Introduction to convolutional ANNs (CNNs)

Yulia Newton, Ph.D.

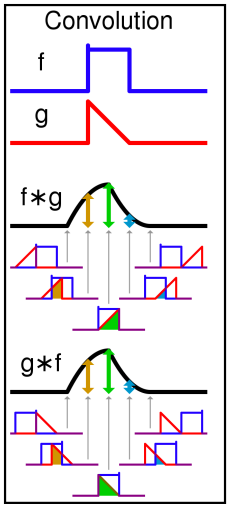
CS156, Introduction to Artificial Intelligence

San Jose State University

Spring 2021

What is convolution?

- “Convolution is a form or shape that is folded in curved or tortuous windings” - Merriam-Webster
- In mathematics, convolution function is an operation on two other functions, which produces a third function



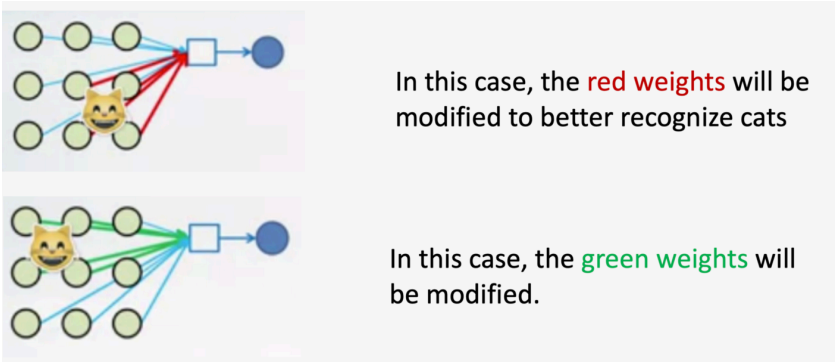
https://en.wikipedia.org

What are convolutional neural networks?

- Convolutional neural nets are a subtype of forward feed artificial neural networks
- Perform great when applied to analyzing image data
- Belong to the class of learning techniques called “representation learning”
  - Automatically discover the set of features needed for the ML task
  - Remember that MLPs use fully connected (dense) layers of neurons
    - Leading to increased model complexity
    - Increased training time
    - Potentially overfitting the model as it uses every feature, even those that are not important to the task

Why dense ANNs are not good at image recognition?

- They are not translation invariant (spacial invariance)
  - What if the picture is in a different part of the picture frame?
  - What if there's a slight rotation of the object you are learning to recognize?
  - What if there are variations on the objects our model is meant to recognize?

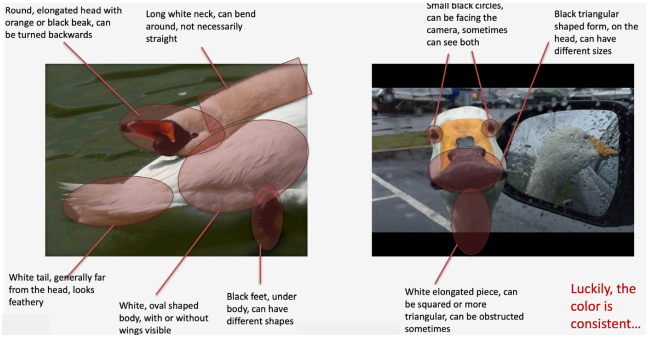


https://towardsdatascience.com

Can we build an ANN model to recognize a swan in a picture? There are challenges!

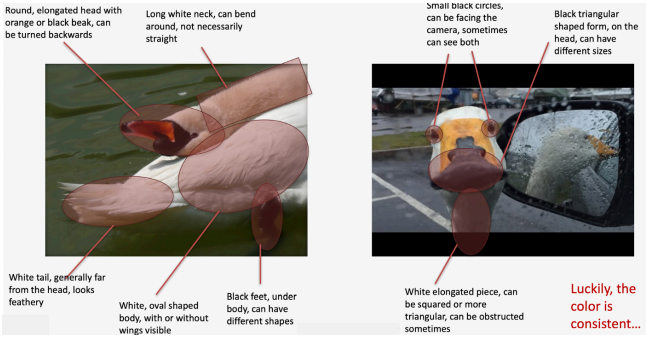
https://towardsdatascience.com

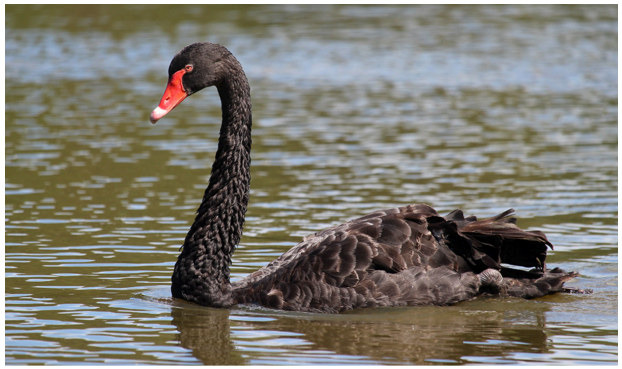
Can we build an ANN model to recognize a swan in a picture? There are challenges!



https://towardsdatascience.com

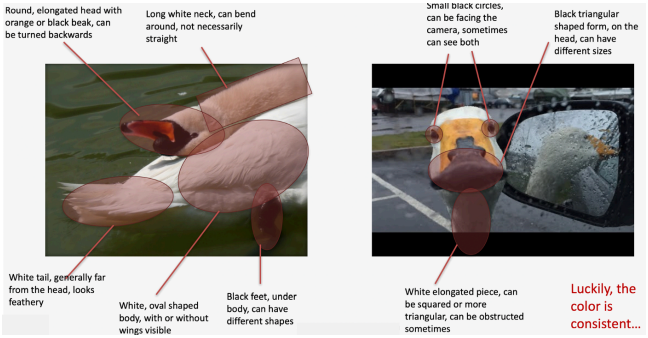
Can we build an ANN model to recognize a swan in a picture? There are challenges!

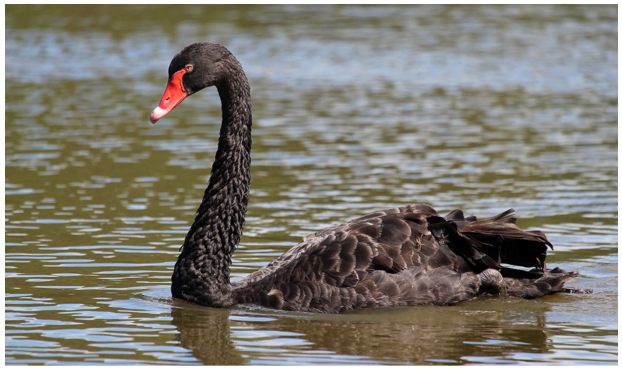




https://towardsdatascience.com

Can we build an ANN model to recognize a swan in a picture? There are challenges!



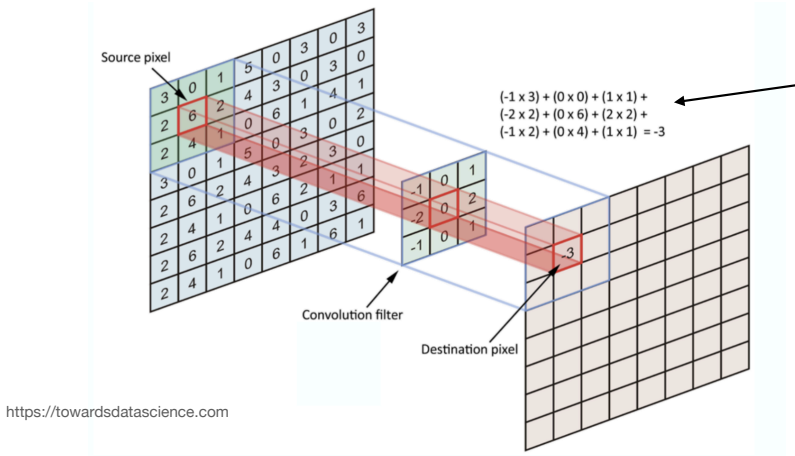


https://towardsdatascience.com

Man in swan tent photographing swans

Main idea behind CNNs

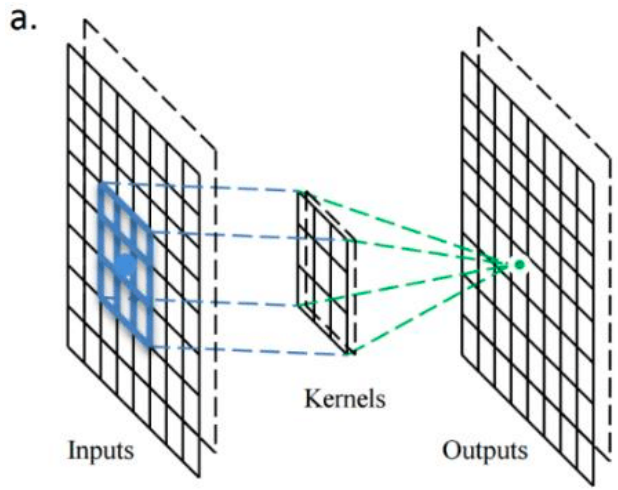
- Nearby features (pixels) are more closely related than more distant features
- CNNs utilize filters - small tiles that are moved across the image in a systematic manner (from left to right, from top to bottom)
  - For each filter, we compute the representation by using a convolution function



*A filter can also be called a **kernel** or a **mask***

Convolution operation

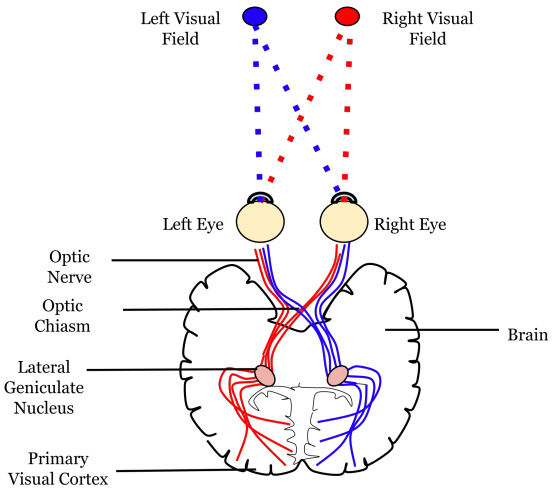
- Convolution operation condenses a group of features into a single feature



"Artificial neural network for bubbles pattern recognition on the images" Journal of Physics Conference Series 2016

Inspiration for CNNs

- CNNs take inspiration from the visual cortex of the brain
  - Small regions of cells are sensitive to specific regions of visual field

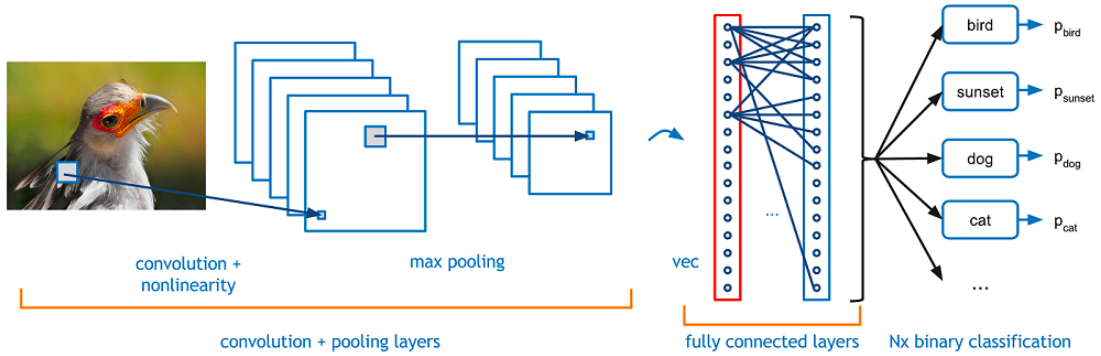


https://en.wikipedia.org

Visualizing a filter passing over an image

- https://i.ibb.co/SxZ9WYs/nagesh-cnn-intro-4.gif

The convolution layer transforms an image into new feature space



Adit Deshpande
https://adeshpande3.github.io

CNN uses

- Image data classification
  - E.g. find all the images that have a train in them
- Image segmentation
  - Partitioning an image into multiple parts
- Signal processing
- Other computer vision problems
- Have been shown to work well in population genetic inference (Flagel *et al.* Mol Biol Evol . 2019)
- CNNs can work well on other types of data that exhibit autocorrelation patterns

Filter size

- A filter can have any size
- Typically, a filters are 3x3 for 2-D images and 3x3x1 for 3-D images

Smaller Filter Sizes	Larger Filter Sizes
We only look at very few pixels at a time. Therefore, there is a smaller receptive field per layer.	We look at lot of pixels at a time. Therefore, there is a larger receptive field per layer.
The features that would be extracted will be highly local and may not have a more general overview of the image. This helps capture smaller, complex features in the image.	The features that would be extracted will be generic, and spread across the image. This helps capture the basic components in the image.
The amount of information or features extracted will be vast, which can be further useful in later layers.	The amount of information or features extracted are considerably lesser (as the dimension of the next layer reduces greatly) and the amount of features we procure is greater.
In an extreme scenario, using a 1x1 convolution is like considering that each pixel can give us some useful feature independently.	In an extreme scenario, if we use a convolution filter equal to the size of the image, we will have essentially converted a convolution to a fully connected layer.
Here, we have better weight sharing, thanks to the smaller convolution size that is applied on the complete image.	Here, we have poor weight sharing, due to the larger convolution size.

https://icecreamlabs.com

Smaller filter vs. larger filter example

Smaller Filter Sizes	Larger Filter Sizes
If we apply 3x3 kernel twice to get a final value, we actually used (3x3 + 3x3) weights. So, with smaller kernel sizes, we get lower number of weights and more number of layers.	If we apply 5x5 kernel once, we actually used 25 (5x5) weights. So, with larger kernel sizes, we get a higher number of weights but lower number of layers.
Due to the lower number of weights, this is computationally efficient.	Due to the higher number of weights, this is computationally expensive.
Due to the larger number of layers, it learns complex, more non-linear features.	Due to the lower number of layers, it learns simpler non linear features.`
With more number of layers, it will have to keep each of those layers in the memory to perform backpropogation. This necessitates the need for larger storage memory.	With lower number of layers, it will use less storage memory for backpropogation.

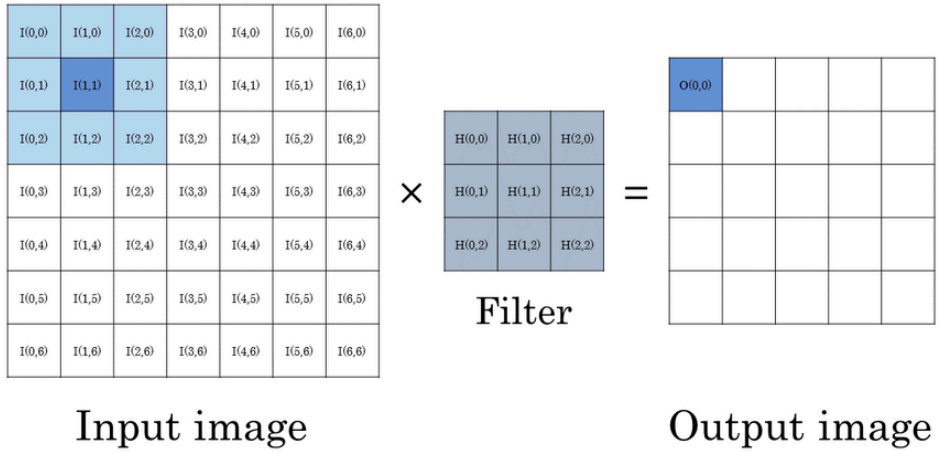
https://icecreamlabs.com

Stride

- A stride value defines how many pixels/steps the filter moves each time it moves across the image
- Very common to use stride = 2
  - Moves 2 pixels at a time

We can apply convolution filters to transform an image without connecting it to an ANN

We call it image convolution



"Streaming Architecture for Large-Scale Quantized NeuralNetworks on an FPGA-Based Dataflow Platform" 2017

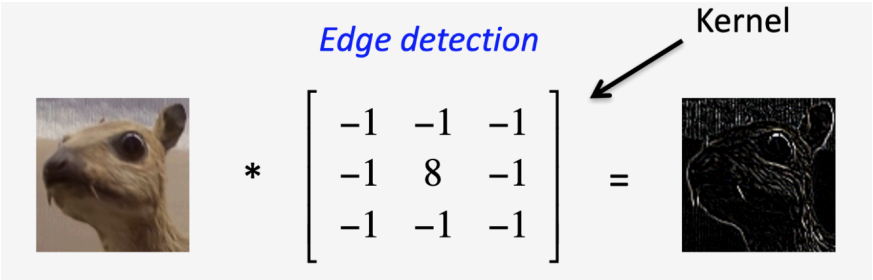
Known image kernels

- Edge detection kernel
- Blur kernel
- Gaussian blur kernel
- Sharpening kernel
- Unsharp kernel

- You can have a custom kernel that fits the best to your data and/or type of learning task

Edge detection kernel

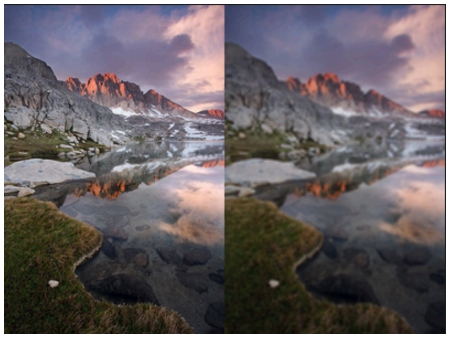
- Detects edges within an image



https://towardsdatascience.com

Blur kernel

- Blurs the image
- Gaussian blur kernel is similar but uses the Gaussian function, which creates a distribution around the center around the center point
  - Pixels near the center contribute more than pixels around the edges of the filter

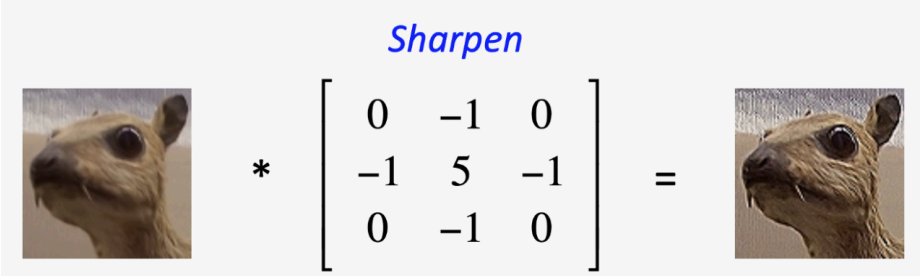


Blur kernel:

0.111	0.111	0.111
0.111	0.111	0.111
0.111	0.111	0.111

Sharpening kernel

- Sharpens the image to bring out the edges by adding contrast to them



https://towardsdatascience.com

Unsharp kernel

- Special technique to sharpen an image
  - First, apply a blur filter to the image
  - Then, subtract the blurred image from the original image

Interesting kernel resource

- https://setosa.io/ev/image-kernels/

Convolutional layer creates features for the ANN

- In convolutional ANNs the first layer is the convolutional layer
  - Once the filter had passed over the image, the convolution layer contains features created by the passage of that filter
- CNNs can have more than one convolutional layer, not necessarily arranged consecutively
- Convolutional layers can take the output of any ANN layer as the input into the convolutional layers
  - The input of a convolutional layer does not need to be the raw data input into the network

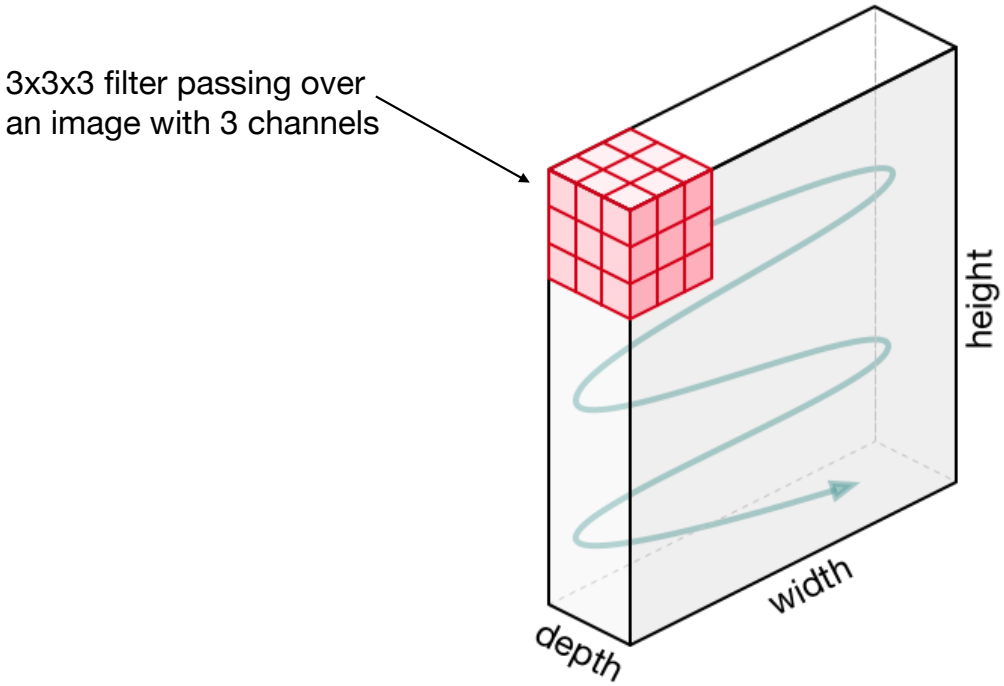
Creating multiple feature maps

- We can apply multiple filters in the same convolutional layer
  - Each filter creates a 2-D feature map
- It's not uncommon in computer vision problems to see hundreds of filters applied in parallel to a single input
  - E.g. if 100 filters are applied to an image then the convolutional layer has a depth of 100
    - 100 feature maps are created
- Diversity of filters allows to better capture specialized feature groups

Processing images with multiple channels

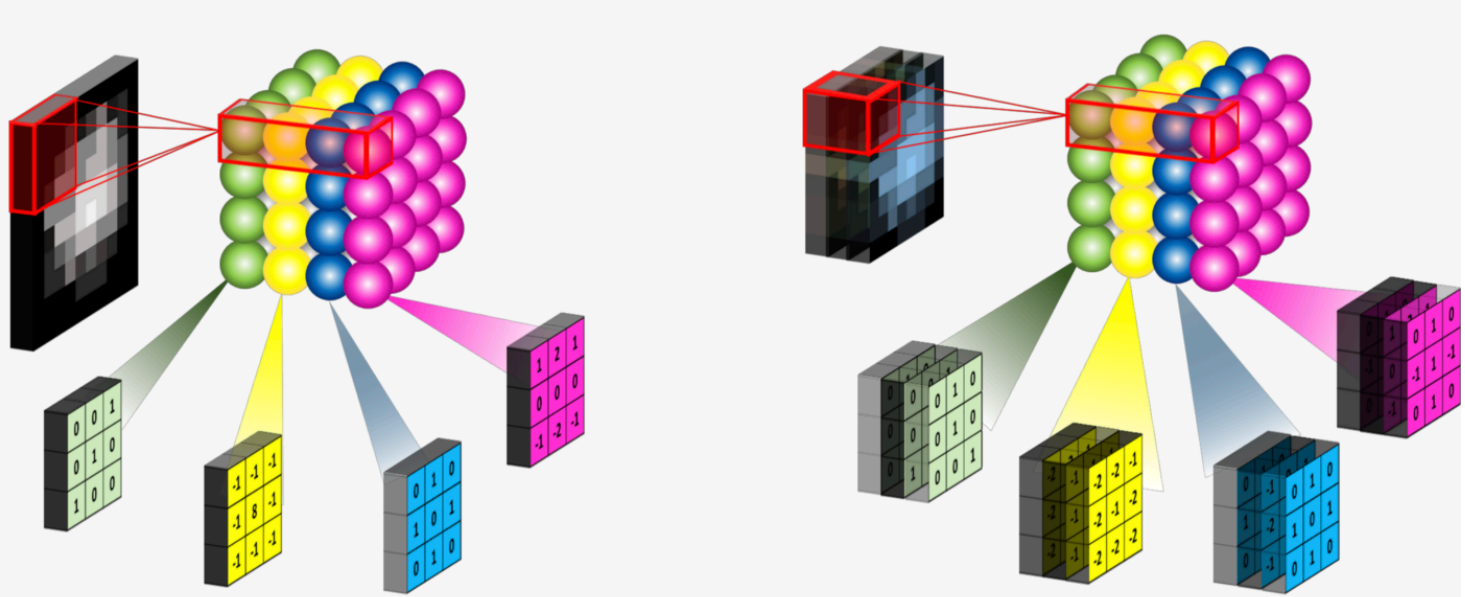
- Each color image typically has multiple channels (RGB)
  - For each single image input we will essentially have three images to process
- A filter must have the same number of channels as the input data
  - We refer to the number of channels as the “depth” of the filter
  - E.g. if the image has 3 channels then the filter applied to that image will have a depth = 3
    - If this is a 3x3 filter then it will be 3x3x3 filter
  - A filter with depth > 1 still produces a single mapped feature value, as if the filter only has 1 channel

Processing images with multiple channels (cont'd)



https://towardsdatascience.com

Processing images with multiple channels (cont'd)

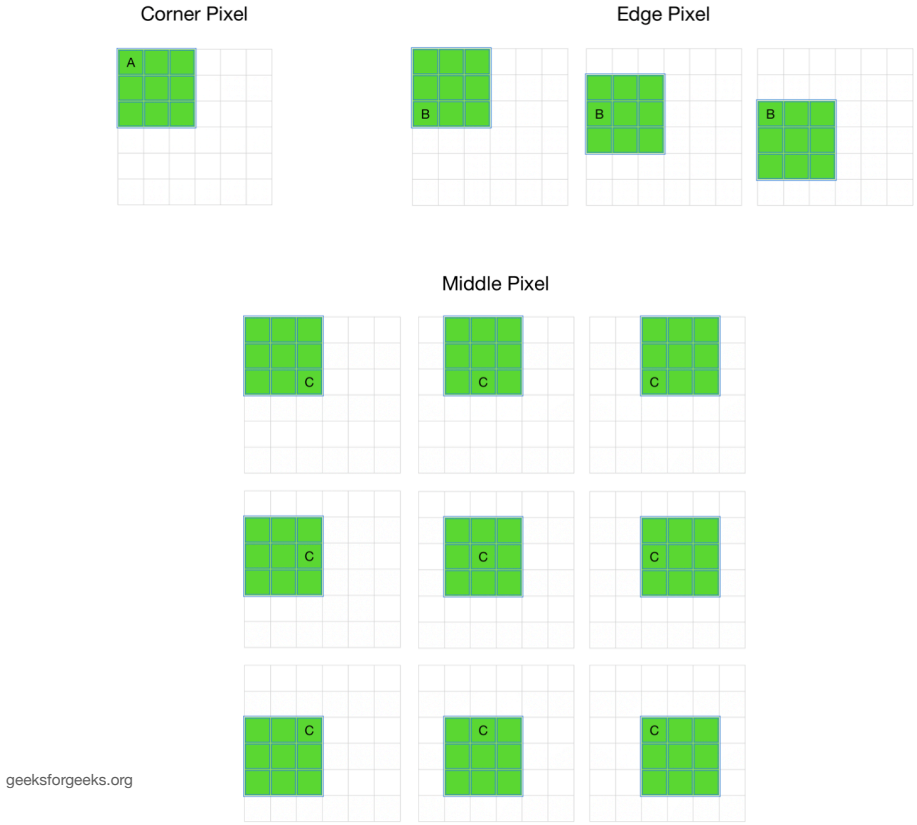


Convolutional layer with four 3x3 filters on a black and white image (just one channel)

Convolutional layer with four 3x3 filters on an RGB image. As you can see, the filters are now cubes, and they are applied on the full depth of the image..

https://towardsdatascience.com

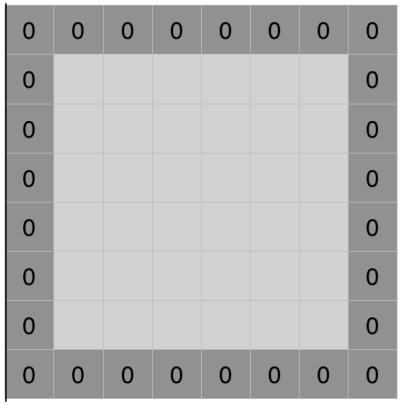
Problem with filters not passing the same number of times over the edge pixels of an image



Here pixel A is only touched 1 time and pixel B is only touched 3 times, while pixel C is touched 9 times - different contribution of different pixels

Dealing with pixels on the edge of the picture frame

- Losing the edge pixels' contributions
- Padding the edges with zeros
  - Adding pixels with zero values around the edge of the image
- Reflection padding
  - Copying the pixels from the edge of the image



geeksforgeeks.org

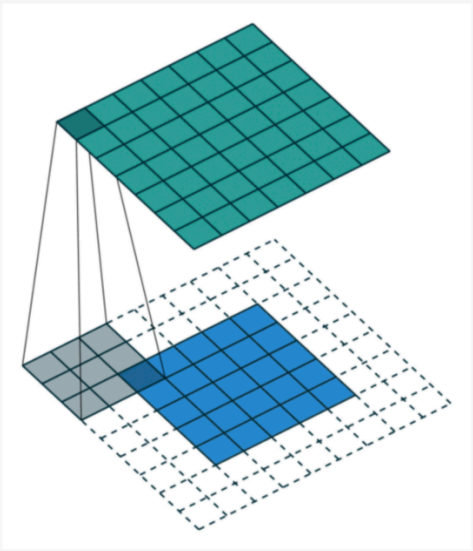
Zero-padding added to image

So what is the size of the convolutional layer?

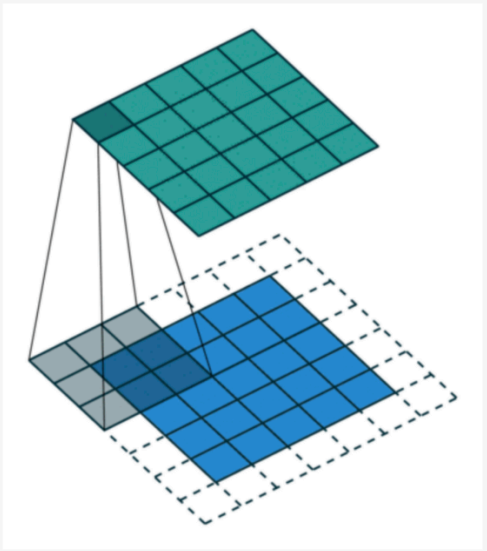
- Let’s say we have a color image 64x64 (3 channels for RGB)
- Let’s say we use 128 3x3x3 filters/kernels
- Stride = 2

- The convolutional layer will be 128x32x32
  - 128 feature maps in the convolutional layer
  - Each feature map is 32x32

The choice of padding affects the size of the output of the convolutional layer



Full padding. Introduces zeros such that all pixels are visited the same amount of times by the filter. Increases size of output.



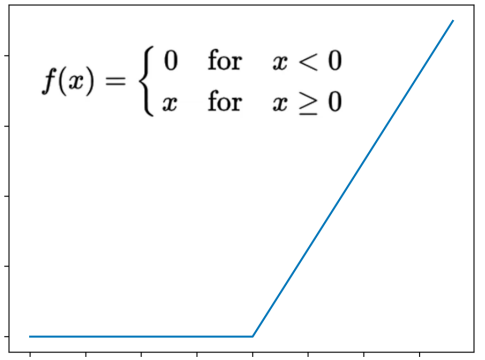
Same padding. Ensures that the output has the same size as the input.

https://towardsdatascience.com

Activation maps

- Often the output of the convolutional layer is put through an activation function
  - ReLU is most commonly used activation function
- Activation functions introduce non-linearity into convolutional layers of CNNs
- ReLU inactivates convolution cells with negative values
- The output of the convolutional layer after an activation function is applied is called an “activation map”

ReLU (Rectifier activation function)



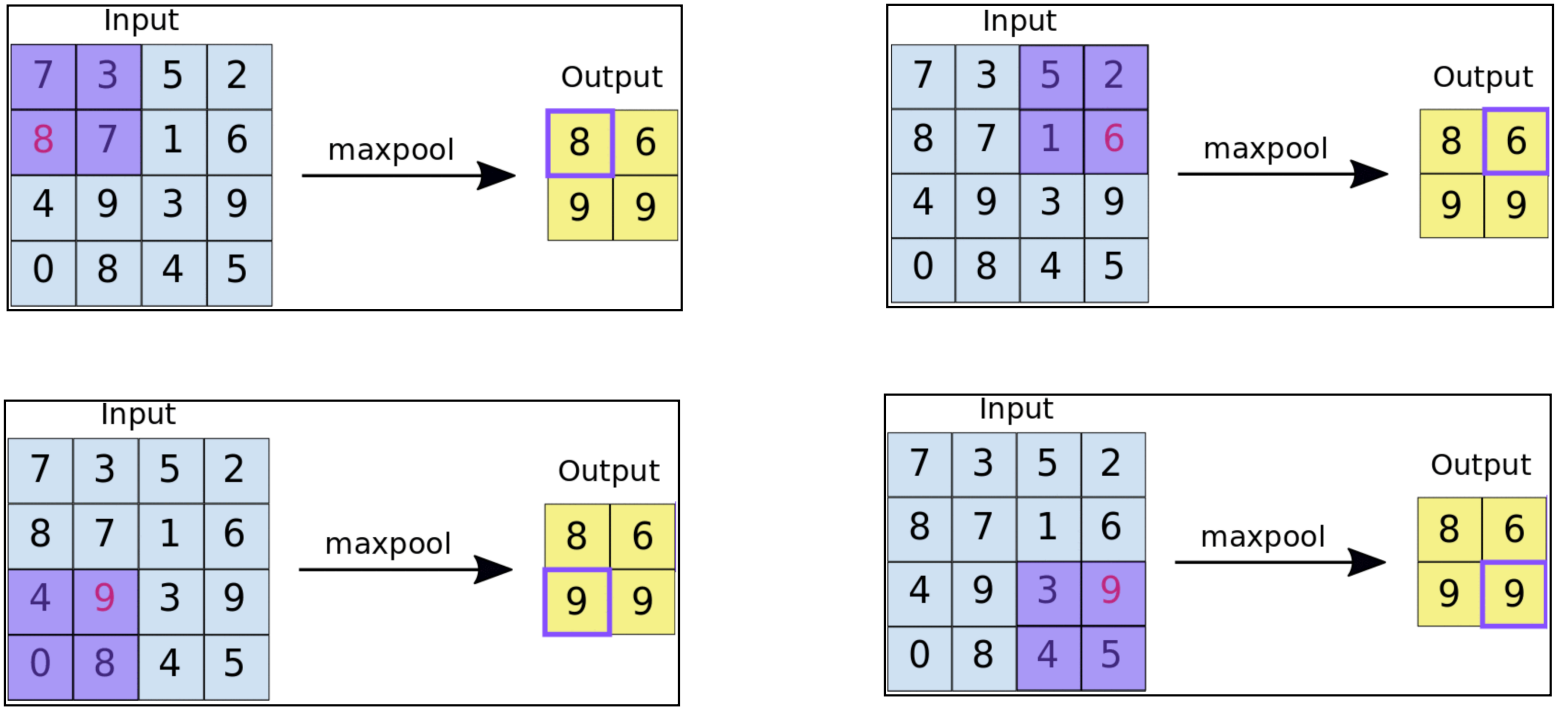
Pooling layer

- After the convolutional layer (with or without applying an activation function) we may have a pooling layer
- A pooling layer downsamples the output of a convolutional layer
  - Saves processing time
  - Speeds up network training
- Introduces additional non-linearity into the model
- Subsampling occurs in sliding windows (similar to convolution)
  - Just like in the convolutional layer the stride parameter specifies how many steps the window moves each time

Types of pooling

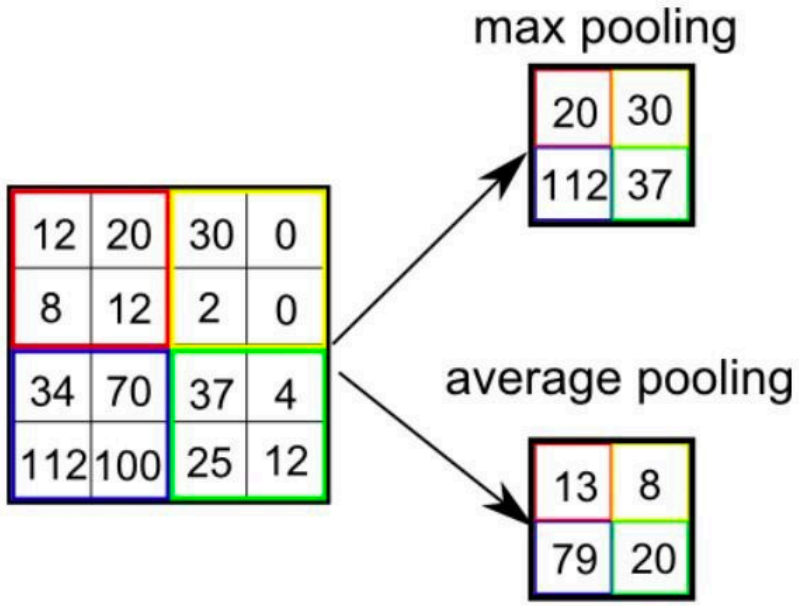
- Max-pooling
  - Obtains the max value of all the values in the sliding window
- Min-pooling
  - Obtains the min value of all the values in the sliding window
- Average-pooling
  - Averages all the values in the sliding window

Pooling layer example (max pooling)



kdnuggets.com

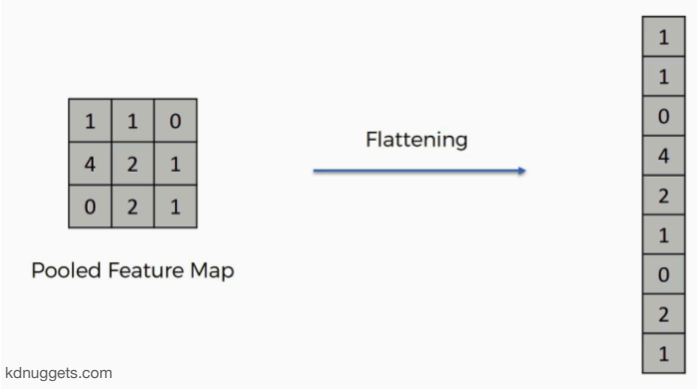
Max pooling vs. average pooling



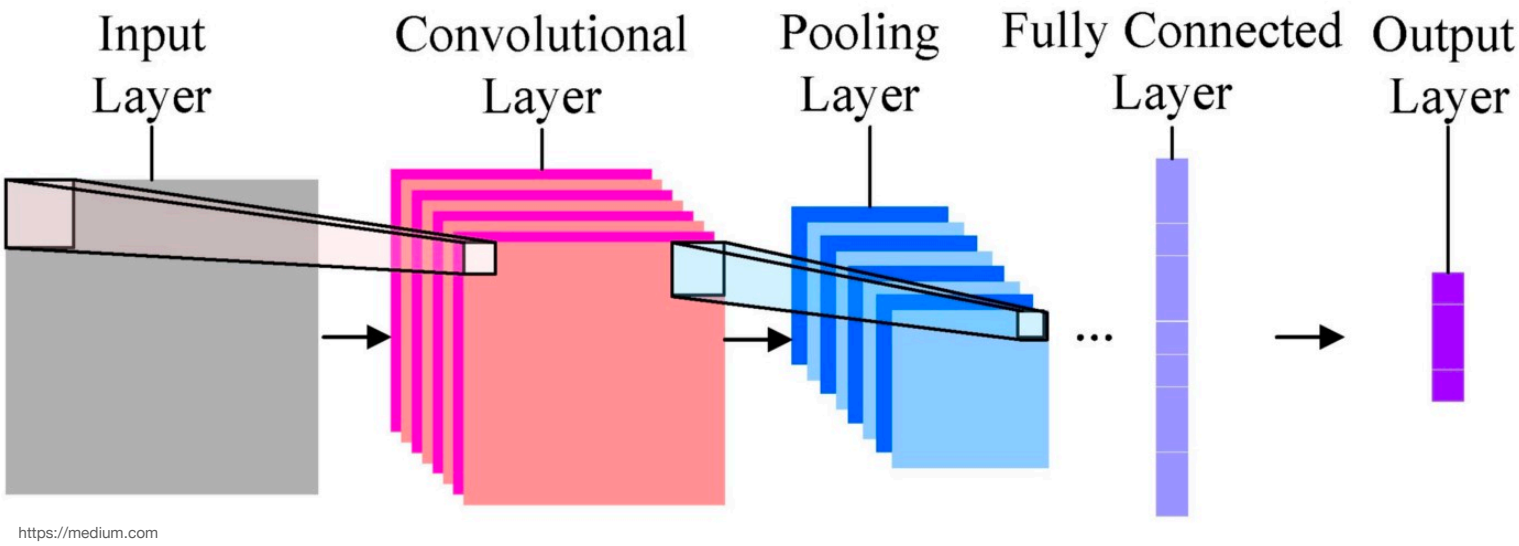
https://medium.com

Flattening layer

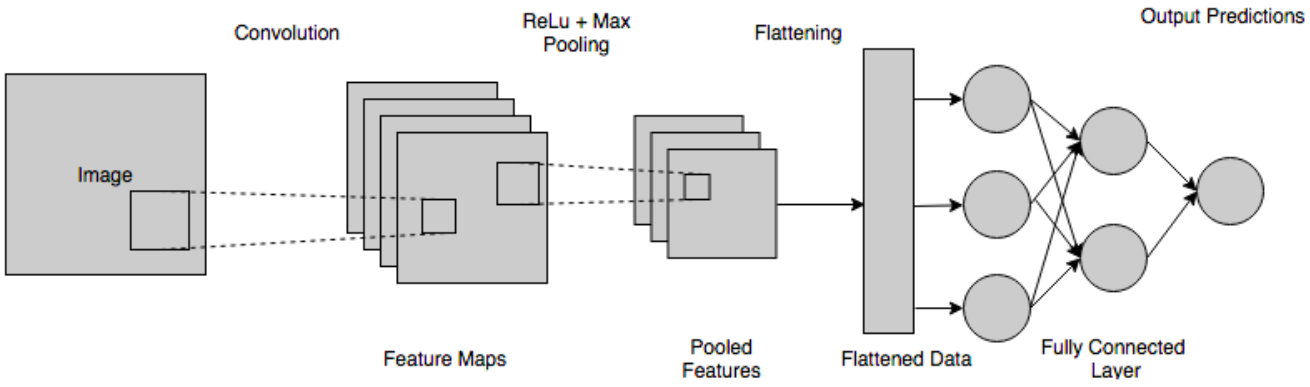
- Usually multiple convolutional layers are applied to the input data
- After a number of convolutional layers, the 3-D representation of the input data is converted into a feature vector, which is passed into a dense neural network (MLP)



Let’s put it all together (simplified diagram)



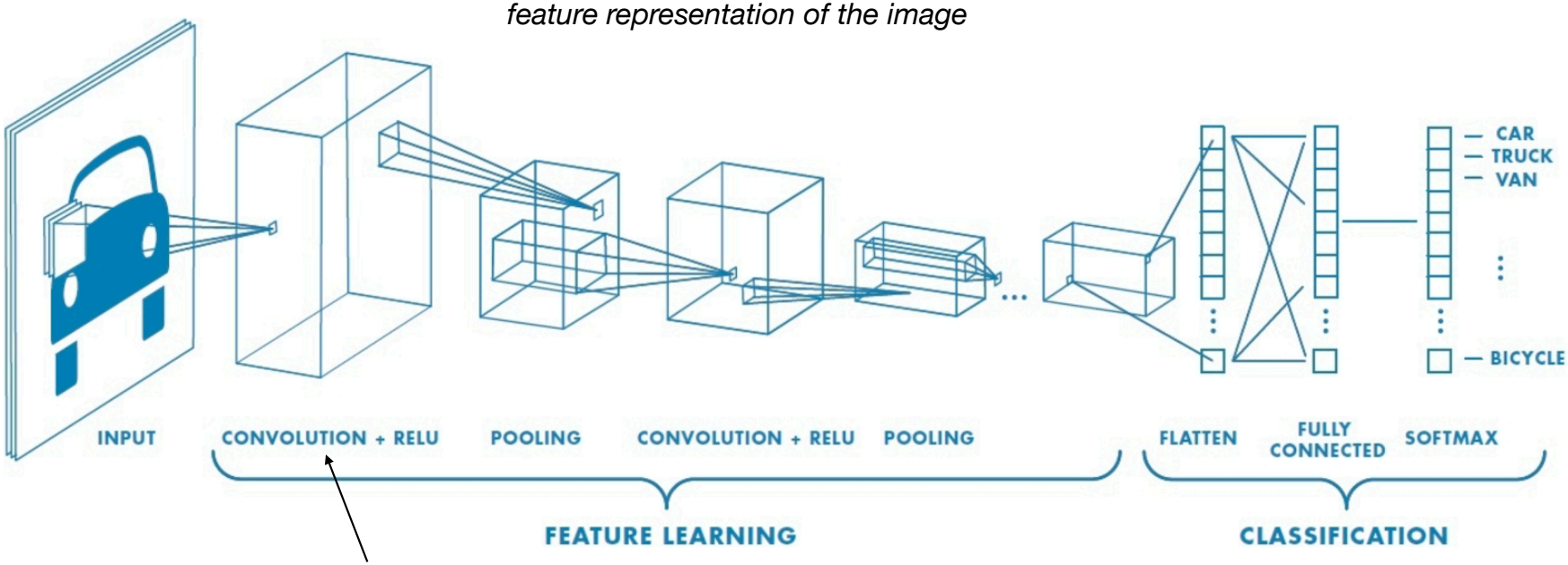
Let’s put it all together (another simple example)



https://medium.com

Let’s put it all together (example of a realistic CNN architecture)

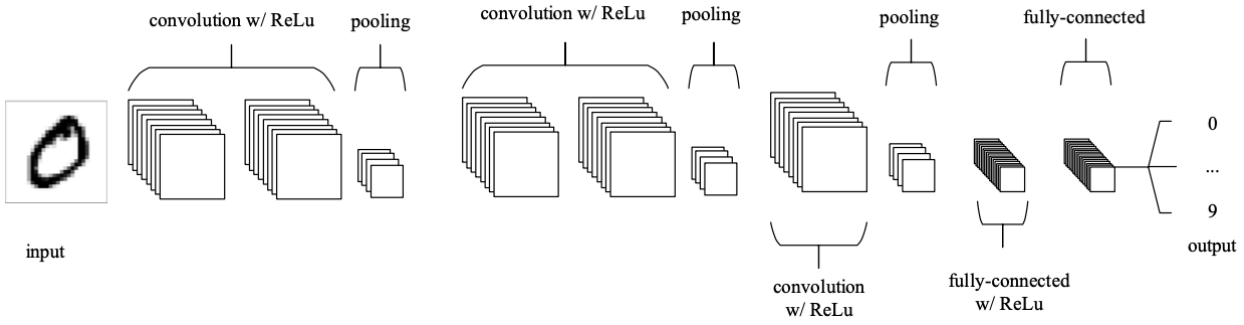
A sequence of convolutional/pooling layers is meant to learn a non-linear feature representation of the image



kdnuggets.com

Usually activation map (ReLU) is not depicted as a separate layer

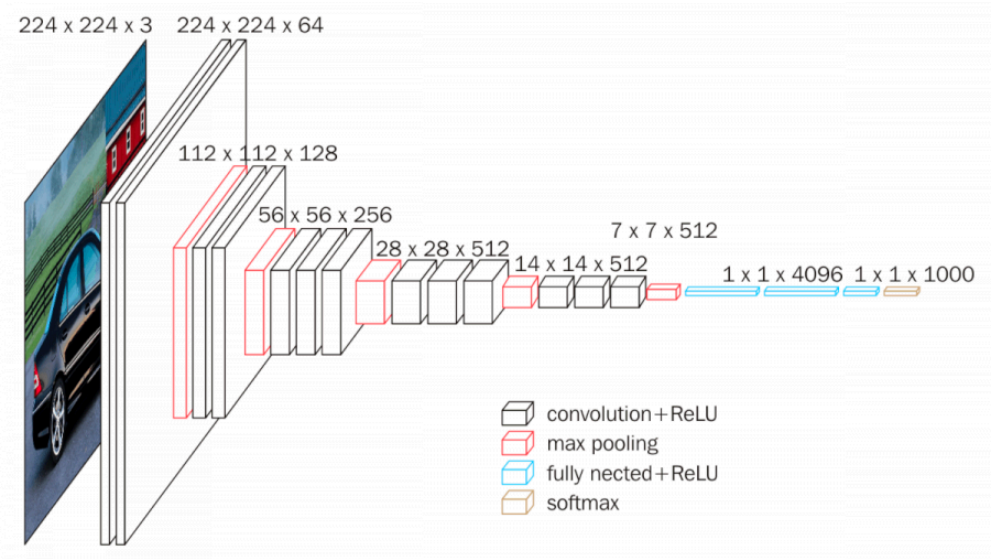
Convolutional layers are often stacked together



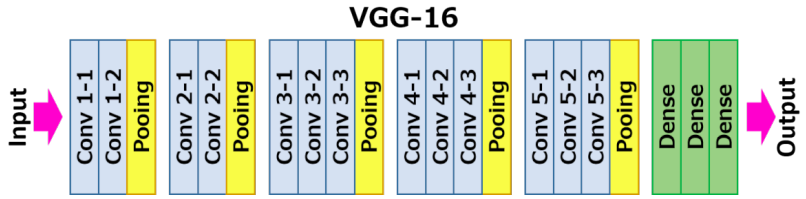
"An Introduction to Convolutional Neural Networks", O'Shea & Nash, 2015

Convolutional layers of VGG16 CNN

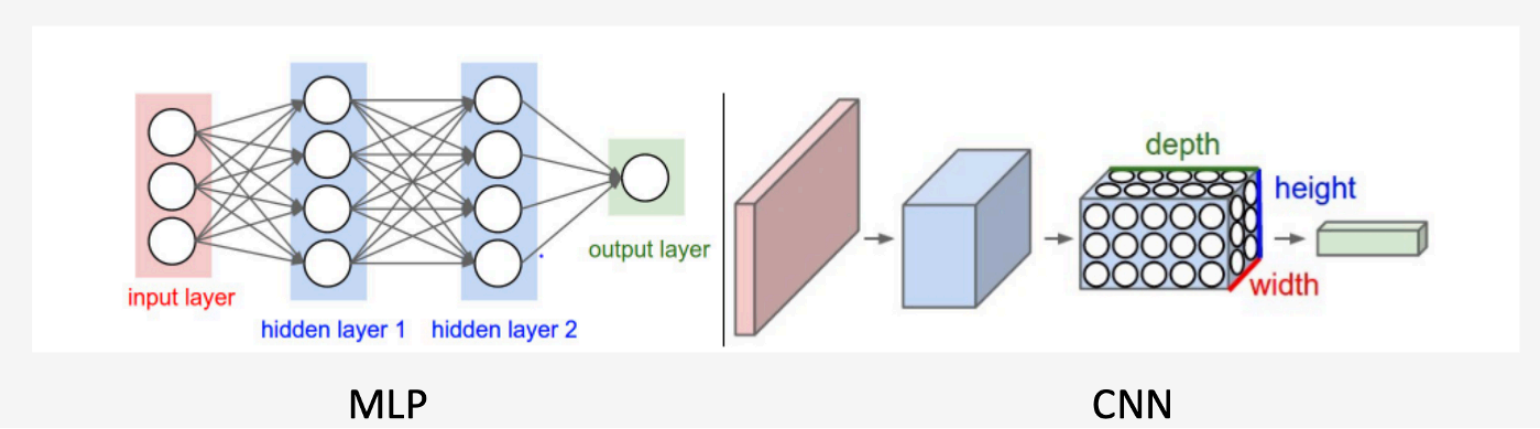
- VGG16 is a CNN proposed by K. Simonyan and A. Zisserman in their 2014 University of Oxford paper “Very Deep Convolutional Networks for Large-Scale Image Recognition” (https://arxiv.org/abs/1409.1556)
  - Image classification model
  - Originally submitted for Large Scale Visual Recognition Challenge 2014 (ILSVRC2014)



https://neurohive.io/en/popular-networks/vgg16/



MLP vs. CNN



https://towardsdatascience.com

Let’s visualize a CNN working in practice

- https://www.cs.ryerson.ca/~aharley/vis/conv/

Regularization in ANNs

- Ridge (L2)
- Lasso (L1)
- Dropout
- Batch normalization
- Gradient scaling/clipping

- Regularization prevents overfitting

L2/L1 regularization

- Ridge (L2) and Lasso (L1) regularization can be applied to the loss/cost function

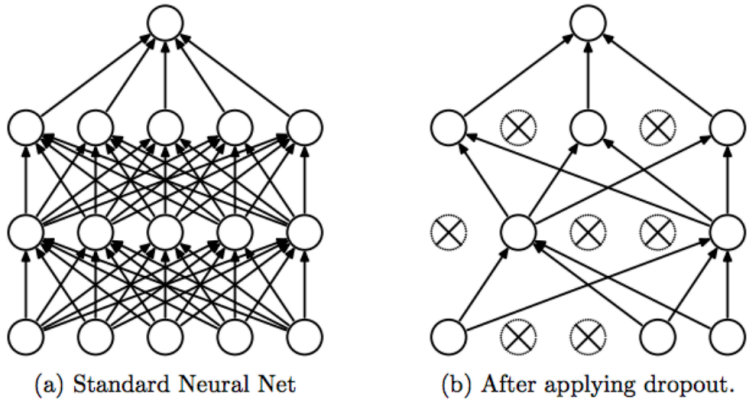
$$\ell_2 : \Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{w}||_2^2 = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\ell_1 : \Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1 = \sum_{\forall i}|w_i|$$

Dropout

- Helps reducing correlation between neuron contributions

- <u>Steps:</u>
  - During training randomly ignore activations by some neurons (with probability *p*)
  - During testing use all activations but scale/reduce them by *p*



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

https://medium.com

Batch normalization

- Batch normalization refers to the step of rescaling all activations in a given ANN layer
  - Map all activations into the common space
  - Comparability between activations of different neurons
  - Consistency of input data into the next layer
  - Reduces variability of outputs in the hidden layers
  - Maintains independence between hidden layers

Gradient scaling/clipping

- Large updates to gradient during training of an ANN can cause numerical overflow (“NaN” or “Inf”)
  - Referred to as “exploding gradient”
  - Can be caused by the choice of learning rate, loss function, data preprocessing and normalization
- Gradient clipping refers to various techniques that prevent the gradient from getting too large
  - Rescaling gradients by some chosen vector norm (gradient scaling)
  - Forcing the gradient values to be within a certain range (gradient clipping)

Let’s look at some image convolution code

- Image convolution (Lena)
  - *Intro_to_image_convolution.ipynb*

- Don’t worry, we will look at implementing classification with convolutional neural networks after we go through introduction to tensorflow and keras