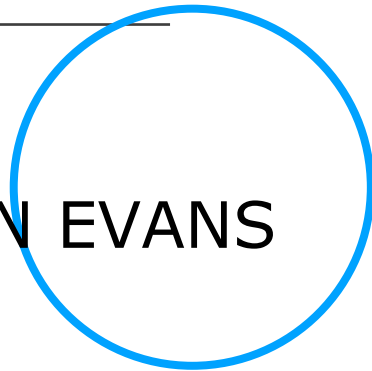


CSE 416 Section 7!

~~Pandemic Special Episode~~

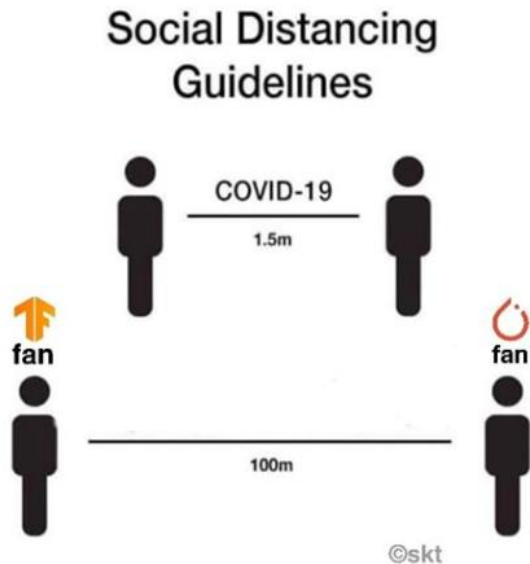
August 6, 2020

- Original slides by HONGJUN JACK WU, modifications by BEN EVANS



Goal for today!

PyTorch Introduction + CNNs



Geoff Hinton after writing the paper on backprop in 1986



how people imagine me
when I tell them I'm
deep learning engineer



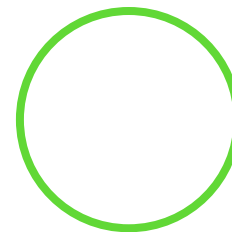
Convolutions

(PART I)

CORRESPONDS TO SLIDE #16 [REFERENCE](#)



CONVOLUTIONS – VERY BITTER MATH



Given two functions f and g , we define the convolution to be as the following integral transform:

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau.$$

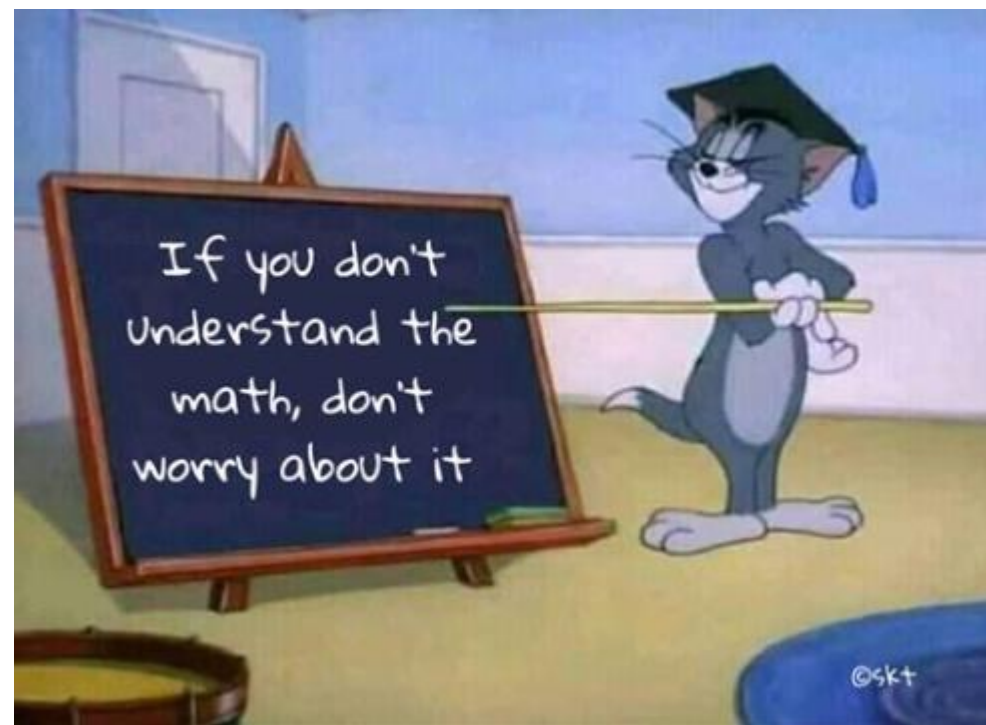
Okay, what the heck.

What does this mean???

I've never seen this in my life...

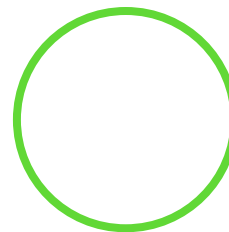
Am I going to fail this class because I have no idea and panikkkkkkkkkking?

No. Hang on.





CONVOLUTIONS



In order to understand this, you will at least need knowledge from some Laplace transformation from Differential Equations (MATH307) and matrix operations from Linear Algebra (MATH308).

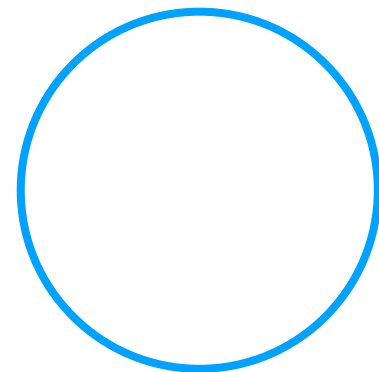
$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau.$$

In my honest opinion the blog we provided to you is still a bit too hard to understand on its own...

But you don't have to understand everything, you just need to understand...well, enough so you get why we care and how CNN works.

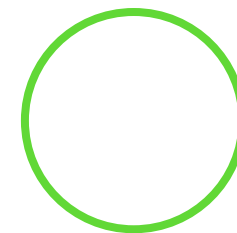
So let's dig into it and learn what you need to know, shall we?

Or if you are super mathy then read this: [Understanding Convolutions](#)





CONVOLUTIONS



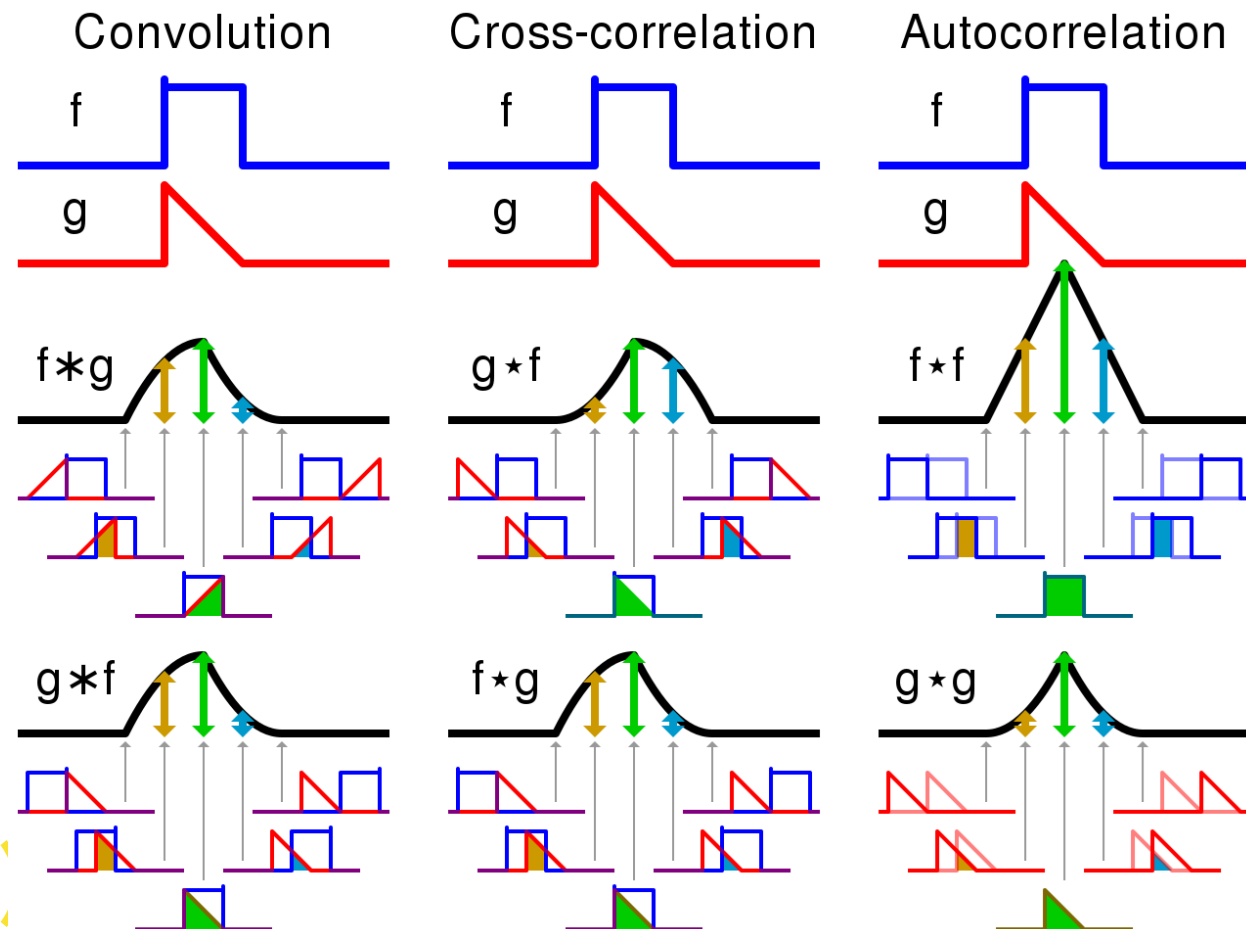
What it is:

In mathematics!

Convolution is a mathematical operation on two functions (f and g) that **produces a third function expressing how the shape of one is modified by the other.**

What does this mean?

This means we can use convolution to extract some high level features!





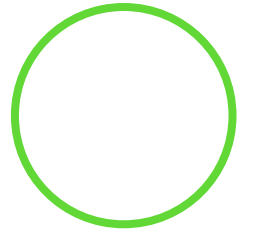
Convolutional Neural Networks

(PART II)

CORRESPONDS TO SLIDE #17 [REFERENCE](#)



WHY DO WE EVEN CARE ABOUT CNN



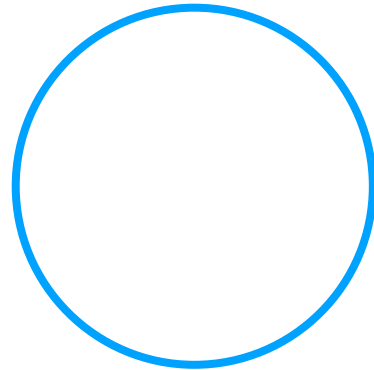
Like seriously, we learned sooooo many ML algorithms, why do we even care about CNN?
(At least when I first learned it I was like oh well fine whatever)

First, where we use it the most? -> Artificial Intelligence aka a fancy name for ML.

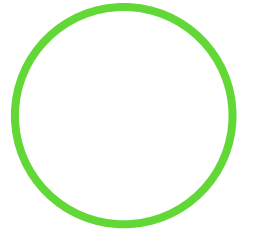
~~(Well, ML is a fancy name for math and statistics except you make more money because of the hype of artificial intelligence than mathematicians and statistians)~~



Like what in AI that we can do cool things? -> How about computer vision? Cooooool.



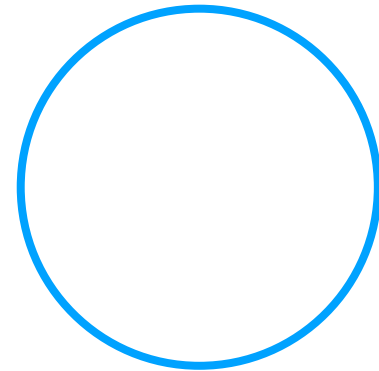
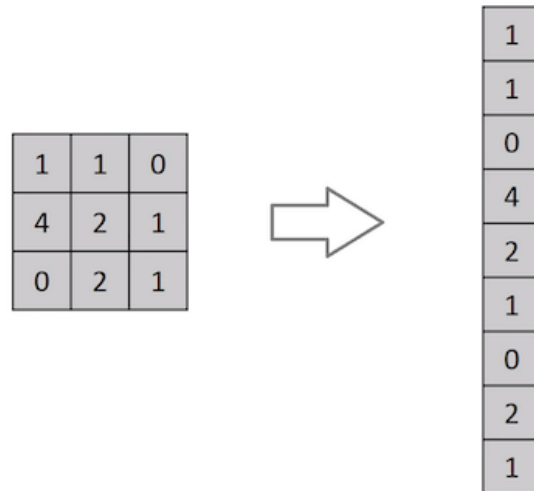
WHY DO WE EVEN CARE ABOUT CNN



What is computer vision in a nutshell?

- > Analyze a picture, or a matrix of numbers(pixels) that makes up an image.
- > Video is just many many images, and image is just many many pixels.

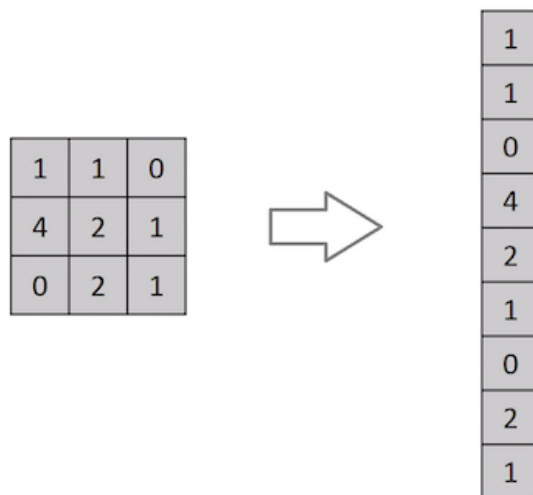
Now, one simple way, naively, one can come up with is say I have a black and white photo, I just convert all the pixels in a matrix into an array (aka flattening) and just feed it to a multi-level perceptron for classification purposes?



WHY DO WE EVEN CARE ABOUT CNN

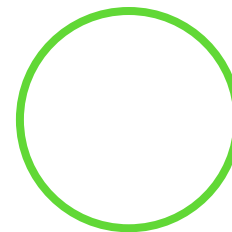
In cases of extremely basic binary images, the method *might* show an *average* precision score while performing prediction of classes.

But would have little to no accuracy when it comes to **complex** images having pixel dependencies throughout.





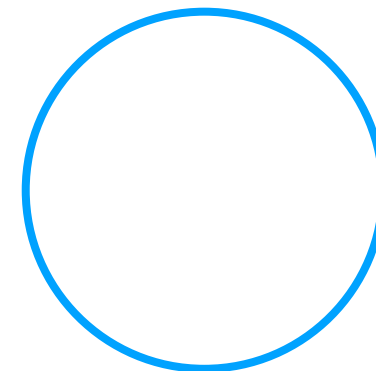
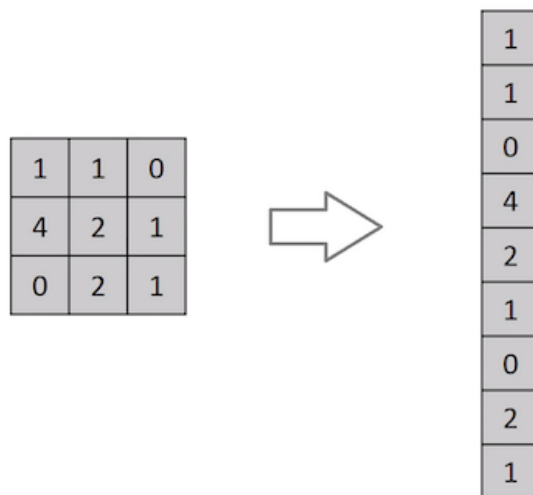
WHY DO WE EVEN CARE ABOUT CNN



A ConvNet is able to successfully capture the **Spatial and Temporal dependencies** in an image through the application of relevant filters.

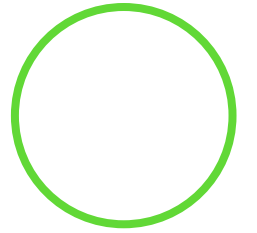
The architecture performs a better fitting to the image dataset due to the **reduction in the number of parameters involved and reusability of weights.**

In other words, the network can be trained to understand the sophistication of the image better.





WHY DO WE EVEN CARE ABOUT CNN




Let's break this down...

A ConvNet is able to successfully capture the **Spatial and Temporal dependencies** in an image through the application of **relevant filters**.

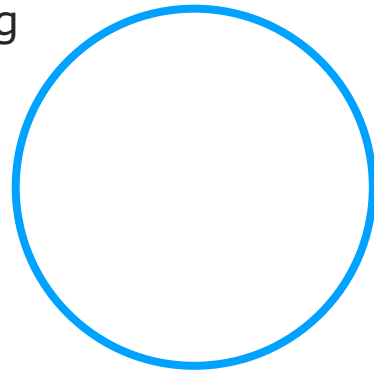
What do filters do?

Well, they filter out some information that we don't want and preserve information that we do want.



So, what should we use to make a filter that filters out all the unnecessary data without losing features which are critical for getting a good prediction?

You guessed it! Convolution!





WHY DO WE EVEN CARE ABOUT CNN



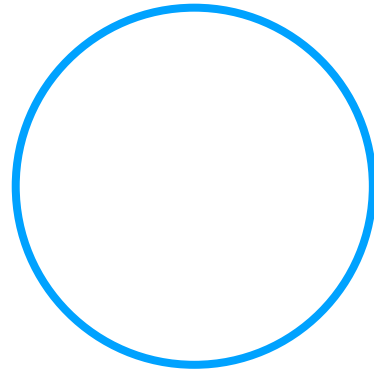
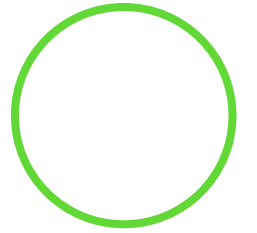
Let's break this down...

The architecture performs a better fitting to the image dataset due to the **reduction in the number of parameters involved** and **reusability of weights**.

Imagine that you are dealing with 8K images (7680x4320) and good luck dealing with every single pixel in that image...

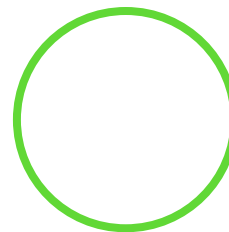
How do we reduce the amount of parameters involved so it is easier to process?

Yes! Convolution!





MAGIC BREAKDOWN



Now, I just boasted in front of you about how great this CNN thing is and to you, I probably look like some fraud that tries to sell you magic. How can this possibly be true?

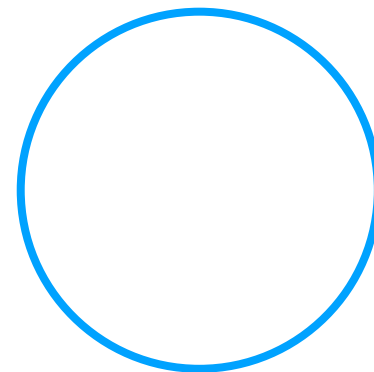
Let me explain how this thing works.

First, you choose a what's called a Kernel, or if that is a fancy name you can just think of it as a filter. It is just a matrix consist of 1 and 0s.

Say I choose:

Kernel/Filter, K =

1	0	1
0	1	0
1	0	1



MAGIC BREAKDOWN

Say I have this 5x5 matrix here and we try to convolute it with the filter we just made, It will look like this:

For each convolved feature simply add all the multiplication results together.
Like this:

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

Kernel/Filter, K =

1	0	1
0	1	0
1	0	1



MAGIC BREAKDOWN

So if we keep filtering, eventually it will look like this:

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

Kernel/Filter, K =

1	0	1
0	1	0
1	0	1

MAGIC BREAKDOWN

The Kernel shifts 9 times because of **Stride Length = 1 (Non-Strided)**, every time performing a **matrix multiplication operation between K and the portion P of the image** over which the kernel is hovering.

Kernel/Filter, K =

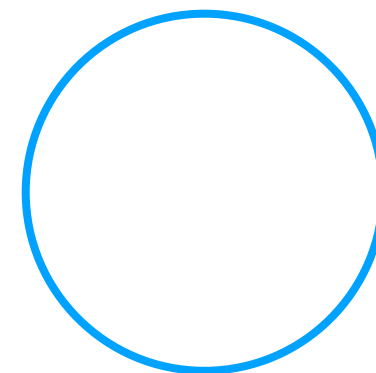
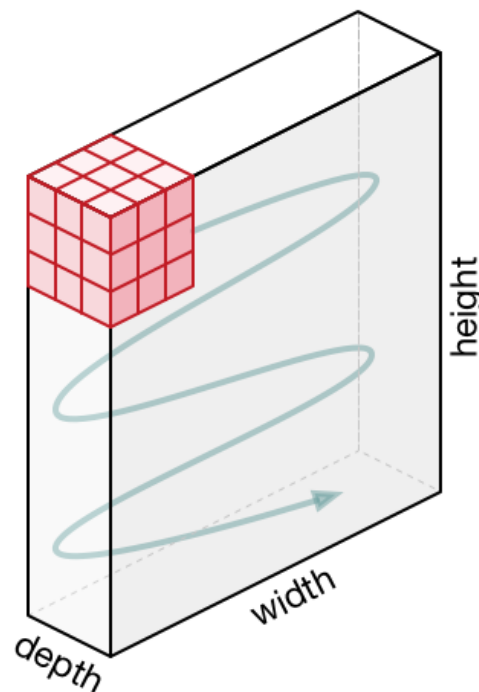
1	0	1
0	1	0
1	0	1

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature



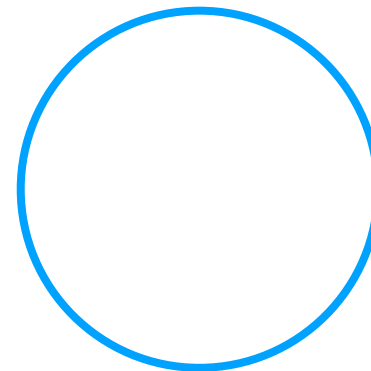
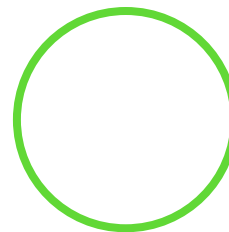
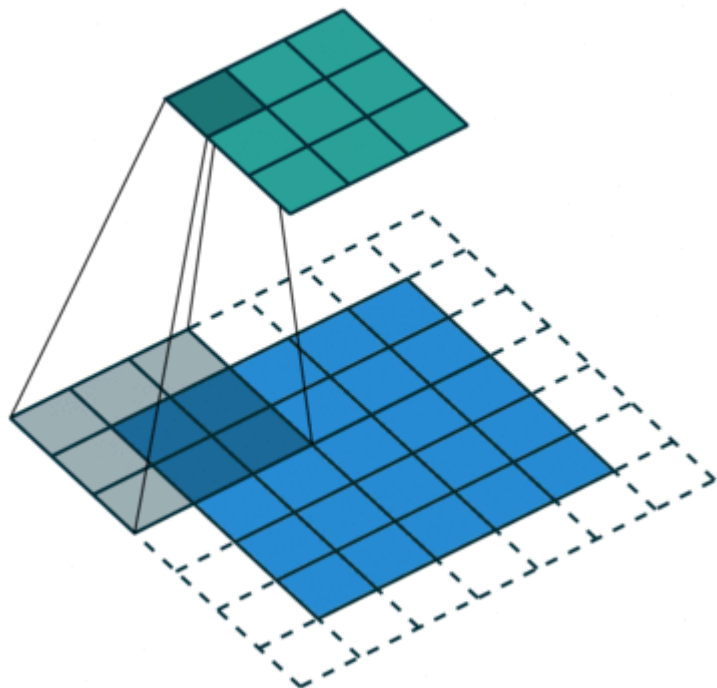


MAGIC BREAKDOWN



Stride Length (Ouch you just introduced another fancy word that we won't be able to remember but that's okay as long as you know what it means you're fine) literally means how many pixels away you go when you do the next filter operation.

For example: Convolution Operation with Stride Length = 2



MAGIC BREAKDOWN

The filter moves to the right with a certain Stride Value till it parses the complete width.

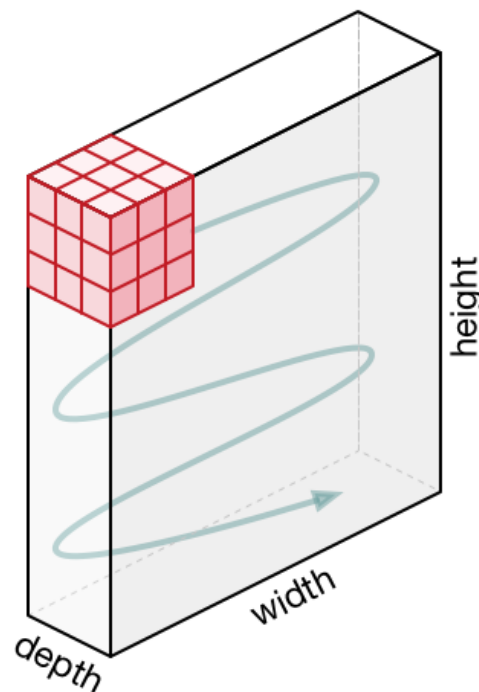
Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

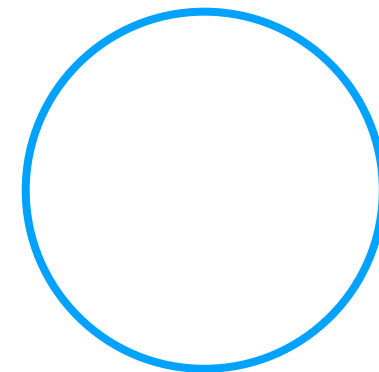
4		

Convolved
Feature



Kernel/Filter, K =

1	0	1
0	1	0
1	0	1





MAGIC BREAKDOWN

In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image.

Matrix Multiplication is performed between K_n and I_n stack ($[K1, I1]; [K2, I2]; [K3, I3]$) and all the results are summed with the bias to give us a squashed one-depth channel Convolved Feature Output.

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...

Input Channel #1 (Red)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



308

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...

Input Channel #2 (Green)

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-498

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...

Input Channel #3 (Blue)

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



164

+

+ 1 = -25

Bias = 1

-25				...
				...
				...
				...
...



I GET HOW FILTER WORKS, NOW WHAT?



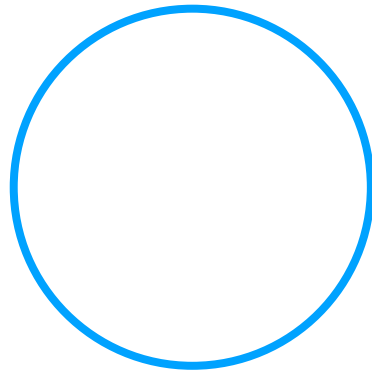
Why do we even do convolution operation???

The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image.

So what?



We are trying to let our model learn an image similar to how we would. Magic?





I GET HOW FILTER WORKS, NOW WHAT?



Magic? Not necessarily.


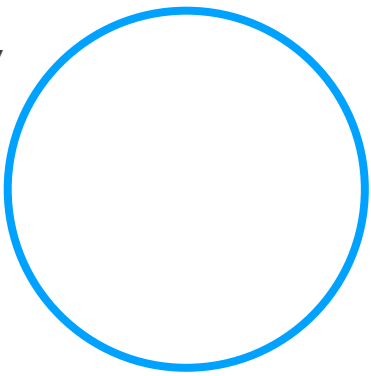

ConvNets need not be limited to only one Convolutional Layer!

Conventionally, the first ConvLayer is responsible for capturing the **Low-Level features**.

Such as edges, color, gradient orientation, etc.

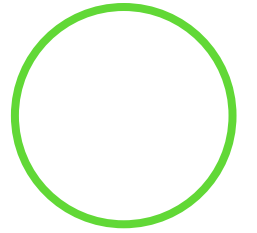
With added layers, the architecture adapts to the **High-Level features** as well.

Giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.





PADDING



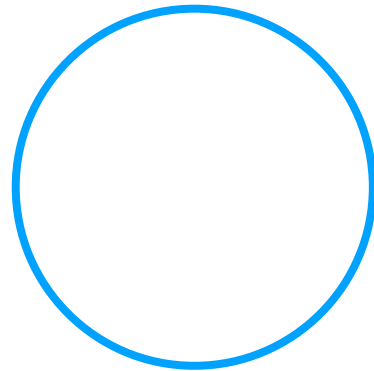
So far, when we do filtering, the result is in which the convolved feature is reduced in dimensionality as compared to the input.

However, sometimes we want the dimensionality is either increased or remains the same.

Houston we got a problem!!!



That is, the original matrix is not big enough for us to filter and obtain a result that we want (dim increase or remain the same).



PADDING

Turns out that pixels in the corners is used less than those in the middle, so we can just fill in 0s to augment the original matrix into a bigger matrix just so that our filter has more space to work with.

This is called Zero-Padding. Very intuitive.

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

Zero-padding added to image

PADDING

Now then, coming back, again, there are two types of results to the operation —

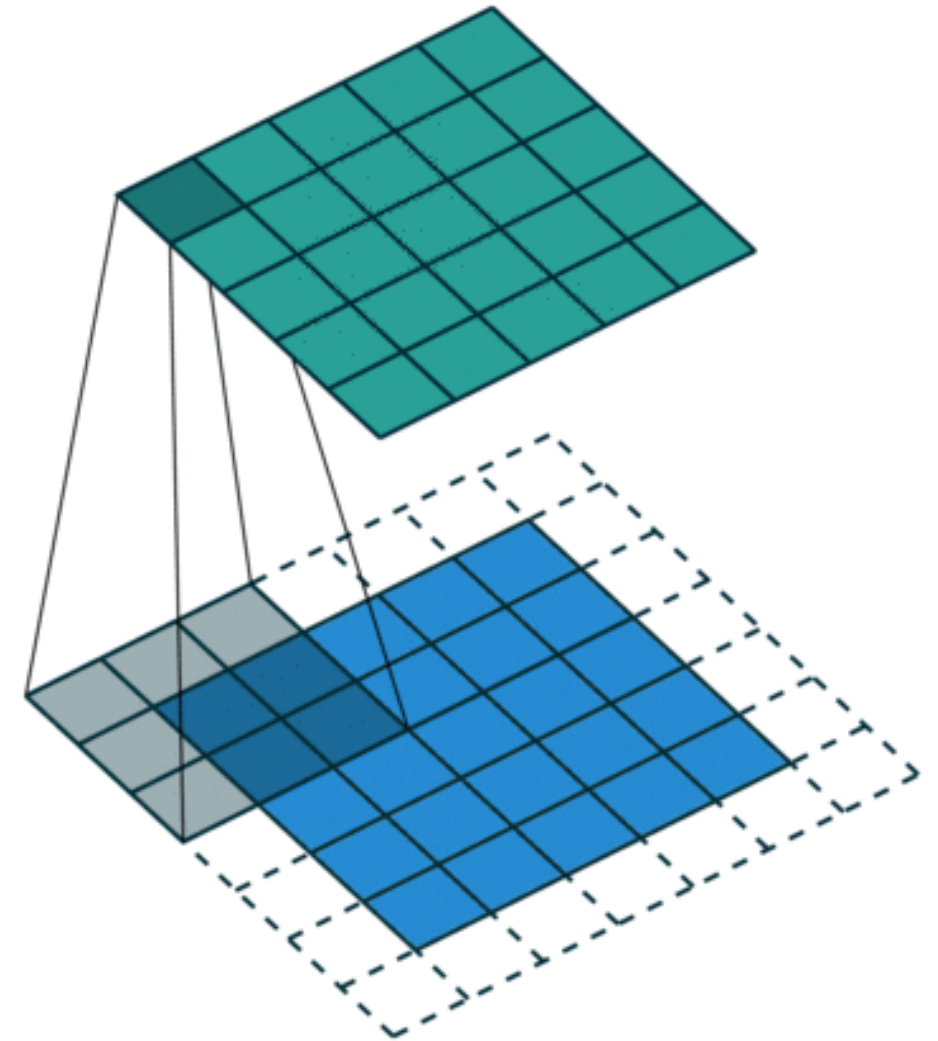
If we want to reduce the dimension all we do is what's called **Valid Padding**, another fancy word that you just need to know, aka what we've been doing before which does not use zero padding.

When we augment the $5 \times 5 \times 1$ image into a $7 \times 7 \times 1$ image and then apply the $3 \times 3 \times 1$ kernel over it, we find that the convolved matrix turns out to be of dimensions $5 \times 5 \times 1$. Hence the name — **Same Padding**.

Image:

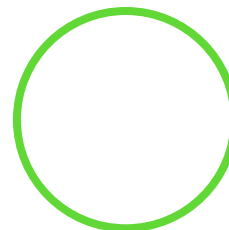
SAME padding: $5 \times 5 \times 1$ image is padded with 0s to create a $6 \times 6 \times 1$ image.

Learn more about padding and watch more animations: [HERE](#)





POOLING LAYER



What is a pooling layer and why I should care about it?

Remember that we want to process an image but there is just wayyy too many data and the complexity of model goes too high and it takes forever to compute?

Well, pooling layers comes to rescue.

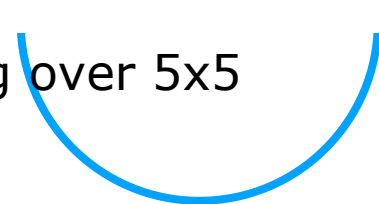
Pooling layer is responsible for reducing the spatial size of the Convolved Feature.



3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

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

Image: 3x3 max pooling over 5x5 convolved feature





POOLING LAYER

Two things they do:

Decrease the computational power required to process the data through dimensionality reduction.

Extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

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

Image: 3x3 max pooling over 5x5 convolved feature



POOLING LAYER

There are two ways to pool, both very intuitive:

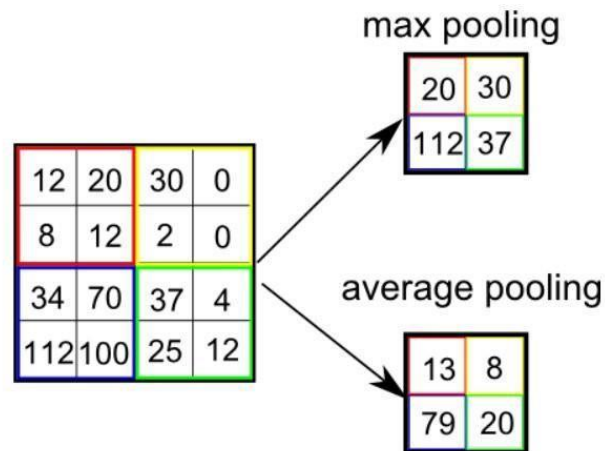
Max Pooling returns the **maximum value** from the portion of the image covered by the Kernel.

- Max Pooling also performs as a **Noise Suppressant**.
- It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction.

Average Pooling returns the **average of all the values** from the portion of the image covered by the Kernel.

- Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism.

Takeaway: **Max Pooling performs a lot better than Average Pooling.**



3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

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

Image: 3x3 max pooling over 5x5 convolved feature

OOOF WRAPPING UP ●

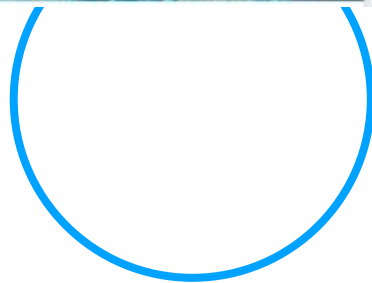
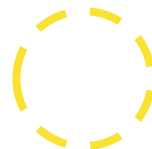
EXPLAINING CONVOLUTION AND POOLING ○

The Convolutional Layer and the Pooling Layer, together form the i -th layer of a Convolutional Neural Network.

Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but **at the cost of more computational power.**

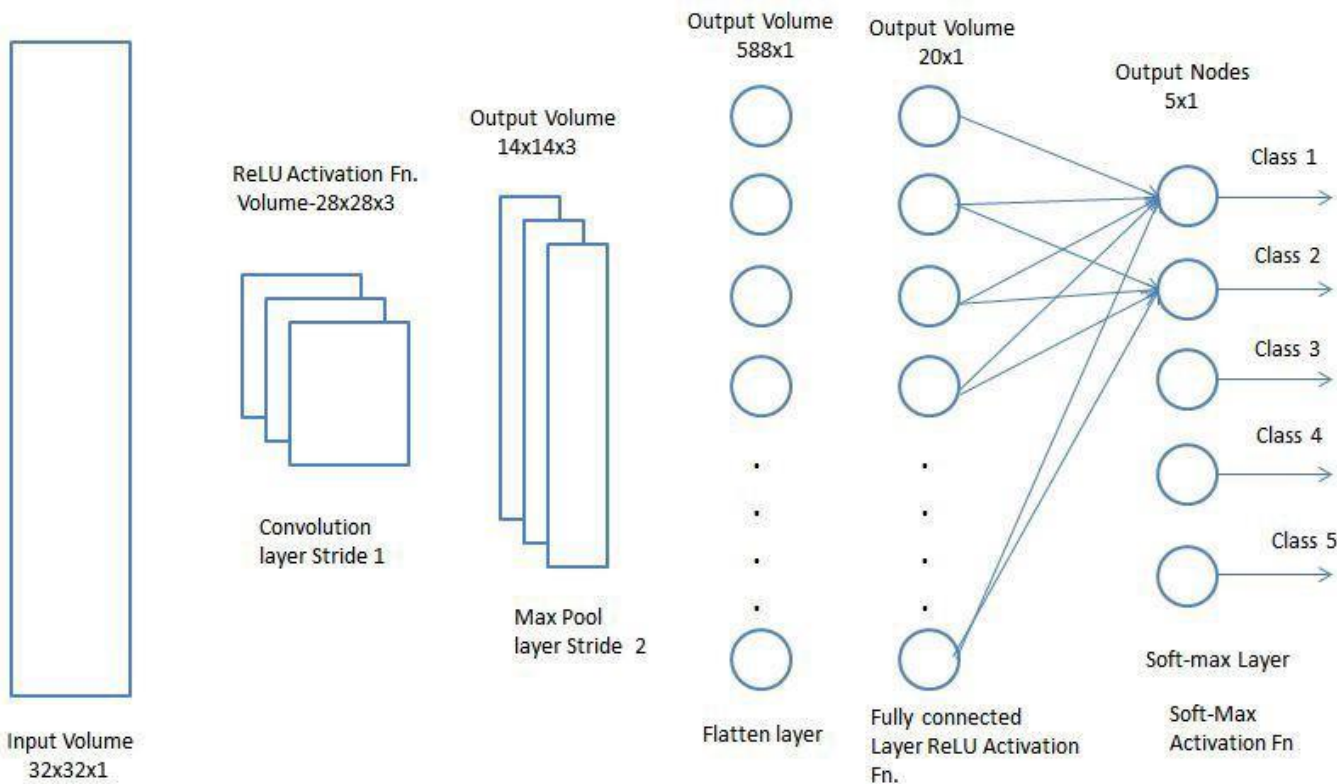
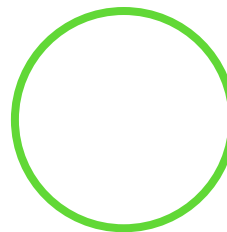
After going through the above process, we have successfully enabled the model to understand the features.

Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.





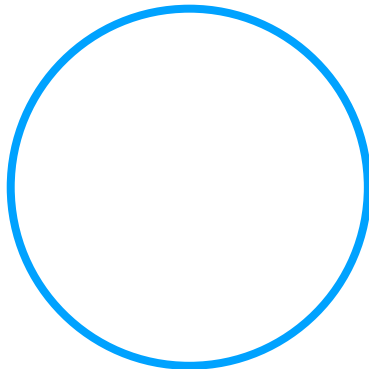
OOF WRAPPING UP



Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer.

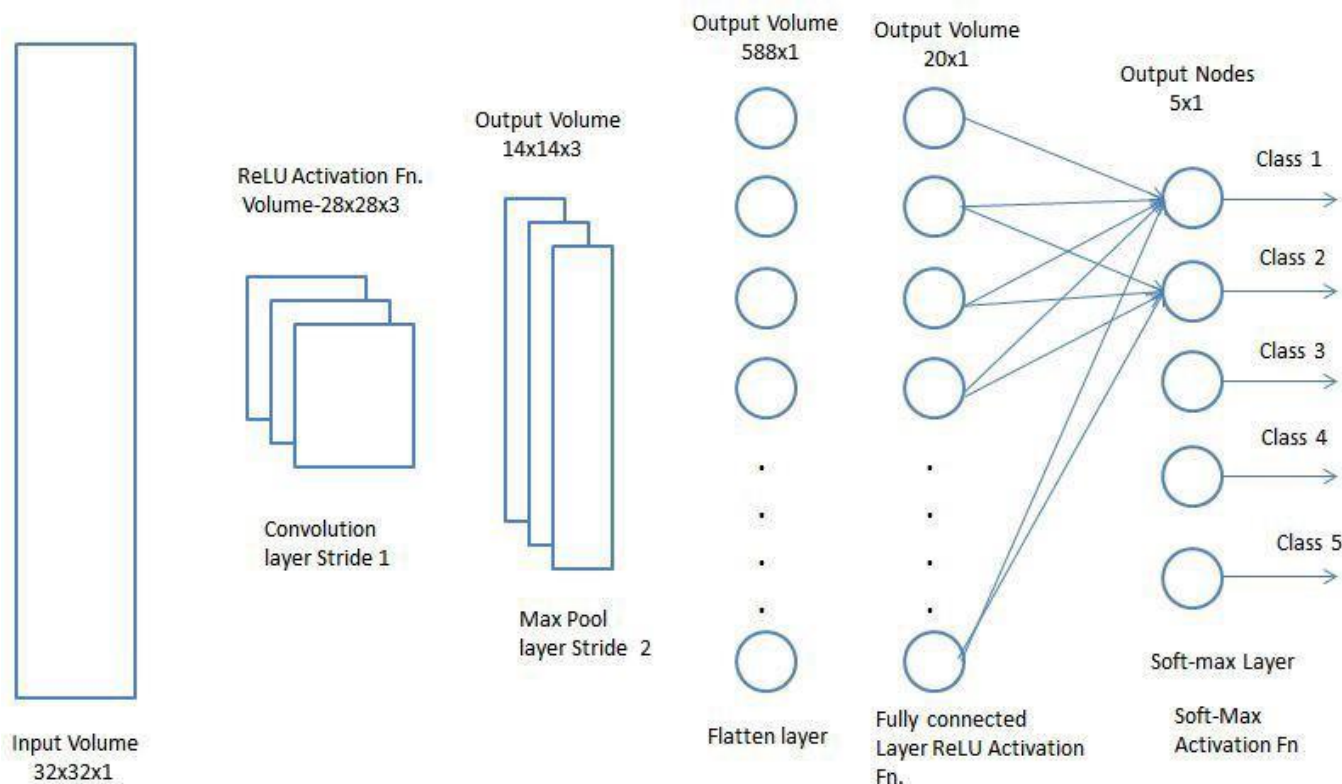
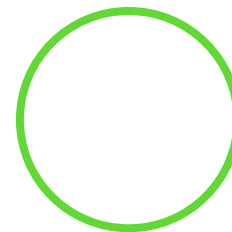
The Fully-Connected layer is learning a possibly non-linear function in that space.

An example neural network.





OOF WRAPPING UP



An example neural network.



Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector.

The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training.

Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **Softmax Classification** technique.



License

This material is originally made by [Hongjun Wu](#) for the course [CSE416: Introduction to Machine Learning](#) in the Spring 2020 quarter taught by [Dr. Valentina Staneva](#), at University of Washington Paul G. Allen School of Computer Science and Engineering.

It was originally made for educational purpose, in a section taught by teaching assistants to help students explore material in more depth.

Any other materials used are cited in the Credits section.

This material is licensed under the [Creative Commons License](#).

Anyone, especially other educators and students, are welcomed and strongly encouraged to study and use this material.