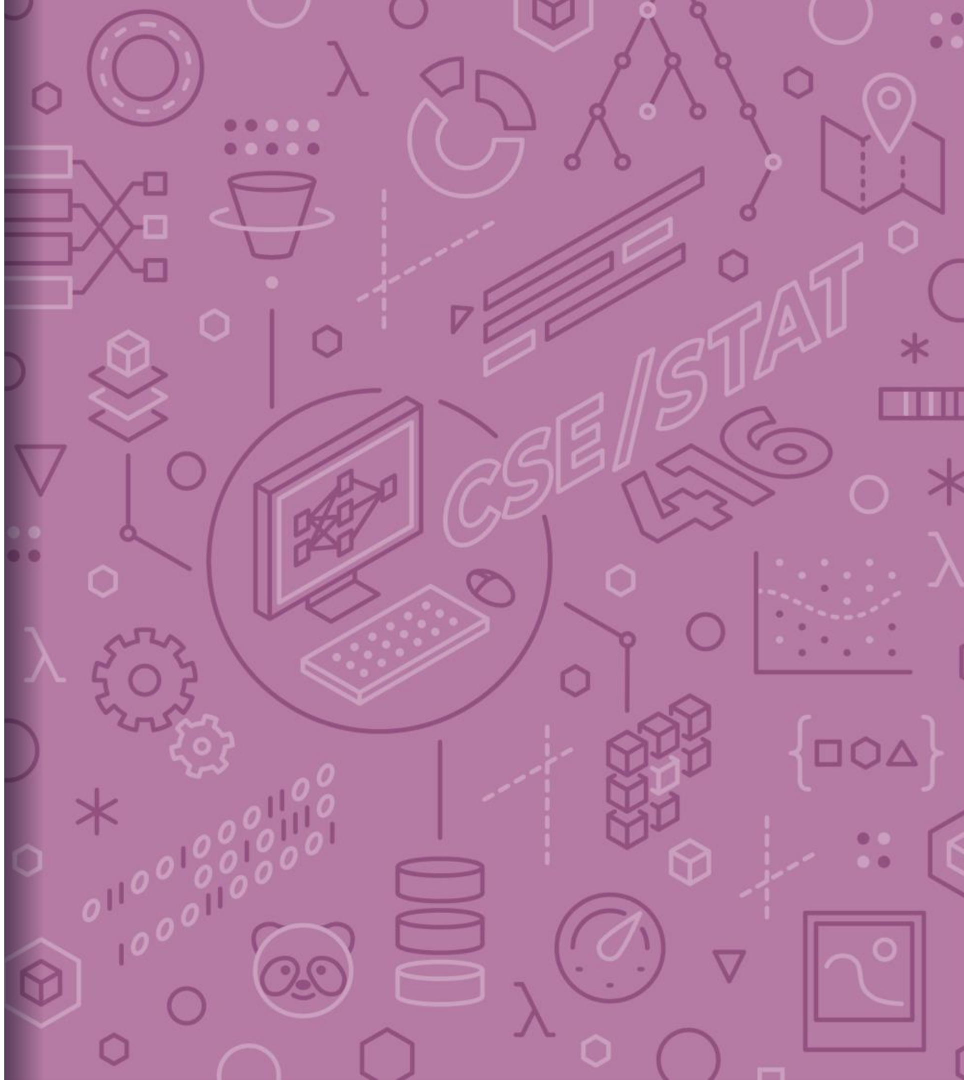
CSE/STAT 416

Convolutional Neural Networks

Hunter Schafer
Paul G. Allen School of Computer Science & Engineering
University of Washington

May 26, 2021



Logistics

Almost the end of the quarter!

- 👩‍🏫 Wed (5/26, today!): Convolutional Neural Networks
- 😎 Thur (5/27): Section on PyTorch
- 🧠 Fri (5/28): LR9 Due
- 🌞 Mon (6/1): Holiday
- 📝 Tue (6/2): HW9 Due (out today, more on this later)
- 👩‍🏫 Wed (6/3): Victory Lap and next steps
  - No pre-lecture video
  - Will have a Checkpoint for review from the quarter
- 😎 Thur (6/4): Review
- 🧠 Fri (6/5): LR10 Due (quarter reflection)
- 📝 Mon – Wednesday (6/7- 6/9): Final exam
  - Will send email in next few days with more info and resources

Deep Learning

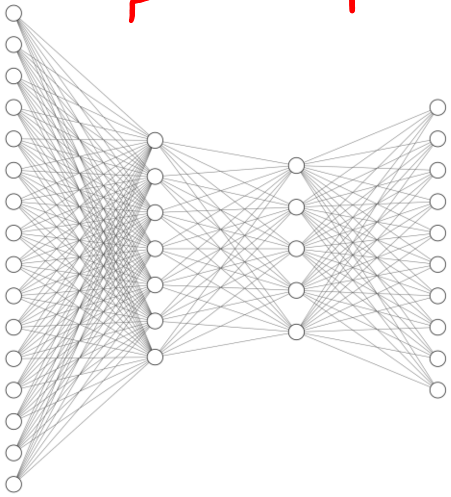
A lot of the buzz about ML recently has come from recent advancements in **deep learning.**

When people talk about "deep learning" they are generally talking about a class of models called **neural networks** that are a loose approximation of how our brains work.

Input

hidden layers

Output

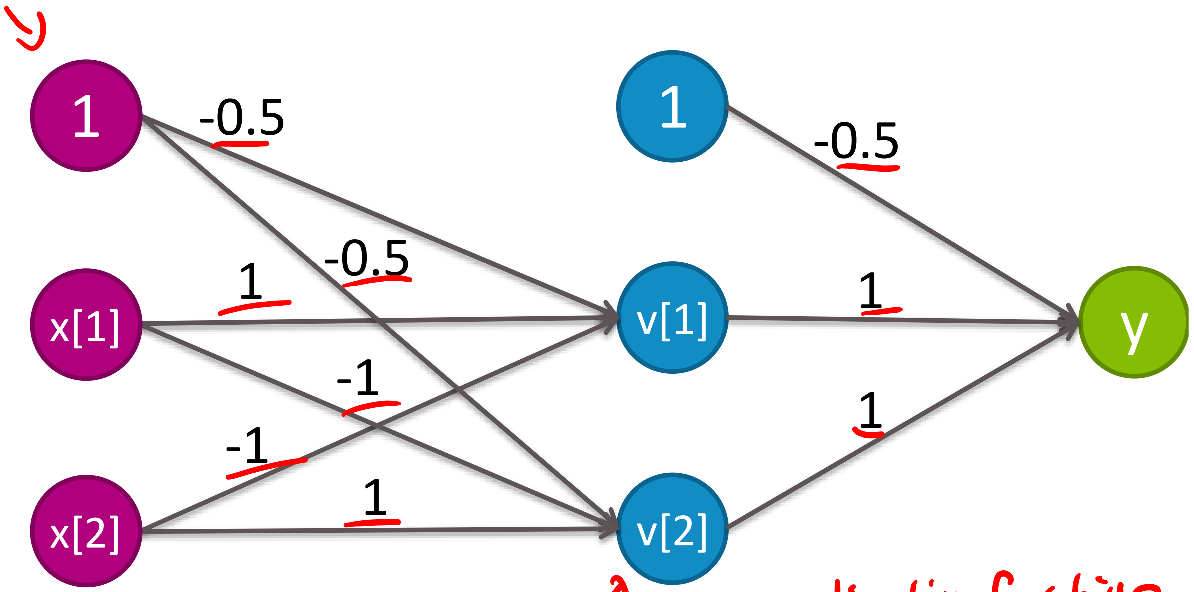


XOR

0.5

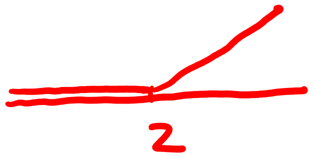
Notice that we can represent

$$x[1]\ XOR\ x[2] = (x[1]\ AND\ !\,x[2])\ OR\ (!\,x[1]\ AND\ x[2])$$



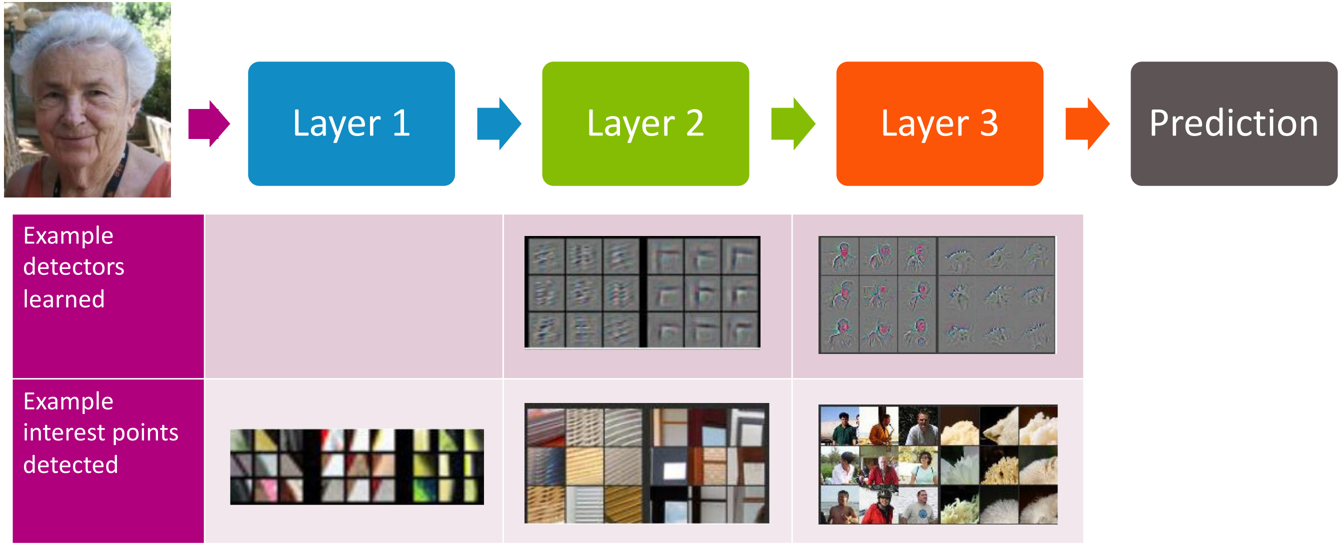
activation function g

$\sum_{j=1} w_j\, x[j] \Rightarrow$



NN to the Rescue

Neural Networks implicitly find these low level features for us!



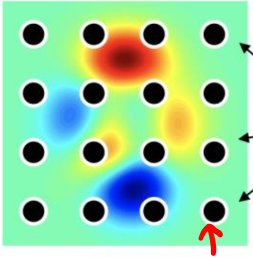
[Zeiler & Fergus '13]

Each layer learns more and more complex features

Hyperparameter Optimization

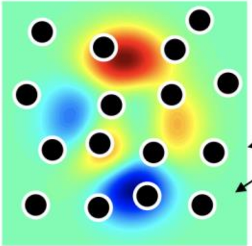
How do we choose hyperparameters to train and evaluate?

Grid search:



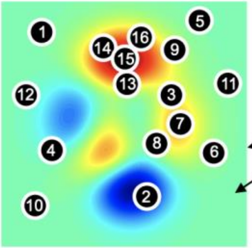
Hyperparameters on 2d uniform grid

Random search:



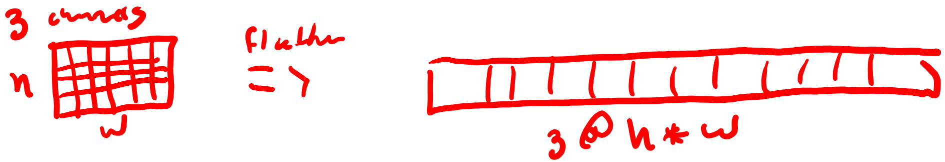
Hyperparameters randomly chosen

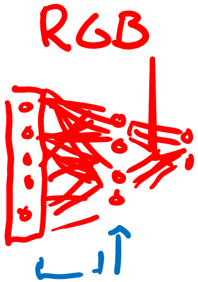
Bayesian Optimization:



Hyperparameters ***adaptively*** chosen

Image Challenges





Images are extremely high dimensional

- CIFAR-10 dataset are very small: 3@32x32
  - # inputs:

$$3 \cdot 32 \cdot 32 = 3072$$

- For moderate sized images: 3@200x200
  - # inputs: RGB

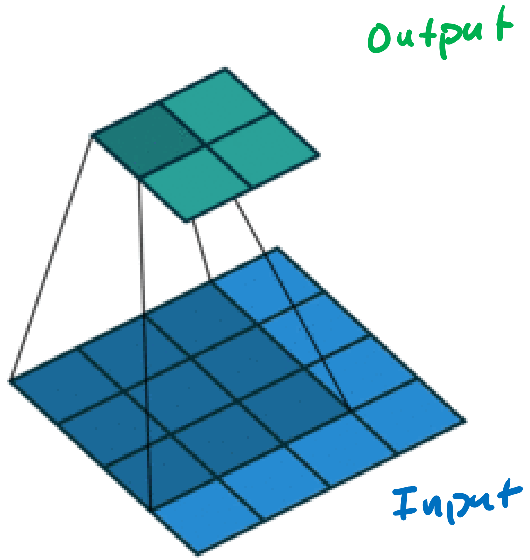
$$3 \cdot 200 \cdot 200 = 120{,}000$$

Images are structured, we should leverage this

Convolutional Neural Networks

Idea: Reduce the number of weights that need to be learned by looking at local neighborhoods of image.

Use the idea of a **convolution** to reduce the number of inputs by combing information about local pixels.



Convolution

Use a **kernel** that slides across the image, computing the sum of the element-wise product between the kernel and the overlapping part of the image

Image

3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

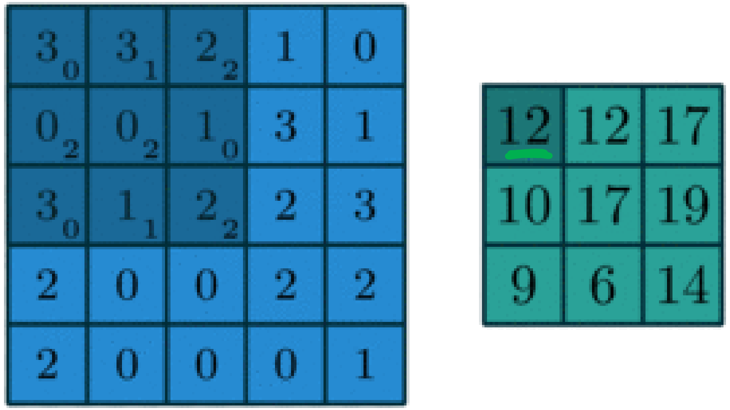
Kernel

0	1	2
2	2	0
0	1	2

$$= \begin{bmatrix} 0 & 3 & 4 \\ 0 & 0 & 0 \\ 0 & 1 & 4 \end{bmatrix} \xrightarrow{\text{sum}} \underline{12}$$

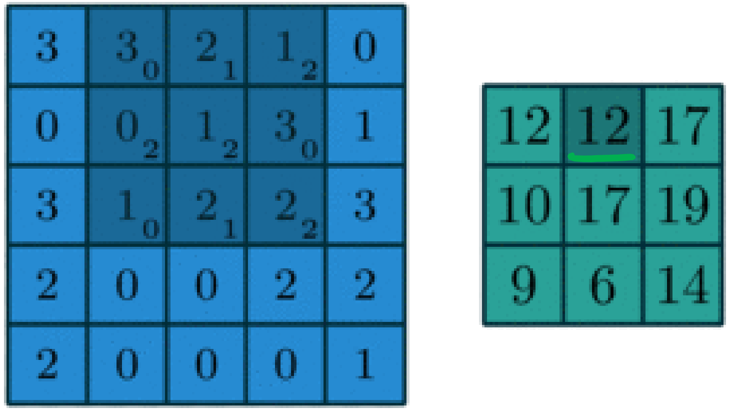
Convolution

The input image (blue), the kernel (dark blue, numbers lower right) slide over the image to produce a result (green)



Convolution

The input image (blue), the kernel (dark blue, numbers lower right) slide over the image to produce a result (green)

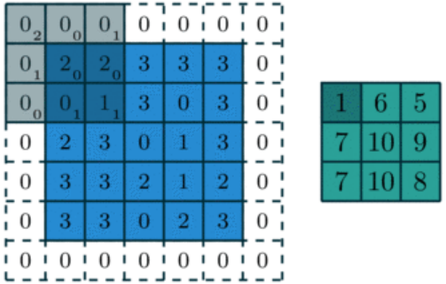


More Convolutions

You can specify a few more things about a kernel

- Kernel dimensions and values
- Padding size and padding values
- Stride (how far to jump) values

For example, a 3x3 kernel applied to a 5x5 image with 1x1 zero padding and a 2x2 stride



Poll Everywhere

Think

1.5 min

pollev.com/cs416

What is the result of applying a convolution using this kernel on this input image?

Use 1x1 zero padding and a 2x2 stride

Image

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Kernel

1	1
0	2

Poll Everywhere

Group

3 min

pollev.com/cs416

What is the result of applying a convolution using this kernel on this input image?

Use 1x1 zero padding and a 2x2 stride

Result size: 3x3

Image

0	0	0	0	0	0
0	1	2	3	4	0
0	5	6	7	8	0
0	9	10	11	12	0
0	13	14	15	16	0
0	0	0	0	0	0

Kernel

1	1
0	2

$$= \begin{pmatrix} 2 & 6 & 0 \\ 23 & 35 & 8 \\ 13 & 29 & 16 \end{pmatrix}$$

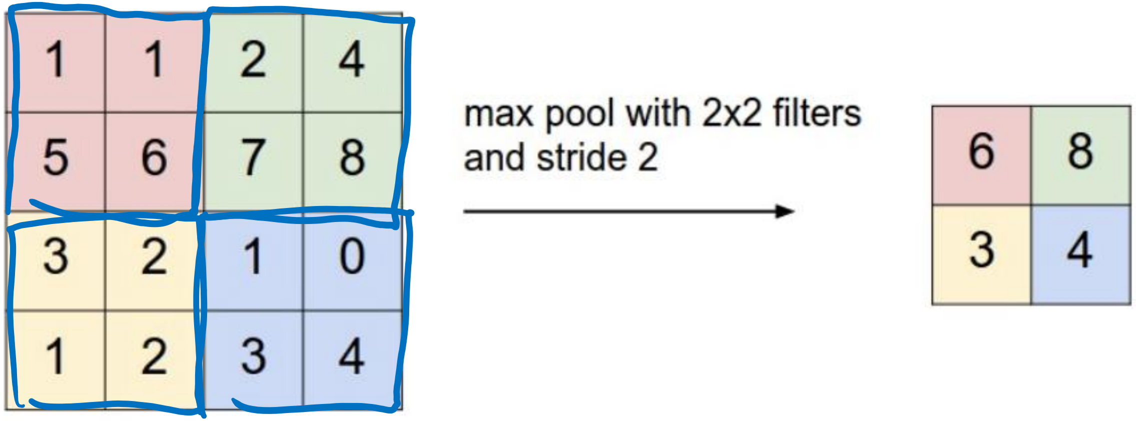
Pooling

Another core operation that is similar to a convolution is a **pool**.

- Idea is to down sample an image using some operation
- Combine local pixels using some operation (e.g. max, min, average, median, etc.)

Typical to use **max pool** with 2x2 filter and stride 2

- Tends to work better than average pool



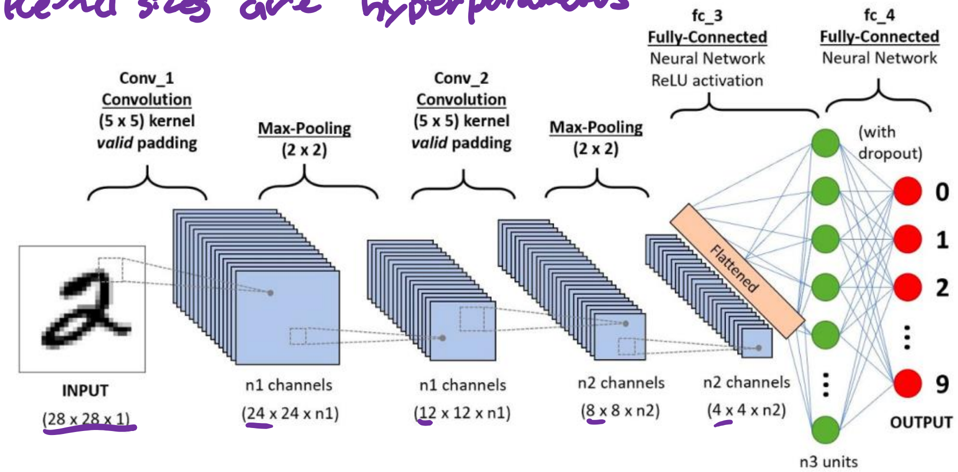
Convolutional Neural Network

Combine convolutions and pools into pre-processing layers on image to learn a smaller, information dense representation.

Example architecture for hand-written digit recognition

- Each convolution section uses many different kernels (increasing depth of channels)
- Pooling layers down sample each channel separately
- Usually ends with fully connected neural network

n1, n2, kernel sizes are hyperparameters

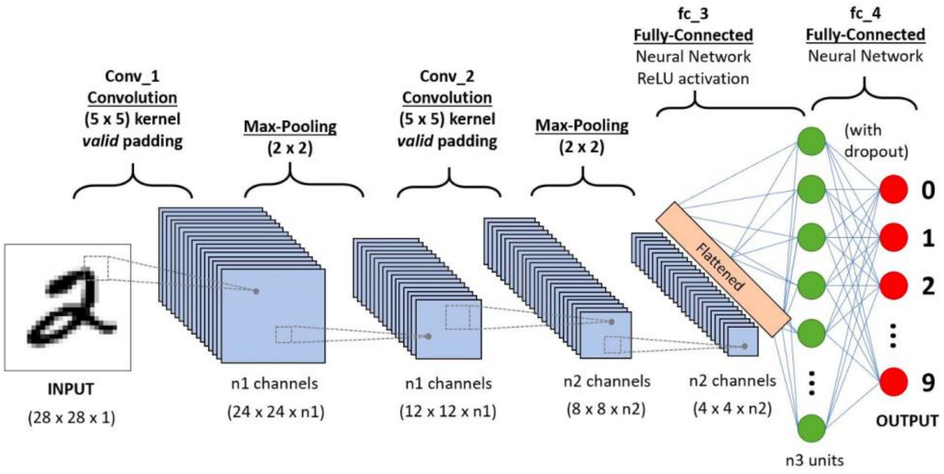


Depth 1 n1 n1 n2 n2

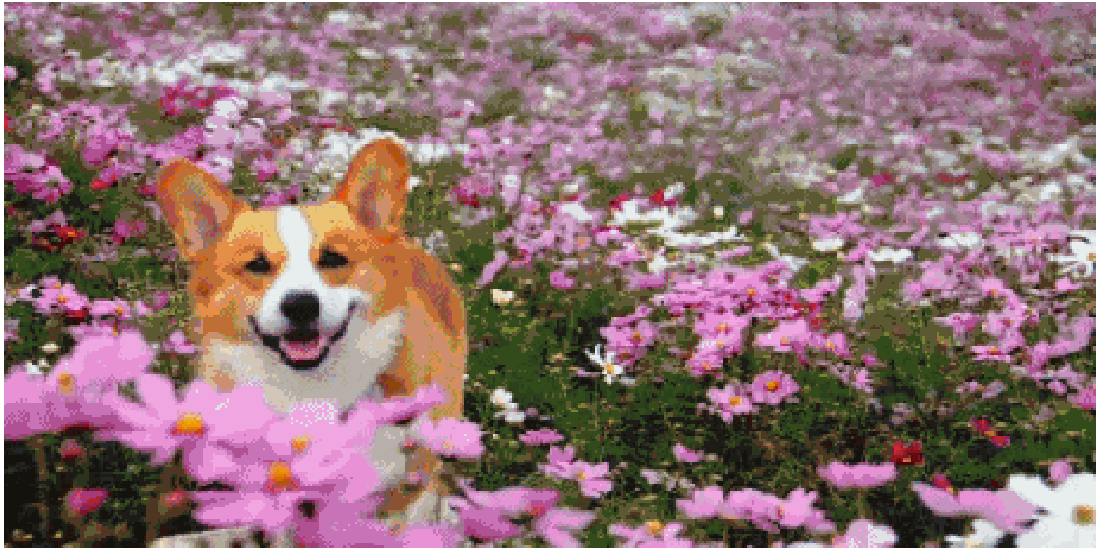
Convolutional Neural Network

Why does this help?

- Only need to learn a small number of values (kernel weights) that get applied to the entire image region by region
  - This is called weight-sharing
  - Gives efficiency + shift invariance
- Pooling helps reduce the number of inputs by “blurring” the image without losing too much info.



Brain Break

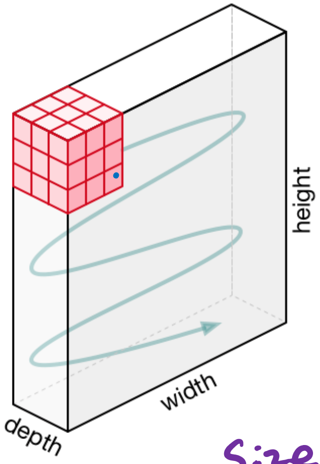


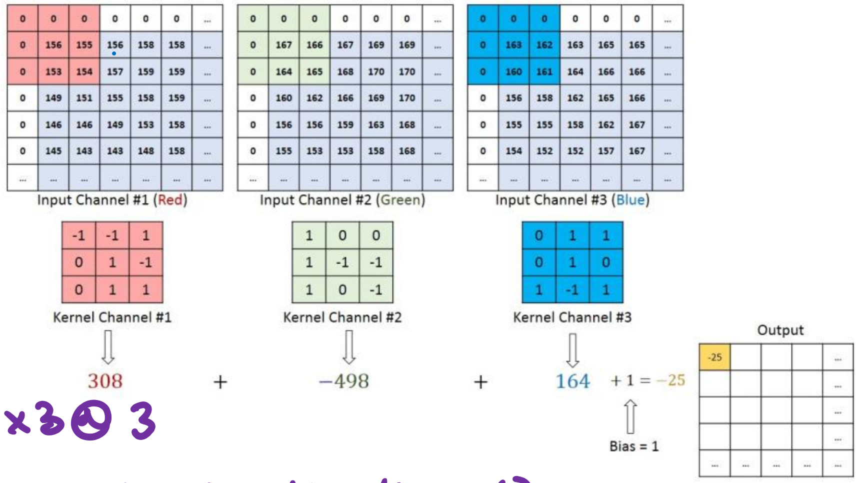
CNN with Color Images

How does this work if there is more than one input channel?

- Usually, use a 3 dimensional **tensor** as the kernel to combine information from each input channel

5





Size: 3x3x3

How many weights in the kernel?

3·3·3=27

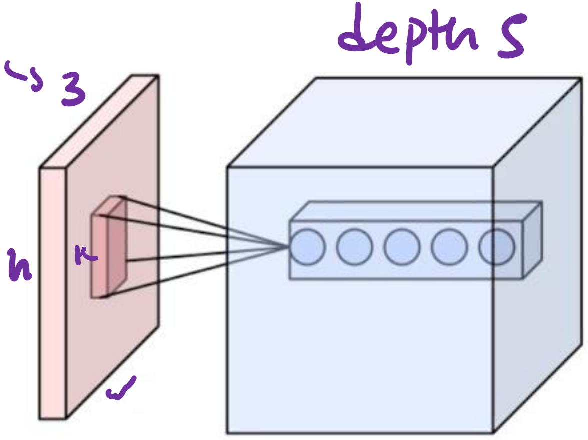
CNN with Color Images

1 kernel (k×k) will need k·k·3 weights
d kernels (k×k) will need k·k·3·d weights

Another way of thinking about this process is each kernel is a neuron that looks at the kernel-size pixels in a neighborhood

If there are 5 output channels in a conv layer, only need to learn the weights for the 5 neurons

- These neurons are a bit different since they look at the pixels that overlap with the window at each position.



d k×k kernels with I input channels will need

$$\underbrace{k \cdot k \cdot I}_{\text{weights in 1 kernel}} \cdot d$$

d kernels

Poll Everywhere

Think

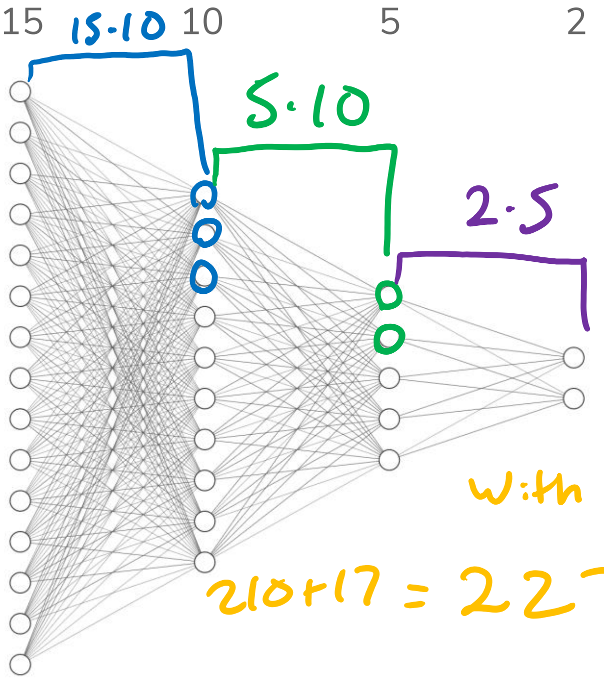
1 min

pollev.com/cs416

Consider a plain neural network below, how many weights need to be learned?

Completely ignore intercept terms

Inputs   Outputs



15·10

5·10

2·5

= 210

With intercept

210+17 = 227

Poll Everywhere

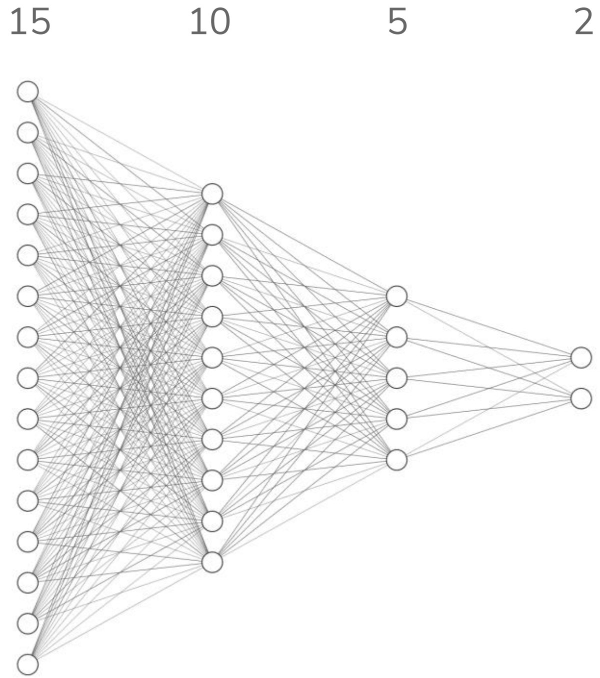
Group

2 min

pollev.com/cs416

Consider a plain neural network below, how many weights need to be learned?

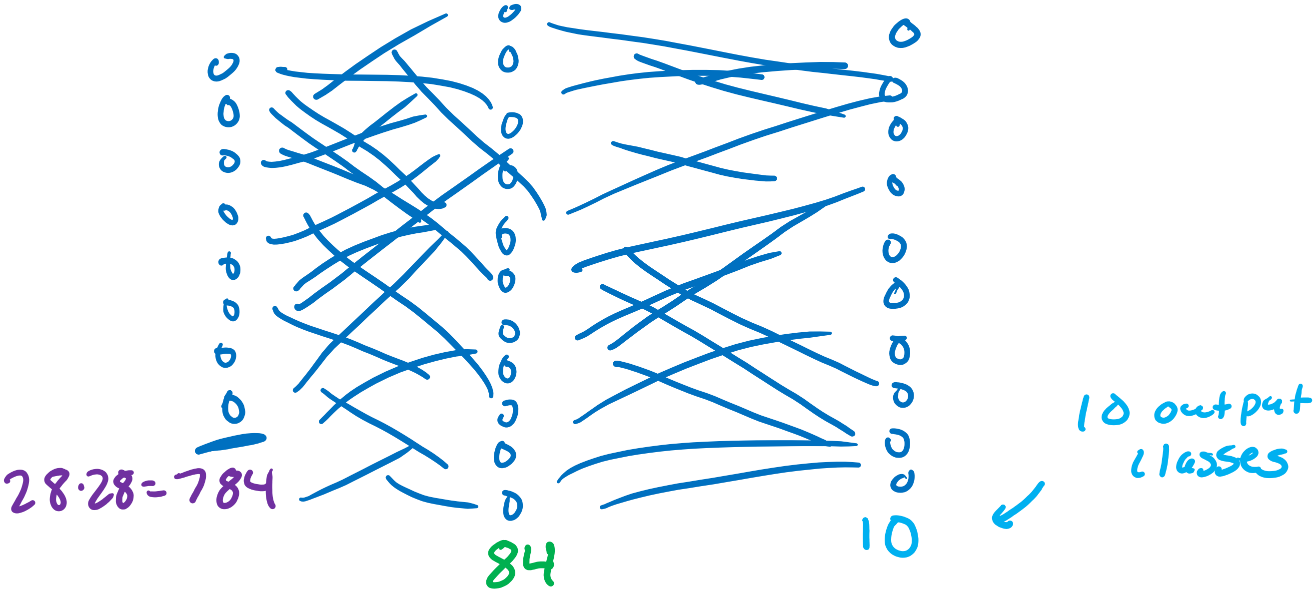
Completely ignore intercept terms



Weight Sharing

Consider solving a digit recognition task on 28x28 images.
Suppose I wanted to use a hidden layer with 84 neurons

Without Convolutions:



Num Weights: $784 \cdot 84 + 84 \cdot 10 = 66{,}696$

Weight Sharing

Total weights learned = 250 + 5000 + 27,720 = 32,970

Consider solving a digit recognition task on 28x28 images. ≪ 66K

Suppose I wanted to use a hidden layer with 84 neurons

With Convolutions (assume n1=10, n2=20)

\# inputs flattened = 4·4·20 = 320

conv_2

$\underbrace{5\cdot5\cdot10}_{\text{\# weights for one kernel}}\cdot\underbrace{20}_{\text{20 kernels}} = 5000$

weights in fc_3 + fc_4

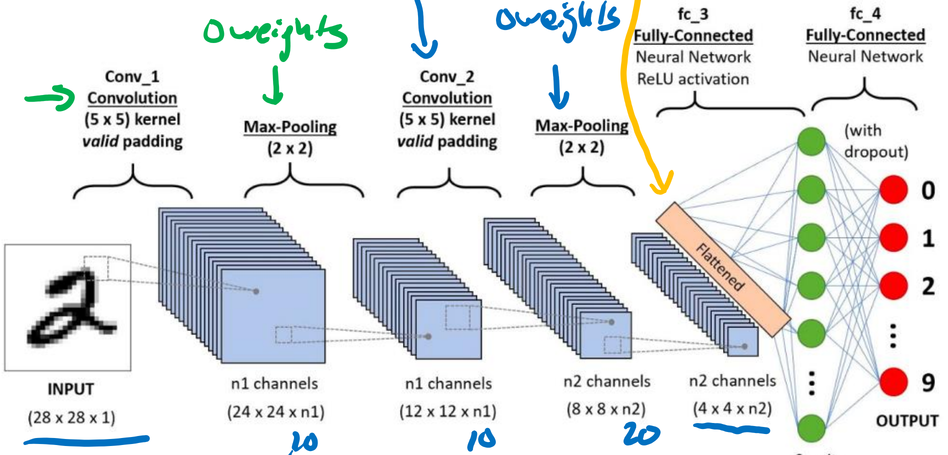
$320\cdot84 + 84\cdot10 = 27,720$

conv_1

55·1·10 = 250

0 weights

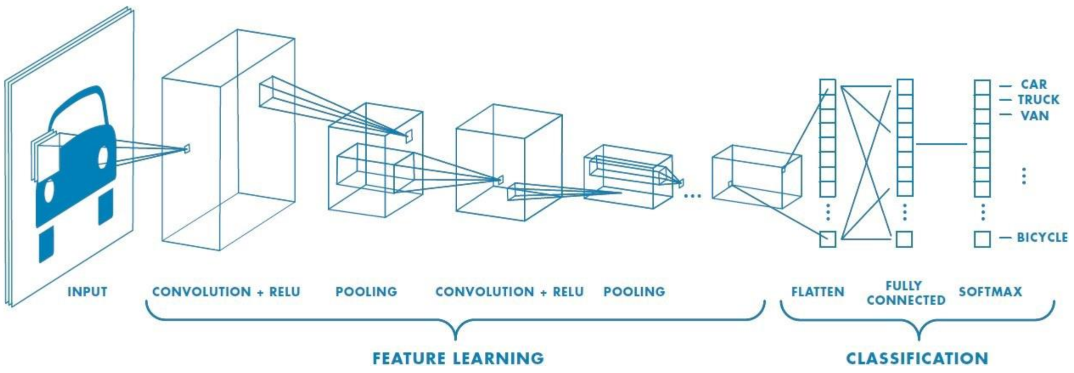
0 weights



General CNN Architecture

CNNs generally (not always) have architectures that look like the following

- A series of Convolution + Activation Functions and Pooling layers. It's very common to do a pool after each convolution.
- Then after some number of these operations, flatten the image to work with the final neural network



Features

The learned kernels are exactly the “features” for computer vision!

They start simple (corners, edges) and get more complex after more layers

Layer 1 → Layer 2 → Layer 3 → Prediction

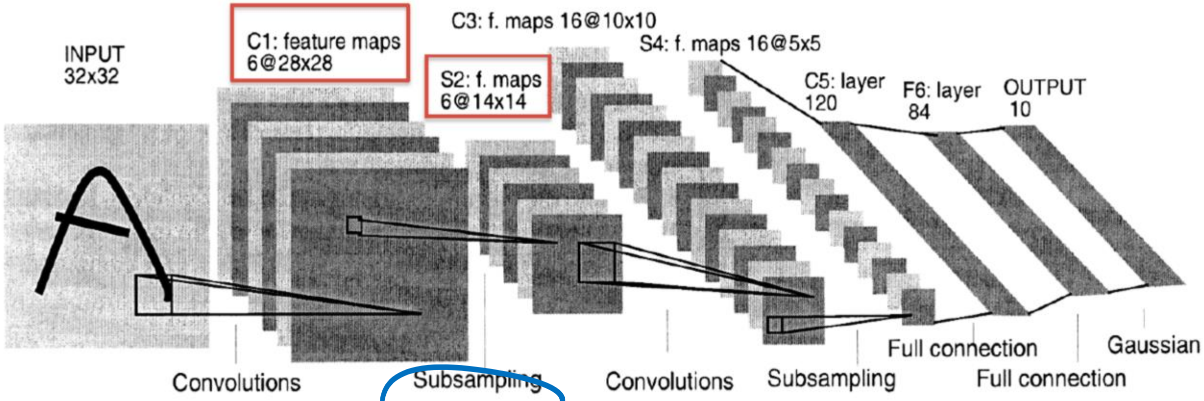
	Layer 1	Layer 2	Layer 3
Example detectors learned			
Example interest points detected			

[Zeiler & Fergus ‘13]

CNN Success

CNNs have had remarkable success in practice

LeNet, 1990s



Pooling

CNN Success

LeNet made 82 errors on MNIST (popular hand-written digit dataset).

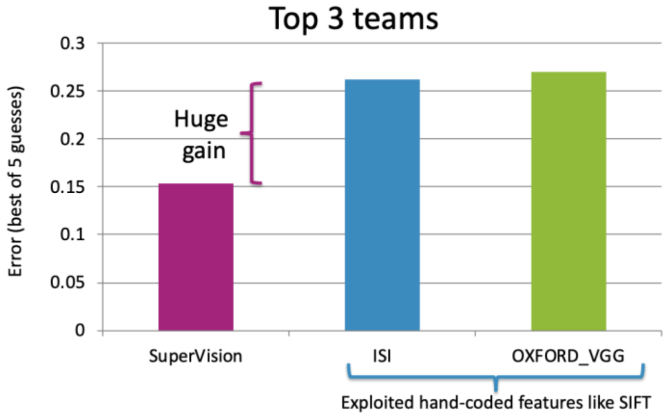
CNN Success

ImageNet 2012 competition:

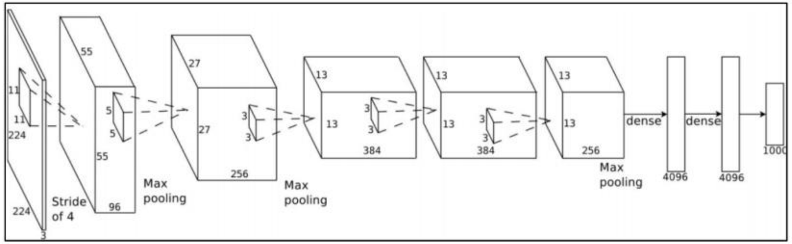
- 1.2M training images
- 1000 categories

Winner: SuperVision

- 8 layers, 60M parameters [Krizhevsky et al. '12]
- Top-5 Error: 17%

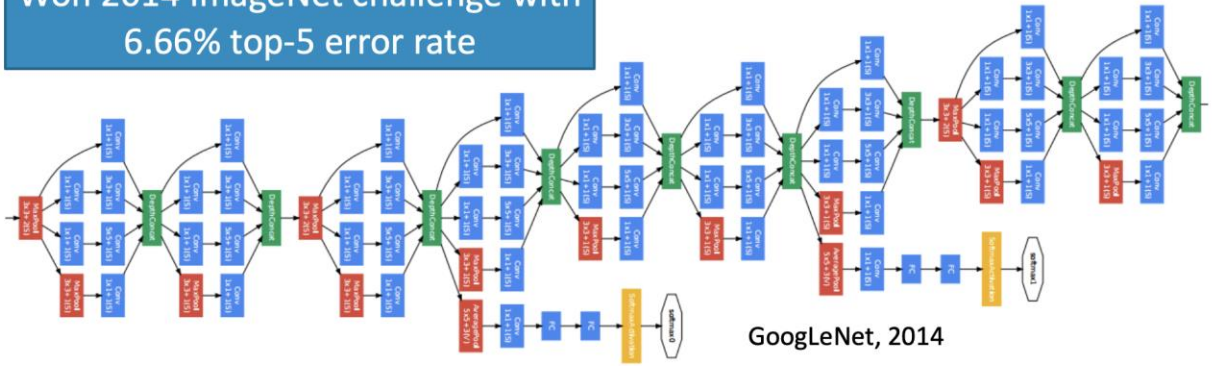


AlexNet



CNN Success

Won 2014 ImageNet challenge with 6.66% top-5 error rate

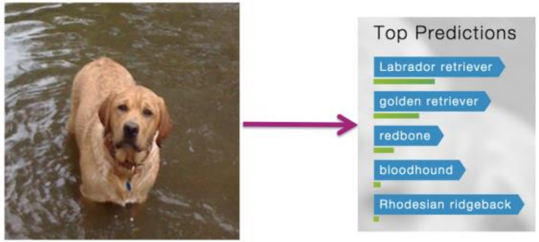


GoogLeNet, 2014

Huge CNN depth has proven helpful in recognition systems… Maybe because images contain hierarchical structure (faces contain eyes contain edges, etc.)

Applications

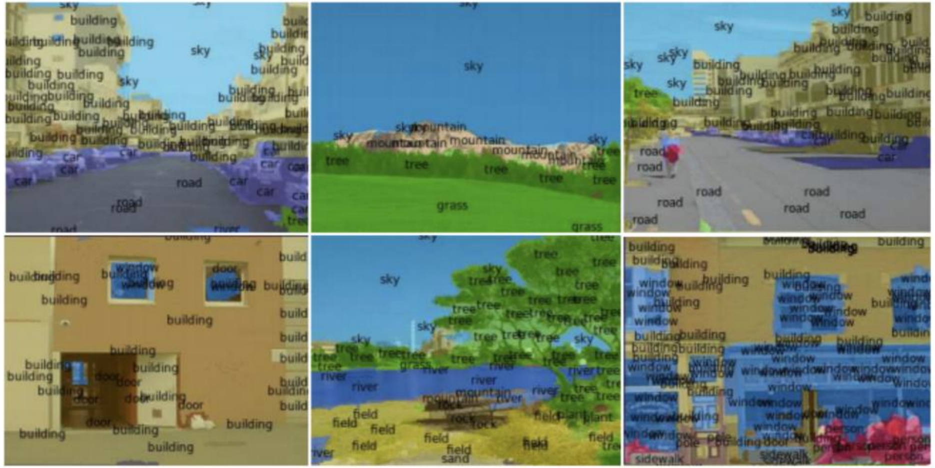
Image Classification



Input: **x**
Image pixels

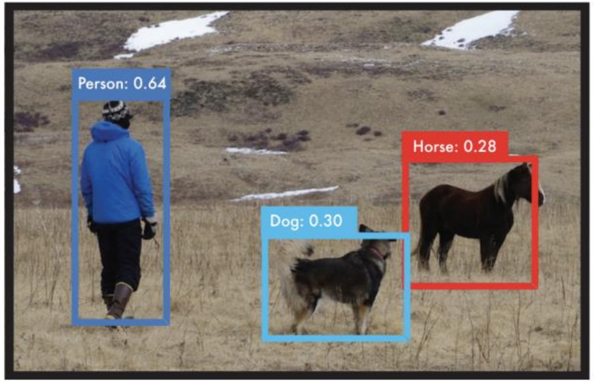
Output: y
Predicted object

Scene Parsing [Farabet et al. ‘13]



Applications

Object Detection [Redmon et al. 2015] (http://pjreddie.com/yolo/)



Product Recommendation

Brain Break

Deep Learning in Practice

Pros

No need to manually engineer features, enable automated learning of features

Impressive performance gains

- Image processing
- Natural Language Processing
- Speech recognition

Making huge impacts in most fields

Cons

Requires a LOT of data

Computationally really expensive

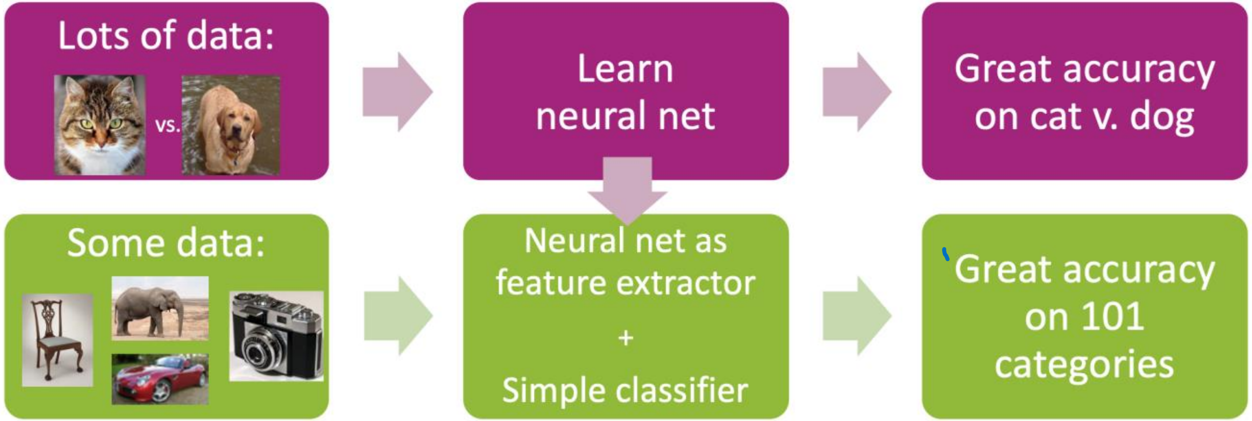
- Environmentally, extremely expensive ([Green AI]())

Hard to tune hyper-parameters

- Choice of architecture (we've added even more hyper-parameters)
- Learning algorithm

Still not very interpretable

A Tale of 2 Tasks



If we don't have a lot of data for Task 2, what can we do?

Idea: Use a model that was trained for one task to help learn another task.

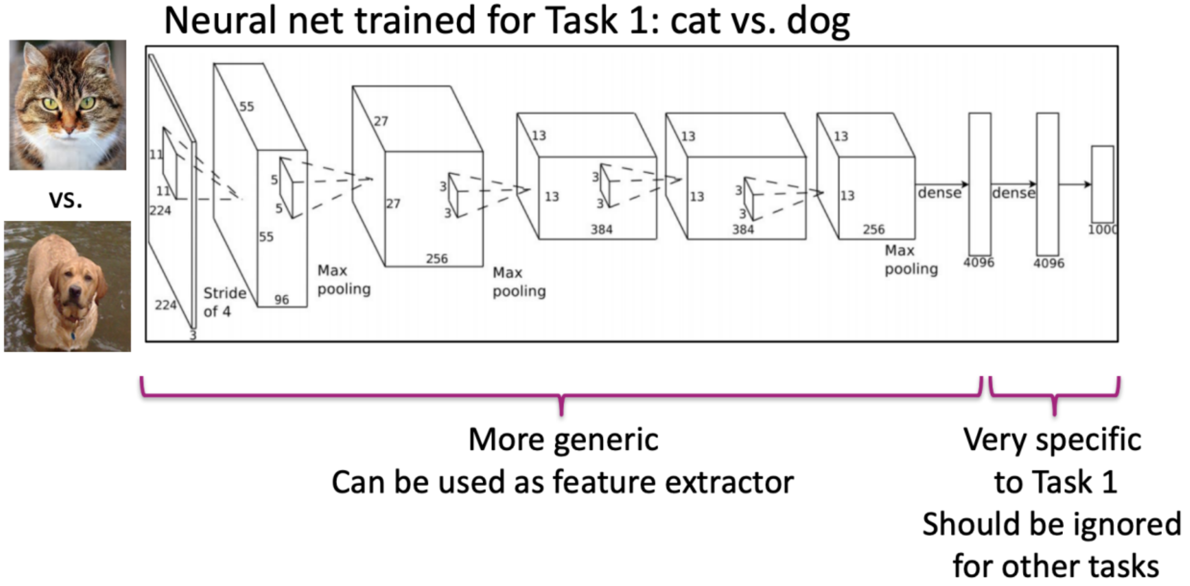
- An old idea, explored for deep learning by Donahue et al. '14 & others

CNNs

What is learned in a neural network?

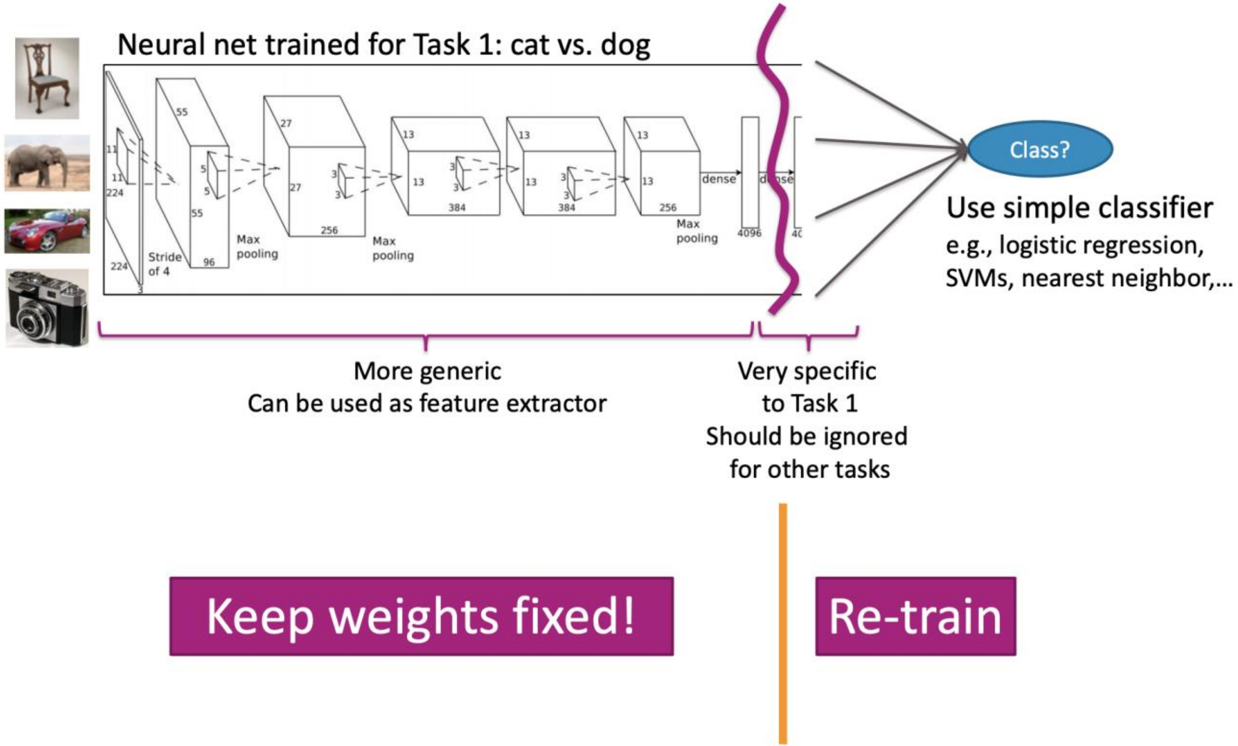
Initial layers are low-level and very general.

- Usually not sensitive/specific to the task at hand



Transfer Learning

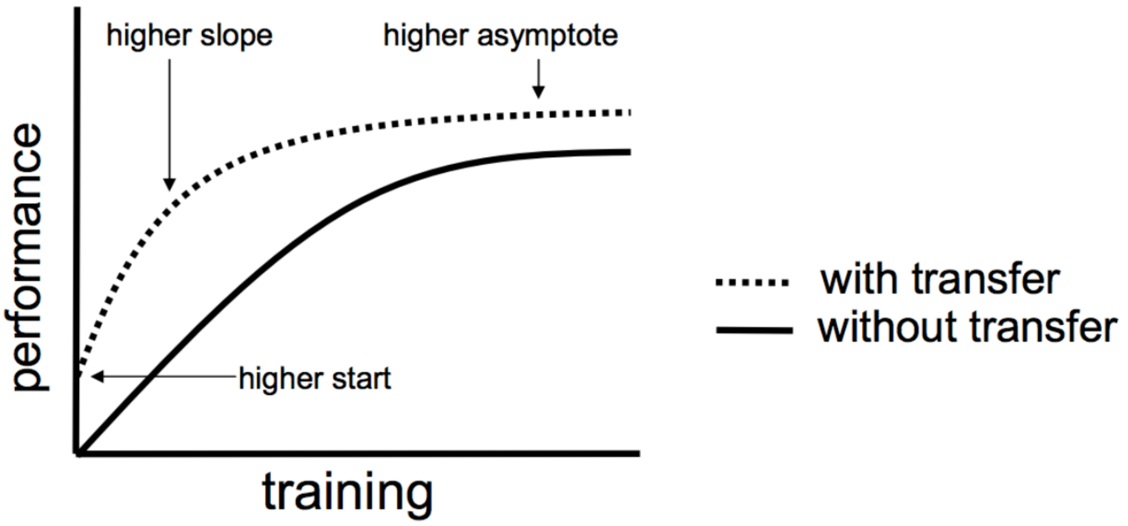
Share the weights for the general part of the network



Transfer Learning

If done successfully, transfer learning can really help. Can give you

- A higher **start**
- A higher **slope**
- A higher **asymptote**



NN Failures

While NNs have had amazing success, they also have some baffling failures.

"No one adds noise to things in real applications"

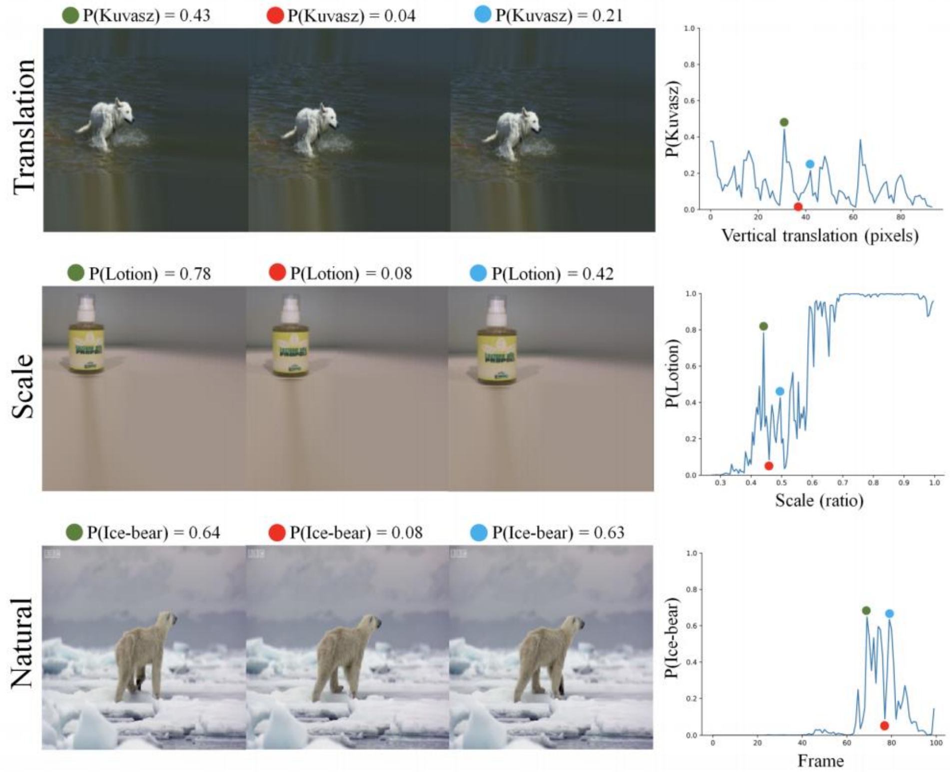
Not true!

- Hackers will hack
- Sensors (cameras) are noisy!

NN Failures

They even fail with "natural" transformations of images

[Azulay, Weiss preprint]

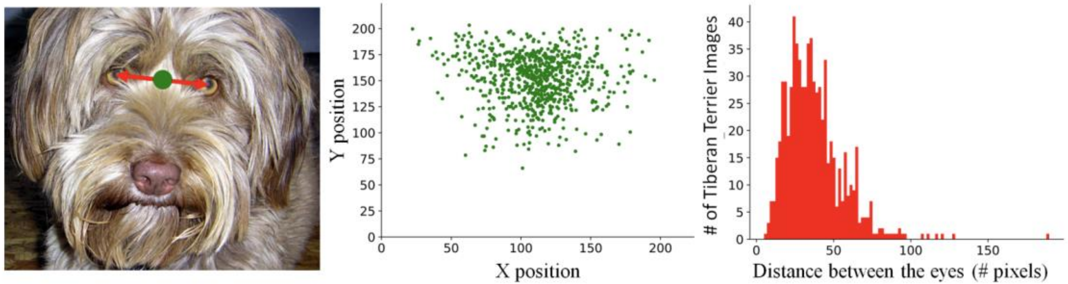


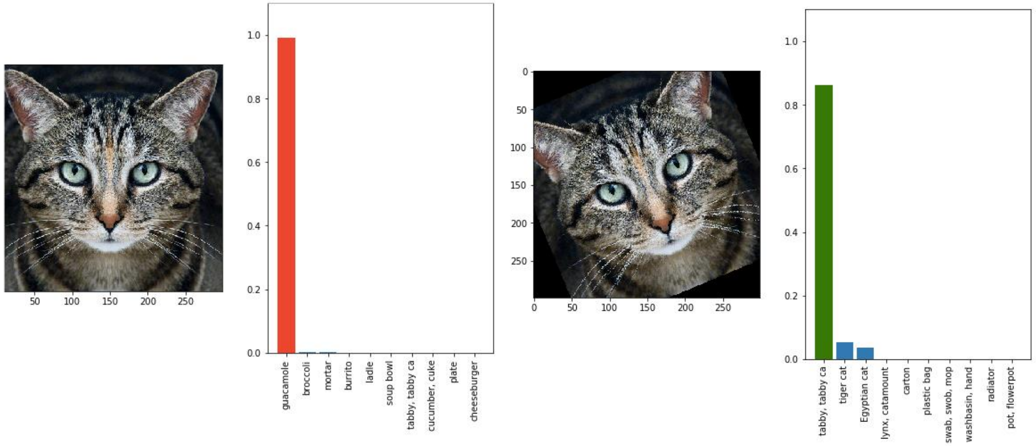
NN Failures

Objects can be created to trick neural networks!

Dataset Bias

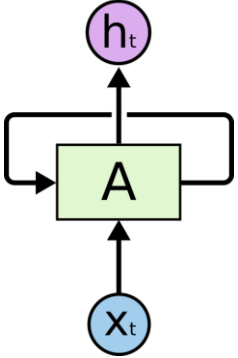
Datasets, like ImageNet, are generally biased





One approach is to augment your dataset to add random permutations of data to avoid bias.

Further Reading



Dealing with Variable Length Sequences (e.g. language)

- Recurrent Neural Networks (RNNs)
- Long Short Term Memory Nets (LSTMs)
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Reinforcement Learning

- Google DeepMind AlphaGo Zero

Generative Adversarial Networks

- How to learn synthetic data

Green AI

HW9

Your last assignment involves using a modern neural network library to make predictions using the CIFAR-10 dataset.

We recommend you use Google Colab for this assignment so that you can use their free GPU

- Nobody

- Google Colab:

Your first task is to read through the PyTorch tutorial to learn how to use their library

- Section tomorrow will introduce some stuff, but reading tutorial and documentation is critical