Introduction to Deep Learning

Nandita Bhaskhar

Content adapted from CS231n and past CS229 teams
April 29th, 2022

Overview

- Motivation for deep learning
- Areas of Deep Learning
- Convolutional neural networks
- Recurrent neural networks
- Deep learning tools

Classical Approaches Saturate!

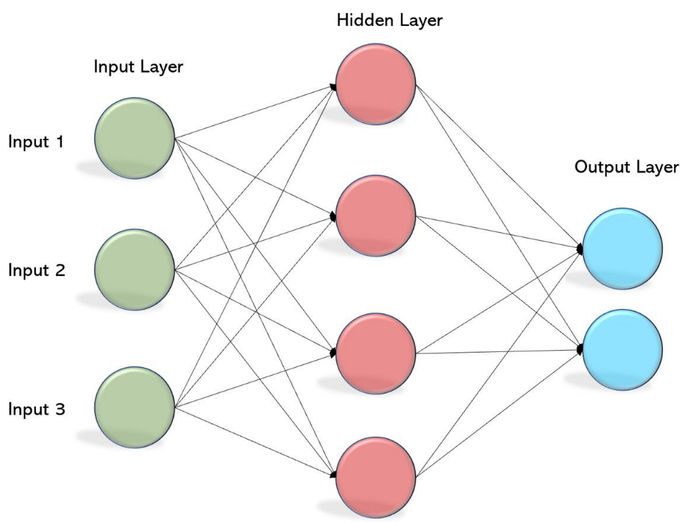
- Computer vision is especially hard for conventional image processing techniques
- Humans are just intrinsically better at perceiving the world!

https://xkcd.com/1425/

What about the MLPs we learnt in class?

Recall:

- Input Layer
- Hidden layer
- Activations
- Outputs

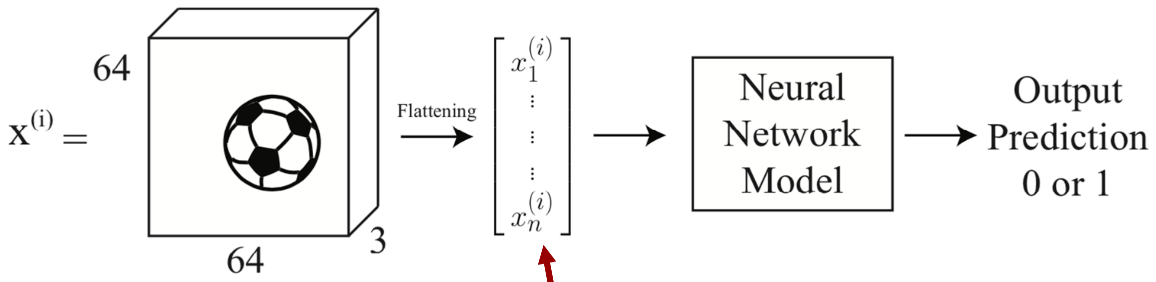


Pic Credit: Becoming Human: Artificial Intelligence Magazine

What about the MLPs we learnt in class?

Expensive to learn. Will not generalize well

Does not exploit the <u>order and local relations</u> in the data!



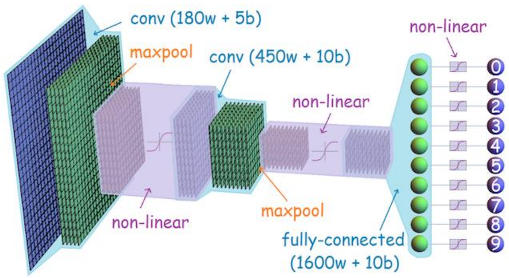
64x64x3=12288 parameters
We also want **many** layers

Overview

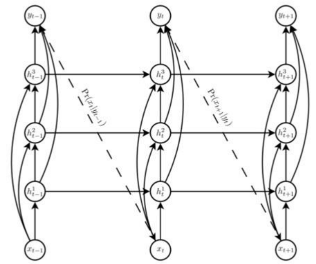
- Motivation for deep learning
- Areas in Deep Learning
- Convolutional neural networks
- Recurrent neural networks
- Deep learning tools

What are different pillars of deep learning?

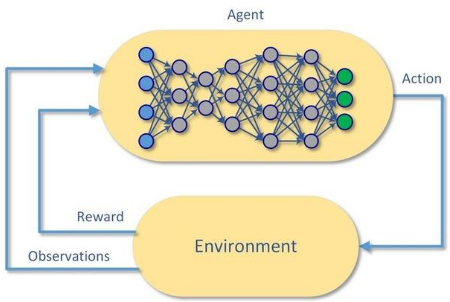
Convolutional NN
Image



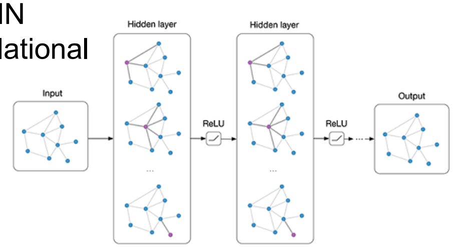
Recurrent NN
Time Series



Deep RL
Control System



Graph NN
Networks/Relational

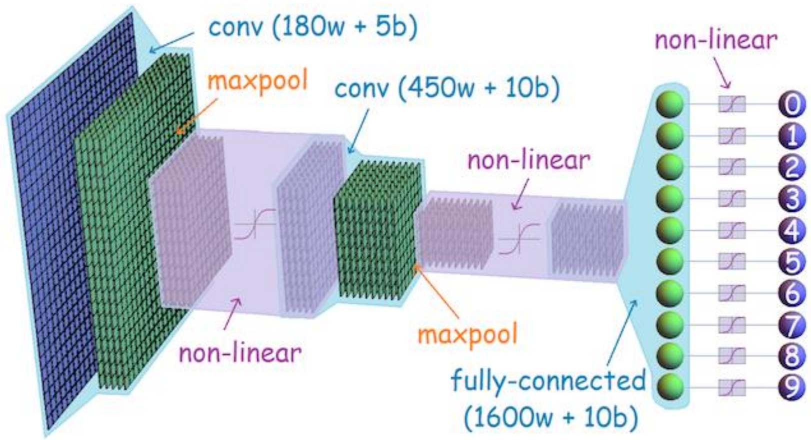


Overview

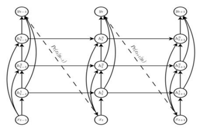
- Motivation for deep learning
- Areas of Deep Learning
- Convolutional neural networks
- Recurrent neural networks
- Deep learning tools

Convolutional Neural Networks

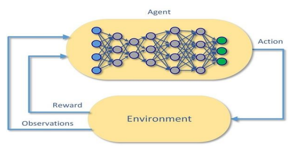
Convolutional Neural Network



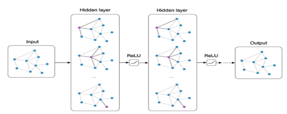
Recurrent NN



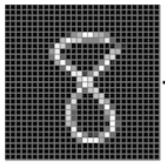
Deep RL



Graph NN

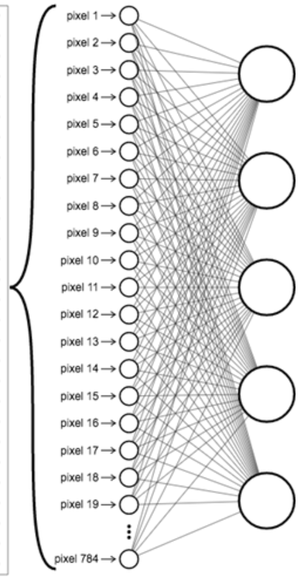


Let us look at images in detail

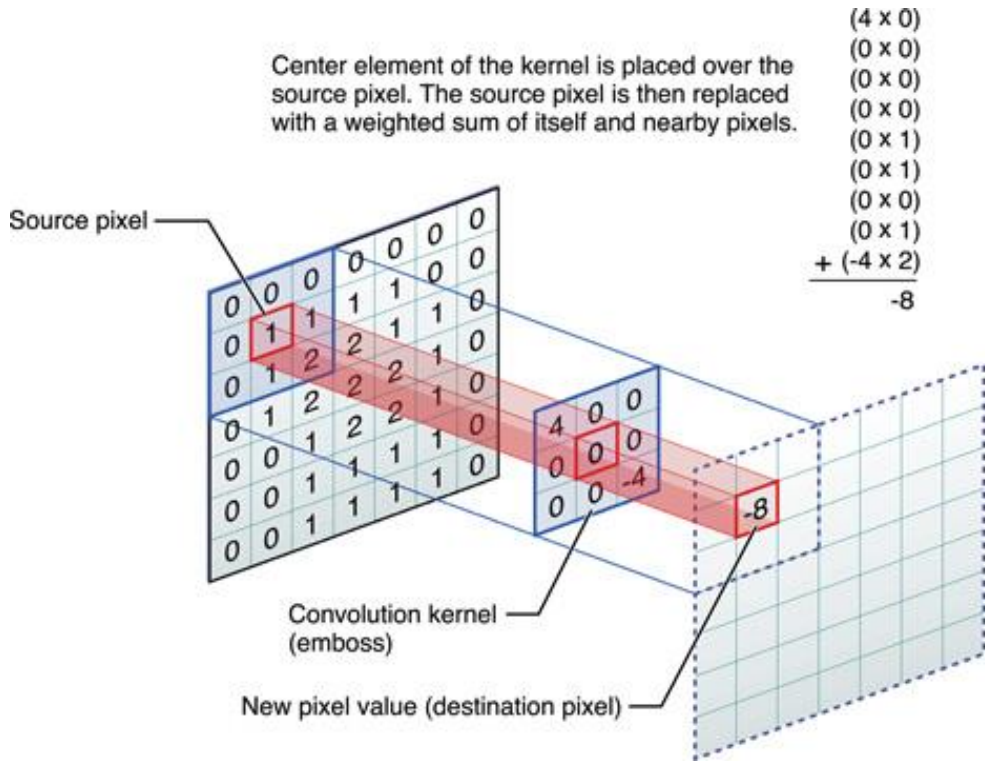


28 x 28
784 pixels

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



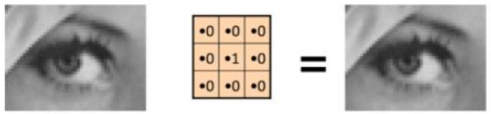
2D Convolution



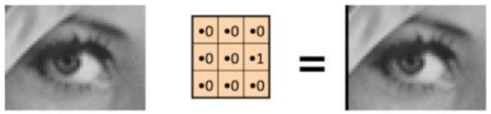
Pic Credit: Apple, Chip Huyen

Convolving Filters

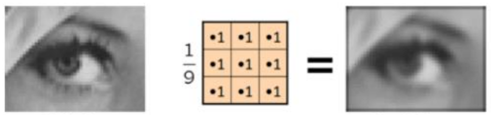
No change:



Shifted right by one pixel:



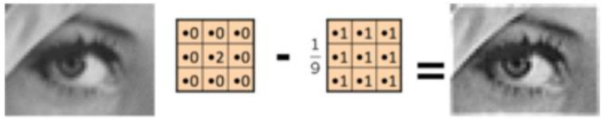
Blurred (you already saw this above):



Note the edge artifact.*

https://ai.stanford.edu/~syyeung/cvweb/tutorials.html

Sharpening



https://ai.stanford.edu/~syyeung/cvweb/tutorials.html

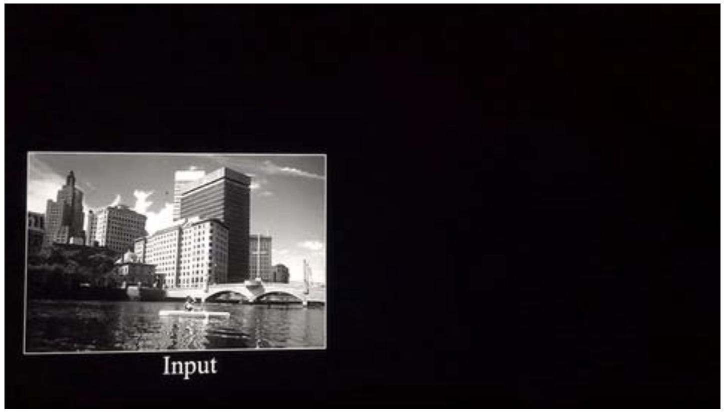
Edge Detection: Laplacian Filters

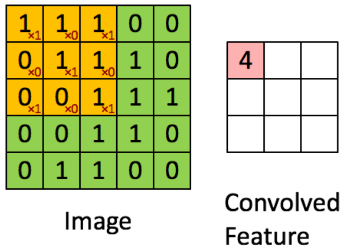
0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Convolving Filters

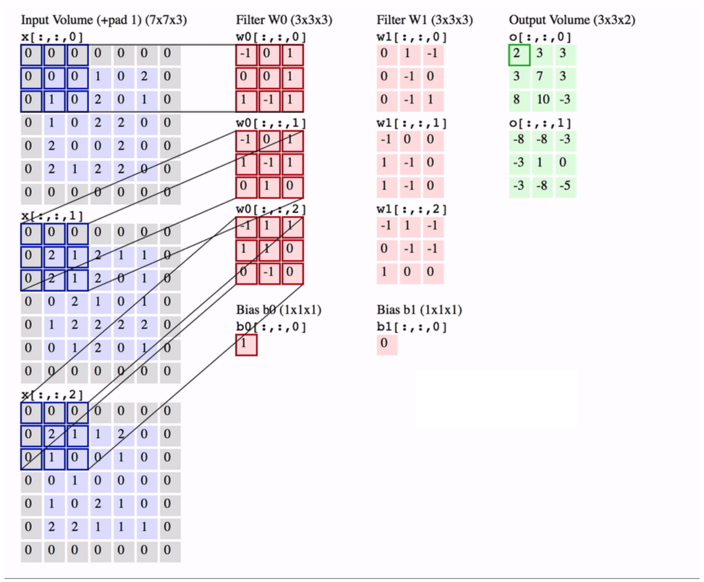
- Why not extract features using filters?
- Better, why not let the data dictate what filters to use?
- Learnable filters!!





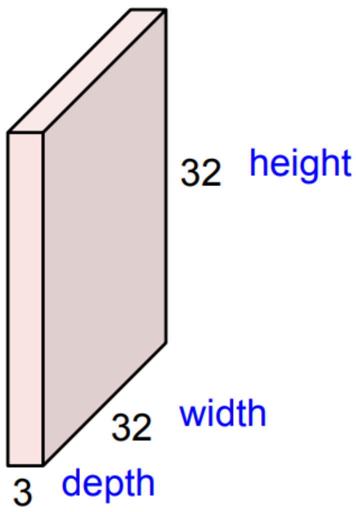
Convolution on multiple channels

- Images are generally RGB !!
- How would a filter work on a image with RGB channels?
- The filter should also have 3 channels.
- Now the output has a channel for every filter we have used.



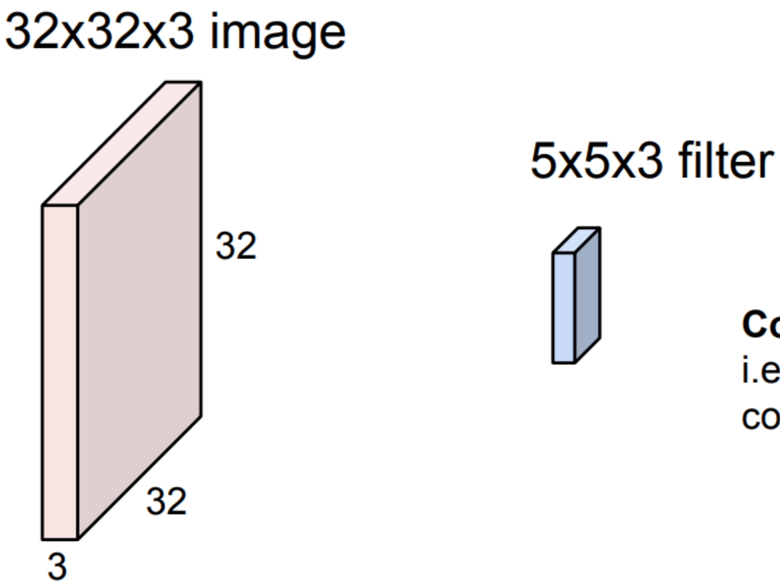
Convolution Layer

32x32x3 image -> preserve spatial structure



Slide Credit: CS231n

Convolution Layer

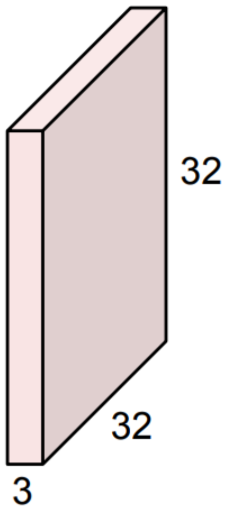


Convolve the filter with the image
i.e. "slide over the image spatially, computing dot products"

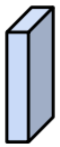
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

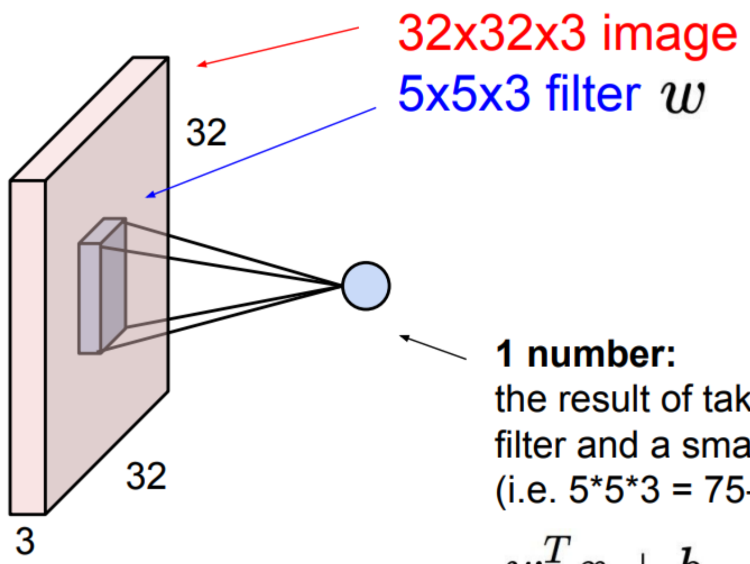


5x5x3 filter



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolution Layer

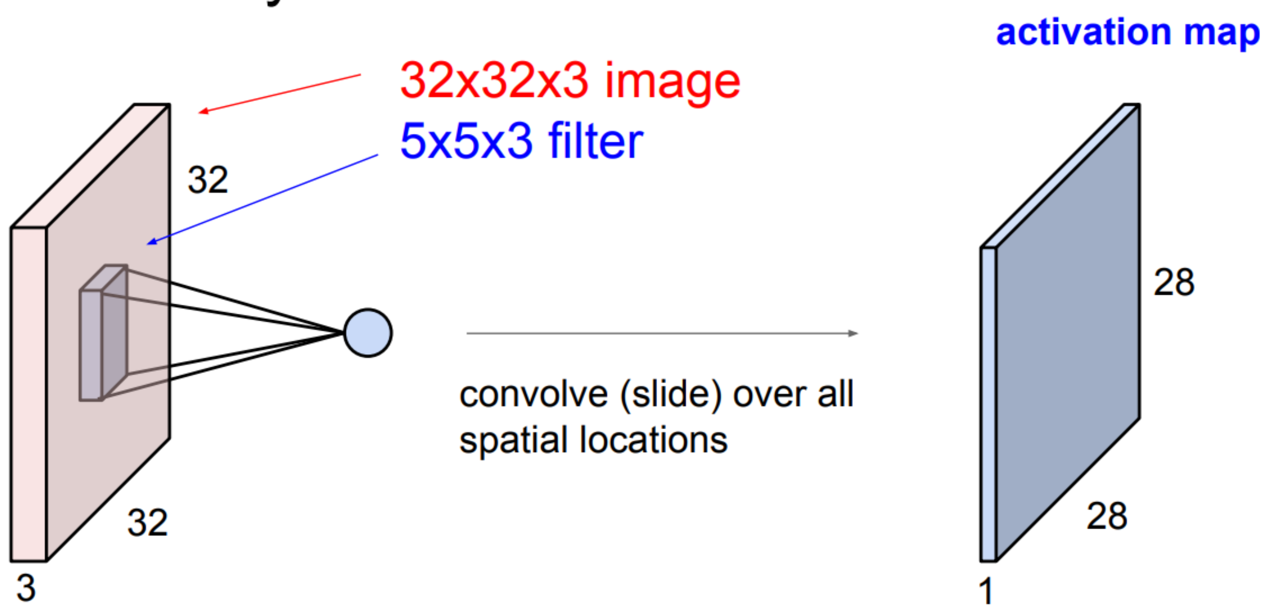


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

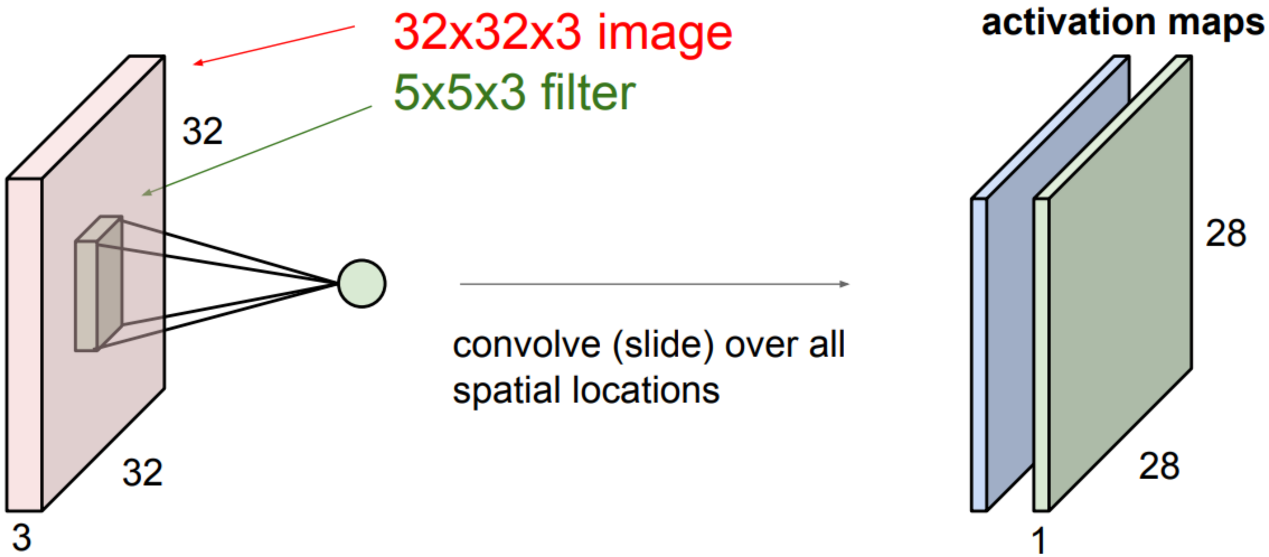
Slide Credit: CS231n

Convolution Layer

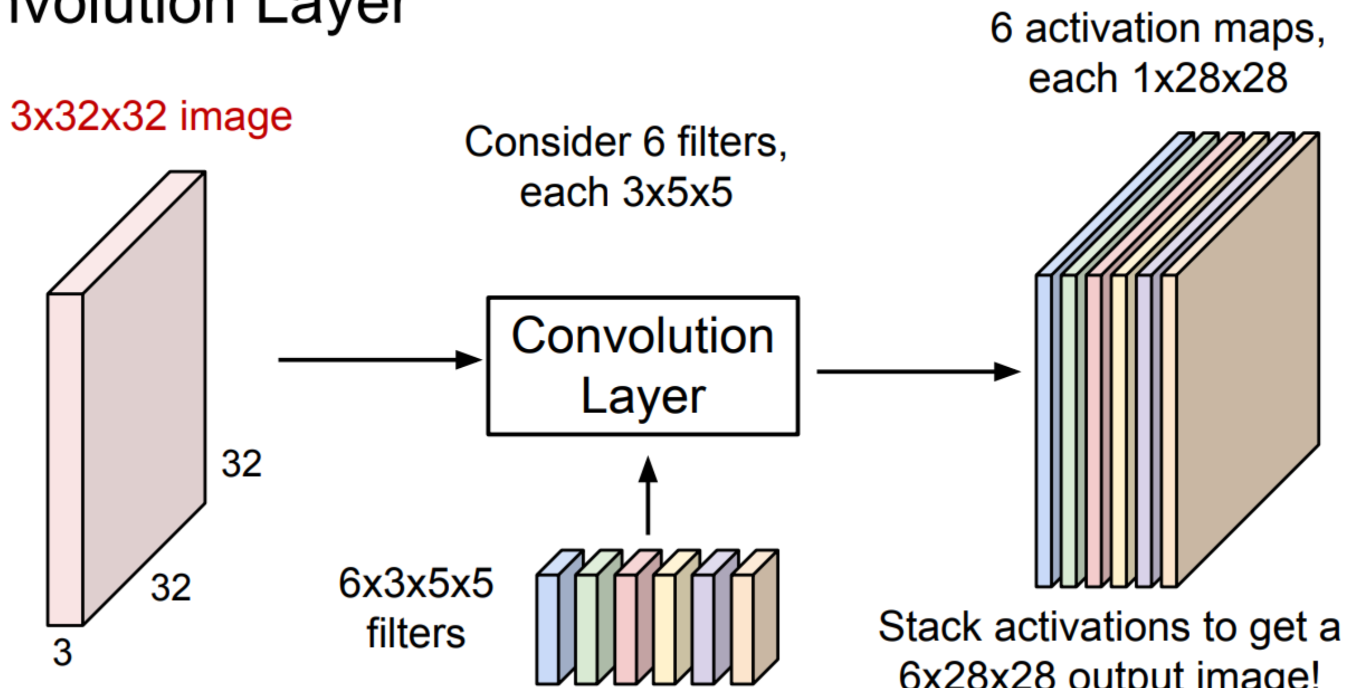


Convolution Layer

consider a second, green filter



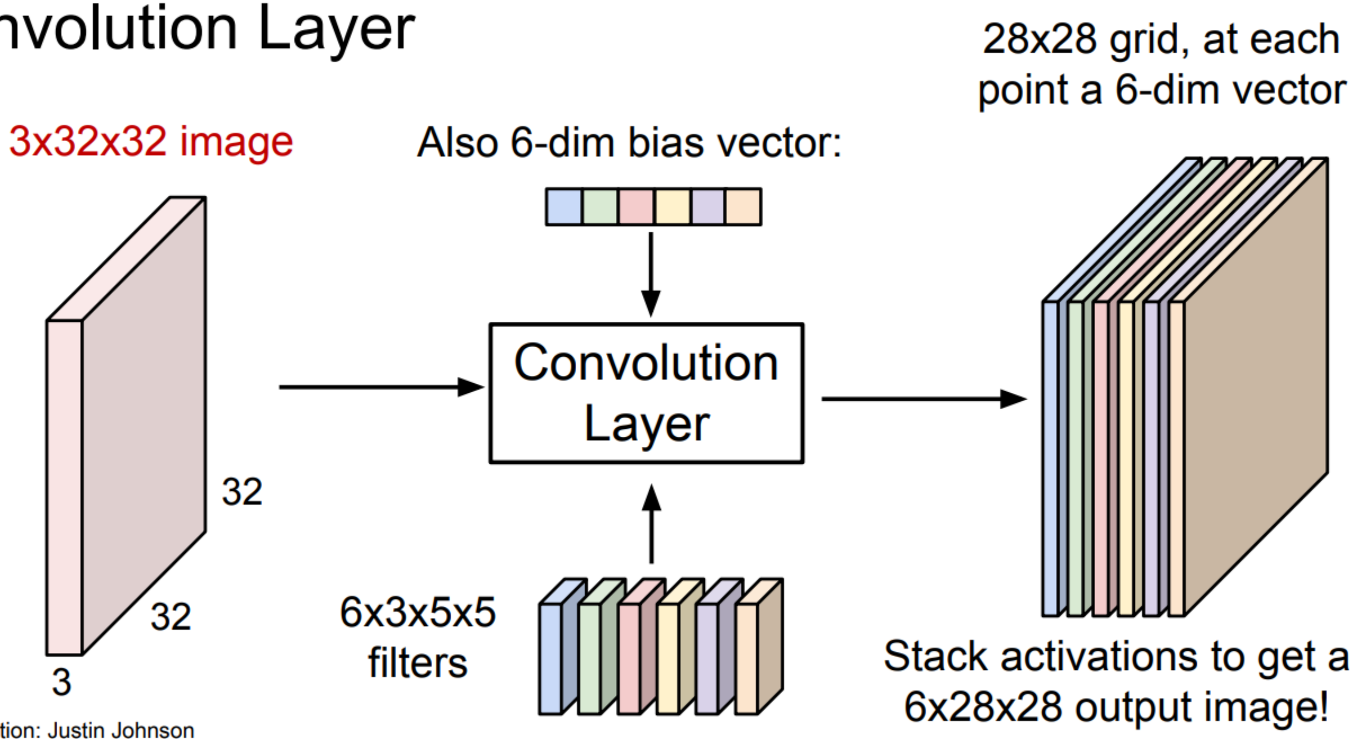
Convolution Layer



Slide inspiration: Justin Johnson

Slide Credit: CS231n

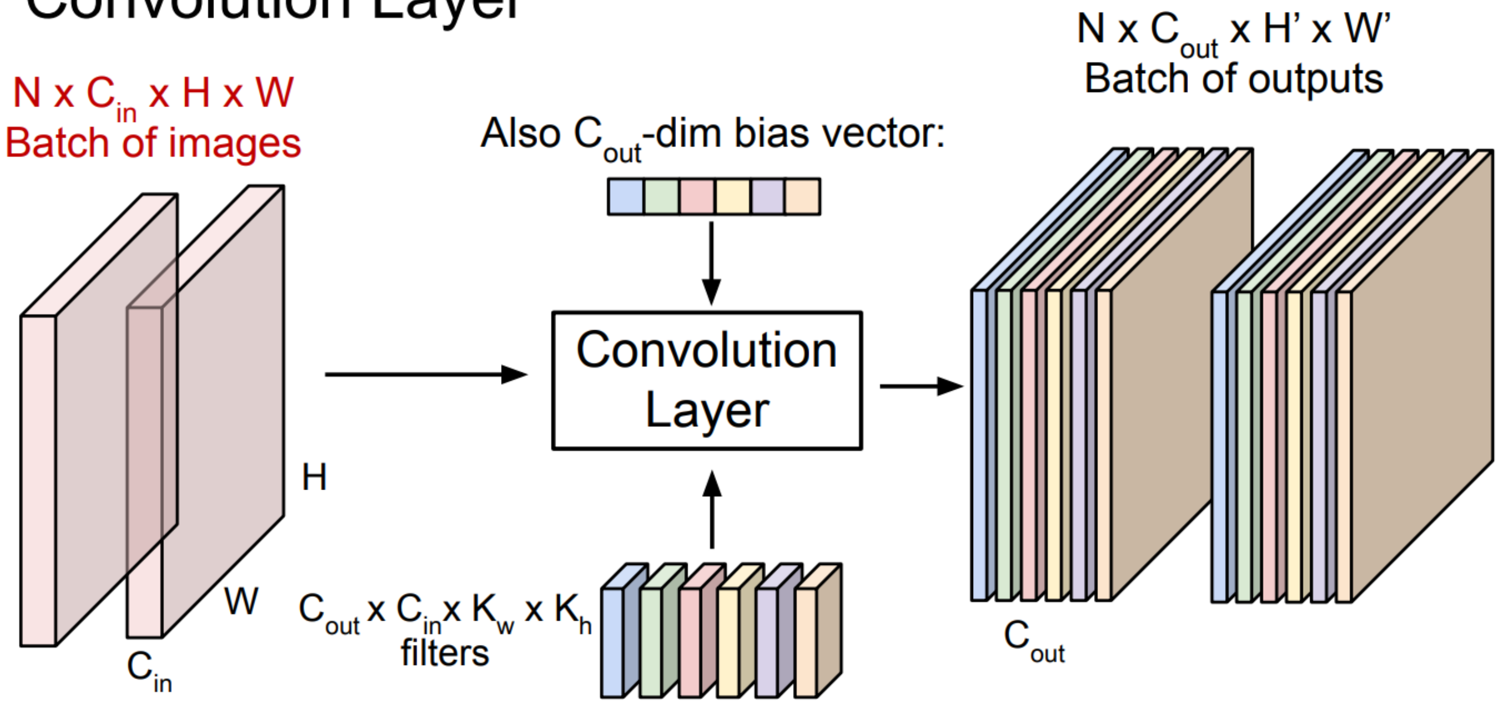
Convolution Layer



Slide inspiration: Justin Johnson

Slide Credit: CS231n

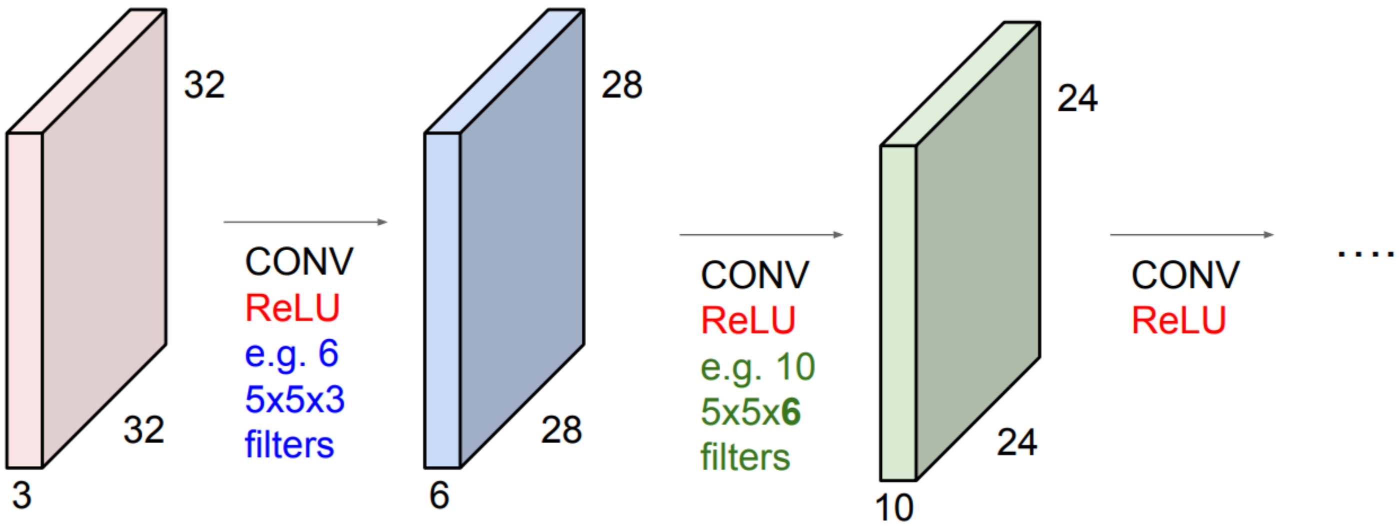
Convolution Layer



Slide inspiration: Justin Johnson

Slide Credit: CS231n

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Slide Credit: CS231n

Convolution layer: summary

Let's assume input is $W_1 \times H_1 \times C$
Conv layer needs 4 hyperparameters:

- Number of filters **K**
- The filter size **F**
- The stride **S**
- The zero padding **P**

This will produce an output of $W_2 \times H_2 \times K$
where:

- $W_2 = (W_1 - F + 2P)/S + 1$
- $H_2 = (H_1 - F + 2P)/S + 1$

Number of parameters: F^2CK and K biases

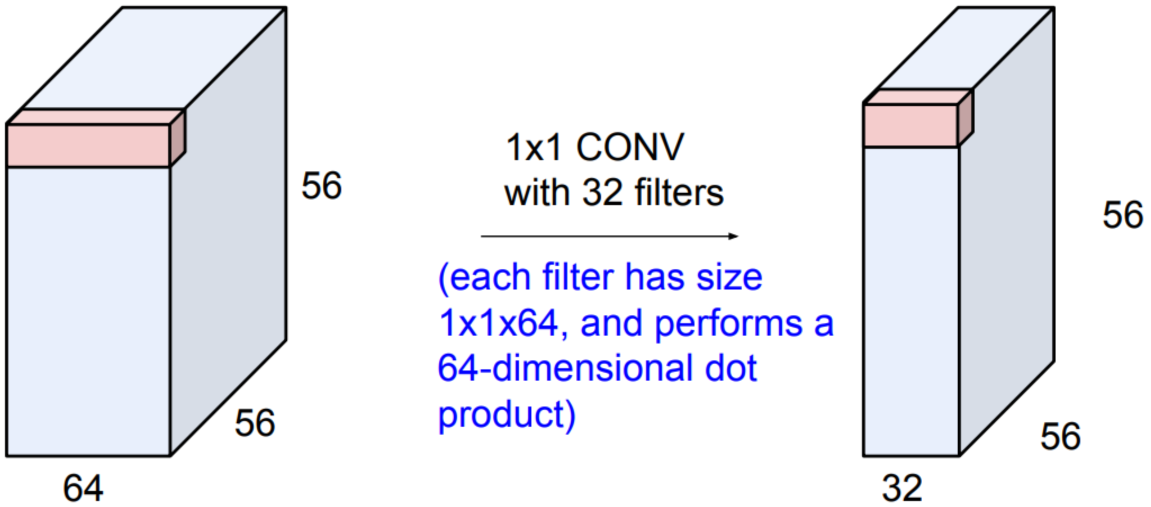
Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)

- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

Slide Credit: CS231n

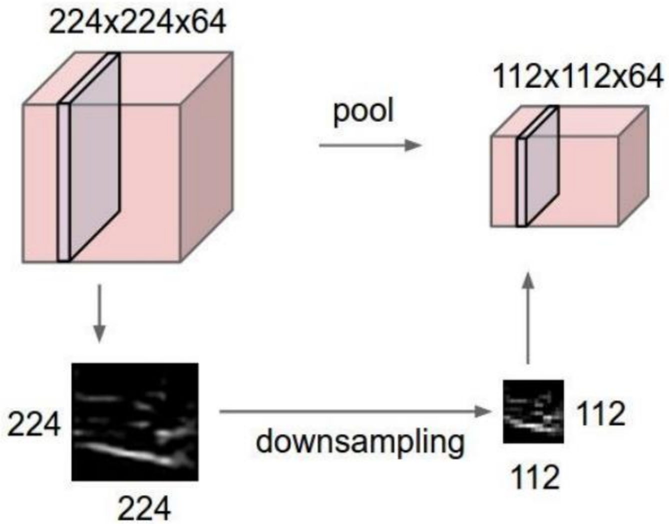
(btw, 1x1 convolution layers make perfect sense)



Slide Credit: CS231n

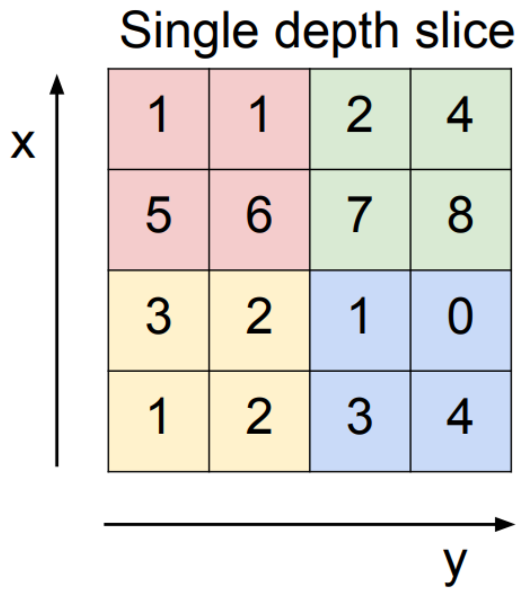
Pooling layer

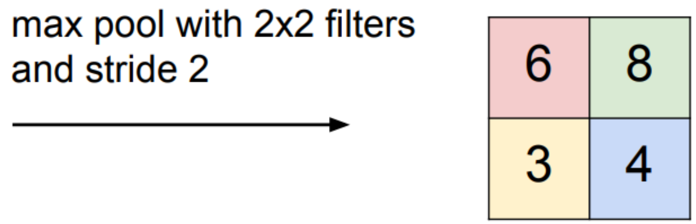
- makes the representations smaller and more manageable
- operates over each activation map independently



Slide Credit: CS231n

MAX POOLING





- No learnable parameters
- Introduces spatial invariance

Slide Credit: CS231n

Pooling layer: summary

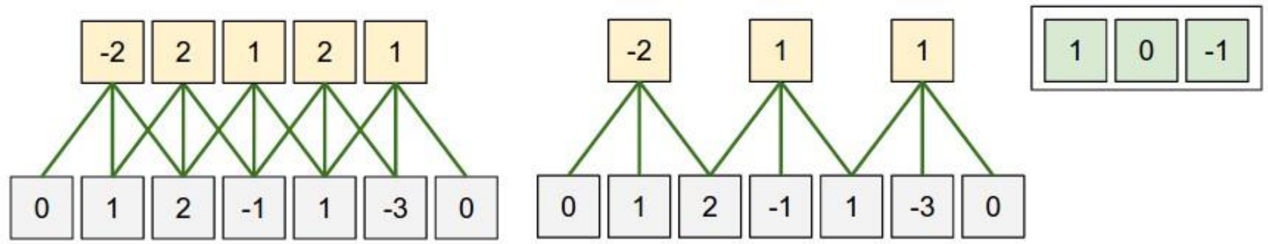
Let's assume input is W_1 x H_1 x C
Conv layer needs 2 hyperparameters:
- The spatial extent **F**
- The stride **S**

This will produce an output of W_2 x H_2 x C where:
- $W_2 = (W_1 - F)/S + 1$
- $H_2 = (H_1 - F)/S + 1$

Number of parameters: 0

Slide Credit: CS231n

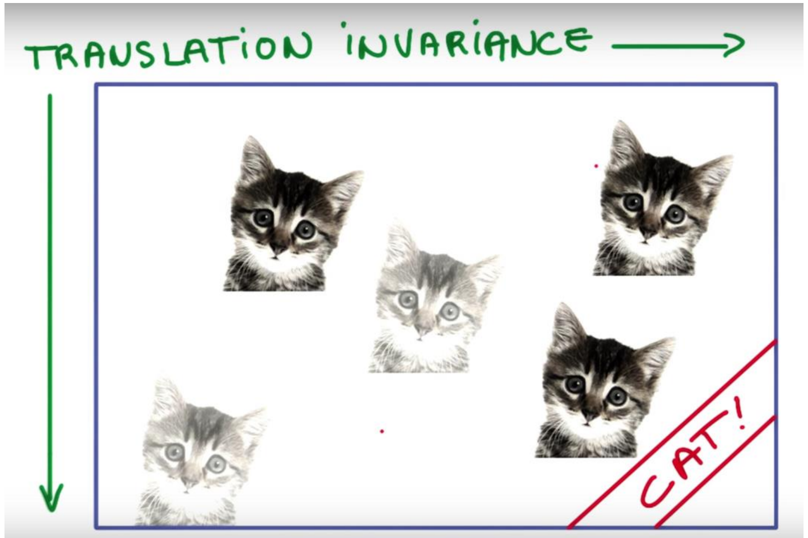
Parameter Sharing



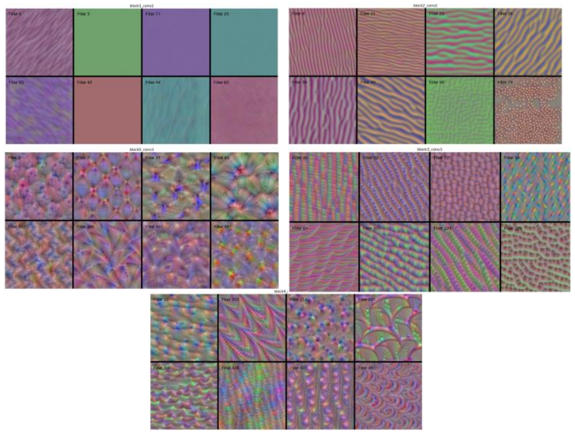
Lesser the parameters less computationally intensive the training. This is a win win as we are reusing parameters.

Translational invariance

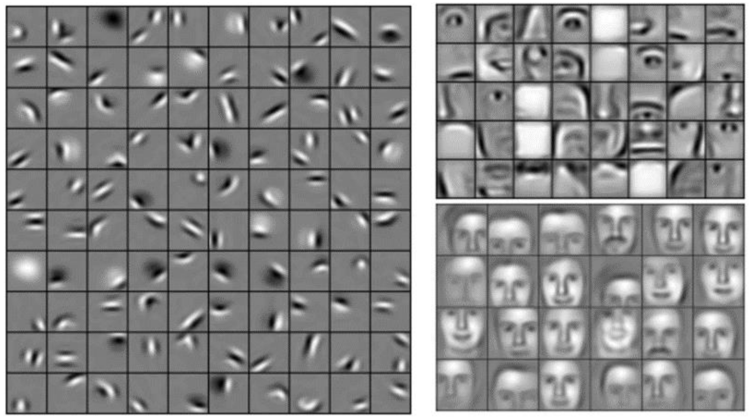
Since we are training filters to detect cats and the moving these filters over the data, a differently positioned cat will also get detected by the same set of filters.



Filteres? Layers of filters?

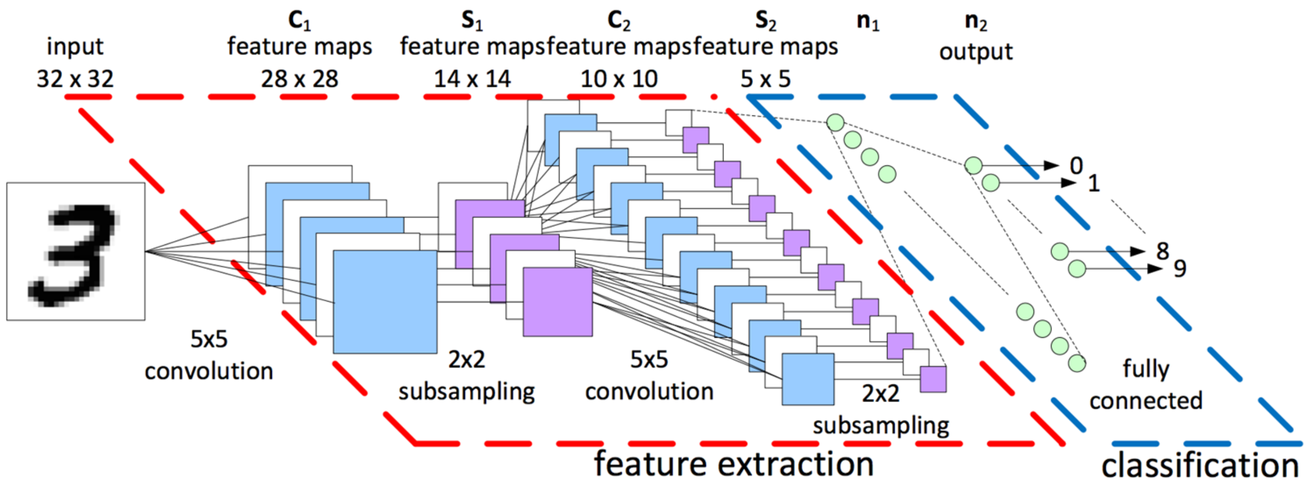


Images that maximize filter outputs at certain layers. We observe that the images get more complex as filters are situated deeper



How deeper layers can learn deeper embeddings. How an eye is made up of multiple curves and a face is made up of two eyes.

How do we use convolutions?



Let convolutions extract features!

Image credit: LeCun et al. (1998)

Fun Fact: Convolution really is just a linear operation

- In fact convolution is a giant matrix multiplication.
- We can expand the 2 dimensional image into a vector and the conv operation into a matrix.

$$\begin{pmatrix} x1 & x2 & x3 \\ x4 & x5 & x6 \\ x7 & x8 & x9 \end{pmatrix} * \begin{pmatrix} k1 & k2 \\ k3 & k4 \end{pmatrix}$$

$$\begin{pmatrix} k1 & k2 & 0 & k3 & k4 & 0 & 0 & 0 & 0 \\ 0 & k1 & k2 & 0 & k3 & k4 & 0 & 0 & 0 \\ 0 & 0 & 0 & k1 & k2 & 0 & k3 & k4 & 0 \\ 0 & 0 & 0 & 0 & k1 & k2 & 0 & k3 & k4 \end{pmatrix} . \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \\ x9 \end{pmatrix}$$

$$\begin{pmatrix} k1\,x1 + k2\,x2 + k3\,x4 + k4\,x5 \\ k1\,x2 + k2\,x3 + k3\,x5 + k4\,x6 \\ k1\,x4 + k2\,x5 + k3\,x7 + k4\,x8 \\ k1\,x5 + k2\,x6 + k3\,x8 + k4\,x9 \end{pmatrix}$$

How do we learn?

We now have a network with:

- a bunch of weights
- a loss function

To learn:

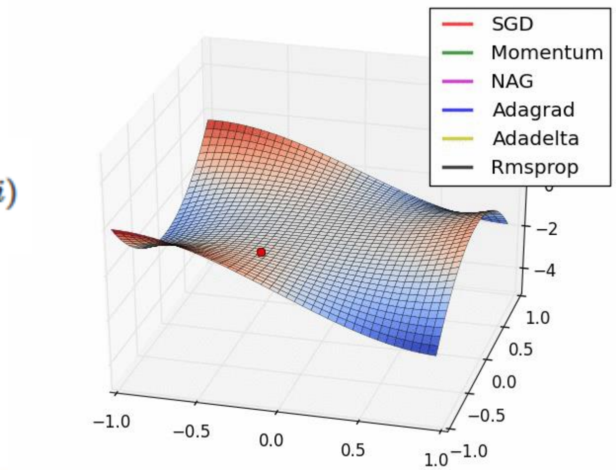
- Just do gradient descent and backpropagate the error derivates

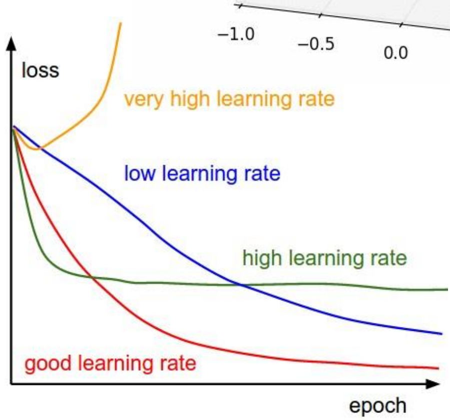
How do we learn?

Instead of $\theta := \theta + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x^{(i)}$

There are “optimizers”

- Momentum: Gradient + Momentum
- Nestrov: Momentum + Gradients
- Adagrad: Normalize with sum of sq
- RMSprop: Normalize with moving avg of sum of squares
- ADAM: RMSprop + momentum





Mini-batch Gradient Descent

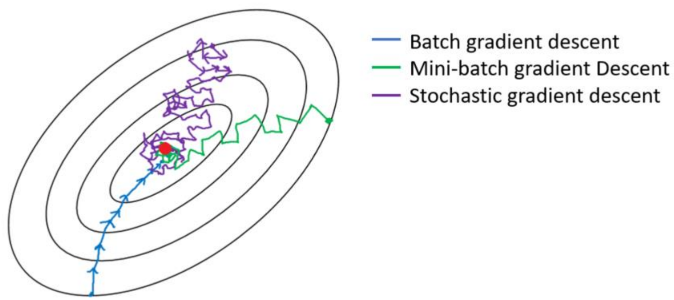
Expensive to compute gradient for large dataset

Memory size

Compute time

Mini-batch: takes a sample of training data

How to we sample intelligently?

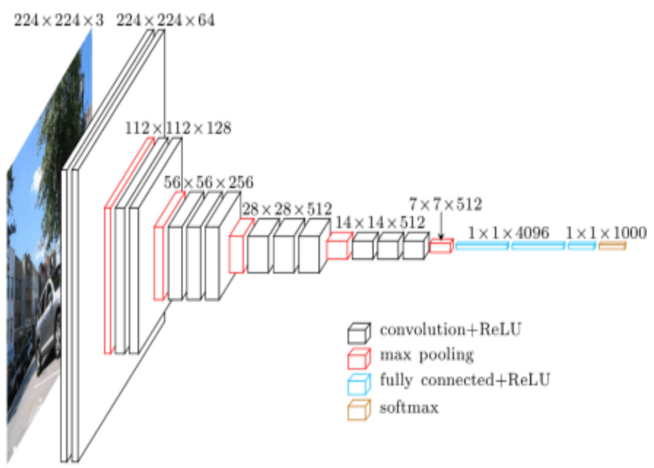


Is deeper better?

Deeper networks seem to be more powerful but harder to train.

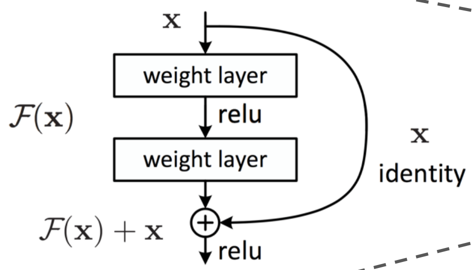
- Loss of information during forward propagation
- Loss of gradient info during back propagation

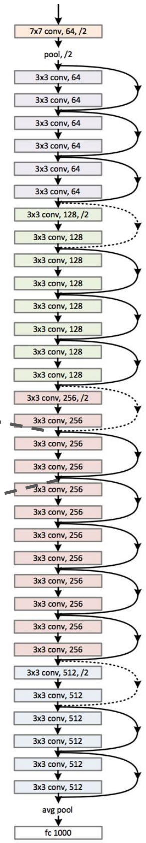
There are many ways to “keep the gradient going”



Solution

Connect the layers, create a gradient highway or information highway.





ResNet (2015)

Image credit: He et al. (2015)

Initialization

- Can we initialize all neurons to zero?
- If all the weights are same we will not be able to break symmetry of the network and all filters will end up learning the same thing.
- Large numbers, might knock relu units out.
- Relu units once knocked out and their output is zero, their gradient flow also becomes zero.
- We need small random numbers at initialization.
- Variance : 1/sqrt(n)
- Mean: 0

Popular initialization setups

(Xavier, He) (Uniform, Normal)

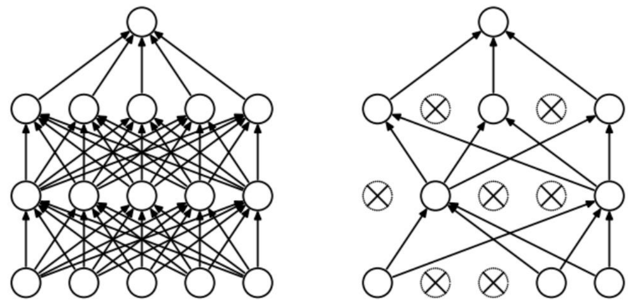
Dropout

- What does cutting off some network connections do?

- Trains multiple smaller networks in an ensemble.

- Can drop entire layer too!

- Acts like a really good regularizer



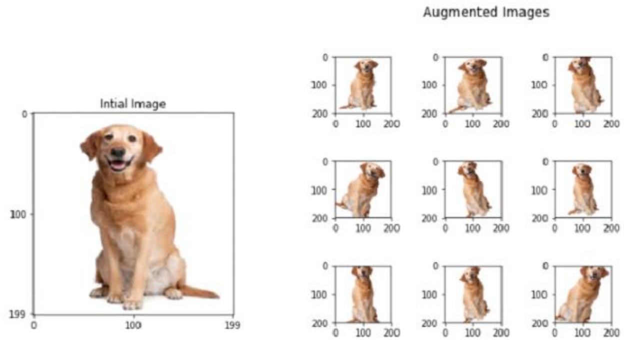
(a) Standard Neural Net (b) After applying dropout.

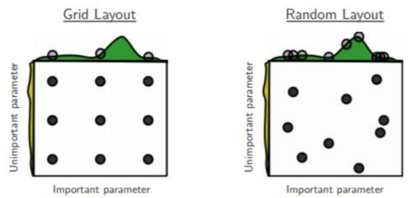
Tricks for training

- Data augmentation if your data set is smaller. This helps the network generalize more.

- Early stopping if training loss goes above validation loss.

- Random hyperparameter search or grid search?



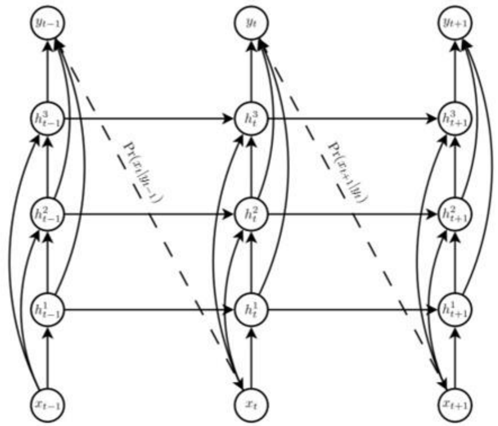


Overview

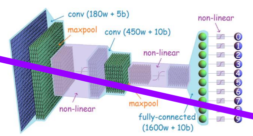
- Motivation for deep learning
- Areas of Deep Learning
- Convolutional neural networks
- Recurrent neural networks
- Deep learning tools

CNN sounds like fun!
What are some deep learning pillars?

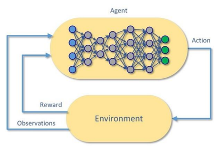
Recurrent NN
Time Series



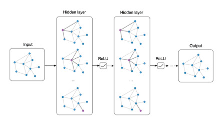
~~Convolutional NN~~



Deep RL

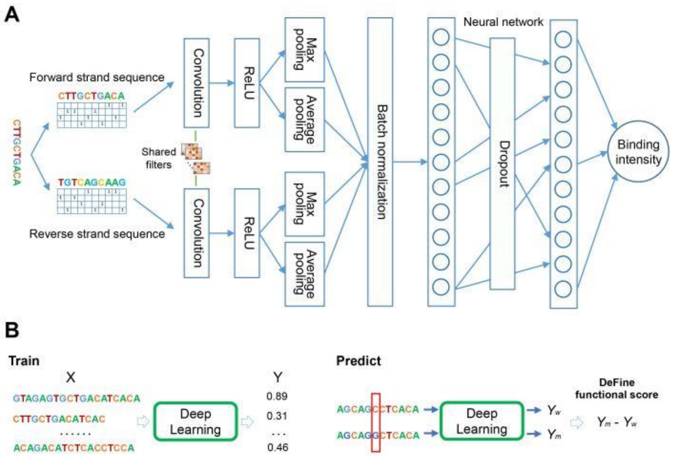


Graph NN



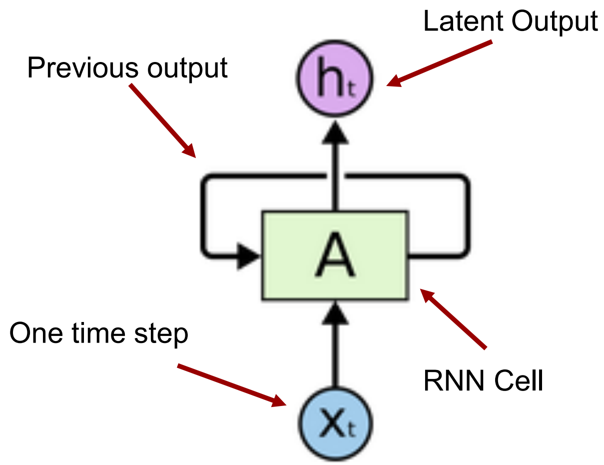
We can also have 1D architectures (remember this)

- CNN works on any data where there is a local pattern
- We use 1D convolutions on DNA sequences, text sequences and music notes
- But what if time series has **causal dependency** or any kind of **sequential dependency**?

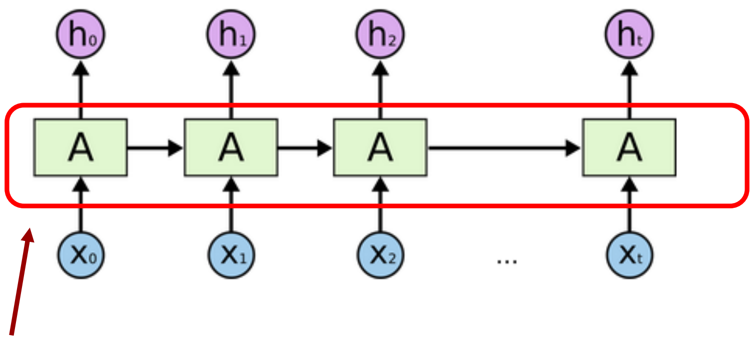


To address sequential dependency?

Use recurrent neural network (RNN)

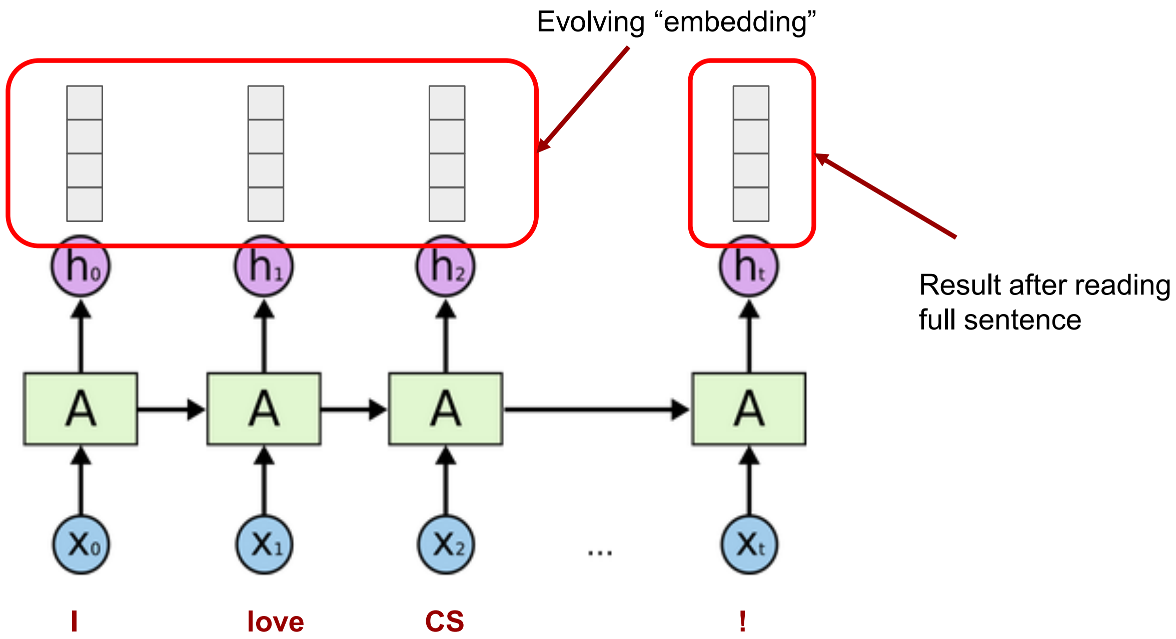


Unrolling an RNN

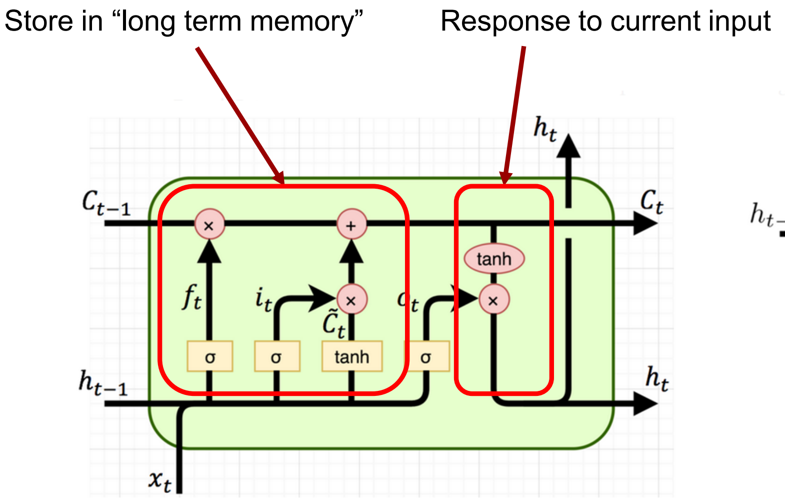


They are really the same cell,
NOT many different cells like kernels of CNN

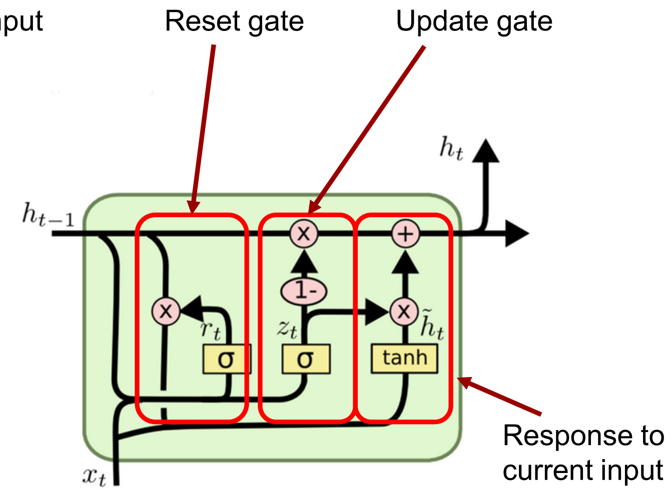
How does RNN produce result?



There are 2 types of RNN cells



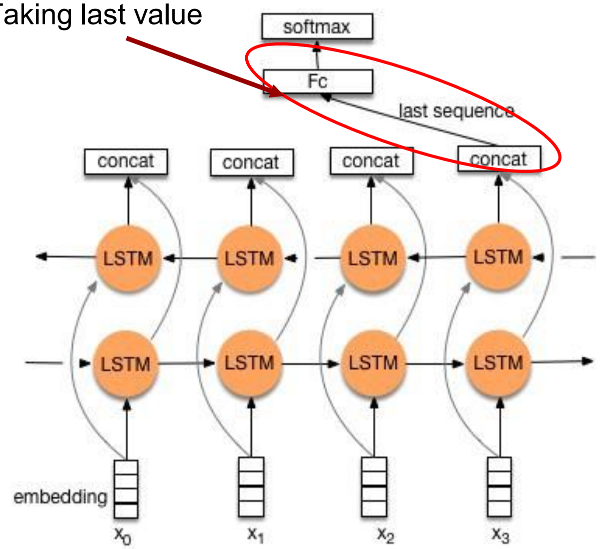
Long Short Term Memory (LSTM)



Gated Recurrent Unit (GRU)

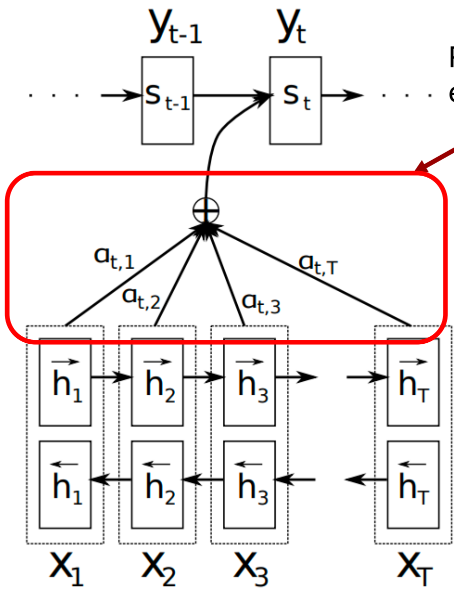
Recurrent AND deep?

Taking last value



Stacking

Pay "attention" to everything



Attention Model

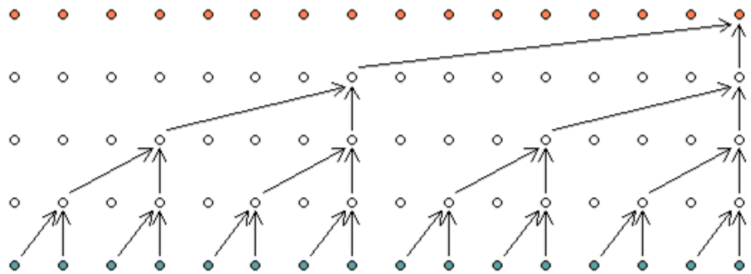
“Recurrent” AND convolutional?

Temporal convolutional network

Temporal dependency achieved through “one-sided” convolution

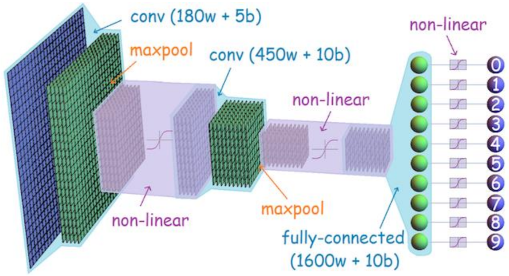
More efficient because deep learning packages are optimized for matrix multiplication = convolution

No hard dependency

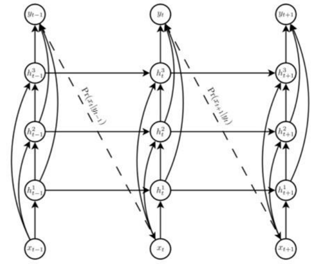


More? Take CS230, CS236, CS231N, CS224N

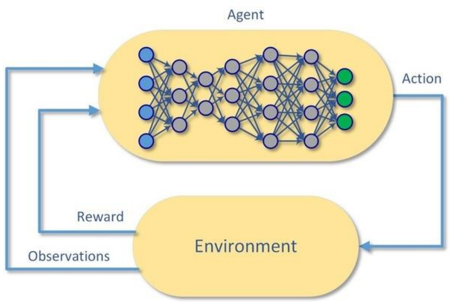
Convolutional NN
Image



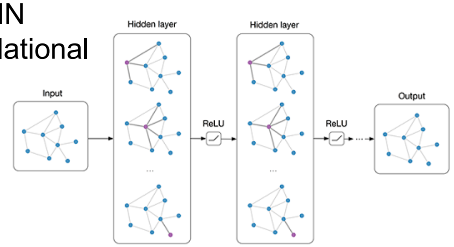
Recurrent NN
Time Series



Deep RL
Control System

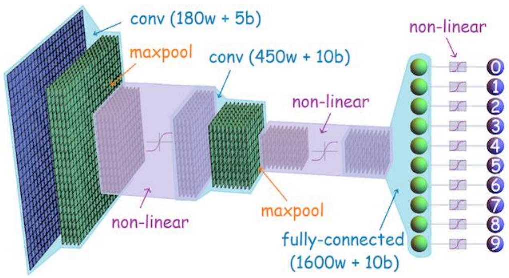


Graph NN
Networks/Relational

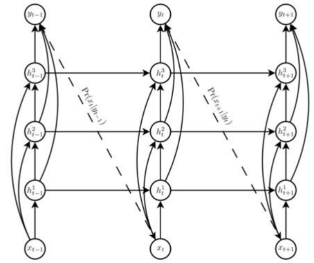


Not today, but take CS234 and CS224W

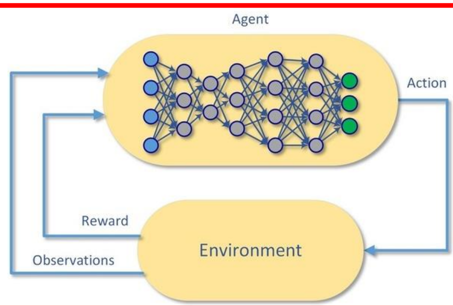
Convolutional NN
Image



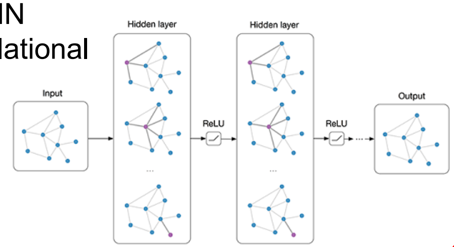
Recurrent NN
Time Series



Deep RL
Control System



Graph NN
Networks/Relational



Overview

- Motivation for deep learning
- Areas of Deep Learning
- Convolutional neural networks
- Recurrent neural networks
- Deep learning tools

Tools for deep learning

Keras

TensorFlow

theano

PYTORCH

Popular Tools

Specialized Groups

Caffe2

mxnet

Microsoft CNTK

Where can I get free stuff?

Google Colab

Free (limited-ish) GPU access

Works nicely with Tensorflow

Links to Google Drive

Azure Notebook

Kaggle kernel???

Amazon SageMaker?

Register a new Google Cloud account

=> Instant $300??

=> AWS free tier (limited compute)

=> Azure education account, $200?

To **SAVE** money

CLOSE your GPU instance

~$1 an hour

Good luck!
Well, have fun too :D