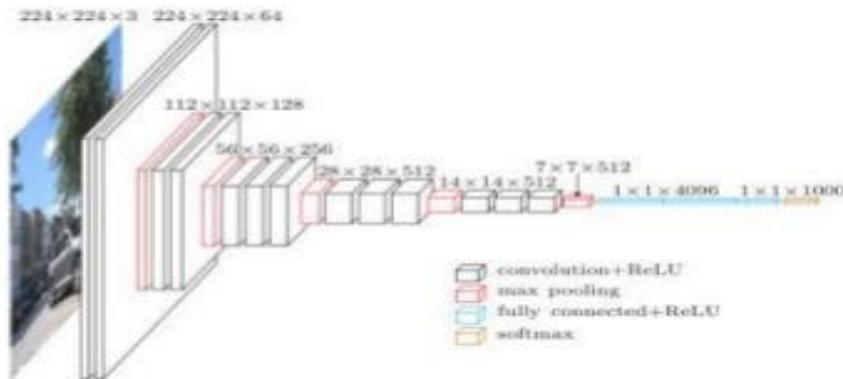
direction of greatest increase in error! Reverse sign get greatest decrease in error.
Use this to make Error go to 0 by adding a fraction (Learning rate) of these values to *w*₁ and *w*₂.

How it works – at scale

- Most **models** have hundreds of thousands of neurons and 100s of millions of parameters (like w_1 and w_2)
- So there are lots of forward and backwards computations
- GPUs do most of the calculations using huge multidimensional matrices

What's a Model?

VGG16 Pre-Trained Model

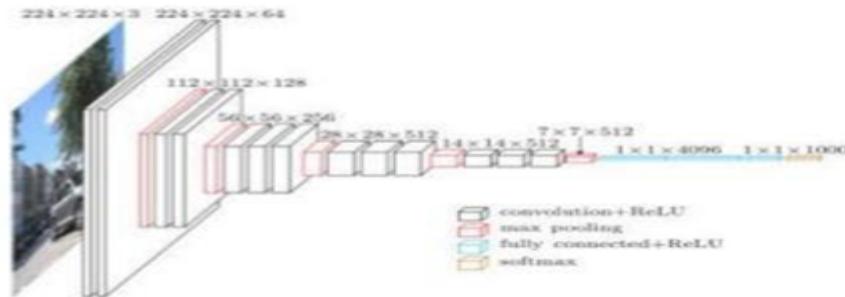


The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

Citation: <https://arxiv.org/pdf/1409.1556.pdf>

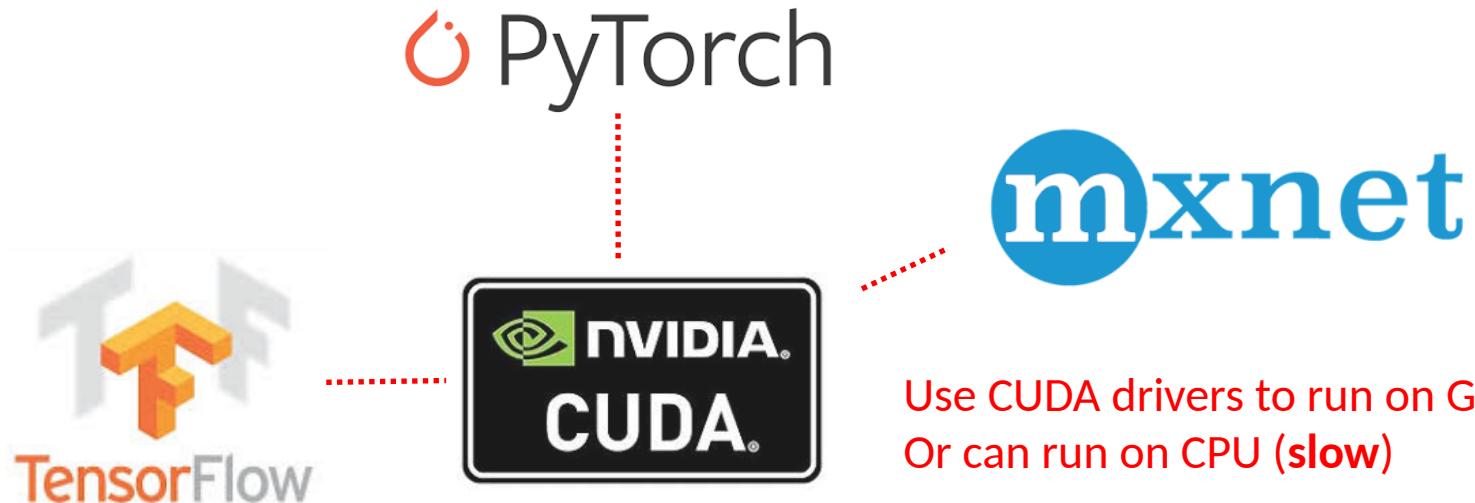
- A pretrained Neural Network.
 - Each layer consists of hundreds of convolutional filters which are in turn built from neurons
 - Each layer connected to next layer
- There are a bunch of these pretrained models, useful for ‘transfer learning’ which is coming up

Training, Testing



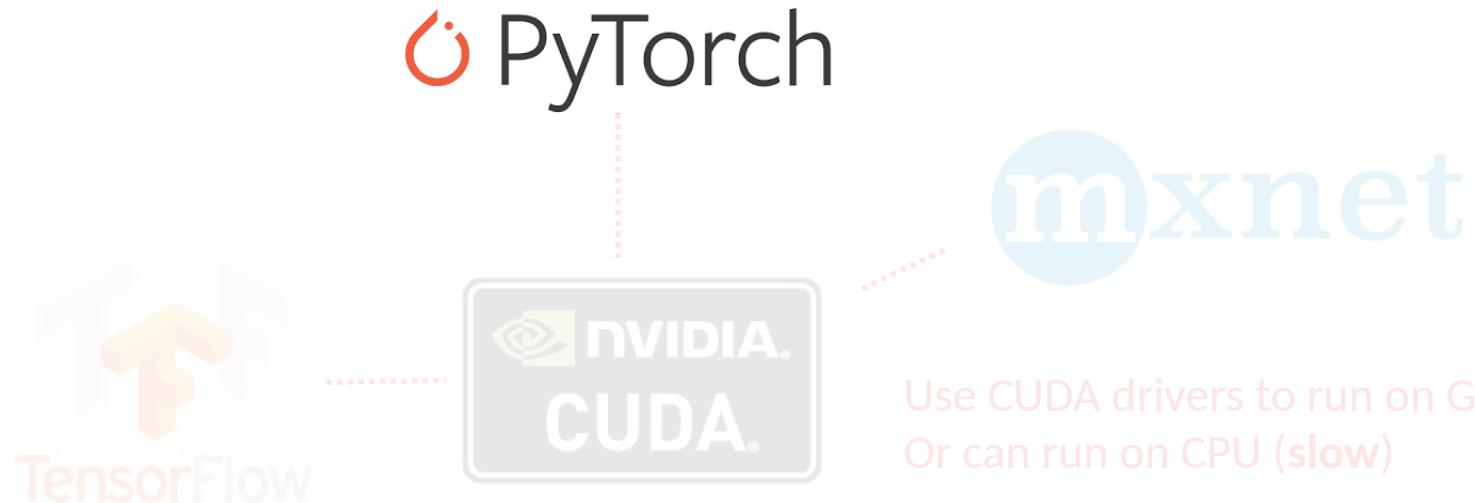
- Need: Data, Architecture (model), Error Function
- Train
 - Define an Error function
 - Feed images through, error is difference between what model thinks image is versus what the image actually is
 - Backpropagate to reduce this error
 - Do this lots of times
 - Eventually error is low for images, ignoring overfitting, model should generalize and recognize new images similar to old
- Test
 - Feed previously unseen images to model, see if it predicts correctly

Frameworks do the heavy lifting



See https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software for others

Frameworks do the heavy lifting



See https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software for others

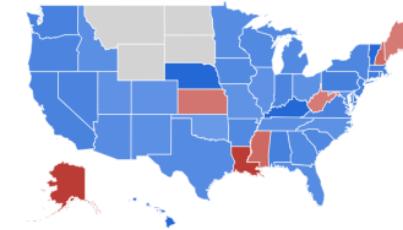
Pytorch -Why?

● Pytorch
Search term

● Tensorflow
Search term

Compared breakdown by subregion

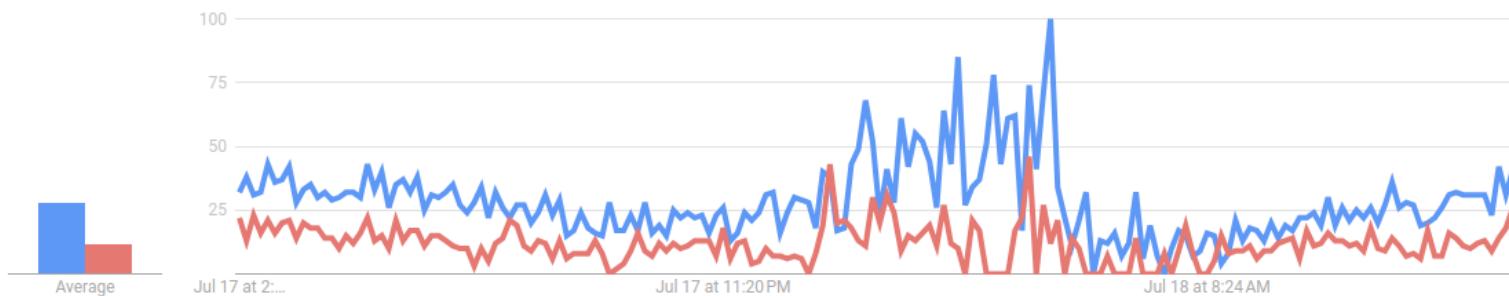
● Pytorch ● Tensorflow



Interest over time ?



Color Intensity represents percentage of searches [LEARN MORE](#)



<https://trends.google.com/trends/explore?date=now%201-d&geo=US&q=Pytorch,Tensorflow&hl=en>

What can DL do?

- It is not Artificial General Intelligence (AGI)! But can handle defined problems in:
 - Sequential Data (speech, character) recognition, prediction, translation, comprehension in the form of sentiment analysis
 - Time series structured data- A big deal. Analyzing data, predicting outcomes. (in store sales, failure prediction, cancer datasets)
 - Reinforcement Learning - learning from the environment (AlphaGo, GPT4)
 - Computer Vision (CV)

Convolutional Networks

Classification



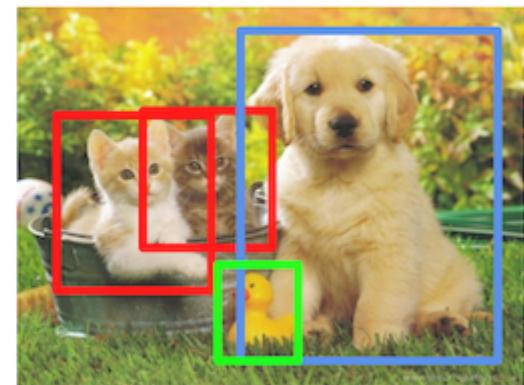
CAT

Classification
+ Localization



CAT

Object Detection



CAT, DOG, DUCK

excel in image recognition, segmentation, localization, medical image diagnosis, video manipulation... even art

Convolutional Networks



+



=



- Neural style transfer – error function optimizes image pixels to look like both images on left

How can I train a model to recognize labeled images I supply?

Called ‘transfer learning’ or ‘fine tuning’, almost certainly what you want to do

Take an existing trained model and reuse most of it to recognize something new.

- Much faster than training model from scratch
 - Need a lot less data as well
 - Works well if the things you want to recognize are similar to the things the model was originally trained on
-
1. Load a fully trained model
 2. Replace the last fully connected (FC) layers with new FC layers that have **number of outputs = number of classes you want to distinguish**
 3. Train this model with your training data
-
- See https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

Transfer Learning

Show Pytorch: <https://pytorch.org/tutorials/beginner/introyt/trainingyt.html#the-training-loop>

Show Fastai (wraps Pytorch): see notebook

Emphasize how to pick a timm model

DL practitioner skillset 1

- Know a bit about GPUs (Nvidia not AMD)
- OS - Linux – period. Just enough to get by
- Need a powerful machine (like Paperspace Gradient)
- Get used to ssh, terminal, and some scripting.
- Get used to Jupyter notebooks (great to teach, tough to debug, awful in version control)
- Git – for everything
- Need to know your vector operations (Linear Algebra)
- A bit about probability

DL practitioner skillset 2

- Python – but not like this

```
# forward pass
out1 = (x * w1 + y * w2)
error = 1 / 2 * (correct - out1) ** 2

print(f"For pass {i}, w1 ={w1}, w2={w2}, error is{error}")

# backward pass derivatives (dout1_dw1 dout1_dw2 defined at
derror_dout1 = -(correct - out1)

# chain rule
derror_dw1 = -derror_dout1 * dout1_dw1 # shows fastest inc
derror_dw2 = -derror_dout1 * dout1_dw2

#adjust weights
w1 = w1+learning_rate * derror_dw1;
w2 = w2+learning_rate * derror_dw2;
```

DL practitioner skillset 2

- Python -more like this

```
# Decode predictions into bboxes.  
for i in range(num):  
    decoded_boxes = decode(loc_data[i], prior_data, self.variance)  
    # For each class, perform nms  
    conf_scores = conf_preds[i].clone()  
  
    for cl in range(1, self.num_classes):  
        c_mask = conf_scores[cl].gt(self.conf_thresh)  
        scores = conf_scores[cl][c_mask]  
        if scores.dim() == 0:  
            continue  
        l_mask = c_mask.unsqueeze(1).expand_as(decoded_boxes)  
        boxes = decoded_boxes[l_mask].view(-1, 4)  
        # idx of highest scoring and non-overlapping boxes per class  
        ids, count = nms(boxes, scores, self.nms_thresh, self.top_k)  
        output[i, cl, :count] = \  
            torch.cat((scores[ids[:count]].unsqueeze(1),  
                      boxes[ids[:count]]), 1)
```

DL practitioner skillset 2

- Python – and realistically like this

```
for i in range(num):
    decoded_boxes = decode(loc_data[i], prior_data, self.variance)

    conf_scores = conf_preds[i].clone()

    for cl in range(1, self.num_classes):
        c_mask = conf_scores[cl].gt(self.conf_thresh)
        scores = conf_scores[cl][c_mask]
        if scores.dim() == 0:
            continue
        l_mask = c_mask.unsqueeze(1).expand_as(decoded_boxes)
        boxes = decoded_boxes[l_mask].view(-1, 4)

        ids, count = nms(boxes, scores, self.nms_thresh, self.top_k)
        output[i, cl, :count] =
            torch.cat((scores[ids[:count]].unsqueeze(1),
                      boxes[ids[:count]]), 1)
```

Lots of beta software
and missing documentation
Heavy use of list
comprehensions, iterators,
generators, matrix
Manipulation

DL practitioner skillset 3

- And... need to know data manipulation routines, image formatting, normalization data cleaning, directory and file manipulation...**but you pick this up as you go**



Pandas



Why is this hard

- When it does not work its hard to troubleshoot 100,000,000 params
- Hyperparam selection, learning rate, architecture, FC layer depth
- Must find sweet spot where your model generalizes well but does not overfit your data
- Split dataset 80% train, 10% test, 10% validation. When accuracy on test set starts to decrease you are probably starting to overfit

Why jump on board?

- A hot field
- Big advances being made with easy ideas
- Cyclic learning rates, learning rate finder, focal loss

Interested?

- Kaggle – data science competitions
 - Lots of how-tos, data scrubbing, AI techniques
 - Both images recognition and structured data
- FastAI Get up to snuff fast!
 - Fastai <https://course.fast.ai/>
 - **Its free.** Teaches coders how to use AI - both machine learning (random forest, boosted trees) and Deep Learning. 24 weeks of state of the art instruction, with a lively forum for support.