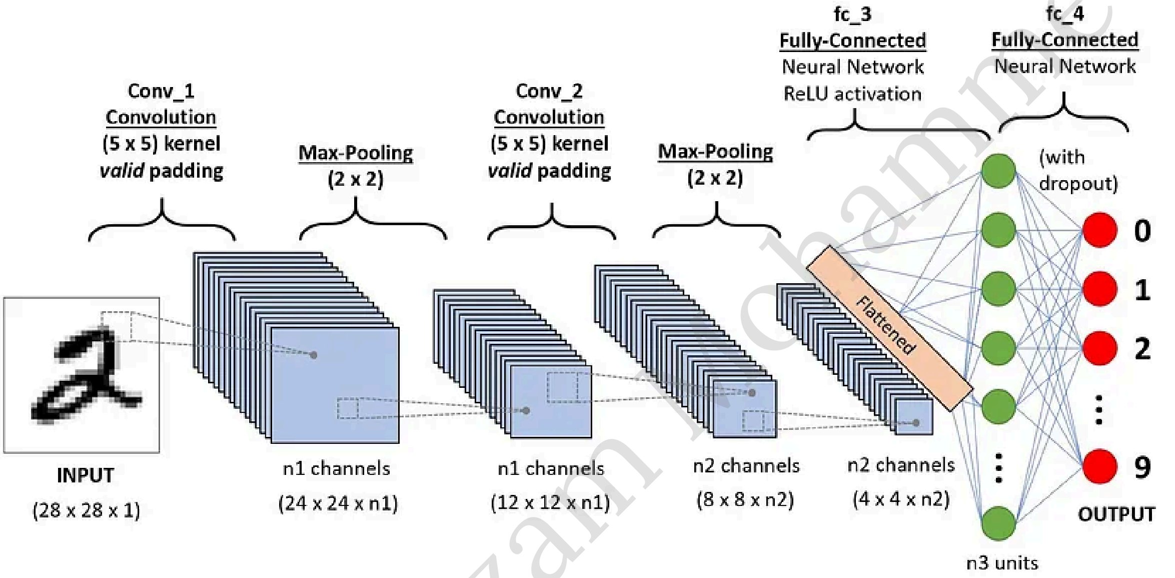
Notes by Izam Mohammed

CNN

Convolutional Neural Network It's a type of artificial neural network designed specifically for processing structured grid data, such as images. The main strength of CNNs lies in their ability to automatically and adaptively learn spatial hierarchies of features from the input data.

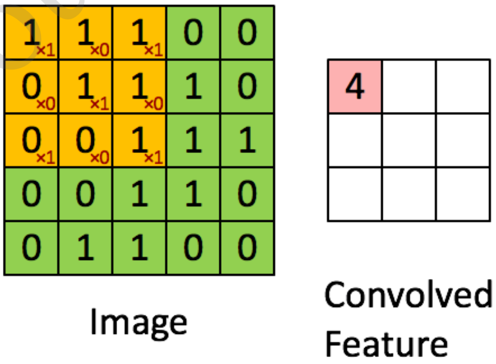


Notes by Izam Mohammed

Terms

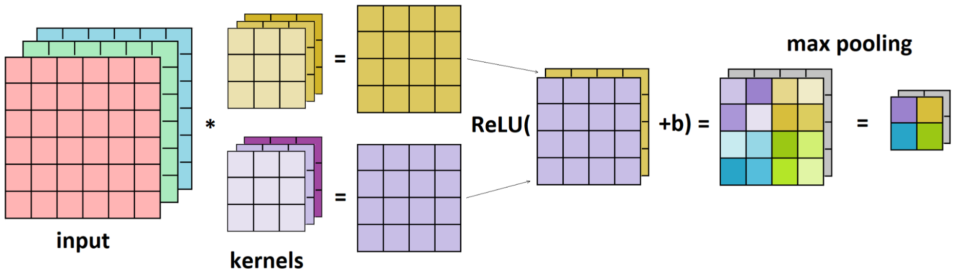
- ➢ *Convolution operation*

  In CNNs, the input data (e.g., an image) and a smaller filter (also called a kernel) move across each other, element-wise multiplying and summing at each position.



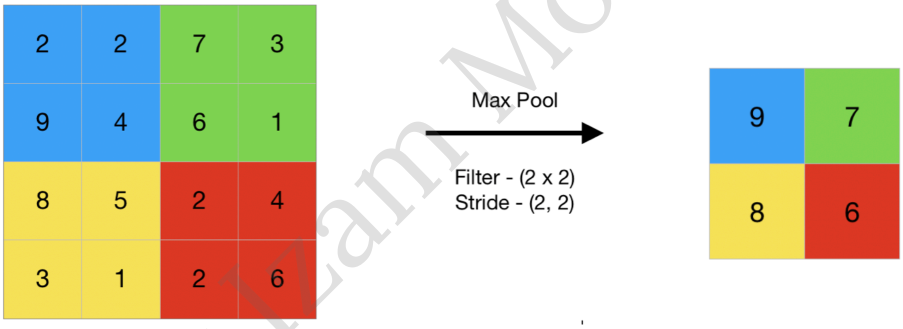
- ➢ *Convolutional layers*

- ○ *Building blocks for CNN which have convolution operation*



- ➢ *Filter/kernel￼*
    - ○ *is a small matrix used for feature extraction. It slides or convolves across the input data, performing element-wise multiplication, adding up, and producing a feature map that highlights specific patterns.*
- ➢ *Pooling layers*

    Pooling layers reduce the spatial dimensions of the input data, effectively downsampling it. This helps in reducing the computational load and controlling overfitting.



Here are the types of pooling

- ○ *Max pool*
- ○ *Min pool*
- ○ *Average pool*
- ➢ *Activation functions*

    Activation functions introduce non-linearities to the neural network, allowing it to learn complex patterns. Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Tanh.
- ➢ *Padding*

    is the addition of extra pixels around the input data. It is used to ensure that the convolutional operations do not reduce the spatial dimensions too quickly, preserving more information.
- ➢ *Stride*

    is the step size with which the filter moves across the input data. A larger stride reduces the spatial dimensions of the output feature map.
- ➢ *The formula for the Convolution Layer*

Notes by Izam Mohammed

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

- *Fully connected layers*

  also known as dense layers, connect every neuron in one layer to every neuron in the next layer. In CNNs, these layers are typically used after convolutional and pooling layers to make final predictions.
- *Flatten*

  Flattening is the process of converting the multidimensional data into a one-dimensional array.
- *Backpropagation*

  It involves updating the model's weights based on the error between predicted and actual outputs.
- *Batch →*

  is a subset of the dataset used to train the model in each iteration.
- *Dropout*

  Dropout is a regularization technique used during training to deactivate some neurons, reducing overfitting randomly.
- *Transfer Learning*

  Transfer learning involves using a pre-trained model on a similar task and fine-tuning it for a specific task.
- *Data Augmentation*

  involves applying various transformations (rotation, scaling, flipping, etc.) to existing data, creating new variations. This helps in increasing the diversity of the training dataset.
- *Vanishing gradient*

  A vanishing gradient occurs when the gradients during backpropagation become extremely small, causing the model to learn very slowly or stop learning altogether.
- *adversarial attacks*

  Adversarial attacks involve deliberately manipulating input data to mislead a model's predictions.

Advantages

- *It is good for capturing spatial hierarchies and local patterns in data*
- *Using shared weights for convolutional layers*
- *robust to variations in the position of features*
- *Feature engineering is not required.*
- *It has outstanding performance in image classification tasks*
- *Transfer learning is possible*

Disadvantages

- *High computational requirements*
- *Difficulty with small datasets*
- *CNNs also require large datasets to achieve high accuracy rates.*
- *Vulnerability to adversarial attacks*
- *Limited ability to generalize to new situations*

Notes by Izam Mohammed

- ➢ *Interpretability is hard*

Applications

- ➢ *Image classification*
- ➢ *Object detection*
- ➢ *Facial recognition*
- ➢ *Image segmentation*
- ➢ *Neural style transfer*
- ➢ *Image super-resolution*
- ➢ *Gesture recognition*
- ➢ *Optical Character Recognition (OCR)*

Networks

- ➢ *LeNet-5 → 1998*
- ➢ *AlexNet → 2012* → ImageNet Classification with Deep CNN.pdf
- ➢ *VGG-16, 19 → 2014* → Very Deep Convolutional Networks .pdf
- ➢ *ResNet-50 → 2016* → Deep Residual Learning for Image Recognition.pdf
- ➢ *Inception (v1, v2), v3 → 2015 → v1 (* Going deeper with convolutions.pdf *)*
- ➢ *MobileNet (v1, v2), v3 → 2019*
- ➢ *EfficientNet → 2019*
- ➢ *DenseNet → 2016*
- ➢ *Faster-RCNN → 2015*
- ➢ *YOLO (v…), v8 → 2023*

Notebooks

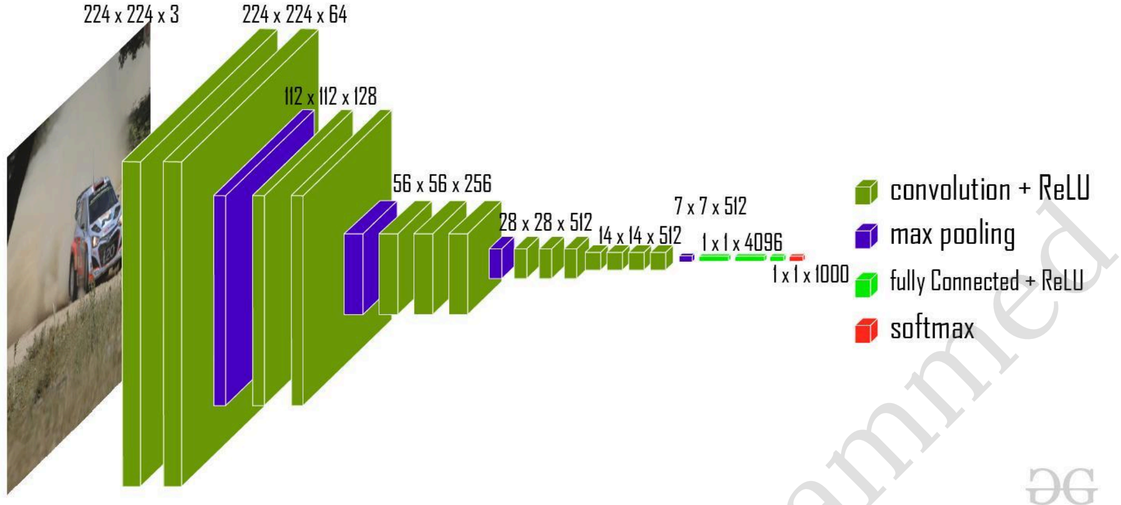
- ➢ CNN_Tensorflow.ipynb
- ➢ CNN_torch.ipynb

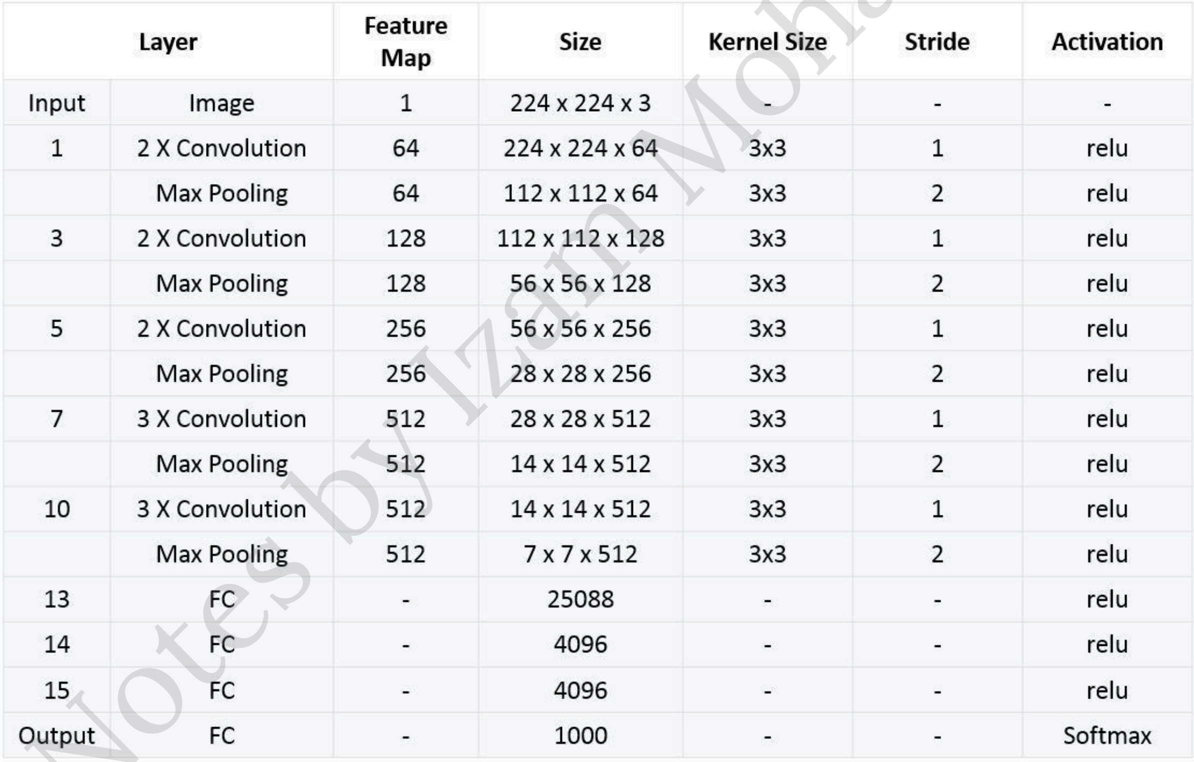
Notes by Izam Mohammed

Notes by Izam Mohammed

VGG-16

Padding - same





Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Notes by Izan Mohammed

Advantages

- ➢ *Simplicity and Uniformity → consists of stacks of 3x3 convolutional layers with small filter sizes, followed by max-pooling layers.*
- ➢ *Transfer Learning Capabilities*
- ➢ *Good Generalization*
- ➢ *Effective Feature Extraction*

Disadvantages

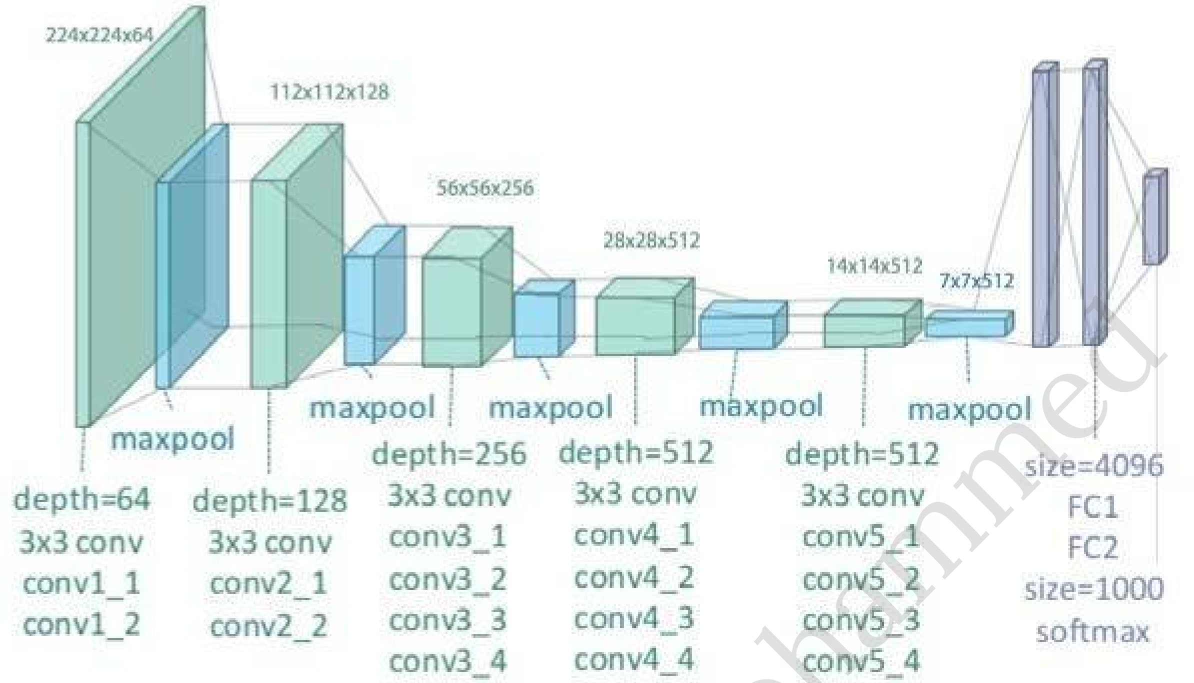
- *computationally intensive, especially compared to more modern architectures like ResNet or EfficientNet*
- *high memory requirements needed during training and inference.*
- *Overfitting on Small Datasets*
- *The computational complexity of VGG16 makes it less suitable for real-time applications*

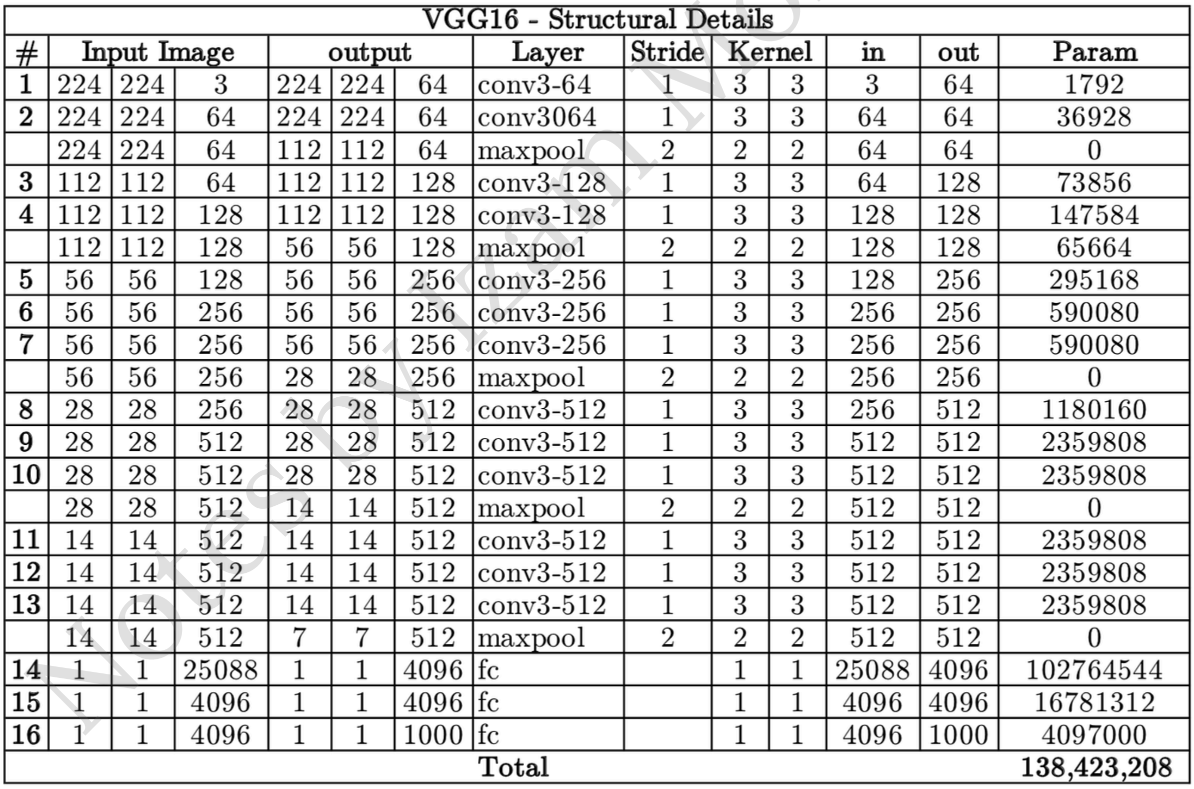
Params → *138 million*

Notebook → VGG16_torch.ipynb

Notes by Izam Mohammed

VGG-19





VGG16 - Structural Details

#	Input Image			output			Layer	Stride	Kernel		in	out	Param
1	224	224	3	224	224	64	conv3-64	1	3	3	3	64	1792
2	224	224	64	224	224	64	conv3064	1	3	3	64	64	36928
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664
5	56	56	128	56	56	256	conv3-256	1	3	3	128	256	295168
6	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
7	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0
8	28	28	256	28	28	512	conv3-512	1	3	3	256	512	1180160
9	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
10	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
	28	28	512	14	14	512	maxpool	2	2	2	512	512	0
11	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
12	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
13	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0
14	1	1	25088	1	1	4096	fc		1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc		1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc		1	1	4096	1000	4097000
							Total						**138,423,208**

Notes by Izam Mohammed

Advantages and disadvantages
**same as vgg16*

Params → *144 million*

ResNet

short for Residual Networks, is a type of neural network architecture that was introduced to address the challenge of training very deep networks

Variations

- *ResNet-18*
    - *18 layers (17 convolutional layers + 1 fully connected layer)*
    - *Basic building block: BasicBlock*
    - *Designed for less computationally intensive tasks or when resources are limited.*
- *ResNet-34*
    - *34 layers (33 convolutional layers + 1 fully connected layer)*
    - *Basic building block: BasicBlock*
    - *Similar to ResNet-18 but with increased depth, providing better representation capabilities.*
- *ResNet-50*
    - *50 layers (49 convolutional layers + 1 fully connected layer)*
    - *Basic building block: Bottleneck*
    - *Introduces the bottleneck architecture to improve efficiency and reduce the number of parameters. It consists of 1x1, 3x3, and 1x1 convolutional layers.*
- *ResNet-101*
    - *101 layers (100 convolutional layers + 1 fully connected layer)*
    - *Basic building block: Bottleneck*
    - *Similar to ResNet-50 but deeper, offering improved representation learning capabilities.*
- *ResNet-152*
    - *152 layers (151 convolutional layers + 1 fully connected layer)*
    - *Basic building block: Bottleneck*
    - *The deepest ResNet model among those listed provides an even stronger representation of learning abilities.*

Advantages

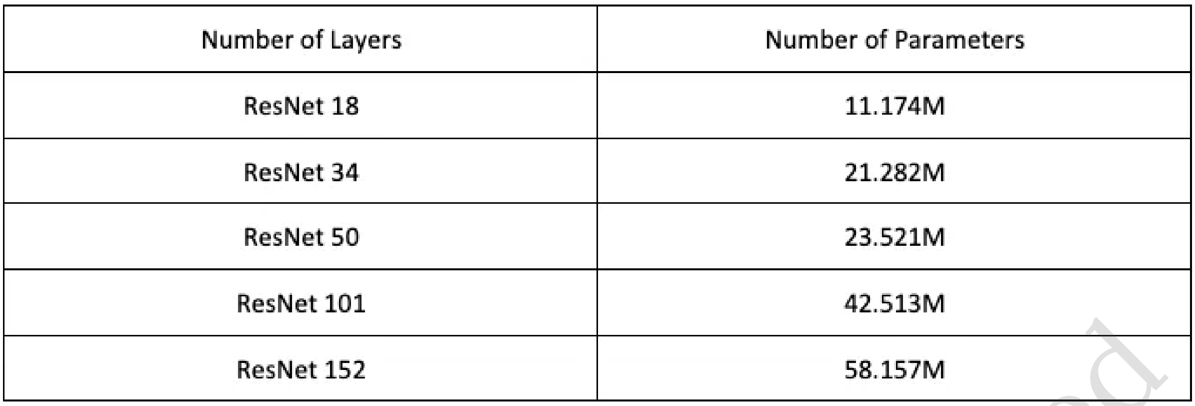
- *Residual connections help with the training of very deep networks*
- *residual connections enable the optimization of deeper networks*
- *Better Performance on Deeper Architectures*
- *Pretrained models are available*
- *Global average pooling is computationally efficient*

Disadvantages

- *Complex and not needed for most of the problems*
- *Hard to interpret*
- *High chance of overfitting in a small dataset*
- *High training time because of the deep network*

Notes by Izam Mohammed

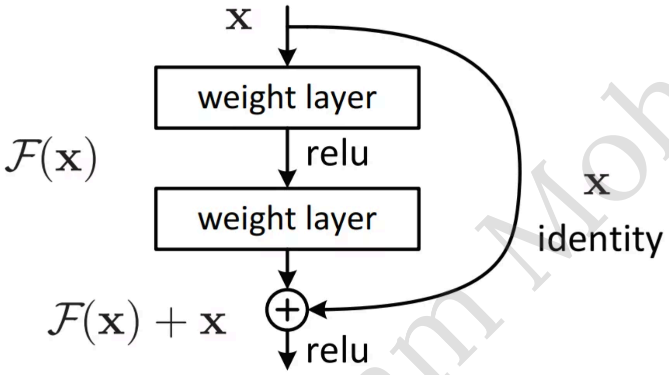
Params



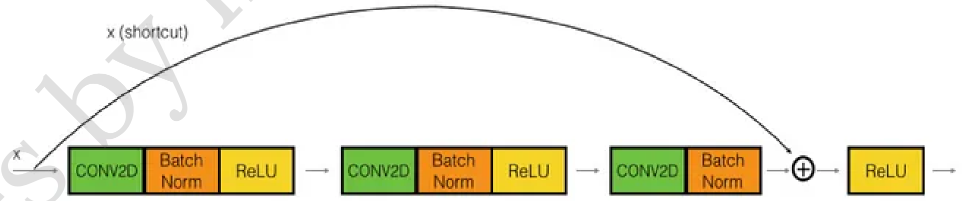
Number of Layers	Number of Parameters
ResNet 18	11.174M
ResNet 34	21.282M
ResNet 50	23.521M
ResNet 101	42.513M
ResNet 152	58.157M

Terms

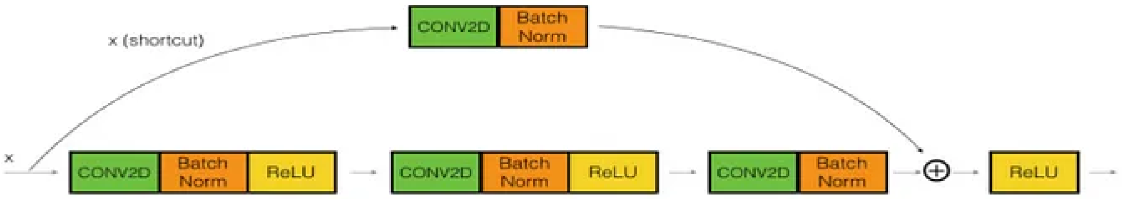
- ➢ *Residual connection*


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

- ➢ *Basic building blocks*
  - ○ *identity*



  - ○ *convolutional*



- ➢ *Global average pooling (GAP)*

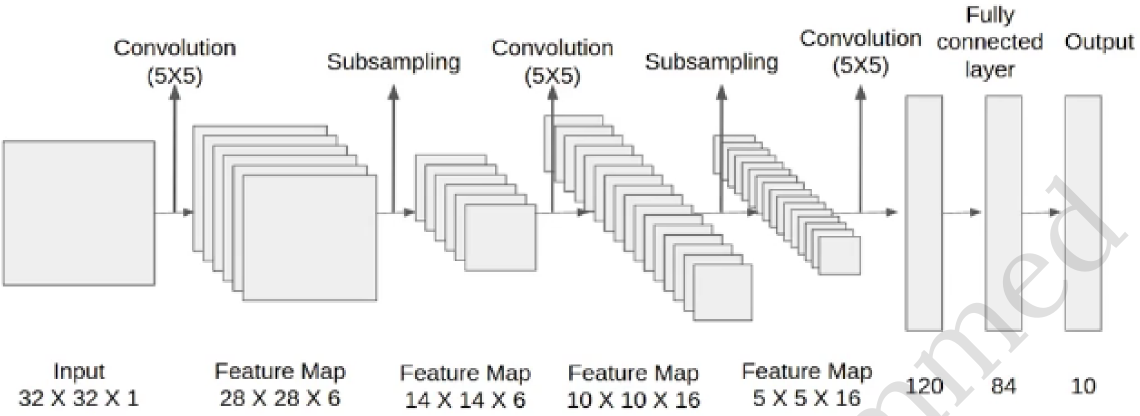
  In global average pooling, instead of using a local window or filter, the average is calculated over the entire feature map. The result is a single value for each feature map, effectively reducing the spatial dimensions to 1x1.

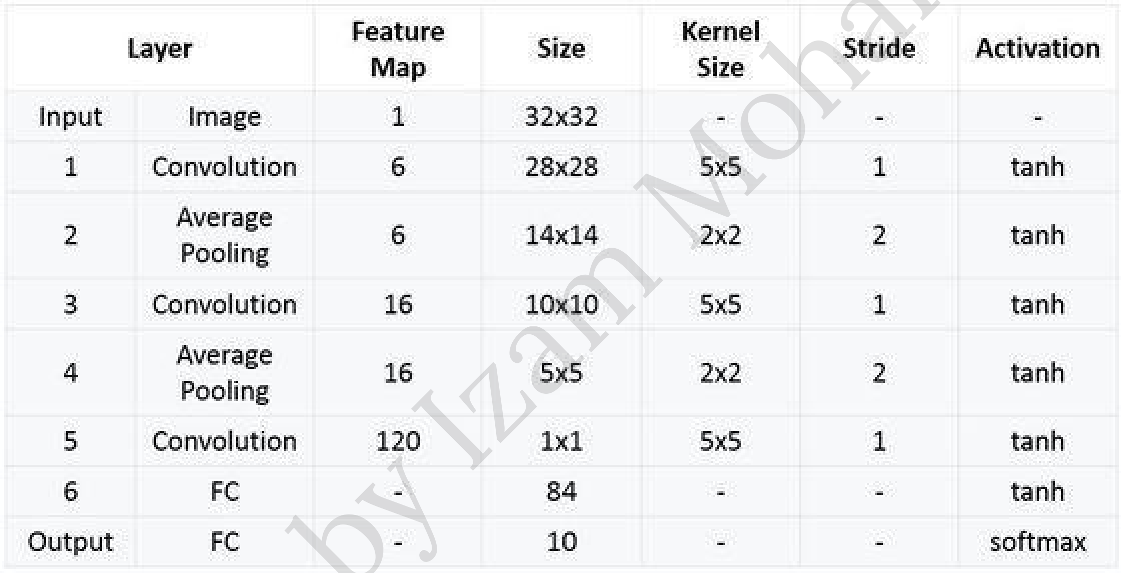
Notebook → ResNet_torch

Notes by Izam Mohammed

LeNet

LeNet-5 was designed back in 1998 for handwritten digit recognition and played a significant role in the development of modern deep learning





Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

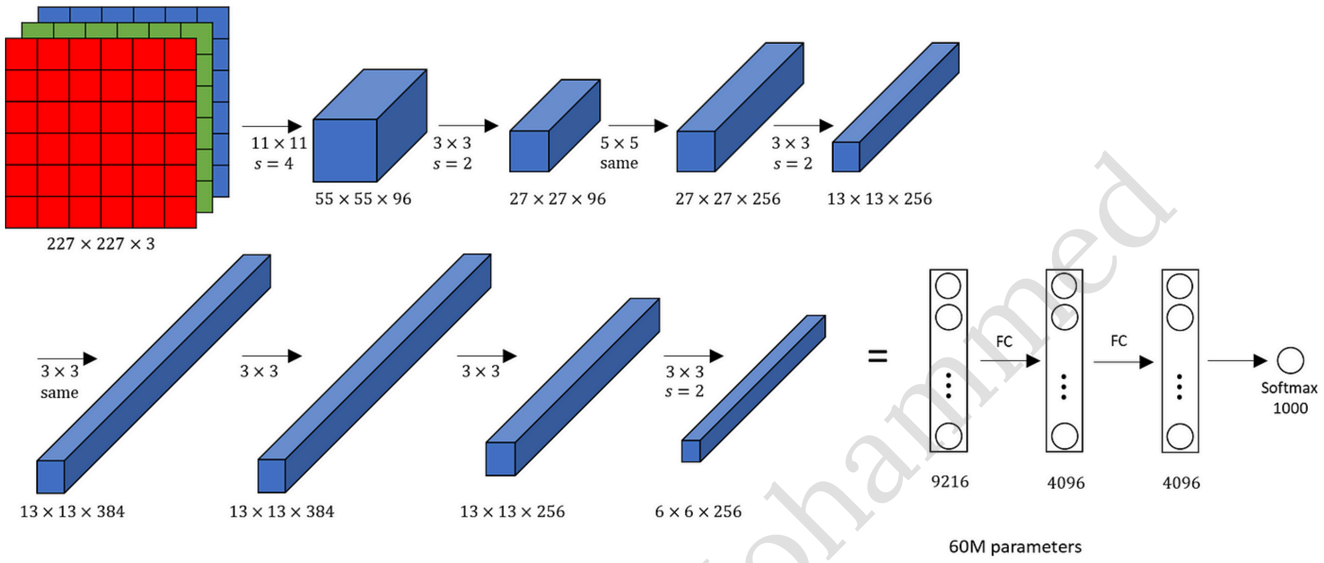
***Total params** → 60k*

***Notebook** →* [LeNet_torch.ipynb]

Notes by Izam Mohammed

AlexNet

AlexNet was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton and won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012



Layer	# filters / neurons	Filter size	Stride	Padding	Size of feature map	Activation function
Input	-	-	-	-	227 x 227 x 3	-
Conv 1	96	11 x 11	4	-	55 x 55 x 96	ReLU
Max Pool 1	-	3 x 3	2	-	27 x 27 x 96	-
Conv 2	256	5 x 5	1	2	27 x 27 x 256	ReLU
Max Pool 2	-	3 x 3	2	-	13 x 13 x 256	-
Conv 3	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 4	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 5	256	3 x 3	1	1	13 x 13 x 256	ReLU
Max Pool 3	-	3 x 3	2	-	6 x 6 x 256	-
Dropout 1	rate = 0.5	-	-	-	6 x 6 x 256	-

Notes by Izam Mohammed

Total params *→ 62.3 million*

Advantages

- *Deep Architecture*
- *Parallelization*
- *Regularization*

Notebook → [AlexNet_torch.ipynb]