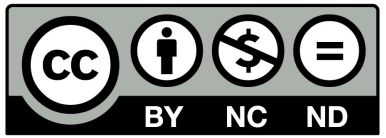
Learn ai

Computer Vision

Convolutional Neural Networks

An AI Commons initiative



What is computer vision?

- Computer Vision refers to any type of algorithm or piece of software that is able to understand and extract low and high level concepts from visual data
- This definition is often extended to encapsulate anything involving images and machine learning

Why computer vision?

Learn ai

- A lot of human intelligence is visual
- Lots of applications
- Fun! Very easy to share results and findings
- Examples
  - Self-driving cars
  - Snapchat filters
  - Medical imaging

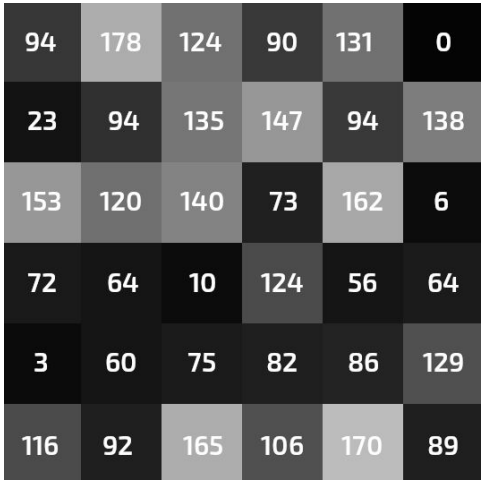
AI

Learning goals

- How computers represent images
- Understand what a Convolutional Layer does
- How CNNs are structured
- How we can use CNNs for image classification

Grayscale images

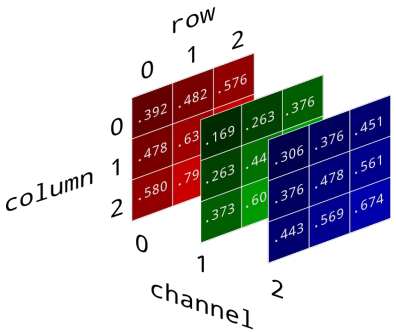
- A single pixel has a brightness value in the range of 0 - 255
  - 0 = black | 255 = white
- An image can be thought of as a 2D array of pixels
- So we can represent an image as a 2D array of numbers

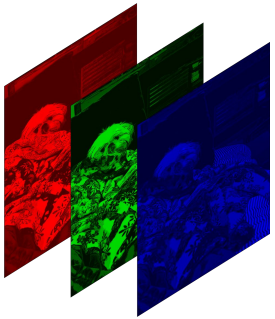


https://medium.com/@somdevsangwan/unblurring-images-for-osint-and-more-part-1-5ee36db6a70b

Colour images

- Similar to grayscale each pixel can be represented by numbers
- Each pixel has 3 values, RED, GREEN and BLUE. Each going from 0 - 255
- We can think of an image as stacking three of these 2D number arrays
- This gives us our image with 3 channels

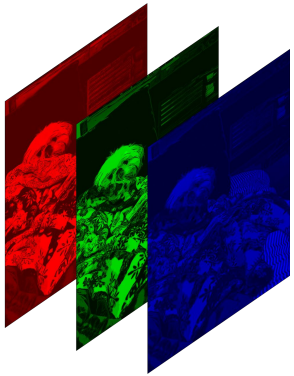


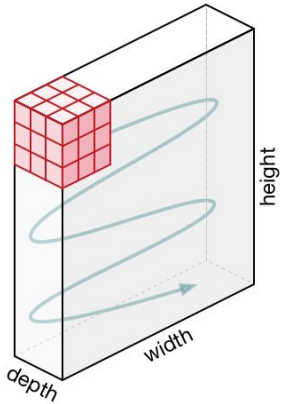


https://www.kdnuggets.com/2019/12/convert-rgb-image-grayscale.html

Colour images

- Since we stack our three colour channels stacks depth-wise to create our image, we can consider an image to be a 3D Matrix
- Luckily computers are very good at working with matrices of numbers





https://machinethink.net/images/vggnet-convolutional-neural-network-iphone/ConvolutionKernel@2x.png

Convolutional layer

- **DEFINITION:** Kernel - A *matrix* of numbers that can be used to detect a *specific feature* in some *section* of the image
- **SYNONYMS:** Filter, Feature Detector

Convolutional layer

DEFINITION: Kernel - A *matrix* of numbers that can be used to detect a *specific feature* in some *section* of the image

- The goal of a **convolutional layer** is to *use* many Kernels to extract *important* visual features from an image
- The hope is that you can determine a lot about an image based on the set of features which are present, and where these features are in the image

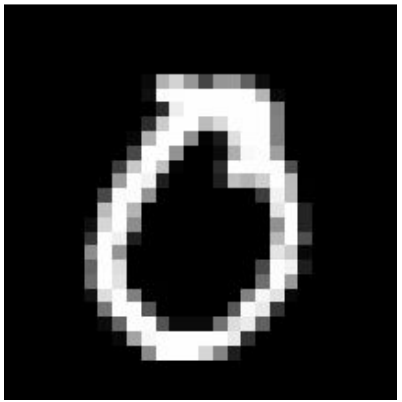
Convolutional layer

DEFINITION: Kernel - A *matrix* of numbers that can be used to detect a *specific feature* in some *section* of the image

- For example:
  - Assume we have Kernels which are able to detect edges, one kernels to find vertical edges, another for horizontal edges and kernels for horizontal edges
  - Let’s say we are trying to classify images as either a 1 or a 0

Convolutional layer

- We can determine whether or not the images below are 1 or 0 by looking at its edges
- The zero has many curved and diagonal edges connected to one another whereas the one has many more straight edges and sharp corners

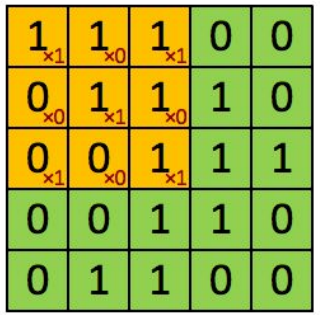


How do kernels detect features?

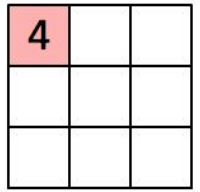
- As we said before a kernel is a matrix of numbers
- We also know that an image can be represented as a matrix of numbers
- So we can perform an operation between the image and the kernel to detect whether or not the feature that the kernel detects is present in this part of the image

1	0	1
0	1	0
1	0	1

Kernel



Image

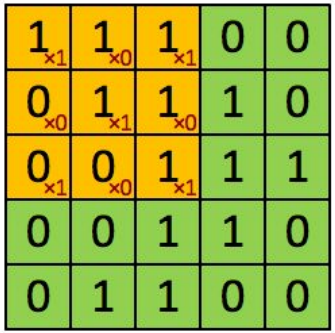


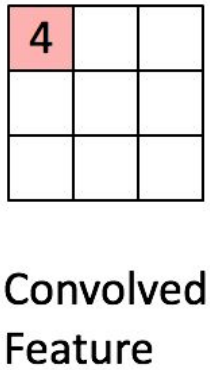
Convolved Feature

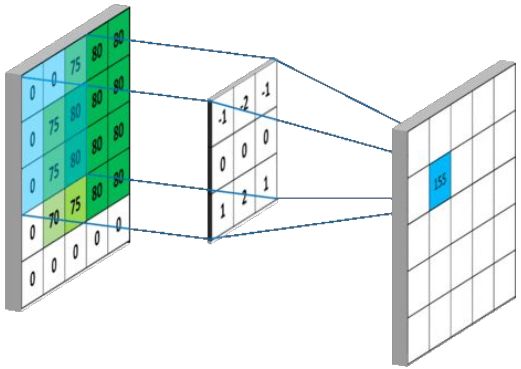
http://deeplearning.stanford.edu/wiki/images/6/6c/Convolution_schematic.gif

How do kernels detect features?

- The kernel is passed over top of the image and *applied* to the pixels it overlaps. This operation results in one number which tells us “how present” the given feature was at that location



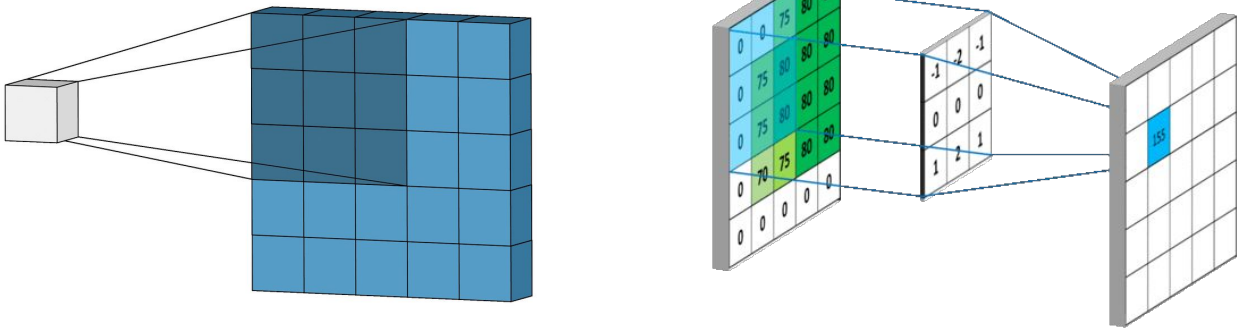




http://deeplearning.stanford.edu/wiki/images/6/6c/Convolution_schematic.gif
https://mlnotebook.github.io/post/CNN1/

How do kernels detect features?

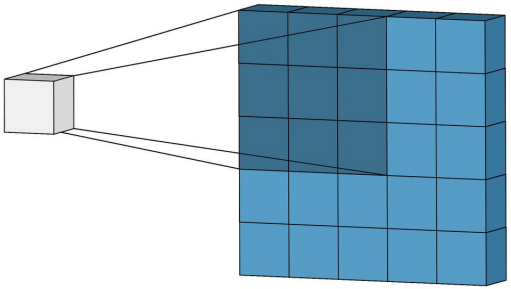
- Notice that this image is a single 2D array. Therefore this animation demonstrates the operation on a grayscale image. Since **colour images are 3D matrices** the diagram is a bit different but the **idea is the same**.

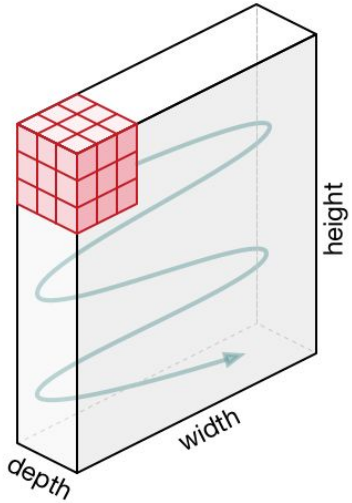


http://deeplearning.stanford.edu/wiki/images/6/6c/Convolution_schematic.gif
https://mlnotebook.github.io/post/CNN1/

How do kernels detect features?

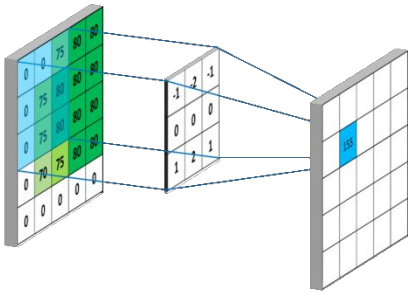
- For an RGB image the pixels have depth since there are three numbers
- Therefore we have 3D kernels which have the same purpose as the 2D kernels, they detect if a feature is present in a section of the image

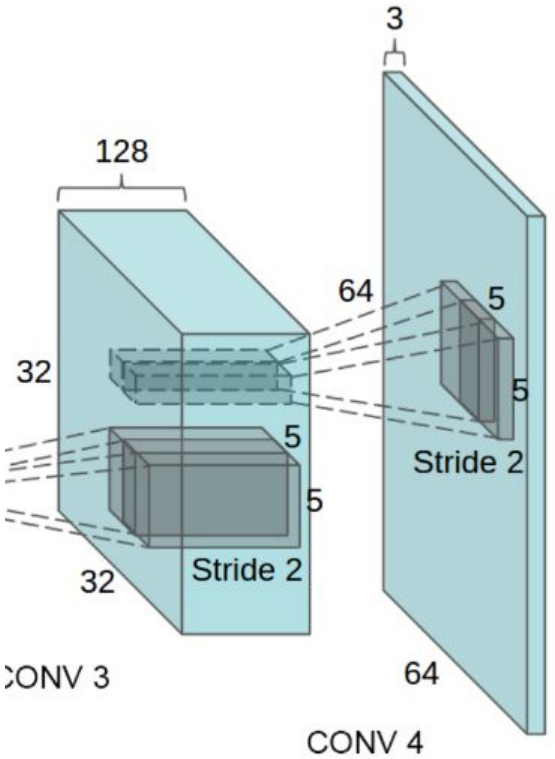




How do kernels detect features?

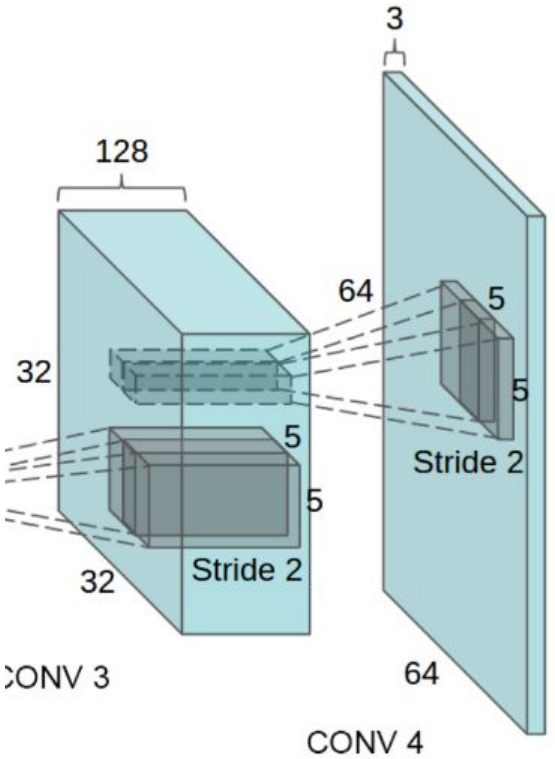
- As seen in the diagram below, each filter generates a single number for each position it can take on the image
- In a Convolutional Layer we will have many filters and therefore there are multiple values for each kernel position. These are stacked depthwise creating a 3D matrix of results





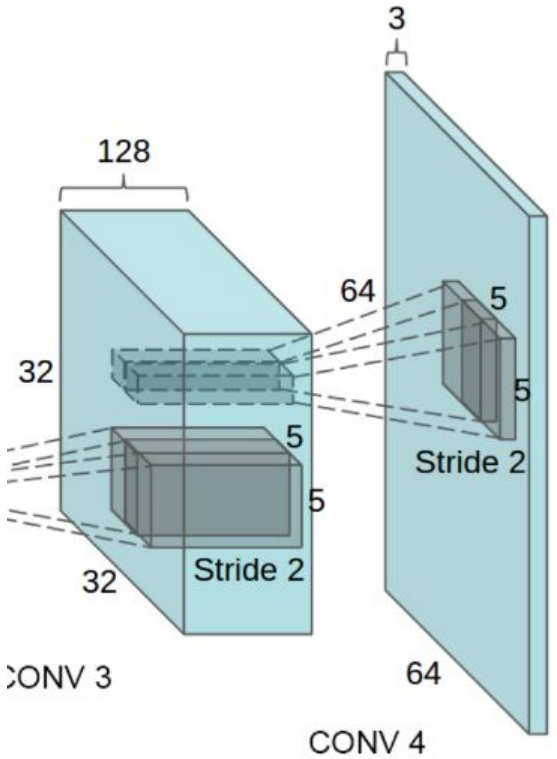
How do kernels detect features?

- From this diagram we can see that the 5x5x3 kernel is being passed over the 64x64x3 image
- Every position it can take results in a single entry in its corresponding depthwise “aisle” in the 32x32x128 block
- Since the left block has depth of 128, we can assume there are 128 kernels applied to the image



How do kernels detect features?

- Each entry in the left block tells us how present each individual kernels feature was at that position in the image
- Therefore the left block is a feature space which has abstracted the image by turning the individual pixels from the image into higher level features

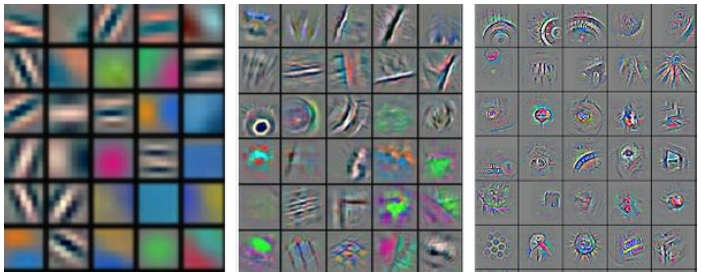


What features are we even detecting?

Learn ai

No idea!

- We have been assuming that the features our model is trying to detect will be helpful
- The model has to learn what the features look like and which ones are important
- It is easy to understand for humans if the features are simple like vertical edges but in practice the features may be undecipherable for humans but can still perform well for the model



Aside about stride

- When moving the filter over the image have the option of how many pixels to move the kernel by as it passes over the image
- Typically* a stride of 2 is used, as a result we only check for features in half of possible positions because the kernel skips one position every time it moves
- **tldr;** the size of feature space halves each time a convolutional is applied
- **(tldr;)$^{-1}$** see appendix

* when stride is used it is typically a stride of 2 but some models favour max pooling layers which we will not mention in this lecture

Meta-features | Connecting convolutional layers

- A single conv layer can be used to obtain a feature map from an image
- This feature map tells us where features live within the image
- **Intuition**: We know where a bunch of small features exist in the image, but in order for us to classify what the image is we will need to connect multiple features together to see if more abstract shapes and patterns exist

Meta-features | Connecting convolutional layers

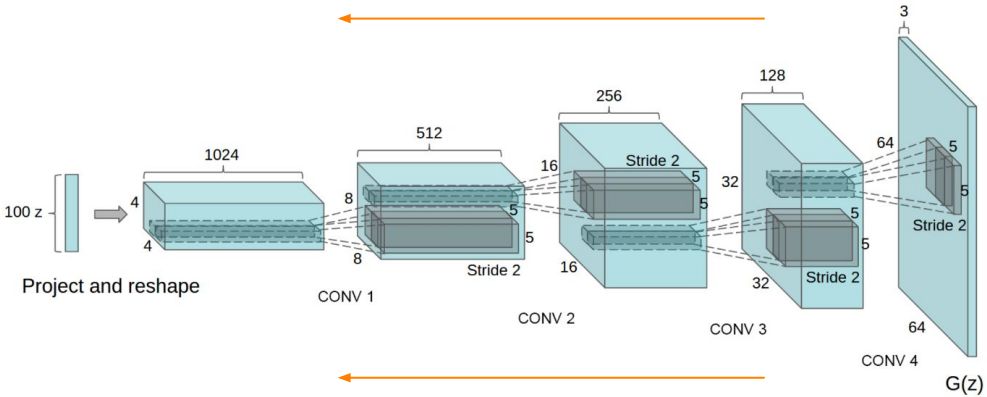
Intuition: We know where a bunch of small features exist in the image, but in order for us to classify what the image is we will need to connect multiple features together to see if more abstract shapes and patterns exist

- To combine many features together we apply a conv layer onto the feature map that was produced from the previous conv layer
- This second conv layer can try to learn more general/abstract features that **combine the presence of the features** seen in our previous layer

Meta-features | Connecting convolutional layers

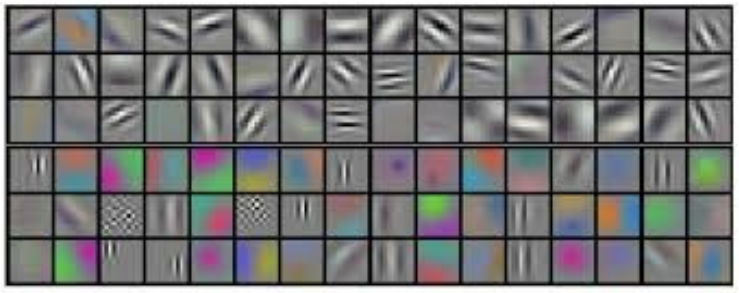
Intuition: We know where a bunch of small features exist in the image, but in order for us to classify what the image is we will need to connect multiple features together to see if more abstract shapes and patterns exist

- Since the feature space halves in size each time we are able to *see more of the image* as we get several layers down in the model
- This also allows us to more easily find where many features can combine together to make meta-features

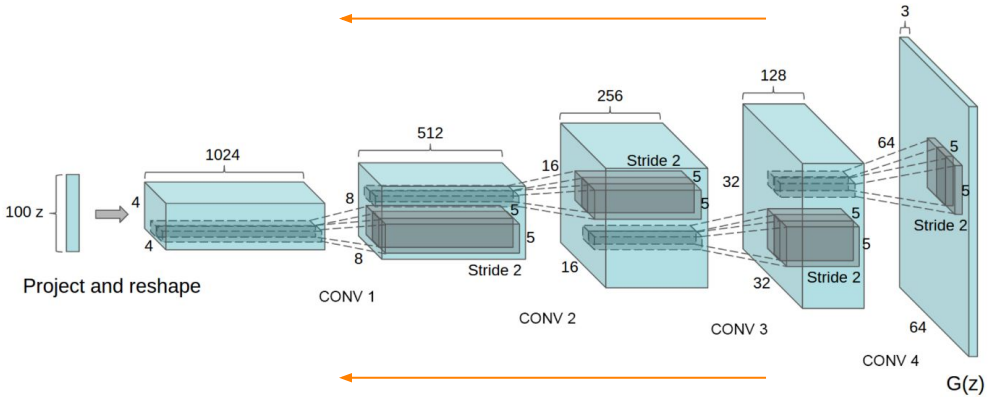


Meta-features | Connecting convolutional layers

- For example: if the first conv layer detects edges then the second layer might learn to connect several edges to create simple shapes. Then a third layer could find many shapes in certain orientations to determine what the image is

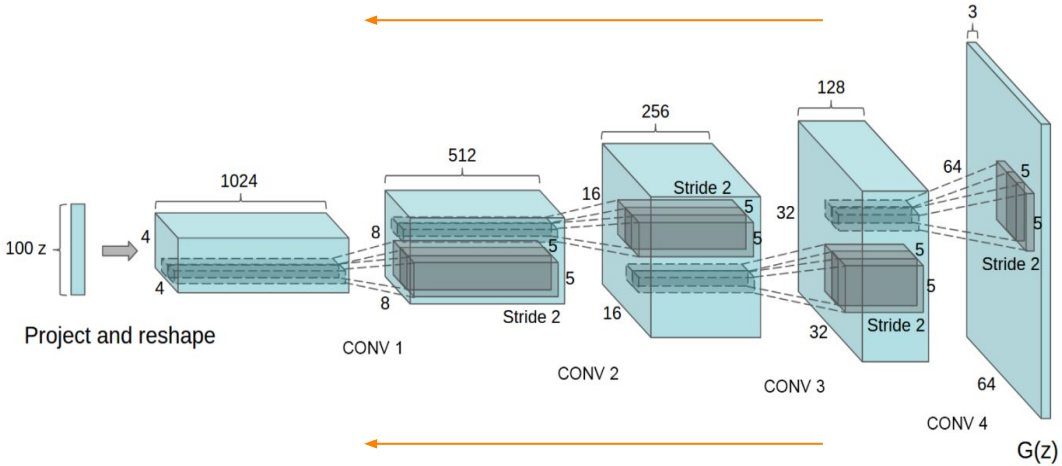


Example of learned kernels:
https://discuss.pytorch.org/t/visualize-feature-map/29597



Meta-features | Connecting convolutional layers

- Overall goal of CNN is to connect several conv layers together to gain insight and information about high level concepts in the image which can be detected through a combination of many layers of well learned feature detectors



Why not use fully connected layers?

- Fully connected layers need each node to look at every pixel to determine its value
    - This can make it tough to isolate certain features in the image and training this type is difficult because nodes need to figure out independently of one another which pixels they should ignore
- The pixels they choose to ignore might not be near each other and they might detect same thing as another node
- At a high level this isn't as effective as making a model which enforces the idea of just looking at certain parts of the image

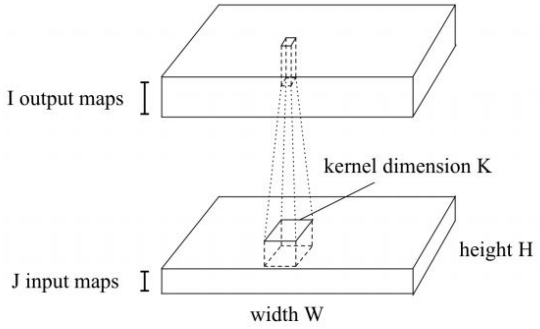
Why not use fully connected layers?

- Many more model parameters
- Note that K < W,H in *all* practical cases
- Therefore:

$$W^2H^2IJ \gg K^2IJ$$

$$W^2H^2IJ > WHK^2IJ$$

- As a result the number of params in a FC Layer is more than a Conv Layer

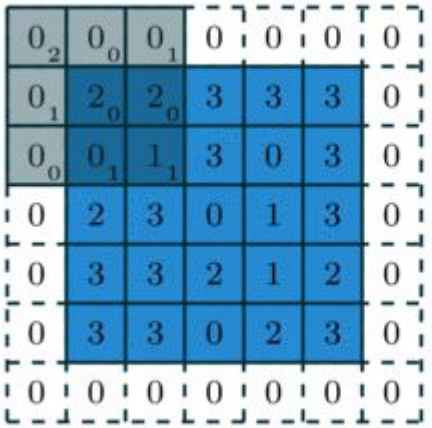


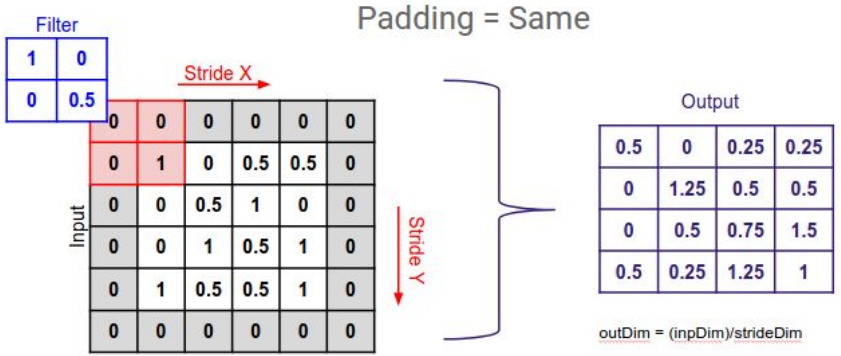
	fully connected layer	convolution layer
# output units	WHI	WHI
# weights	W^2H^2IJ	K^2IJ
# connections	W^2H^2IJ	WHK^2IJ

Details about kernel application

Padding: The process of adding zeros around an image to ensure that the kernel application results in an output of the desired dimensions

- Typically same padding is used allowing image to stay the same size with stride 1 or half in size with stride 2



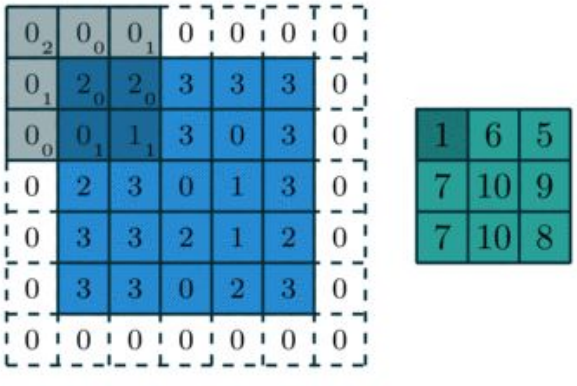


Further reading:
https://medium.com/@ayeshmanthaperera/what-is-padding-in-cnns-71b21fb0dd7
https://www.geeksforgeeks.org/cnn-introduction-to-padding/

Details about kernel application

Stride: The amount of pixels you shift the kernel

- Using a stride of 1 maintains same image dimensions (with same padding) and a stride of 2 halves the image dimensions
- Typically in a CNN you either use a max pooling layer to half the size of the images or you have a stride of two on



1	6	5
7	10	9
7	10	8

<-- Stride of 2 in this image

Further reading:
https://deepai.org/machine-learning-glossary-and-terms/stride
https://www.quora.com/What-does-stride-mean-in-the-context-of-convolutional-neural-networks

Further Readings | (CNNs and GANs)

- https://medium.com/@jonathan_hui/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09
- https://cs.nyu.edu/~yann/2010f-G22-2565-001/diglib/lecun-98.pdf
- https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- https://csc413-2020.github.io/assets/readings/L05a.pdf
- https://csc413-2020.github.io/assets/readings/L05b.pdf
- https://csc413-2020.github.io/assets/slides/lec10.pdf
- https://sgfin.github.io/files/notes/CS321_Grosse_Lecture_Notes.pdf (pg 153-162)

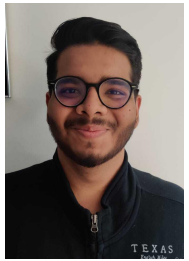
Contributors

Elias Williams

Addison Weatherhead

Isha Sharma

Kevin Zhu



Pranjal Kumar

Gilles Fayad