CSE/STAT 416

Convolutional Neural Networks

Amal Nanavati
University of Washington
July 27, 2022

Adapted from Hunter Schafer's slides

Administrivia

- Timeline:
  - **Next Week**: Clustering
  - **Following Week**: Dimensionality Reduction, Recommender Systems
  - **Then**: Course Recap & Final
- Deadlines:
  - HW5 released TODAY, due Tues 8/2 11:59PM
  - Learning Reflection 6 due Fri, 7/29 11:59PM

HW5
Walkthrough

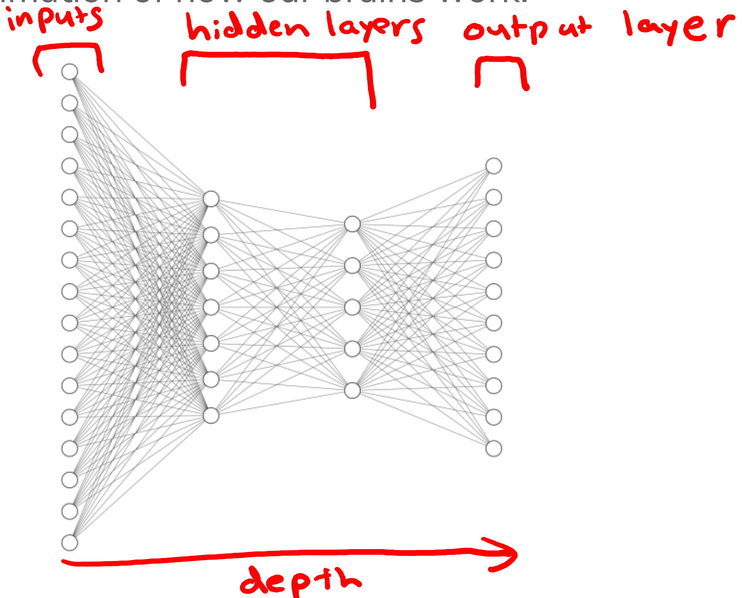
- Nobody
- Google Colab:

Recap: Neural Networks

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.

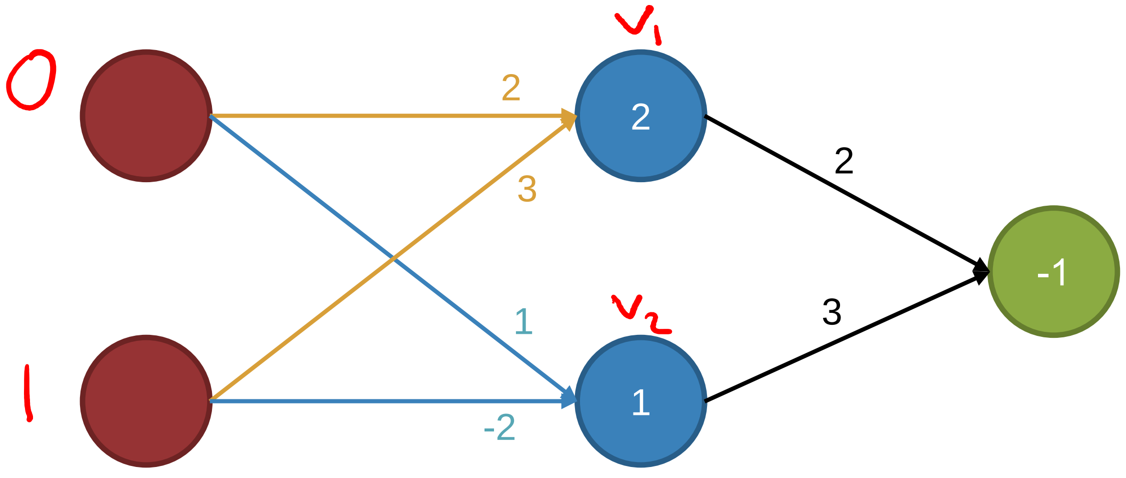


Poll Everywhere

Think

2 mins

- Compute the output for input (0, 1). There is a sign activation function on the hidden layers and output layer.



$$v_1 = \text{sign}(2 + 2\cdot 0 + 3\cdot 1) = \text{sign}(5) = 1$$

$$v_2 = \text{sign}(1 + 1\cdot 0 - 2\cdot 1) = \text{sign}(-1) = 0$$

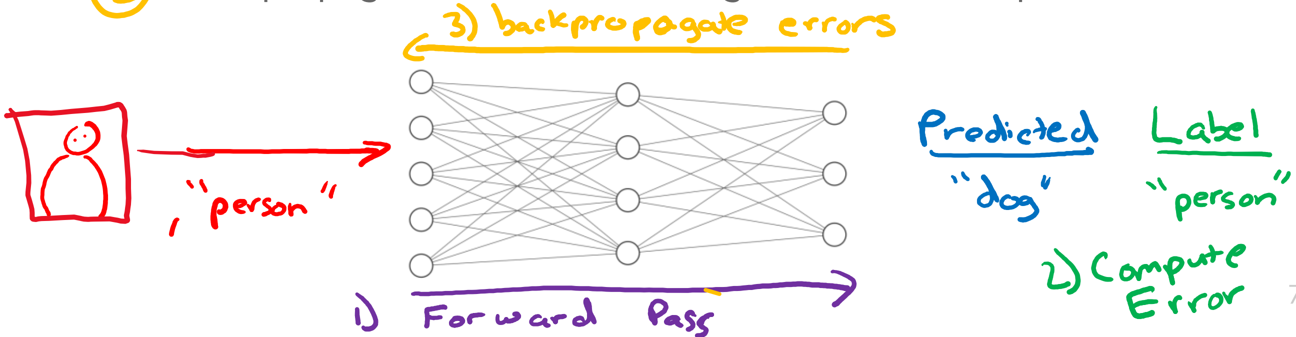
$$y = \text{sign}(-1 + 2\cdot 1 + 3\cdot 0) = \text{sign}(1) = 1$$

Backpropagation

What does gradient descent do in general? Have the model make predictions and update the model in a special way such that the new weights have lower error.

To do gradient descent with neural networks, we generally use **backpropagation.**

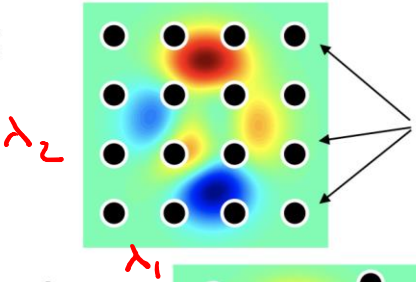
1. Do a forward pass of the data through the network to get predictions
2. Compare predictions to true values
3. Backpropagate errors so the weights make better predictions



Hyperparameter Optimization

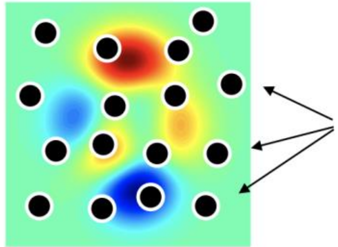
How do we choose hyperparameters to train and evaluate?

Grid search:



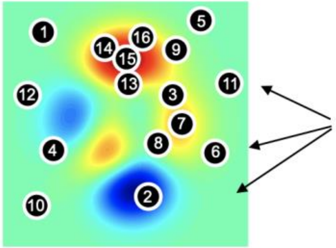
Hyperparameters on 2d uniform grid

Random search:



Hyperparameters randomly chosen

Bayesian Optimization:



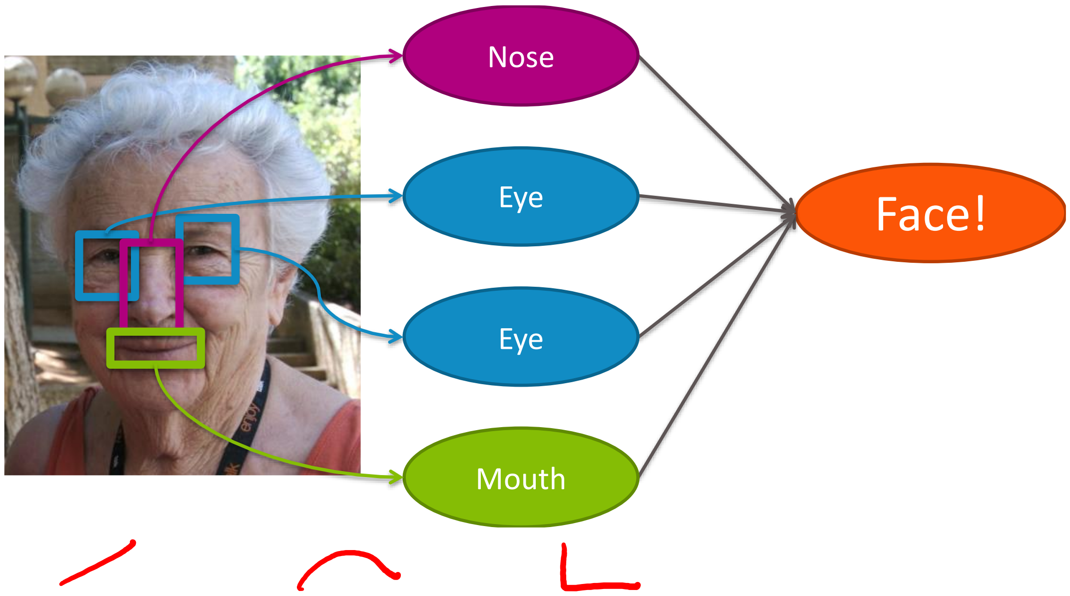
Hyperparameters ***adaptively*** chosen

Applying NNs to Computer Vision

Image Features

Features in computer vision are local detectors

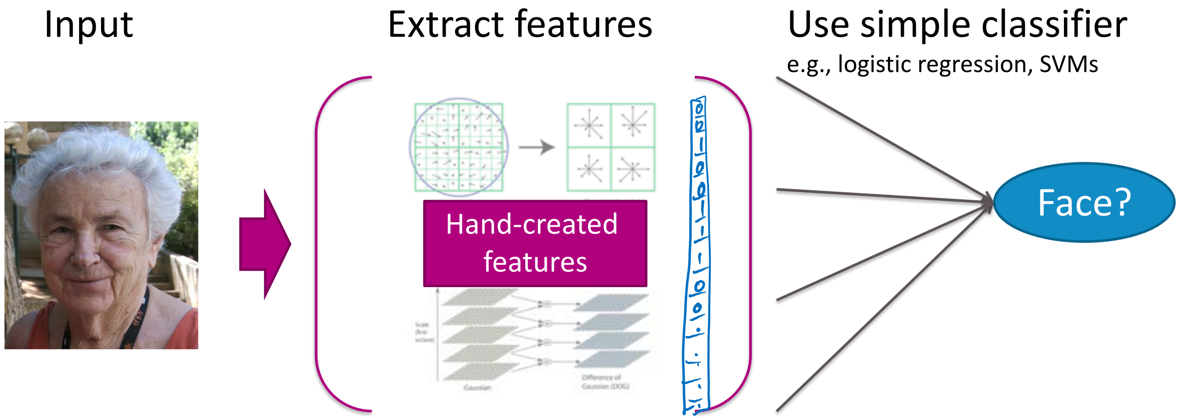
- Combine features to make prediction



In reality, these features are much more low level (e.g. Corner?)

The Past

A popular approach to computer vision was to make hand-crafted features for object detection



Relies on coming up with these features by hand (yuck!)

NNs to the Rescue

Neural Networks implicitly find these low level features for us!

Layer 1 → Layer 2 → Layer 3 → Prediction

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

[Zeiler & Fergus '13]

Each layer learns more and more complex features

Poll Everywhere

Think

1 min

Dall-E2 : 3.5B

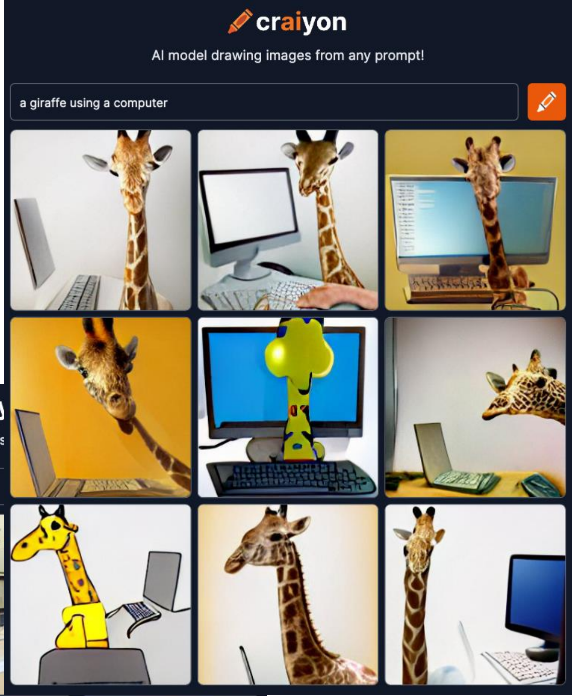
Dall-E : 12B

pollev.com/cs416

- The models we have seen so far have ≤ 100 parameters (weights, biases). How many parameters do you think DALL-E Mini has?

		Class Votes
(a)	0.4B	1
(b)	1B	2
(c)	12B	2
(d)	90B	3
(e)	175B	6

https://www.craiyon.com/
(formerly Dall-E Mini)



Convolutions

Image Challenges

Images are extremely high dimensional

3 channels/features (RGB)
width
heigth

- CIFAR-10 dataset are very small: 3@32x32
  - # inputs: $3 \cdot 32 \cdot 32 = 3072$ input neurons

    Hidden Layer of Size 4:

    $3072 \cdot 4 + 4 = 12{,}291$ parameters

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

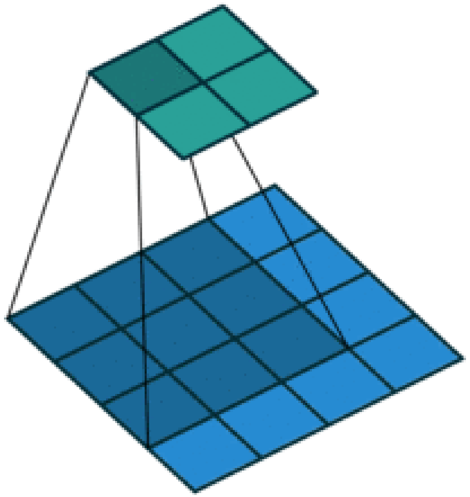
    $3 \cdot 200 \cdot 200 = 120{,}000$

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

$$3 \cdot 0 + 3 \cdot 1 + 2 \cdot 2 + 0 \cdot 2 + 0 \cdot 2 + 1 \cdot 0 + 3 \cdot 0 + 1 \cdot 1 + 2 \cdot 2 = 12$$

Image

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

Kernel

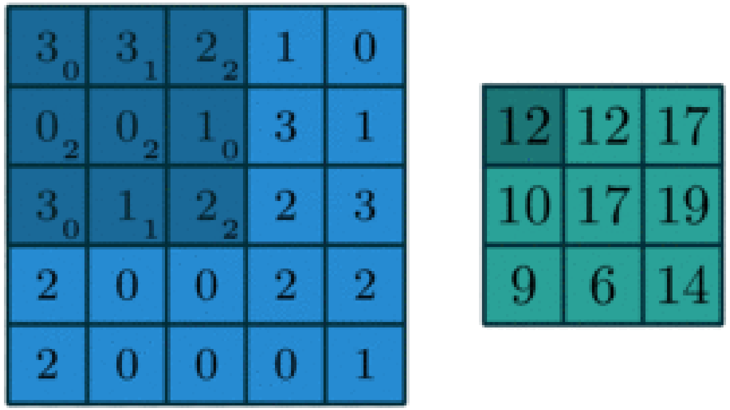
0	1	2
2	2	0
0	1	2

Output

$$= \begin{bmatrix} 12 & & \\ & & \\ & & \end{bmatrix}$$

Convolution

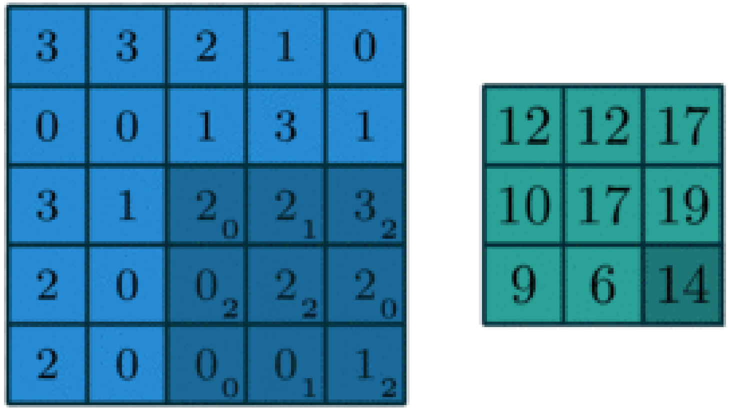
The input image (blue), the kernel (dark blue, numbers lower right) slide over the image to produce a result (green)



Convolution

End of one convolution with a particular kernel

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



Special Kernels

The numbers in the kernels determine special properties

Maintains same image

↓

Identity

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Edge Detection

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Sharpen

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Box Blur

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

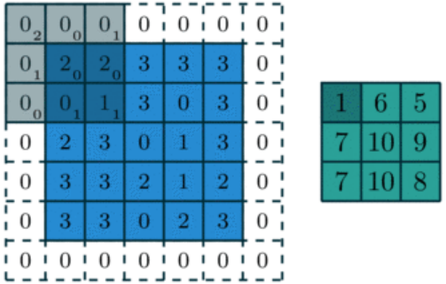
Convolutional Neural Networks (CNNs) learn the right weights for each kernel they use! Generally not interpretable!

Hyper-parameters of a Single Convolution

You can specify a few more things about a kernel

- Kernel dimensions — 5x5, 3x3, 10x10
- Padding size and padding values — typically 0
- 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

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

1:30

Poll Everywhere

Group

3 min

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

Use 1x1 zero padding and a 2x2 stride

Result: 3×3

Image

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

Kernel

1	1
0	2

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

Convolutional Neural Networks

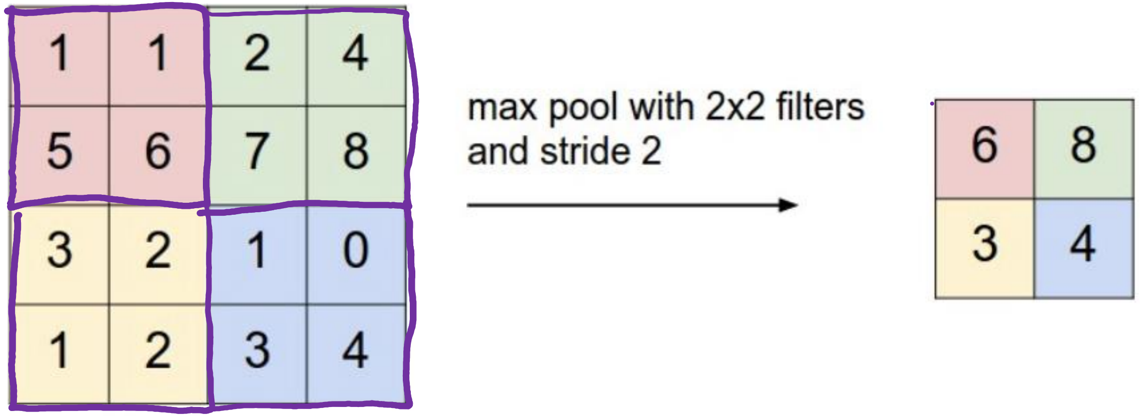
Pooling

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



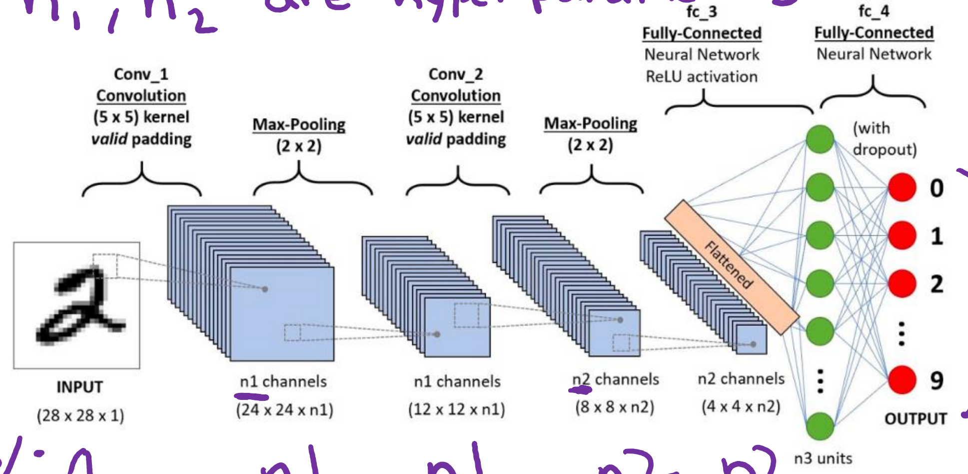
Convolutional Neural Network

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 downsample each channel separately
- Usually ends with fully connected neural network

n_1, n_2 are hyperparameters



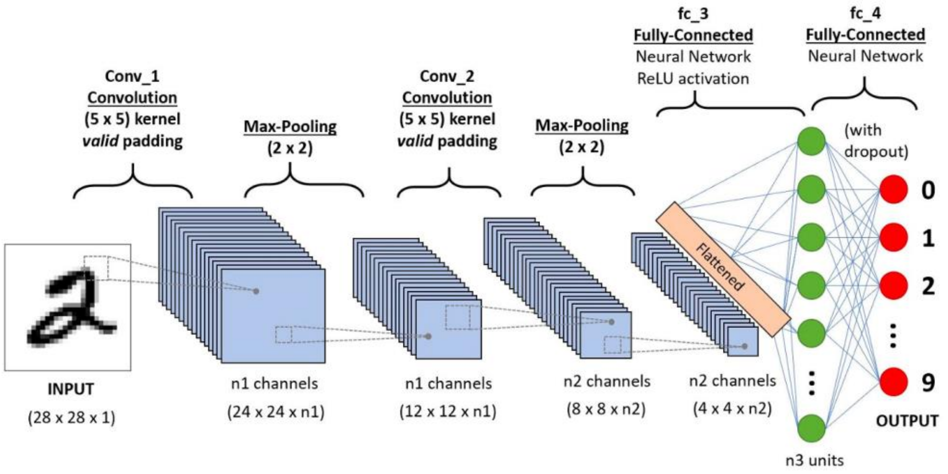
what digit is in the image?

features/channels: 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 lets us focus on features from larger and larger regions of the original image.

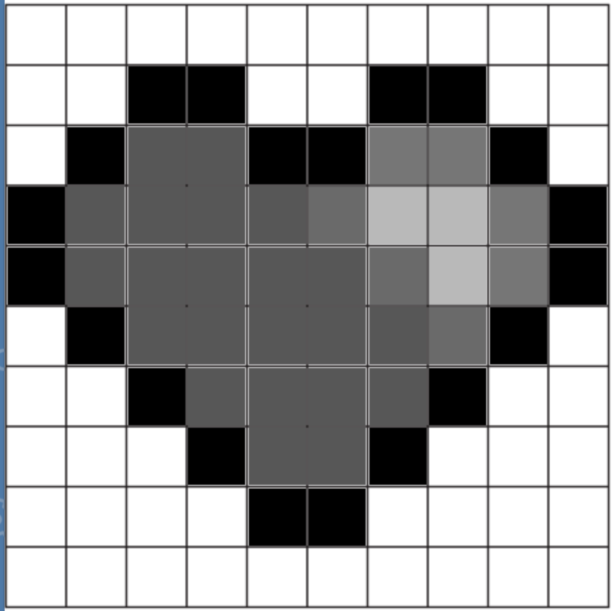


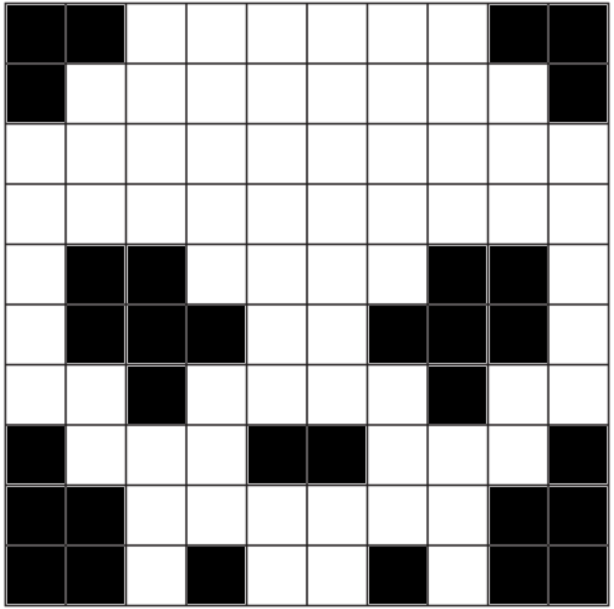
Poll Everywhere

Think

2 min

- **Input**: 10x10x1 image (grayscale image of 10x10 pixels)
- **Convolution**: 5x5 kernel, stride 1
- **MaxPool**: 2x2, stride 2
- What is the size of the resulting image?



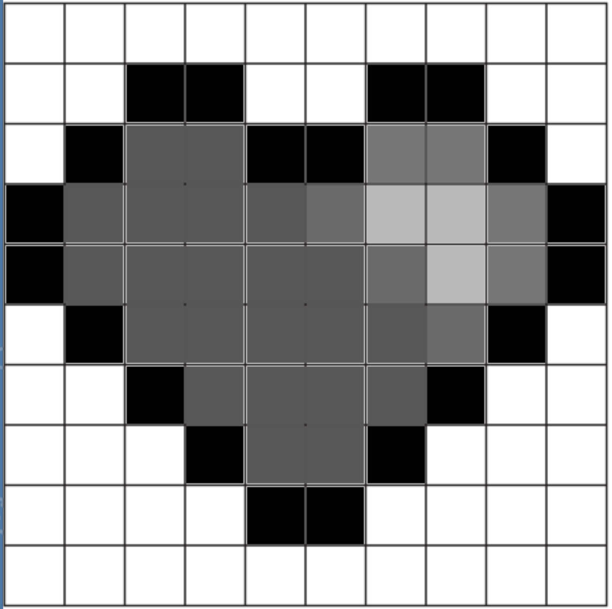


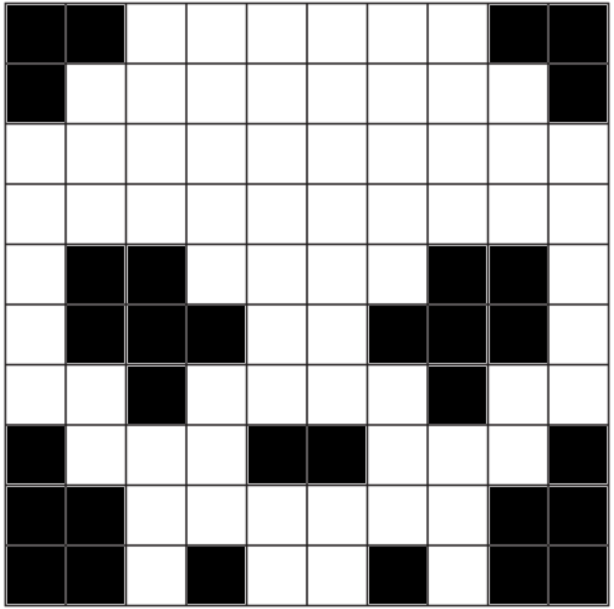
Poll Everywhere

Group

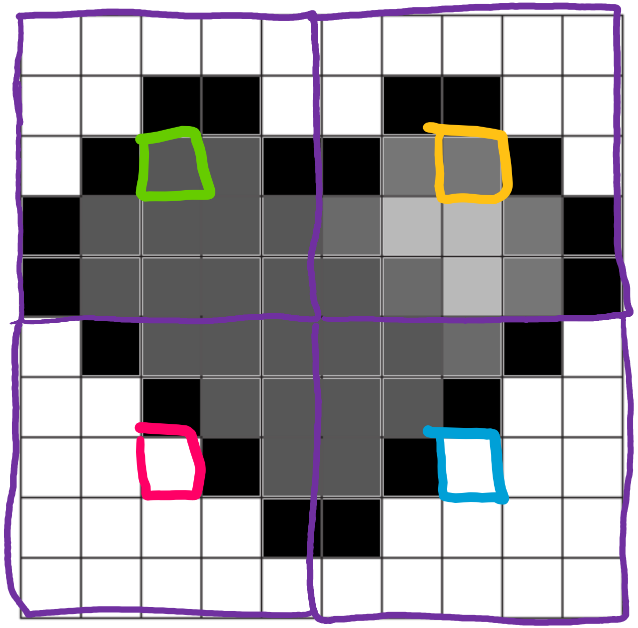
2 min

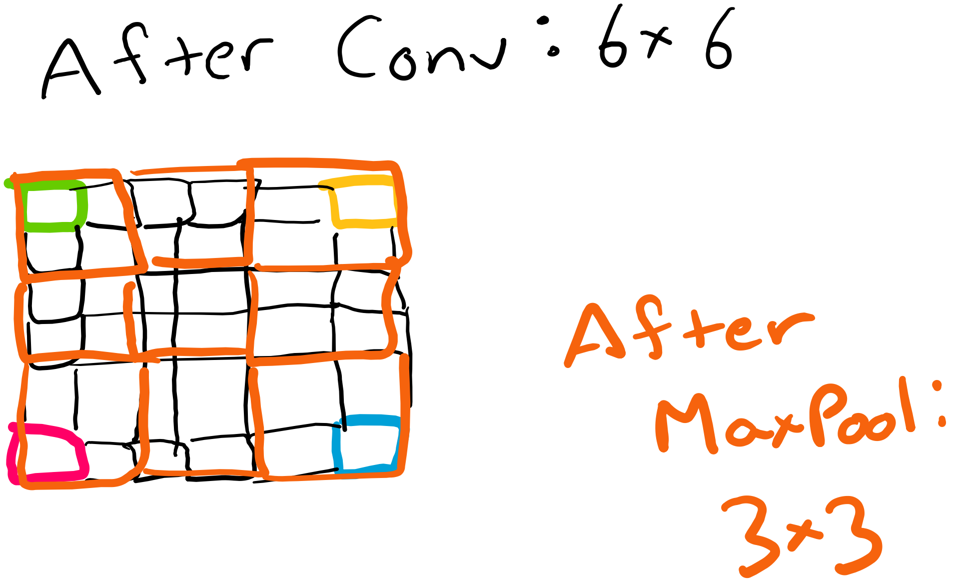
- **Input**: 10x10x1 image (grayscale image of 10x10 pixels)
- **Convolution**: 5x5 kernel, stride 1
- **MaxPool**: 2x2, stride 2
- What is the size of the resulting image?





- **Input**: 10x10x1 image (grayscale image of 10x10 pixels)
- **Convolution**: 5x5 kernel, stride 1
- **MaxPool**: 2x2, stride 2
- What is the size of the resulting image?





Brain Break

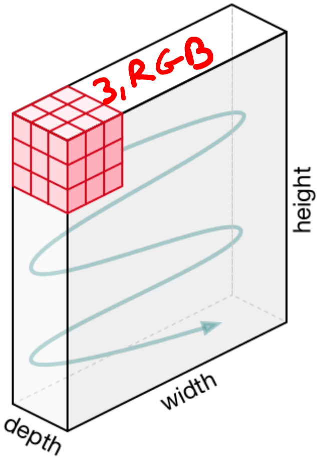
3:28

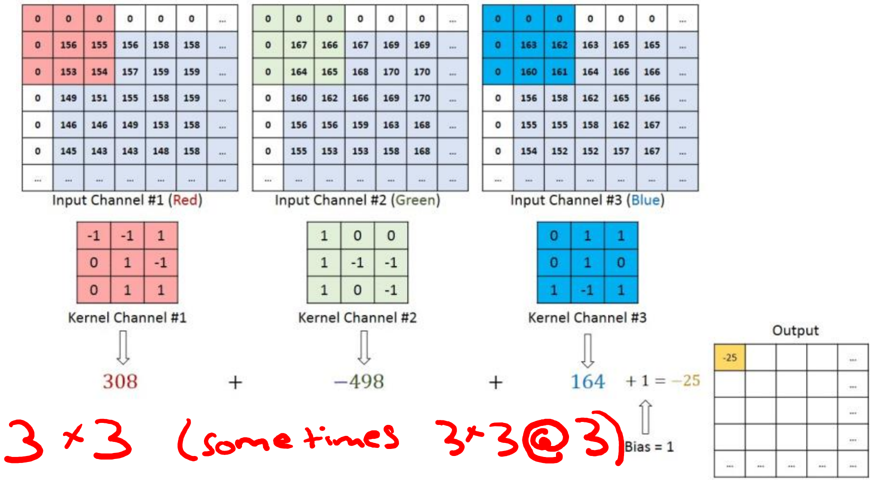
Number of Weights / Parameters

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





Kernel : 3×3×3 (sometimes 3×3@3)

w′×h′×1

weights : 3×3×3 = 27

(with bias) = 3×3×3+1 = 28

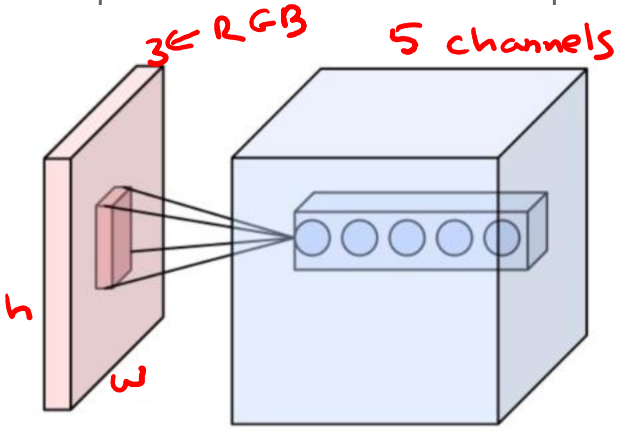
1 kernel: $k_1 * k_2 * 3$ weights
d kernels: $k_1 * k_2 * 3 * d$ weights

CNN with Color Images

Another way of thinking about this process is each kernel is a (hidden-layer) 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.



If: $k_1 = 5$, $k_2 = 5$

params =

$3 \cdot 5 \cdot 5 \cdot 5 =$

375

Binary Classification

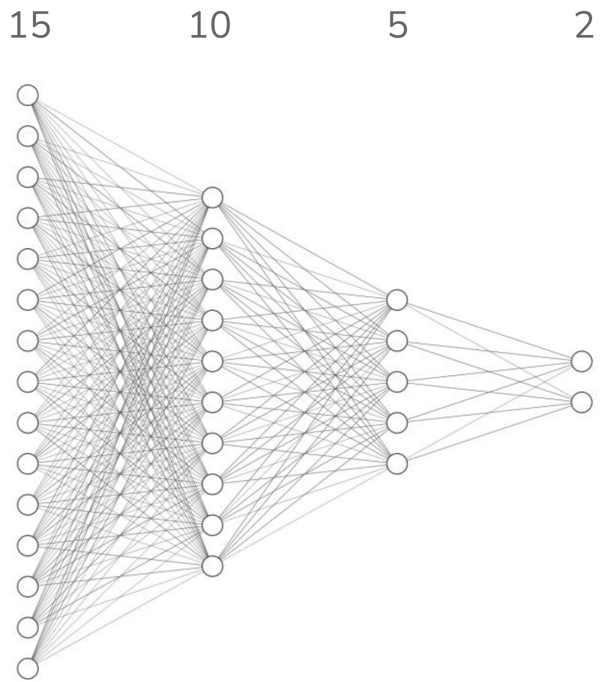
Poll Everywhere

Think

1 min

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

Completely ignore intercept (bias) terms



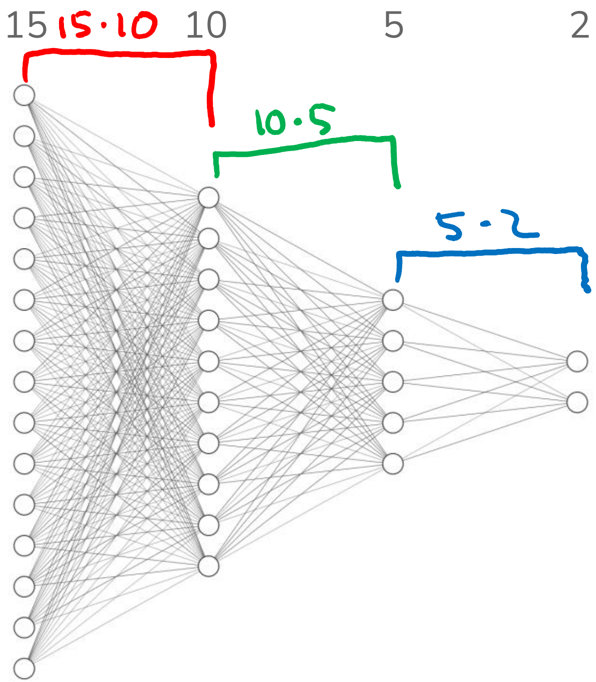
Poll Everywhere

Group

2 min

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

Completely ignore intercept terms



Without Intercept

15·10 + 10·5 + 5·2 =

150 + 50 + 10 =

210

With Intercept

210 + 10 + 5 + 2 =

227

Weight Sharing

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

Without Convolutions:

Num Weights:

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

Total : 250 + 5000 + 27,720 = 32,970 << 66K!

Weight Sharing

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

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

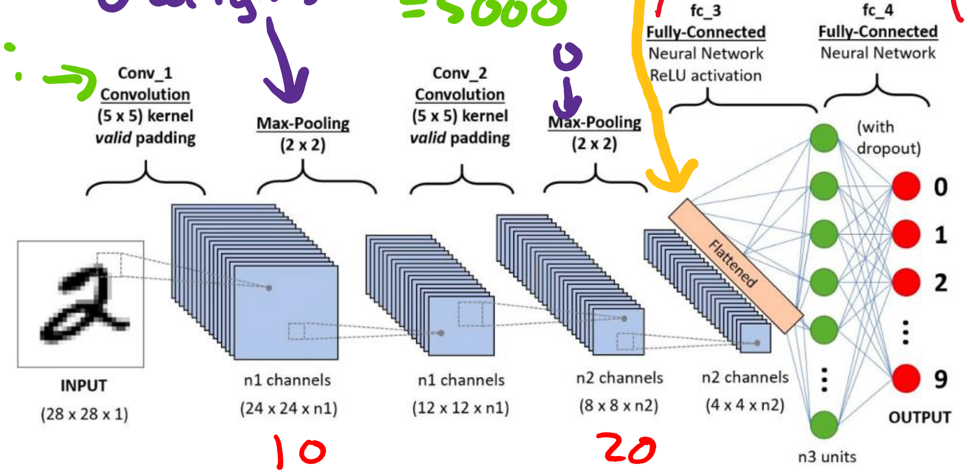
inputs flattened: 4·4·20 = 320

FC component: 320·84 + 84·10 = 27,720

MaxPool: 0 weights

Conv2: 5·5·10·20 = 5000

Conv1: 5·5·10 = 250

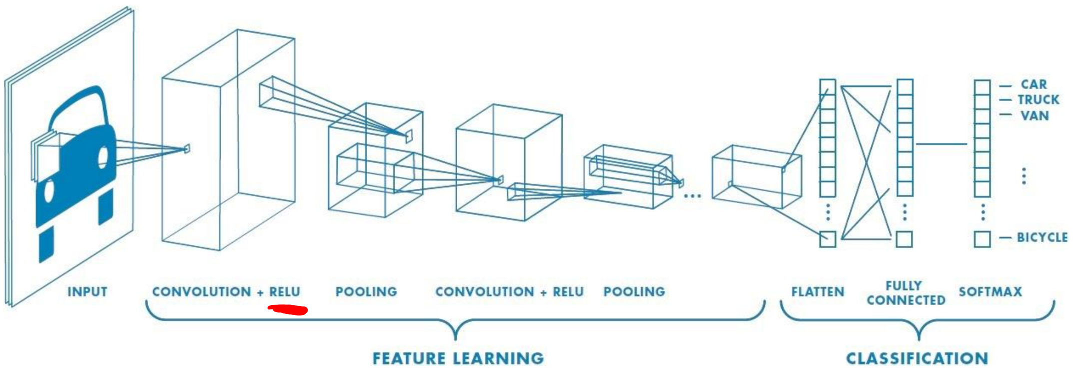


CNN Applications & Transfer Learning

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.
- Each set of operations lowers the size of the image but increases the number of features.
- 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

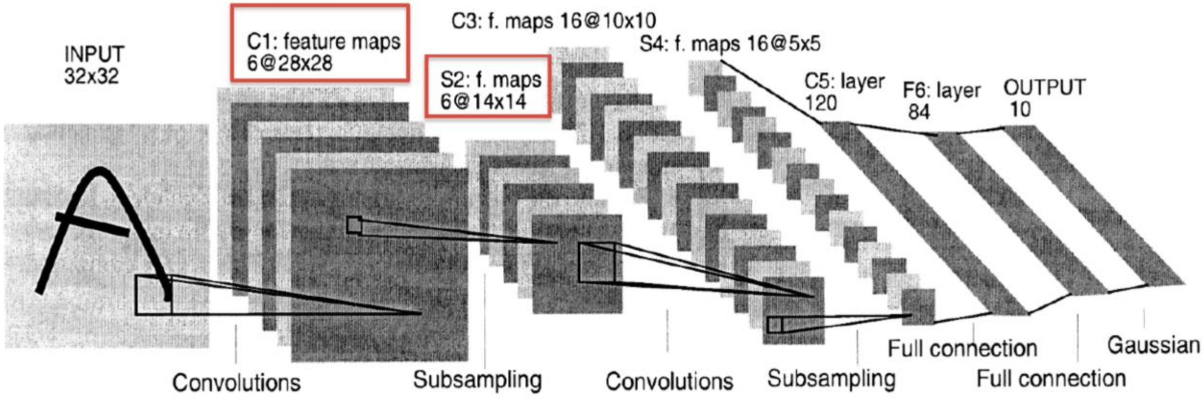
	Layer 1	Layer 2	Layer 3
Example detectors learned			
Example interest points detected			

[Zeiler & Fergus ‘13]

CNN Success

CNNs have had remarkable success in practice

LeNet, 1990s



CNN Success

LeNet made 82 errors on MNIST (popular hand-written digit dataset of size 60K). 99.86% accuracy

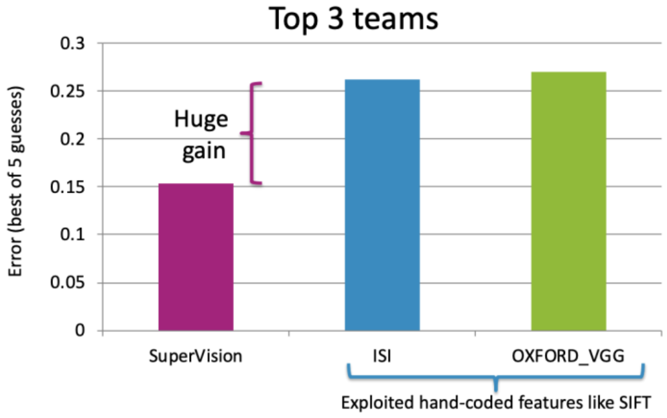
CNN Success

ImageNet 2012 competition:

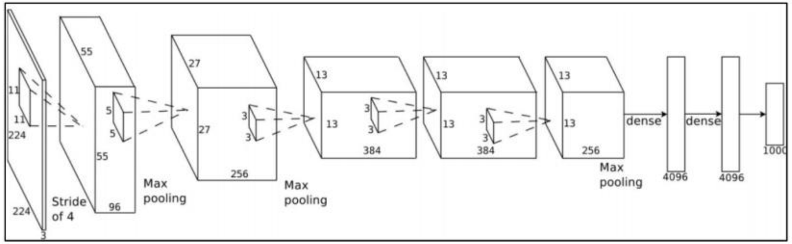
- 1.2M training images
- 1000 categories

Winner: SuperVision

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

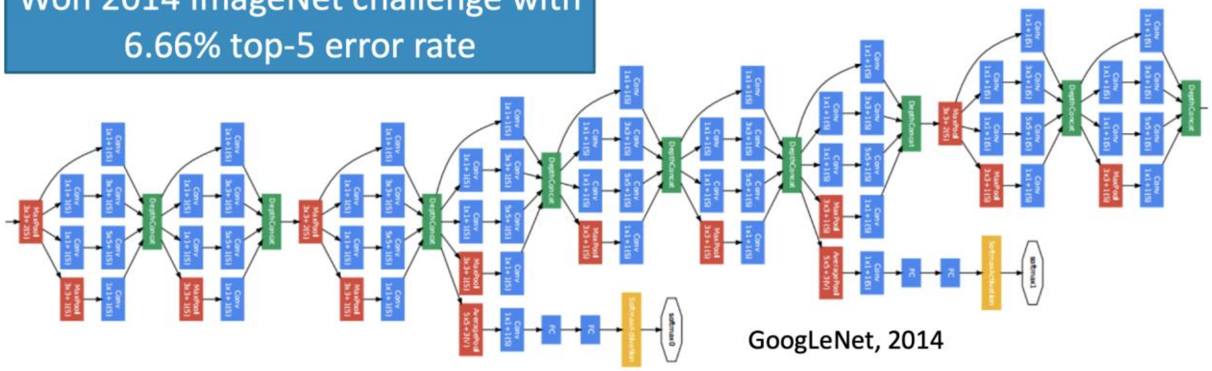


AlexNet



CNN Success

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

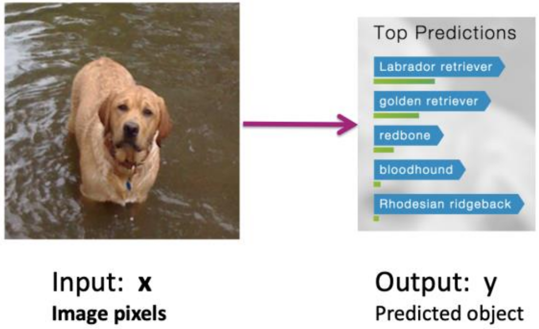


GoogLeNet, 2014

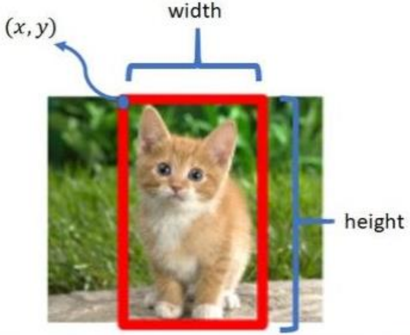
Huge CNN depth has proven helpful in recognition systems... Maybe because images contain hierarchical structure (faces contain eyes contain edges, etc.)

Applications

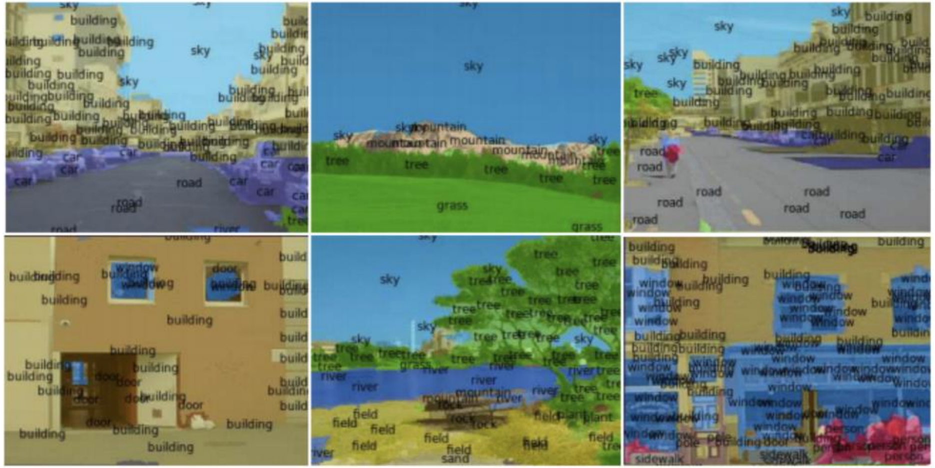
Image Classification



Object Localization

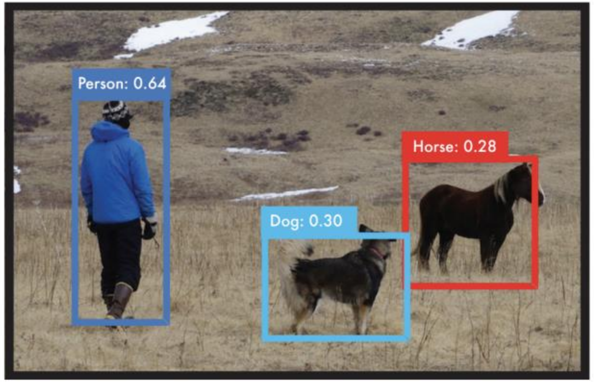


Scene Parsing [Farabet et al. ‘13]



Applications

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



Product Recommendation

Poll Everywhere

Think

1.5 mins

For each of the Computer Vision Tasks below, what do you think the output layer of the neural network would look like? What would each output neuron represent?

- **Image Classification**: Given an image with a single object, output the class of the object.
- **Object Localization**: Given an image with a single object, output the class **and** bounding box (x,y,w,h) of the object.
- **Object Detection**: Given an image with possibly multiple objects, output the bounding box ***and*** class for ***each*** object.

Poll Everywhere

Group

3 min

For each of the Computer Vision Tasks below, what do you think the output layer of the neural network would look like? What would each output neuron represent?

- **Image Classification**: Given an image with a single object, output the class of the object.
- **Object Localization**: Given an image with a single object, output the class **and** bounding box (x,y,w,h) of the object.
- **Object Detection**: Given an image with possibly multiple objects, output the bounding box ***and*** class for ***each*** object.

Image Classification: Given an image with a single object, output the class of the object.

Output Layer:

C neurons,

C is # classes

- each neuron represents probability that the image is of that class

Object Localization: Given an image with a single object, output the class **and** bounding box (x,y,w,h) of the object.

Output Layer:

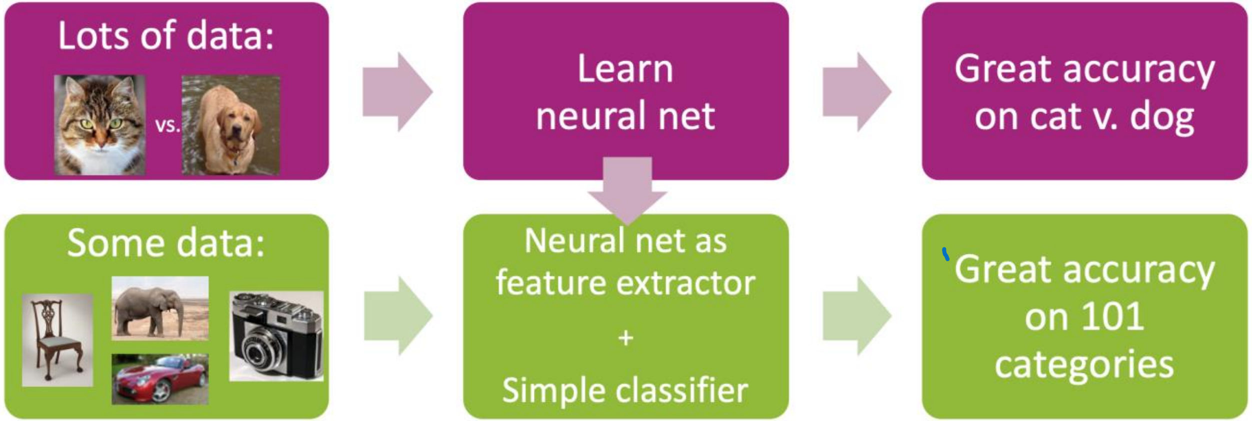
C + 4 neurons,

- first C are same
- Also have w, h, c_x, c_y

Object Detection: Given an image with possibly multiple objects, output the bounding box ***and*** class for ***each*** object.

Search Youtube for the YOLO algorithm explained!

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

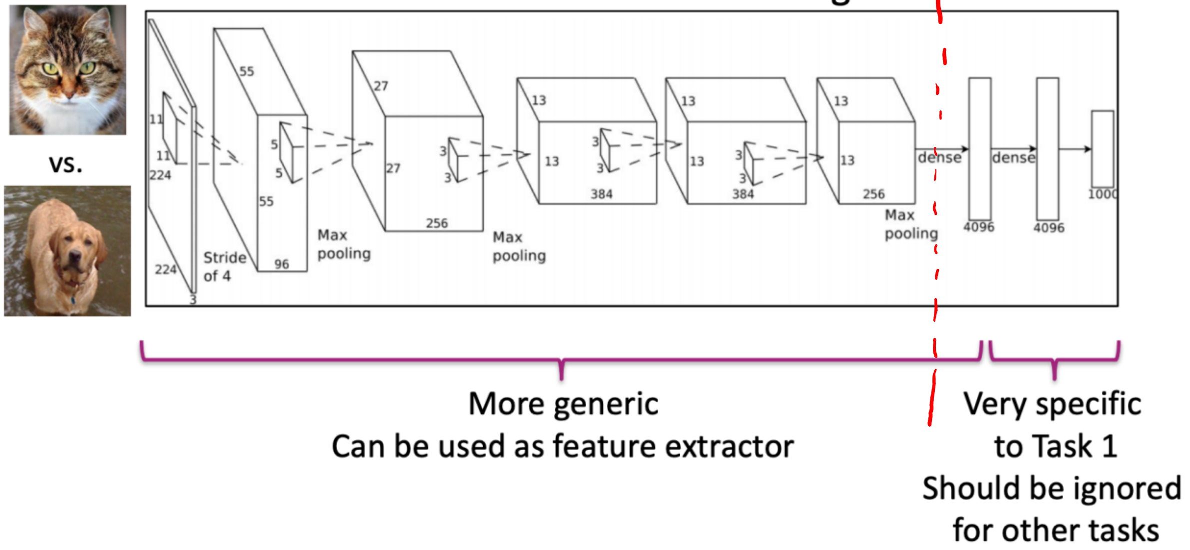
CNNs

What is learned in a neural network?

Initial layers are low-level and very general.

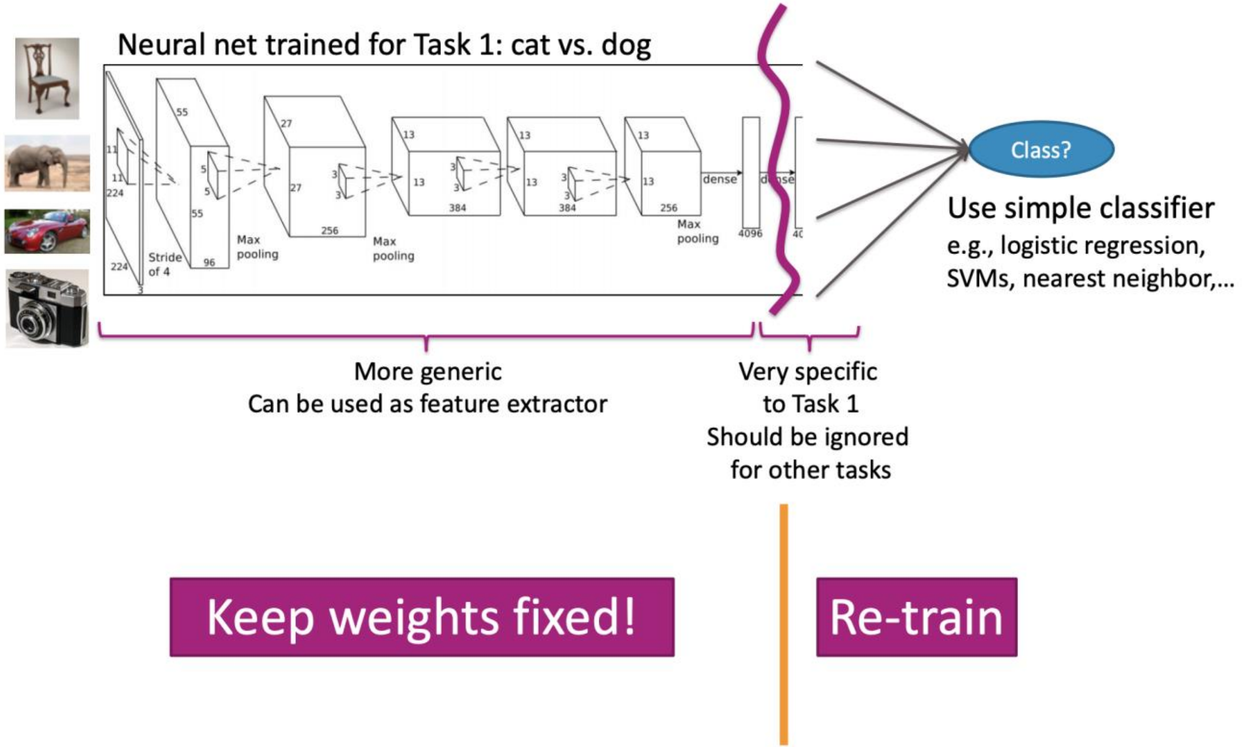
- Usually not sensitive/specific to the task at hand

Neural net trained for Task 1: cat vs. dog



Transfer Learning

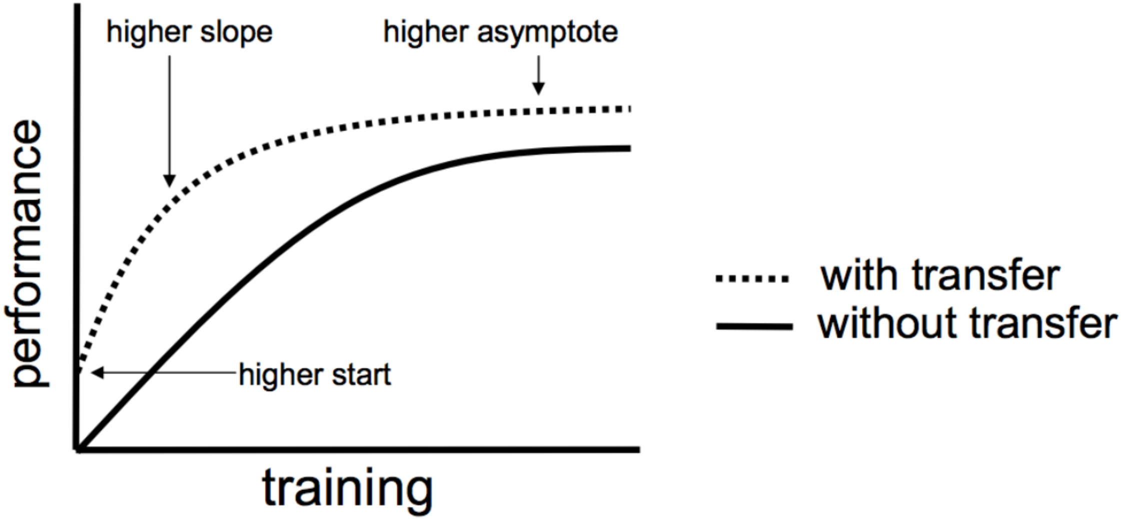
Share the weights for the general part of the network



Transfer Learning

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

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



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)
  - Size of kernels, stride, 0 padding, number of conv layers, depth of outputs of conv layers,
- Learning algorithm

Still not very interpretable

NN Failures

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

"No one adds noise to things in real applications"

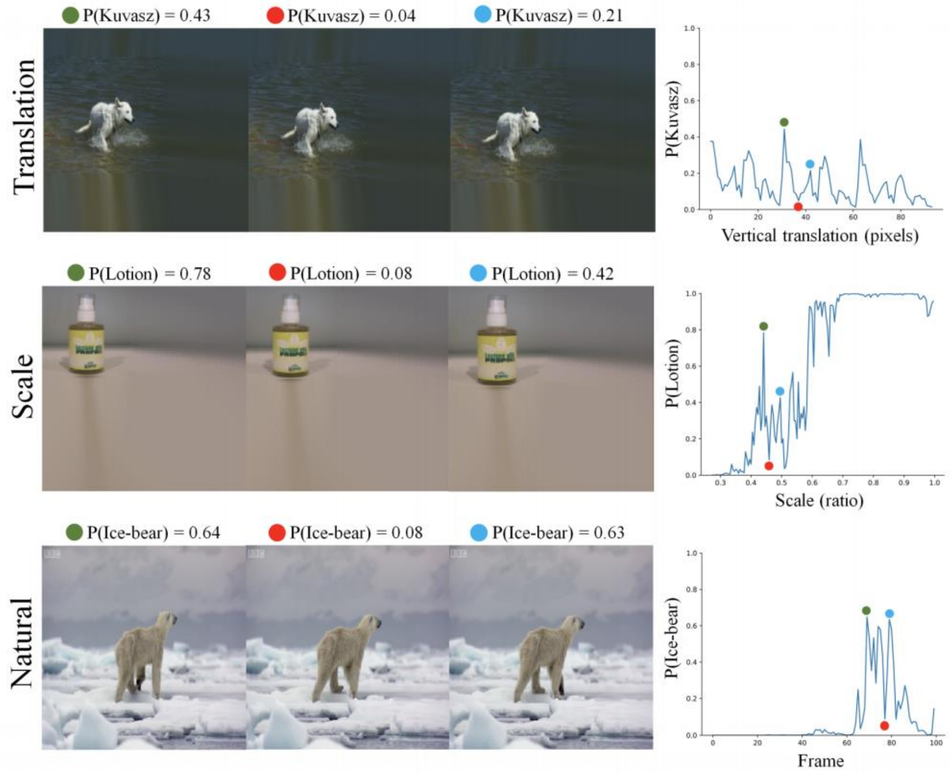
Not true!

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

NN Failures

They even fail with “natural” transformations of images

[Azulay, Weiss https://arxiv.org/abs/1805.12177]

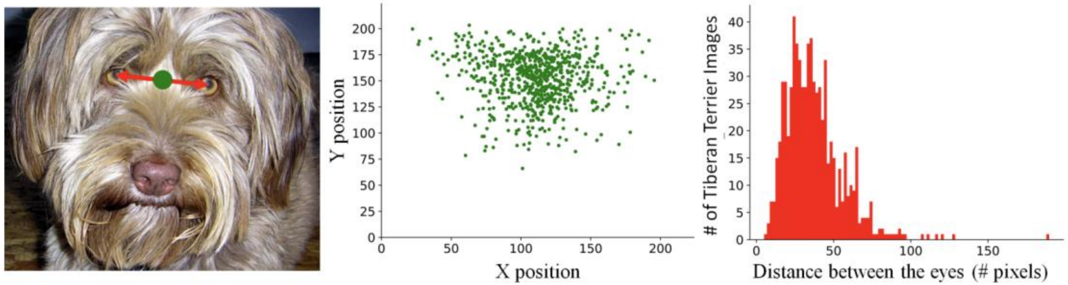


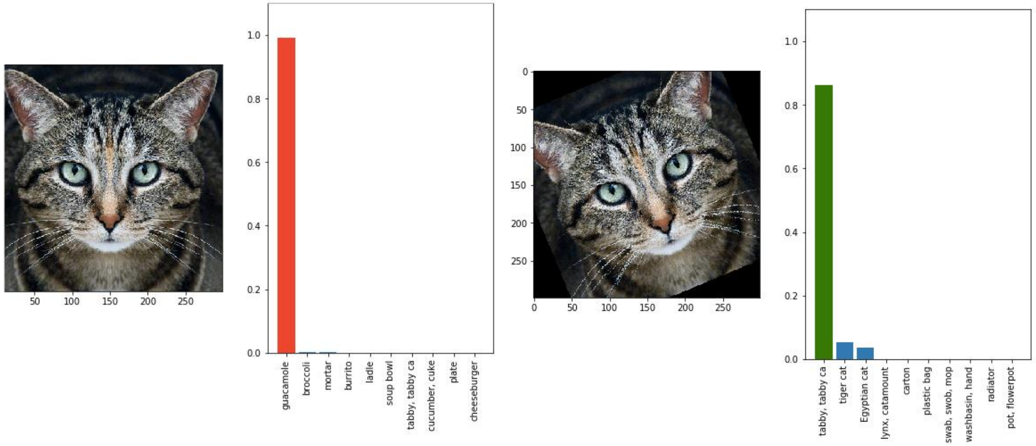
NN Failures

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.

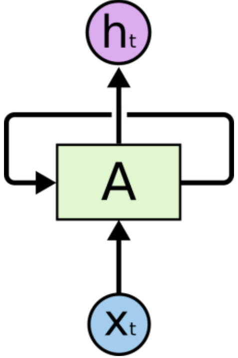
Demo: Adversarial Neural Networks to Promote Fairness

https://godatadriven.com/blog/towards-fairness-in-ml-with-adversarial-networks/

Dataset: Adult UCI

- Predict whether a person's income will be > $50K or ≤ $50K based on factors like:
  - Age
  - Education level
  - Marital status
  - Served in Armed Services?
  - Hours per week worked
  - Occupation sector
  - Etc.

Further Readings on Deep Learning



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

Recap

Theme: Details of convolutional neural networks

Ideas:

- Convolutions
- MaxPool
- Number of Parameters in a (C)NN
- Weight Sharing
- CNN Applications
- Transfer Learning
- NN Failures
- Using NNs to promote algorithmic fairness