Convolutional Neural Networks

Fundamental Concepts & Hands-on Examples

Slide Deck and Code

http://bit.ly/cnn_ds_dphi20

Session Agenda

Introduction

Understanding Convolutional Neural Networks

CNN Layer Operations

Transfer Learning Brief

Hands-on Examples

CNN Applications


Introduction

About Me

Dipanjan Sarkar

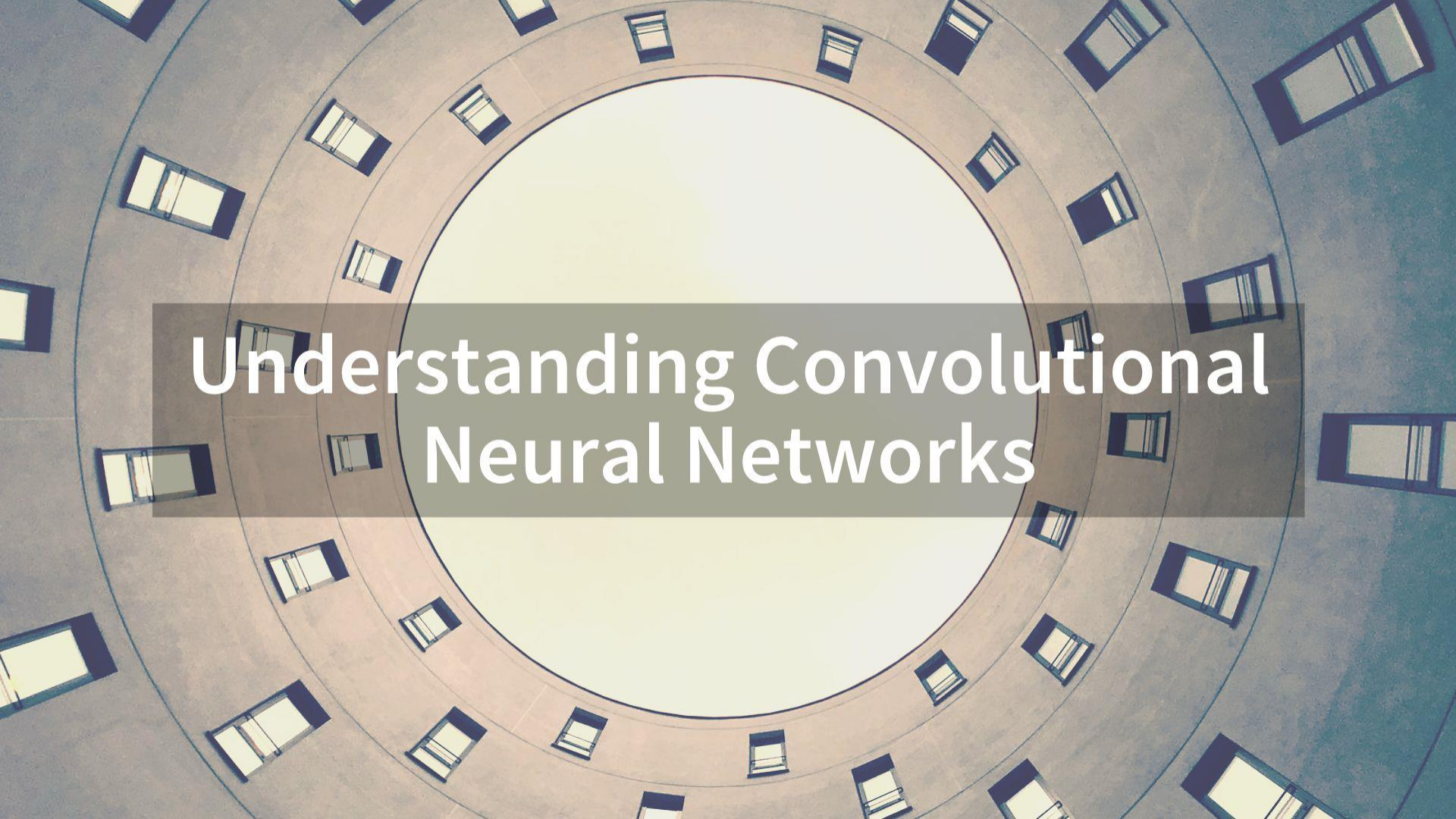
Data Science Lead, Author, Google Developer Expert - ML

APPLIED MATERIALS®

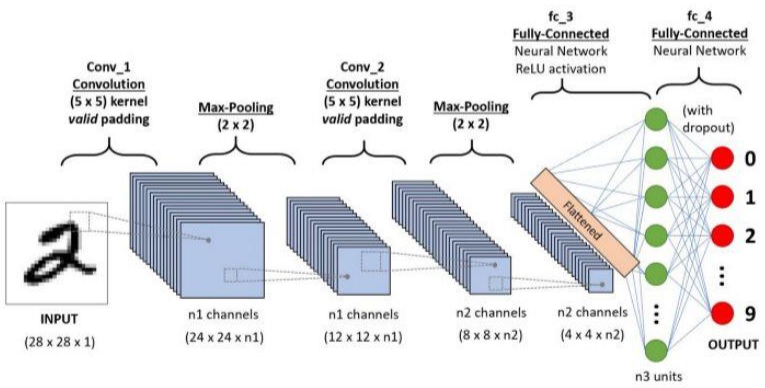
Springboard

Experts

Machine Learning

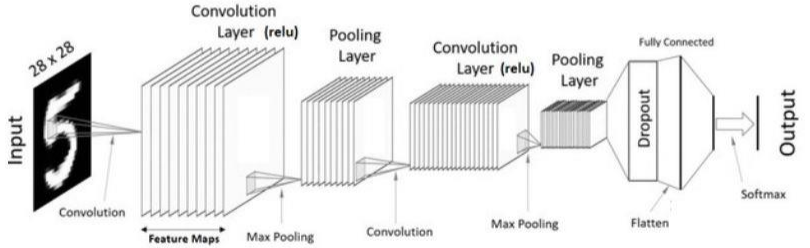

Understanding Convolutional
Neural Networks

Convolutional Neural Networks (CNNs)



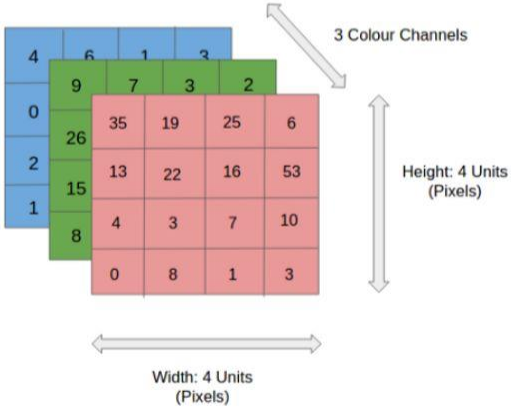
- CNNs have a layered architecture of several layers to learn hierarchical spatio-temporal features
- Convolution Layers use convolution filters to build feature maps (feature extraction)
- Pooling Layers help in reducing dimensionality after convolutions (compression)
- Non-linear activation functions are applied in the network as usual
- Dropout or BatchNormalization Layers may be used to prevent model overfitting
- FC Layers in final stages help with flattening and prediction

CNN - Main Layer Components

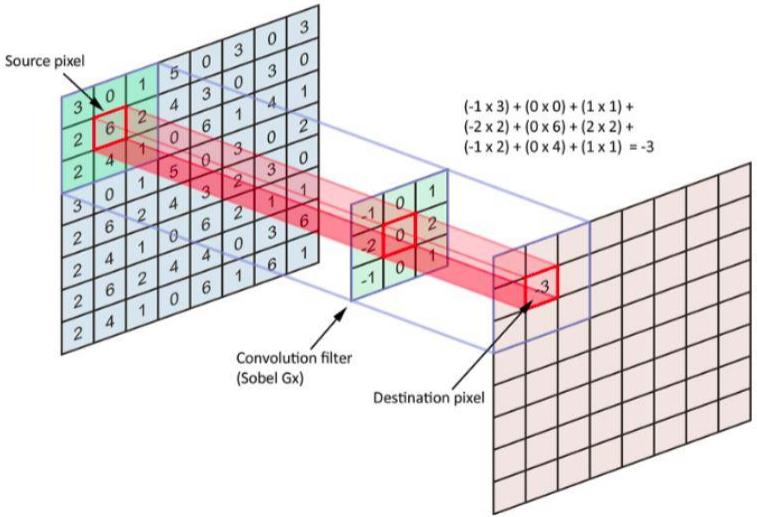


- **CNNs have a stacked layered architecture of several convolution and pooling layers**
- **Convolution layer**
  - Consists of several filters or kernels
  - Passed over the entire image in patches and computes a dot product
  - Result is summed up into one number per operation (dot product)
- **Pooling layer**
  - Downsamples feature maps from conv layers
  - Typically max-pooling is used which selects the max-pixel value out of a patch of pixels
- **Activation layer**
  - Feature maps \ pooled outputs are sent through non-linear activations
  - Introduces non-linearity and helps train via. backpropagation

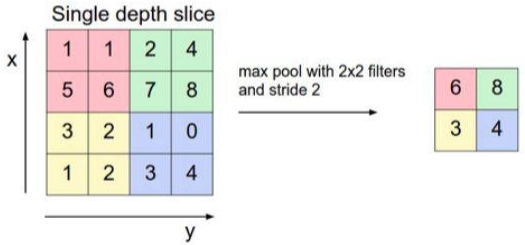
CNN - Conv - Pool Operations



Source Image

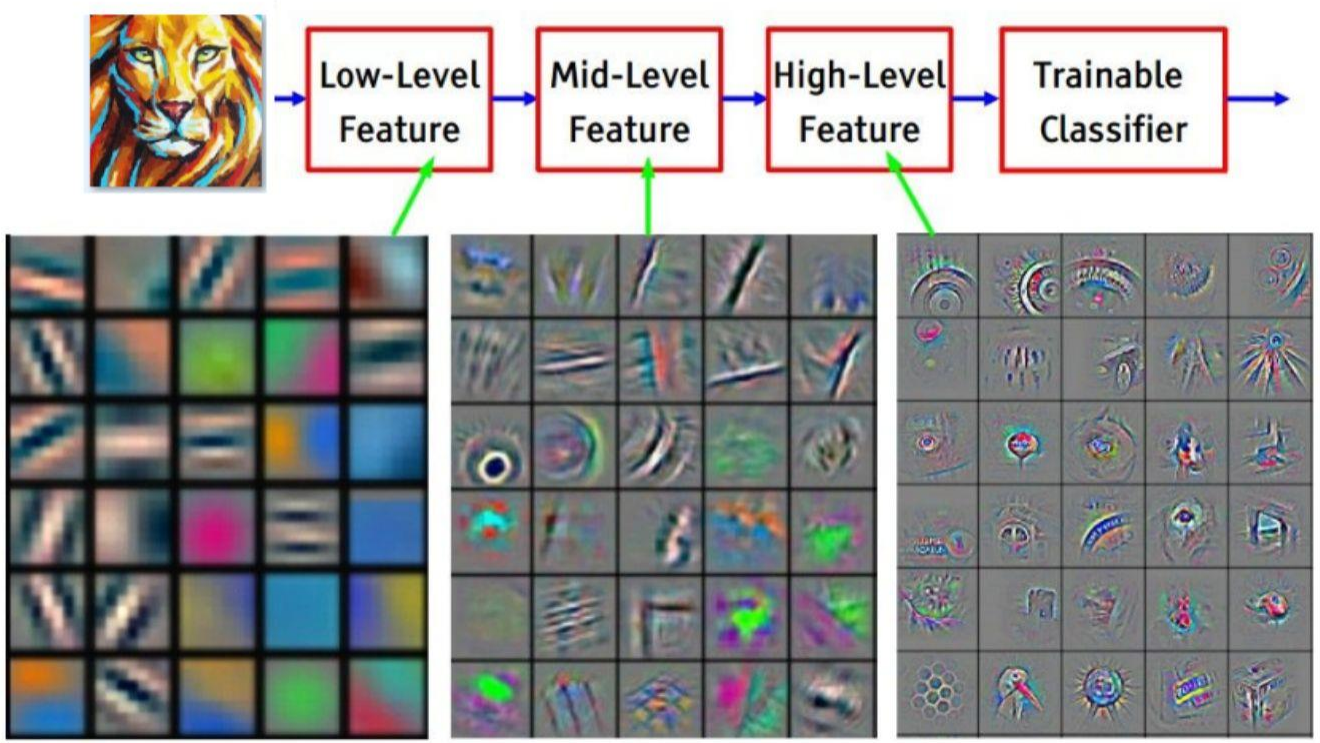


Convolution Layer



Pooling Layer

CNN - Why several layers & filters?



CNN Layer Operations

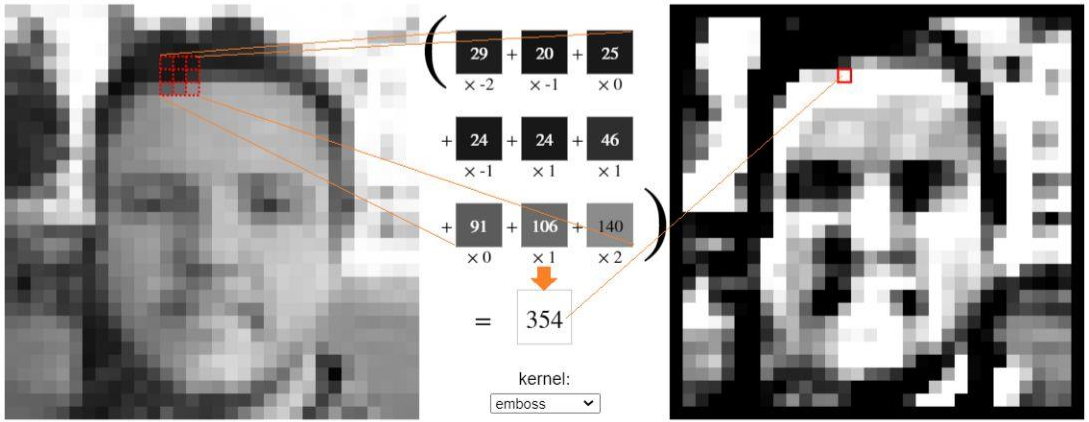
Convolution Layer - What is a convolution operation?

emboss

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

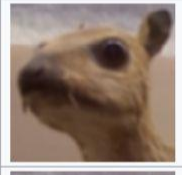
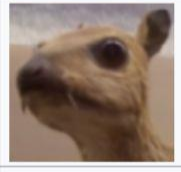
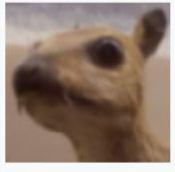
$$\begin{pmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right.



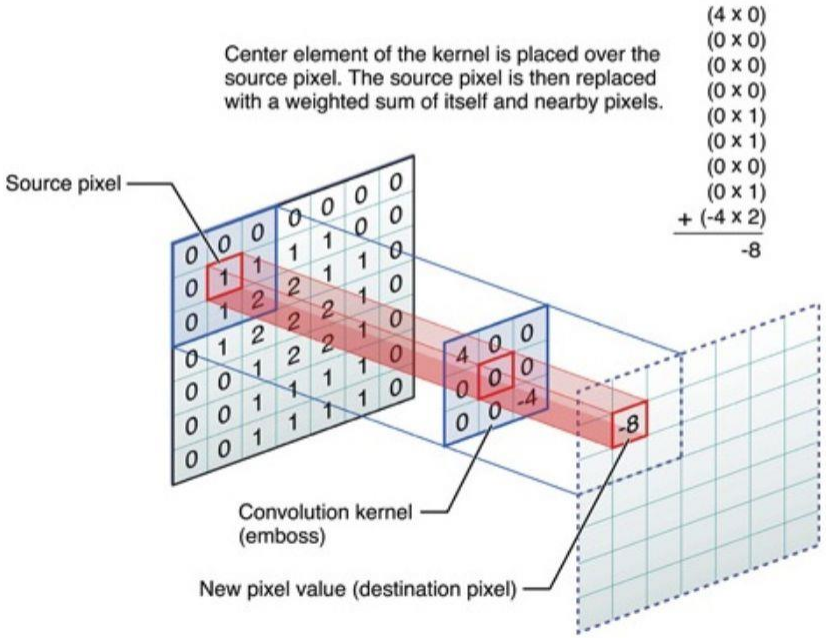
Convolution Layer - Types of Kernels

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Operation	Kernel ω	Image result g(x,y)
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Types of Kernels: https://en.wikipedia.org/wiki/Kernel_(image_processing)

Convolution Layer - Key Aspects



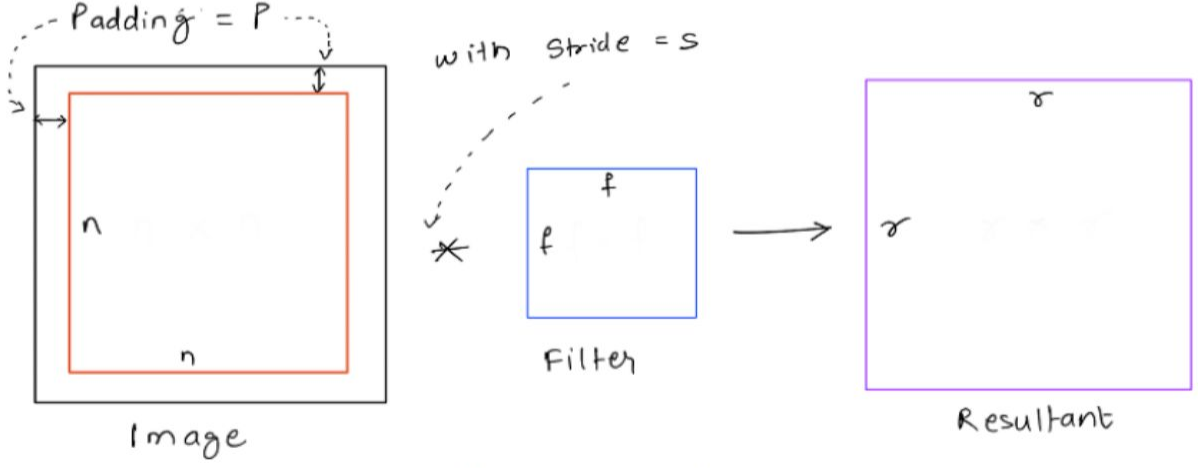
- **Kernel is also known as a filter or a feature detector**
  - Can be 3x3, 5x5 or higher filled with random or specific values based on type of filter
  - Wide variety of filters help in detecting a wide variety of features like edges, corners, textures, patterns etc.
  - Conv layers usually have multiple filters to extract multiple feature maps
- **Feature map is the output of the convolution operation**
  - Conv2d(input image, filter) gives us the convolved features
  - Elementwise multiplication of the two matrices
  - Works on the image patch-by-patch
  - Kernel size is the size of the filter applied on the image patch
  - Stride is the number of pixels to move horizontally and vertically
- **Activation layer**
  - Feature maps are sent through non-linear activations
  - Introduces non-linearity and helps train via. backpropagation

Convolution Layer - Key Aspects

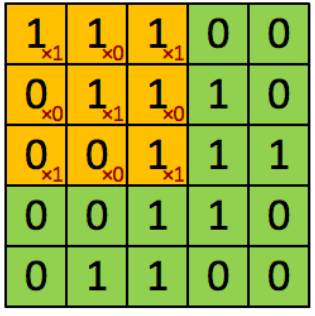
```
tf.keras.layers.Conv2D(
    filters, kernel_size, strides=(1, 1), padding='valid', data_format=None,
    dilation_rate=(1, 1), groups=1, activation=None, use_bias=True,
    kernel_initializer='glorot_uniform', bias_initializer='zeros',
    kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None,
    kernel_constraint=None, bias_constraint=None, **kwargs
)
```

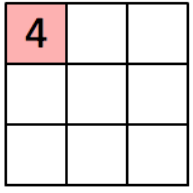
- Filters: Number of filters \ kernels to use
- Kernel Size: Size of the filter \ kernel e.g 3x3
- Activation: Type of non-linear activation to use e.g, relu
- Stride: Number of pixels to move horizontally & vertically after each convolution operation
- Padding: Valid or Same
  - **Valid:** reduces the size of the image after convolving due to loss of the edges
  - **Same:** pads extra pixels around the image to retain original image dimensions after convolution

Simple Convolution - Operations on 1D Tensors

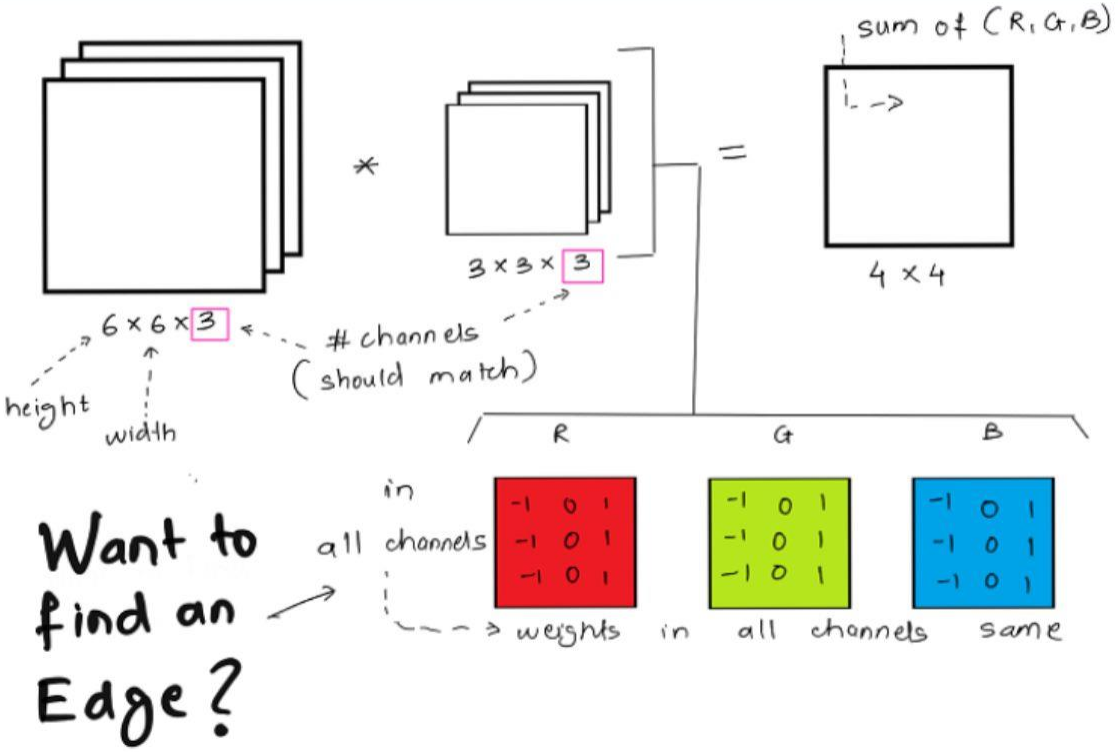


Source: https://towardsdatascience.com/cnn-part-i-9ec412a14cb1



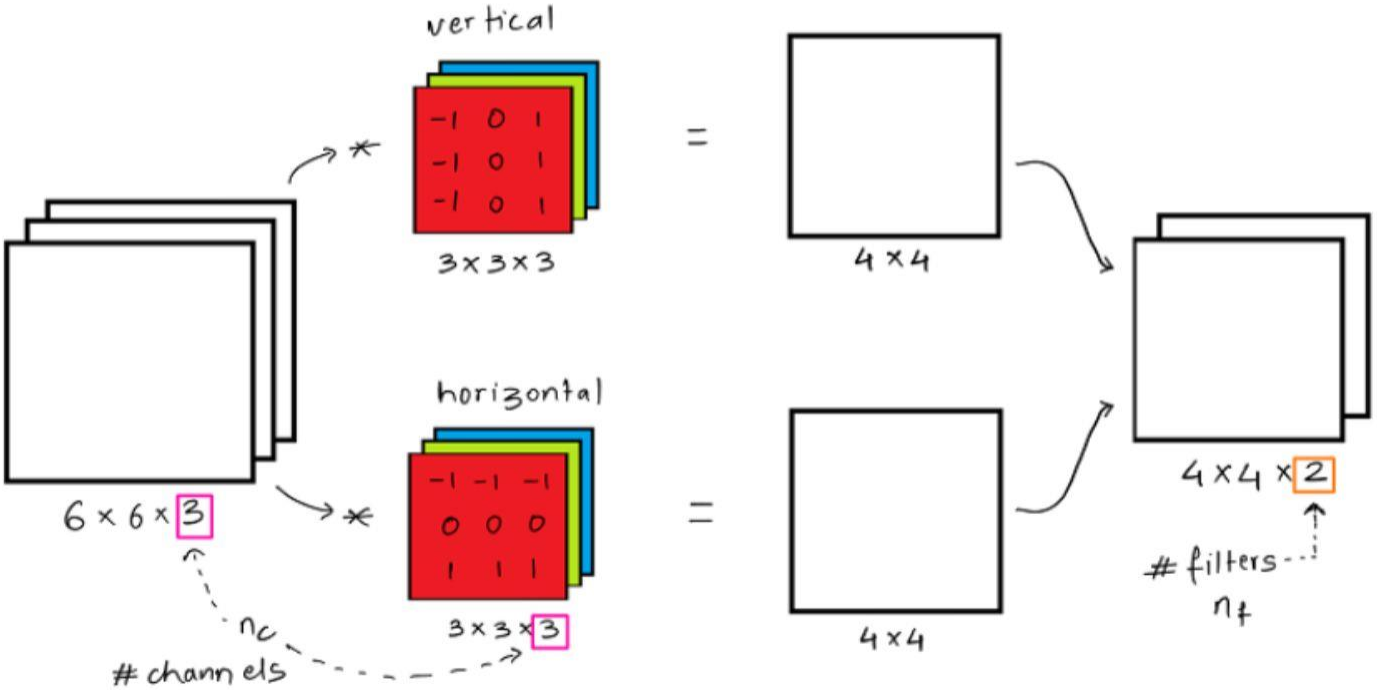


Volume Convolution - Operations on 3D Tensors



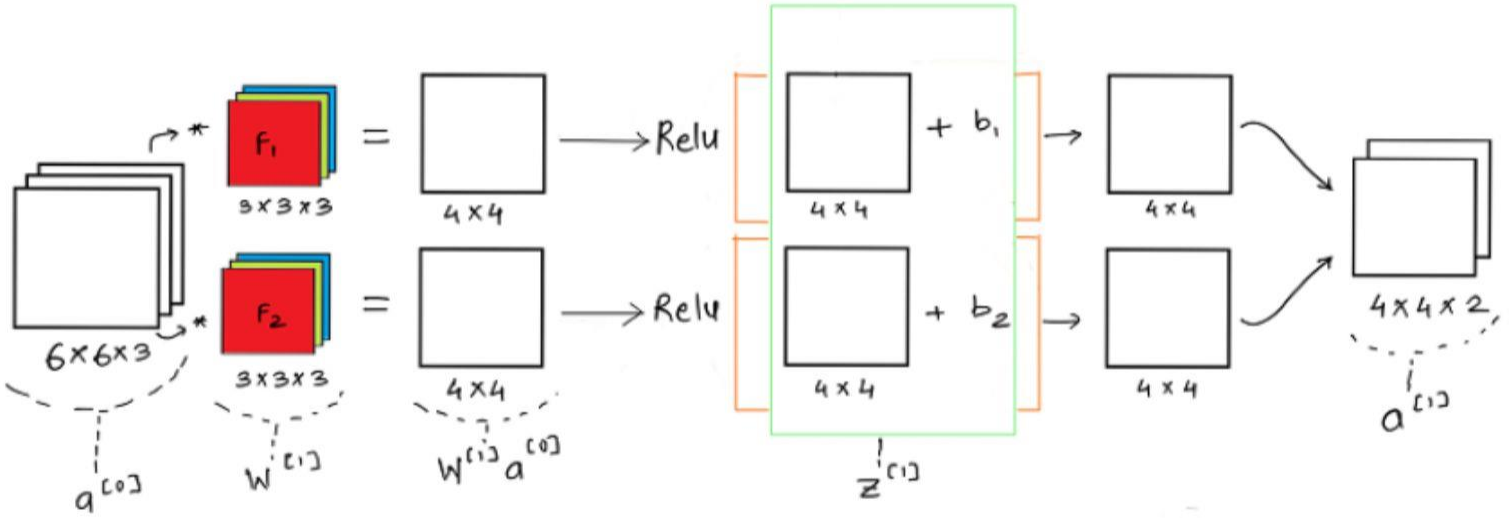
Source: https://towardsdatascience.com/convolutional-neural-network-ii-a11303f807dc

Volume Convolution - Multiple Filters



Source: https://towardsdatascience.com/convolutional-neural-network-ii-a11303f807dc

Convolution Layer - Operations Flow



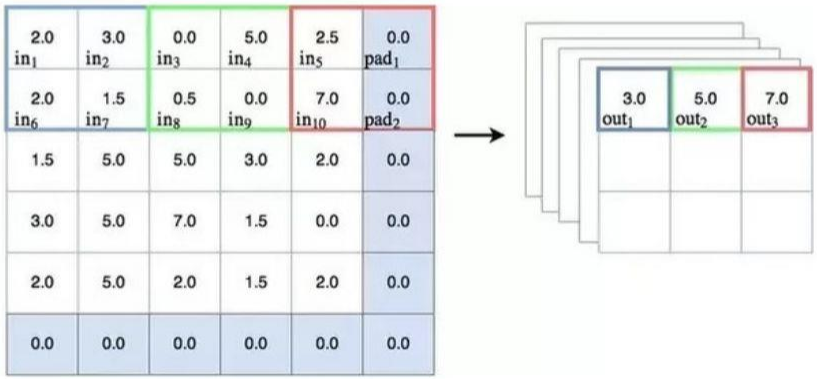
In any given neural network $z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$

$a^{[1]} = g(z^{[1]})$

Relu function

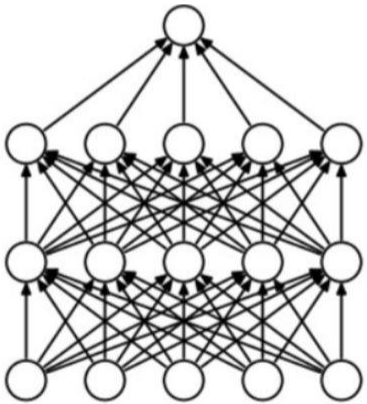
Source: https://towardsdatascience.com/convolutional-neural-network-ii-a11303f807dc

Pooling Layer - Key Aspects

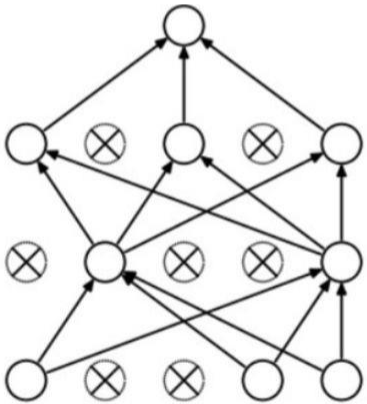


- Helps in reducing the number of parameters by down-sampling
- Reduction in parameters helps in faster training and prevent overfitting
- Various types of pooling are available like sum, mean, max
  - Max pooling is preferred as it is faster and usually gives a better performance than mean or sum pooling
  - Helps enhance key aspects of feature maps
- Pool size is the number of image pixels to consider for downsampling e.g, 2x2

Dropout Layer - Key Aspects



Neural Net



applying dropout

- Technique where randomly selected neurons are ignored during training
- Helps in preventing overfitting as a regularization mechanism

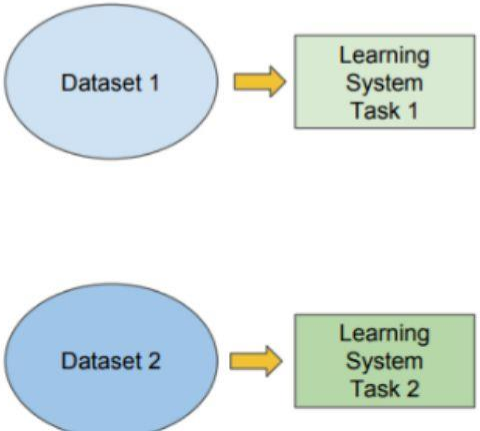

CNN Layer Operations
Hands-on Examples


Transfer Learning Brief

Traditional ML vs. Transfer Learning

Traditional ML

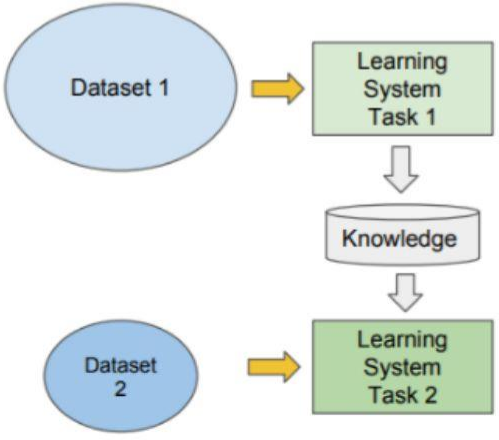
- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



vs

Transfer Learning

- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data



The power of Transfer Learning

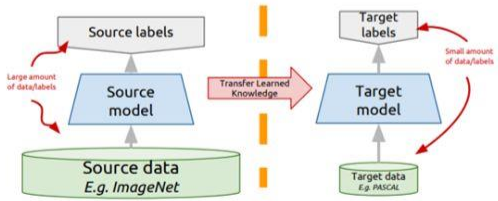
Transfer learning: idea

Instead of training a deep network from scratch for your task:

- Take a network trained on a different domain for a different **source task**
- Adapt it for your domain and your **target task**

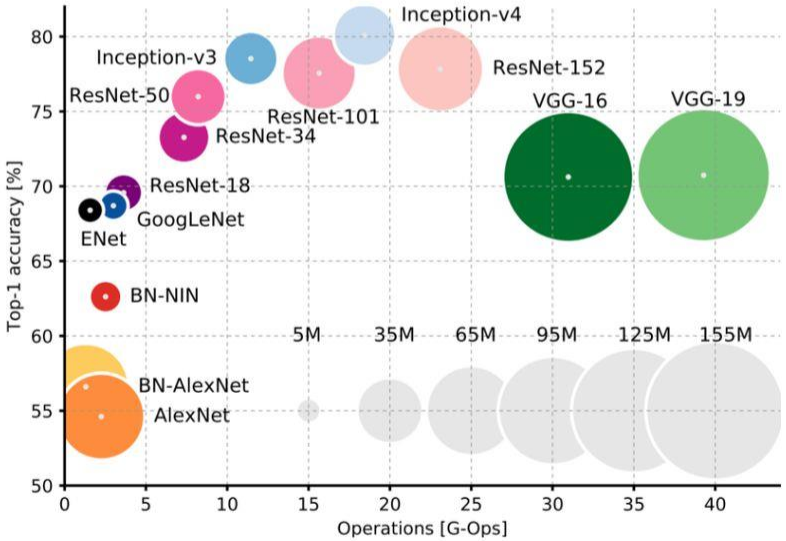
Variations:

- Same domain, different task
- Different domain, same task



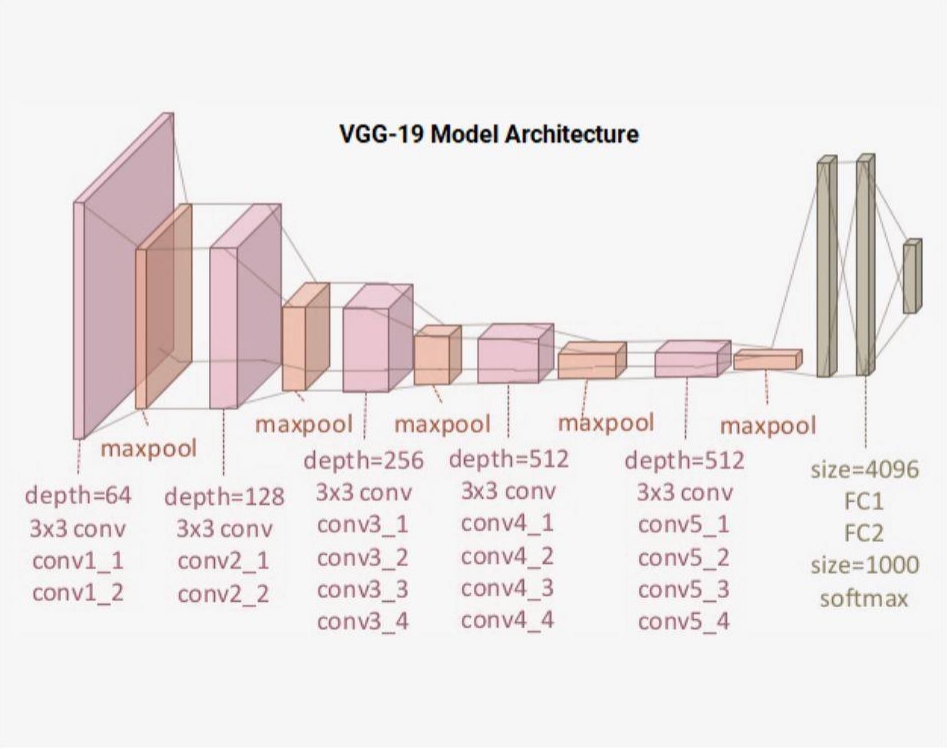
- **Leverage a pre-trained deep learning model (which was trained on a large dataset — like ImageNet)**
- **Solve the problem of apparel classification in Fashion-MNIST by applying and transferring its knowledge in the context of our problem**
- **Two approaches**
  - Frozen pre-trained model as a feature extractor
  - Fine-tuning pre-trained model on our data

Transfer Learning - Advantage of Pre-trained Models



- Training DL models from scratch can take a lot of time
- Pre-trained models have been trained on a lot of data and specialized for performing specific tasks
- Can be used for both feature extraction and inference
- Models are available for computer vision and natural language processing tasks

Pre-trained CNN - VGG-19



- VGG-19 model is a 19-layer (convolution and fully connected) deep learning network built on the ImageNet database
- 16 convolution layers using 3 x 3convolution filters along with max pooling layers
- 2 fully connected hidden layers of 4096 units in each layer followed by a dense layer of 1000 units
- We focus on intermediate layers for extracting the right representations


CNN Classifiers
Hands-on Examples


CNN Applications

CNN Applications

1. **Classification & Regression**
   Used extensively in computer vision problems with image, video. Can also be used for audio and text

2. **Object Detection**
   Can be used to detect objects in images and video

3. **Image Segmentation**
   Useful for segmenting key entities in an image

4. **Speech Processing & Modeling**
   Can be used for processing and modeling on speech e.g speech recognition

5. **Style Transfer**
   Generative learning to transfer the style of an image on the content of an image

6. **Search & Retrieval**
   Learns efficient feature representations which can be used for similarity analysis & search

7. **Image Super Resolution**
   Improves the resolution of a base image with poor resolution

8. **Image Captioning**
   CNN coupled with a generative language model can be used for image caption generation

References

- **Visuals & Content**
  - http://cs231n.stanford.edu/
  - https://www.youtube.com/playlist?list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF
  - https://www.kaggle.com/kanncaa1/convolutional-neural-network-cnn-tutorial
  - https://setosa.io/ev/image-kernels/
  - https://en.wikipedia.org/wiki/Kernel_(image_processing)
  - https://aishack.in/tutorials/image-convolution-examples/
  - https://towardsdatascience.com/cnn-part-i-9ec412a14cb1
  - https://towardsdatascience.com/convolutional-neural-network-ii-a11303f807dc
  - https://www.coursera.org/learn/introduction-tensorflow

Stay in Touch!

LinkedIn

https://www.linkedin.com/in/dipanzan

GitHub

https://github.com/dipanjanS

Medium

https://medium.com/@dipanzan.sarkar