Convolutional Neural Networks (CNN)

An Introduction to Deep Learning in Computer Vision

DepartmentofAI

July 21, 2025

Overview

1. Introduction
2. Architecture
3. LeNet-5
4. AlexNet
5. VGGNet
6. GoogLeNet (Inception)
7. ResNet
8. DenseNet
9. Conclusion

UNIT II: Convolutional Neural Networks (CNN)

- Introduction, Striding and Padding, Pooling Layers
- Structure, operations and prediction of CNN with layers
- CNN - Case study with MNIST, CNN VS Fully Connected

References

- Giancarlo Zaccone, Md. Rezaul Karim, Ahmed Menshawy "Deep Learning with TensorFlow: Explore neural networks with Python", Packt Publisher, 2017.
- Antonio Gulli, Sujit Pal "Deep Learning with Keras", Packt Publishers, 2017.
- Francois Chollet "Deep Learning with Python", Manning Publications, 2017.

What is a Convolutional Neural Network?

Definition: A Convolutional Neural Network (CNN) is a type of deep learning algorithm designed to process and analyze visual data, such as images and videos.

Key Features:

- CNNs are inspired by the organization of the animal visual cortex.
- They excel in capturing spatial and hierarchical patterns in data.
- CNNs have revolutionized computer vision tasks, including image classification, object detection, and image segmentation.

Importance of CNNs in Machine Learning and Computer Vision

Handling Complex Data: CNNs are capable of handling high-dimensional and complex visual data, making them suitable for various computer vision tasks.

Feature Extraction: CNNs automatically learn hierarchical representations of data, enabling them to extract meaningful features without manual feature engineering.

Achieving State-of-the-Art Performance: CNNs have consistently achieved remarkable performance on benchmark datasets, surpassing traditional machine learning methods.

Transfer Learning: CNNs can leverage pre-trained models to accelerate training and improve performance on new tasks, even with limited labeled data.

Limitations of Fully Connected Neural Networks (FNN)

Scaling Issues:

- Traditional Fully Connected Neural Networks struggle with scaling up to large input data such as high-resolution images or videos.
- This is due to the large number of connections and weights, leading to high computational requirements and longer training times.

Lack of Spatial Awareness:

- FNNs do not take into account the spatial hierarchy and structure in images.
- Each pixel in the image is treated independently, ignoring its context within the image.
- This leads to inefficiencies and poor performance on tasks like image recognition and object detection where spatial relationships are key.

Limitations of Fully Connected Neural Networks (FNN)

- **Complex Problems and Increased Weights:** For complex problems, we need multiple hidden layers in our FNN. This compounds the problem of having many weights.
- **Learning Difficulties:** Having too many weights makes learning more difficult as the dimension of the search space is increased.
- **Resource Intensive:** More weights result in more time/resource consuming training.
- **Overfitting:** The abundance of weights increases the likelihood of overfitting.
- **Color Images:** The problem is further compounded for color images. Each pixel in a color image is represented by 3 values (RGB color mode), increasing the channel size to 3.
- **Pixel Representation:** For example, an image represented by 64 x 64 x 3 (rows x columns x channels) results in 12,288 values.
- **Weight Explosion:** The number of weights with 500 hidden neurons would be 12,288 x 500 = 6,144,000.

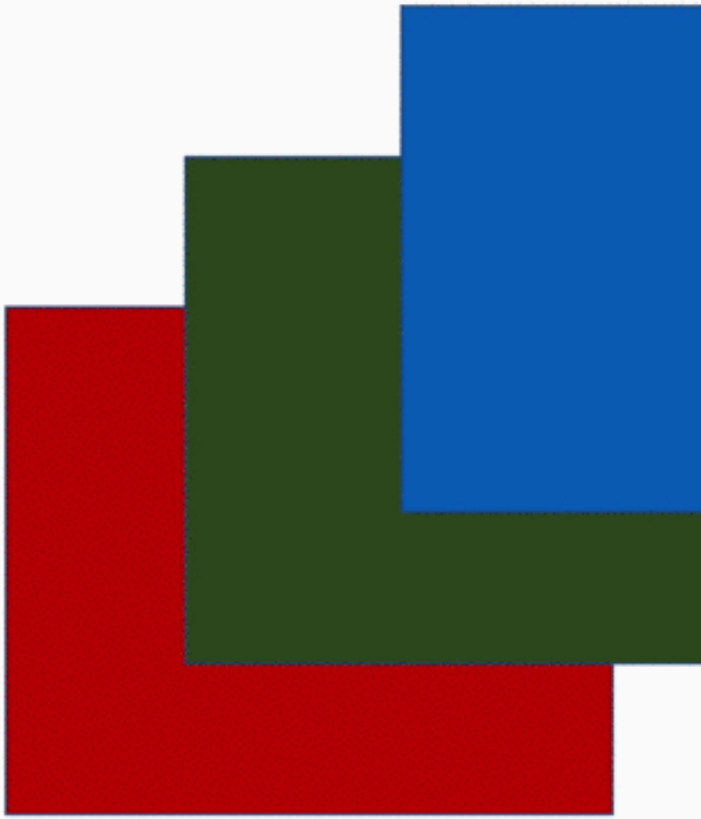
Limitations of Fully Connected Neural Networks (FNN)

An Introduction to Deep Learning in Computer Vision

Fully Connected Neural Networks face challenges when processing high-dimensional image data:

- **Input Image:** Consider a 64 pixel by 64 pixel grayscale image.
- **Pixel Representation:** Each grayscale pixel is represented by 1 value, usually between 0 to 255, where 0 is black, 255 is white, and values in between are shades of gray.
- **Channel Size:** Since each grayscale pixel is represented by 1 value, we say the channel size is 1.
- **Total Values:** Thus, the image is represented by 64 x 64 x 1 = 4,096 values (rows x columns x channels).
- **Input Nodes:** Let's assume the next layer has 500 nodes.
- **Total Weights:** Since FNNs are fully connected, we have 4,096 x 500 = 2,048,000 weights.

Image representation



RGB and Grayscale Images

RGB Images

- RGB stands for Red, Green, and Blue
- RGB images are composed of three color channels: red, green, and blue
- By combining these colors in varying intensities, it is possible to produce any color
- Each pixel contains three values (R, G, B)

Grayscale Images

- Grayscale images consist of shades of gray
- Unlike RGB images, grayscale images have only one channel
- Values can range from 0 (black) to 255 (white)
- Each pixel represents the intensity of light
- Often used in image processing to reduce complexity

Architecture of CNN

A typical Convolutional Neural Network consists of four primary layers:

1. **Input Layer:** This layer receives the raw pixel data from images.
2. **Convolution Layer:** This layer applies a set of learnable filters to the input, producing a feature map.
3. **Pooling Layer:** This layer reduces the spatial dimensions (width and height) of the input volume, reducing computational complexity and controlling overfitting.
4. **Fully Connected Layer:** This layer takes the output of the previous layers, flattens it, and feeds it into a standard neural network for classification or regression.

Basic Convolutional Network Structure

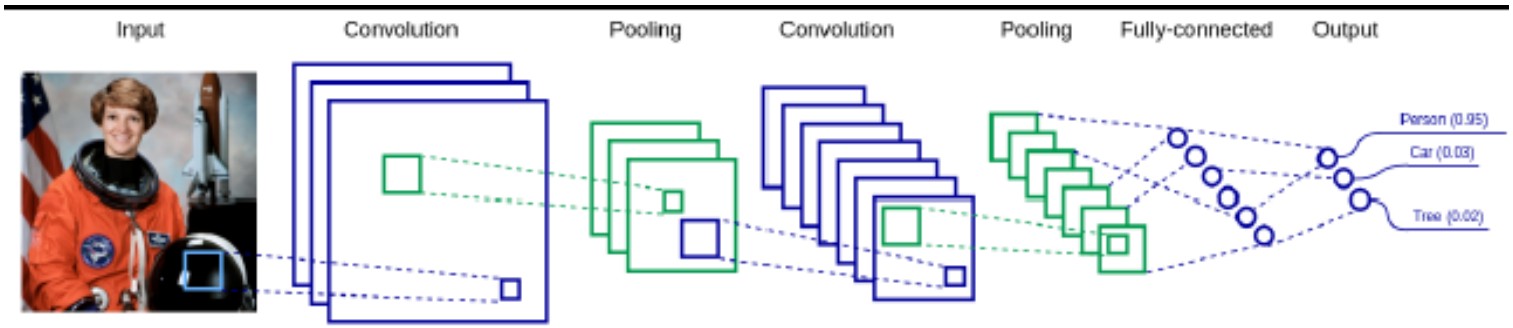


Figure: Basic convolutional network with convolutional and fully-connected layers in blue, and pooling layers in green.

Input Layer in Convolutional Neural Networks

The Input Layer is the starting point of the CNN architecture:

- **Data Entry:** This layer is where the CNN receives input in the form of images.
- **Dimensions:** The input is a multi-dimensional array (also known as a tensor) of pixel values.
- **Color Images:** For color images, each pixel is represented by three numbers (RGB values), and thus the input is a 3-dimensional matrix.
- **Grayscale Images:** For grayscale images, each pixel is represented by a single number (intensity), making the input a 2-dimensional matrix.

Convolution Layer in Convolutional Neural Networks

- **Components:** The Convolution layer is composed of multiple filters (also known as kernels).
- **Dimension:** For a 2D image, the filters are also 2D.
- **Filter Example:** Consider a 3x3 filter (total of 9 values), where values are randomly set to 0 or 1.
  - **Convolution:** This process starts by placing the 3x3 filter on the top-left corner of the image.
  - **Operation:** Multiply the filter values by the corresponding pixel values, then sum the results.
  - **Filter Movement:** Move the filter one pixel to the right and repeat the process.
  - **Continue Process:** Once reaching the top-right corner, move the filter one pixel down and repeat the process.
  - **End:** The process ends when the filter reaches the bottom-right corner of the image.

3 *3 Filter

An Introduction to Deep Learning in Computer Vision

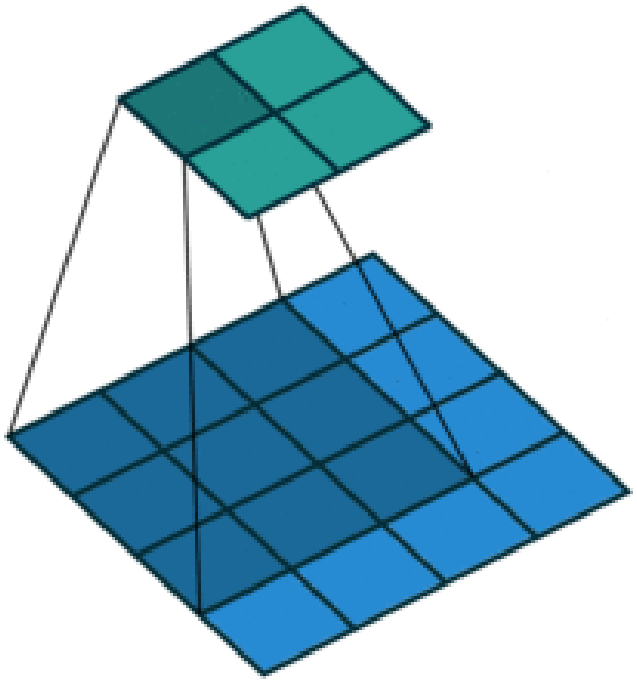


Figure: 3*3Filter

Convolution Operator Parameters

Filter Size:

- Filter size can be 5x5, 3x3, and so on.
- Larger filter sizes should be avoided as the learning algorithm needs to learn the filter values (weights).
- Odd-sized filters are preferred to even-sized filters due to their nice geometric property of having all input pixels around the output pixel.

Convolutional Filters in Deep Learning

Convolutional Filter

- Also known as a kernel
- Is a matrix used in Convolutional Neural Networks (CNNs)
- Performs a mathematical operation known as convolution on input data, such as images

Purpose of a Convolutional Filter

- Extracts features like edges, shapes, or patterns from the input data
- Essential for tasks like image recognition in deep learning

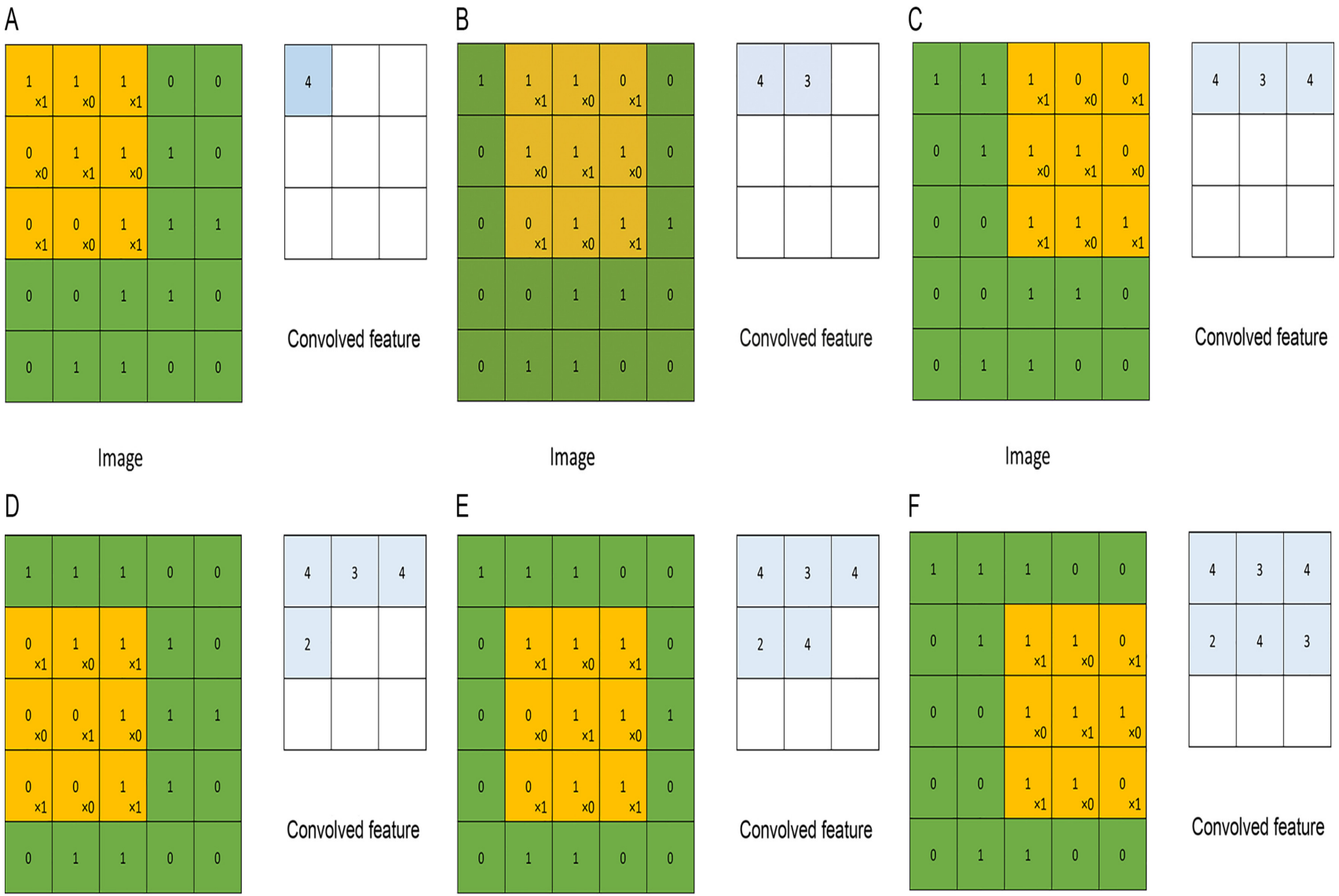
Convolution

255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0

*

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

Convolution Operation



Convolution Operation

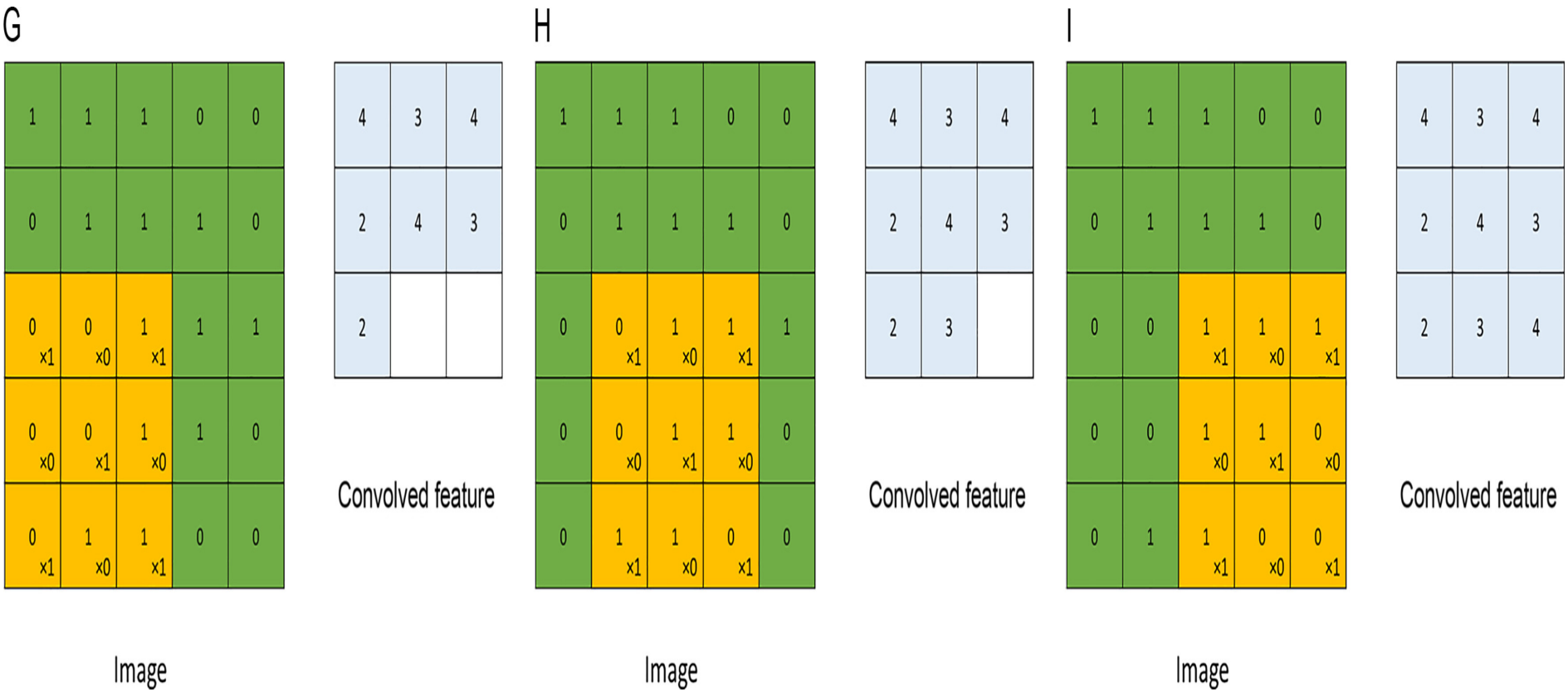


Image Size After Convolution

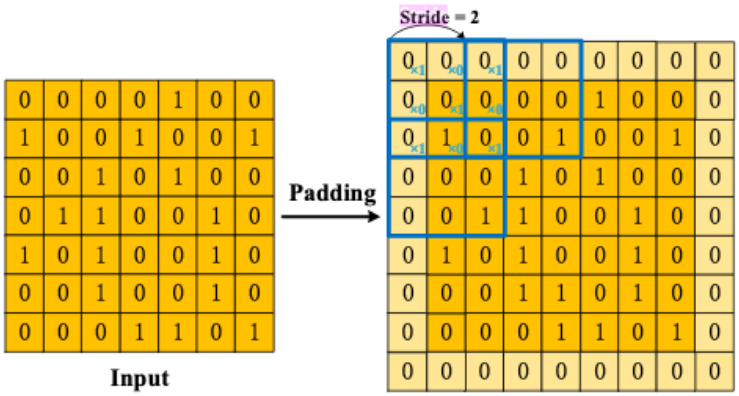
Assume an image size of $n \times n$ and a filter size of $f \times f$.

- Without padding, the image size after convolution is: $(n - f + 1) \times (n - f + 1)$
- With zero padding (padding of p), the image size after convolution is: $(n + 2p - f + 1) \times (n + 2p - f + 1)$
- With strided convolution (stride of s), the image size after convolution is: $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$

Convolution Operator Parameters

Padding:

- After applying a 3x3 filter to a 4x4 image, the resulting image size goes down to 2x2.
- If we want to keep the image size the same, we can use padding.
- Padding involves adding zeros around the input image before applying the filter.
- For example, if the padding is 1x1, we add 1 zero in every direction.
- If the padding is 2x2, we add 2 zeros in every direction, and so on.
- Applying a 3x3 filter with a padding of 1 to a 5x5 image results in a 5x5 image.



Convolution Operator Parameters

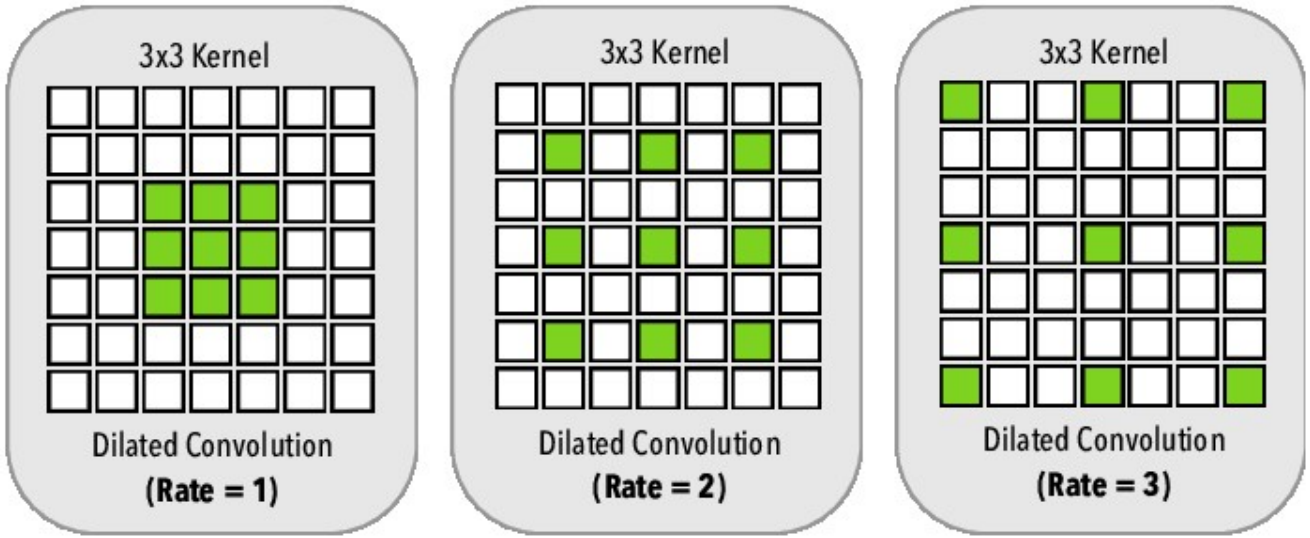
Stride:

- Stride determines how many pixels the filter moves to the right or down.
- Stride 1 means moving the filter one pixel to the right or down.
- Stride 2 means moving the filter two pixels to the right or down.
- Applying a 3x3 filter with a stride of 2 to a 5x5 image results in a 2x2 image.

Convolution Operator Parameters

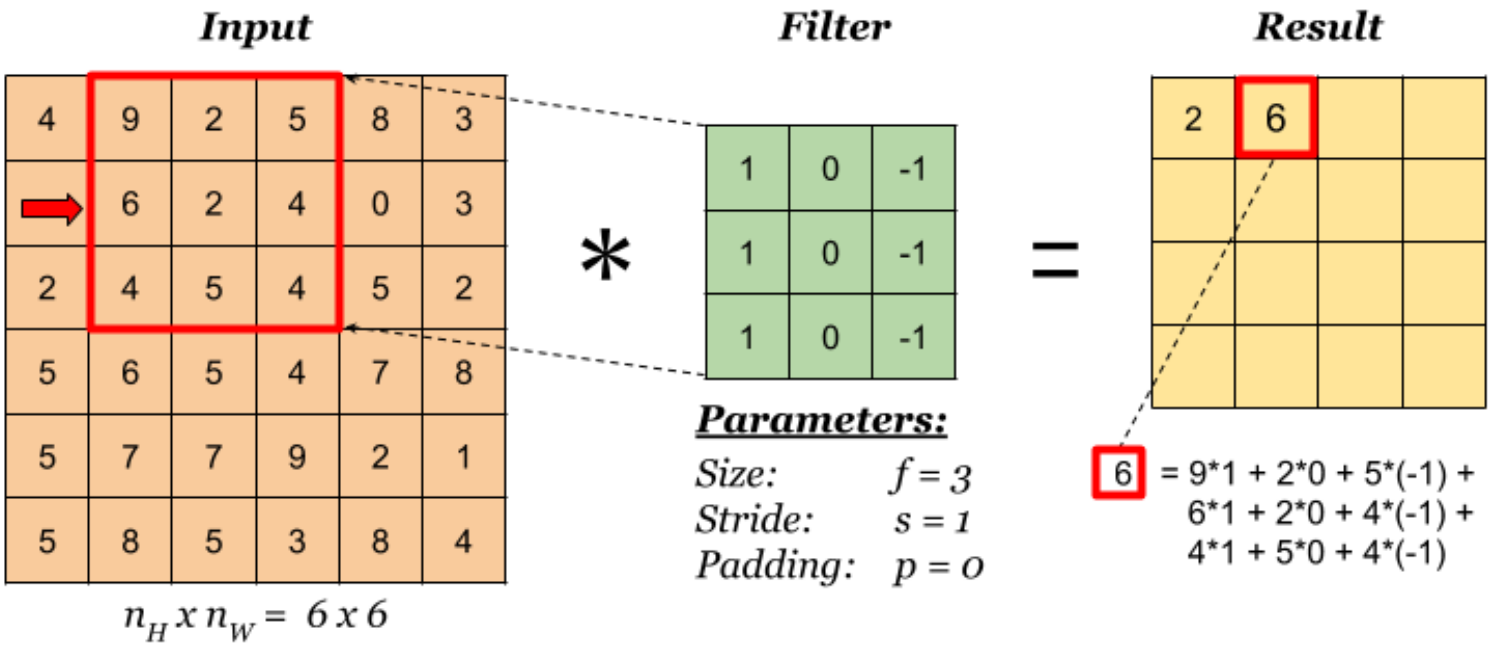
Dilation:

- When applying a 3x3 filter, the output is affected by the pixels in a 3x3 subset of the image.
- Dilation allows us to have a larger receptive field, i.e., a larger portion of the image that affects the filter's output.
- If the dilation is set to 2, every other pixel in a 5x5 subset of the image affects the output.
- Applying a 3x3 filter with a dilation of 2 to a 7x7 image results in a 3x3 im



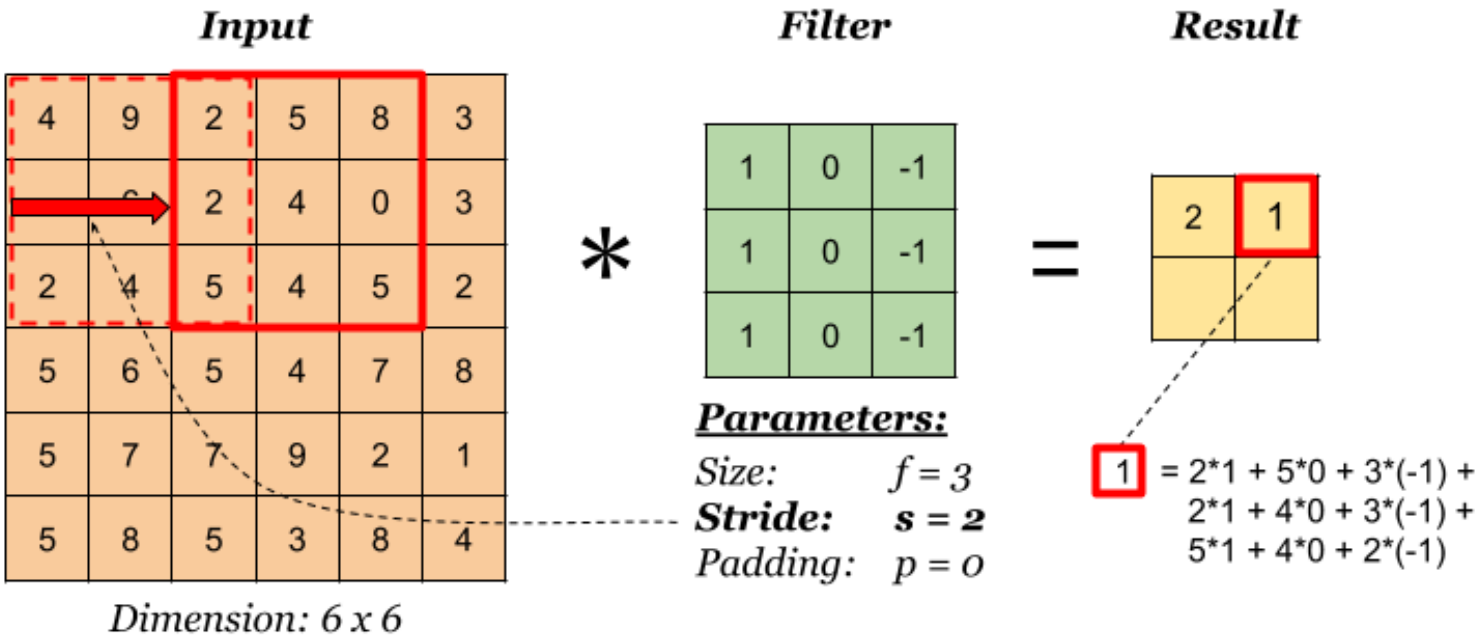
Stride

move the filter right one position



Stride

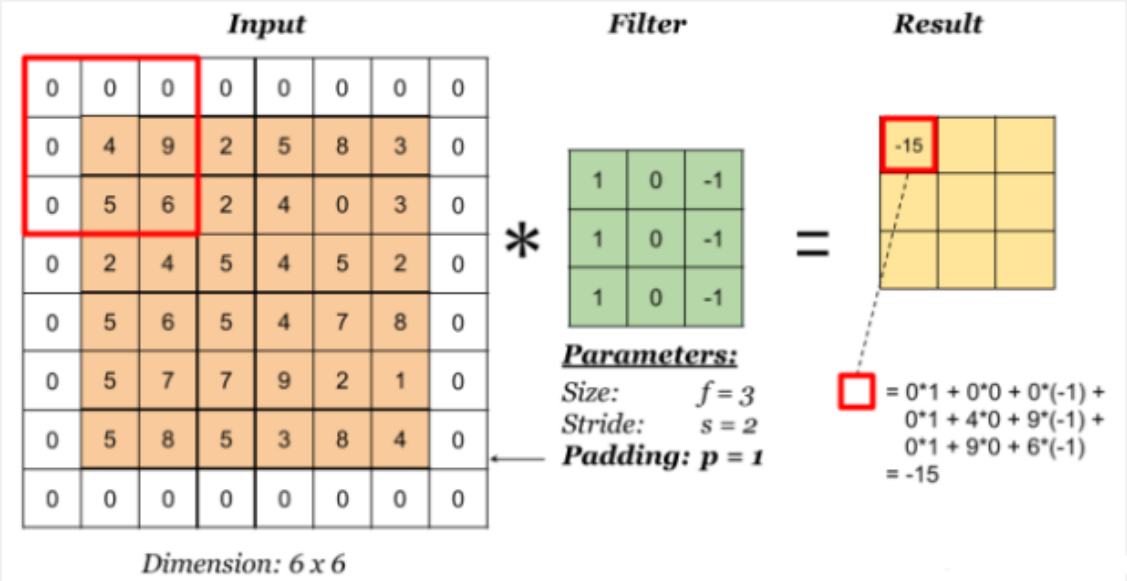
Stride governs how many cells the filter is moved in the input to calculate the next cell in the result.



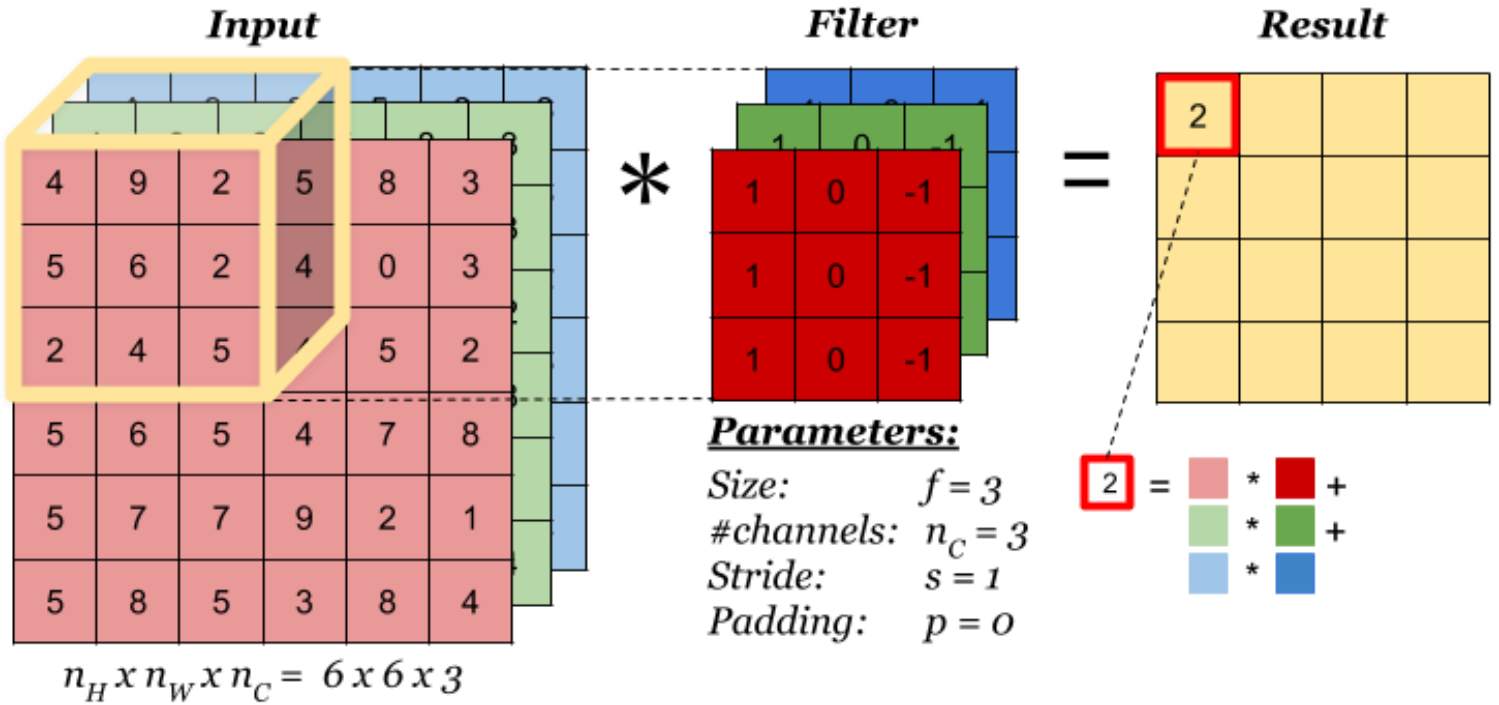
Benefits of Padding in Convolution

Padding offers several advantages in the convolution operation:

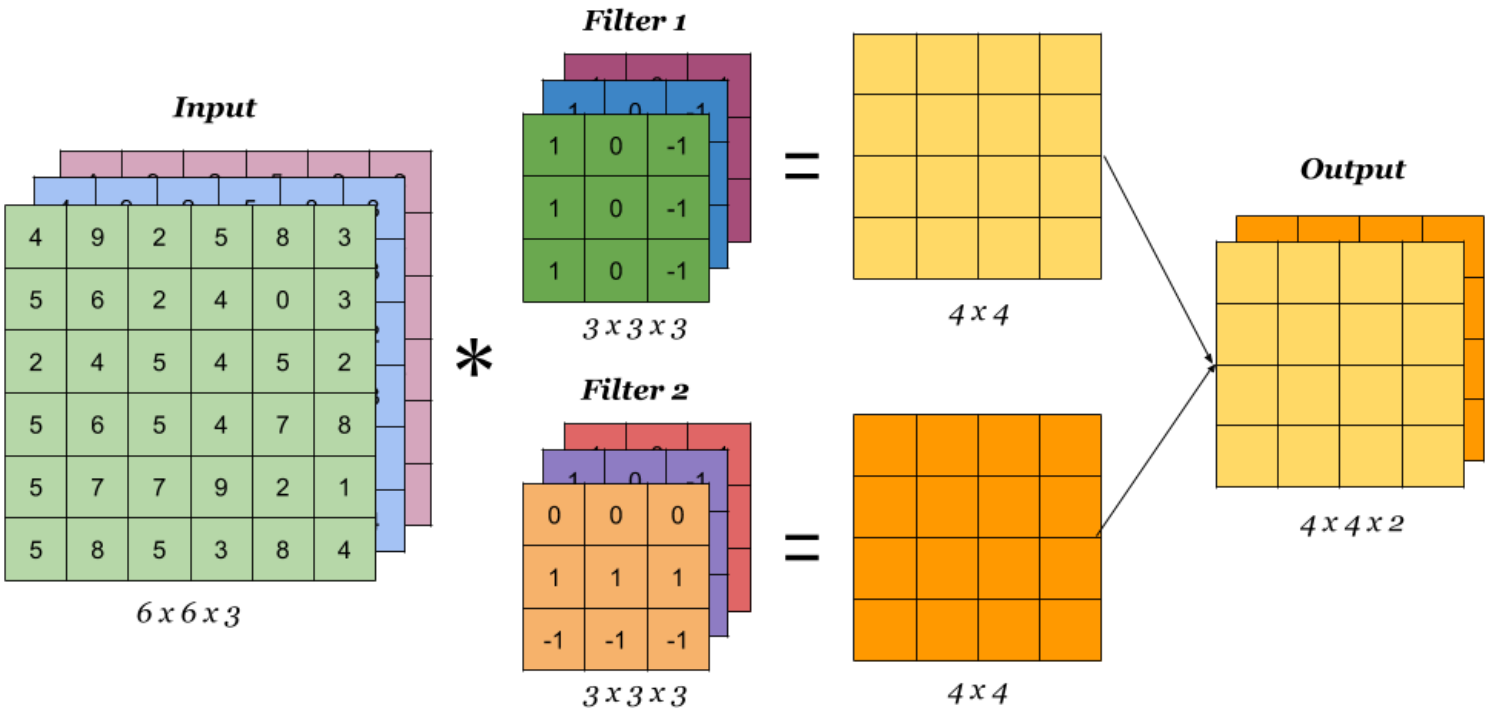
- enables the use of a convolutional layer without necessarily shrinking the height and width of the volumes, which is crucial for building deeper networks. Otherwise, the height/width would reduce as we progress to deeper layers.
- helps preserve more of the information at the image border. Without padding, very few values at the next layer would be influenced by pixels at the edges of an image.



Convolution on volume



Multiple filters



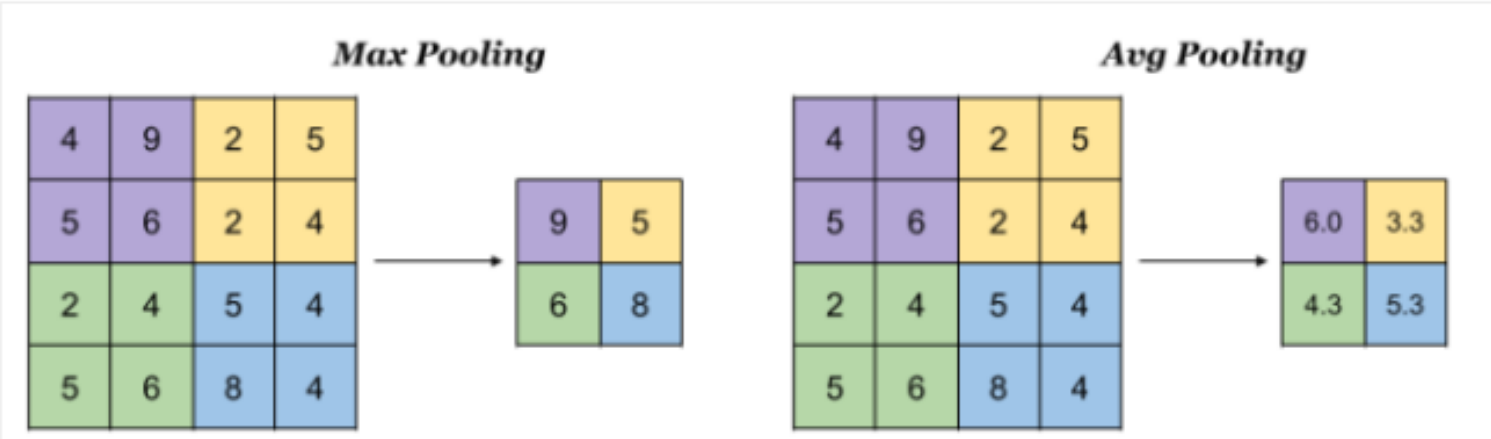
Pooling Layer

The Pooling Layer plays a crucial role in the architecture of a Convolutional Neural Network:

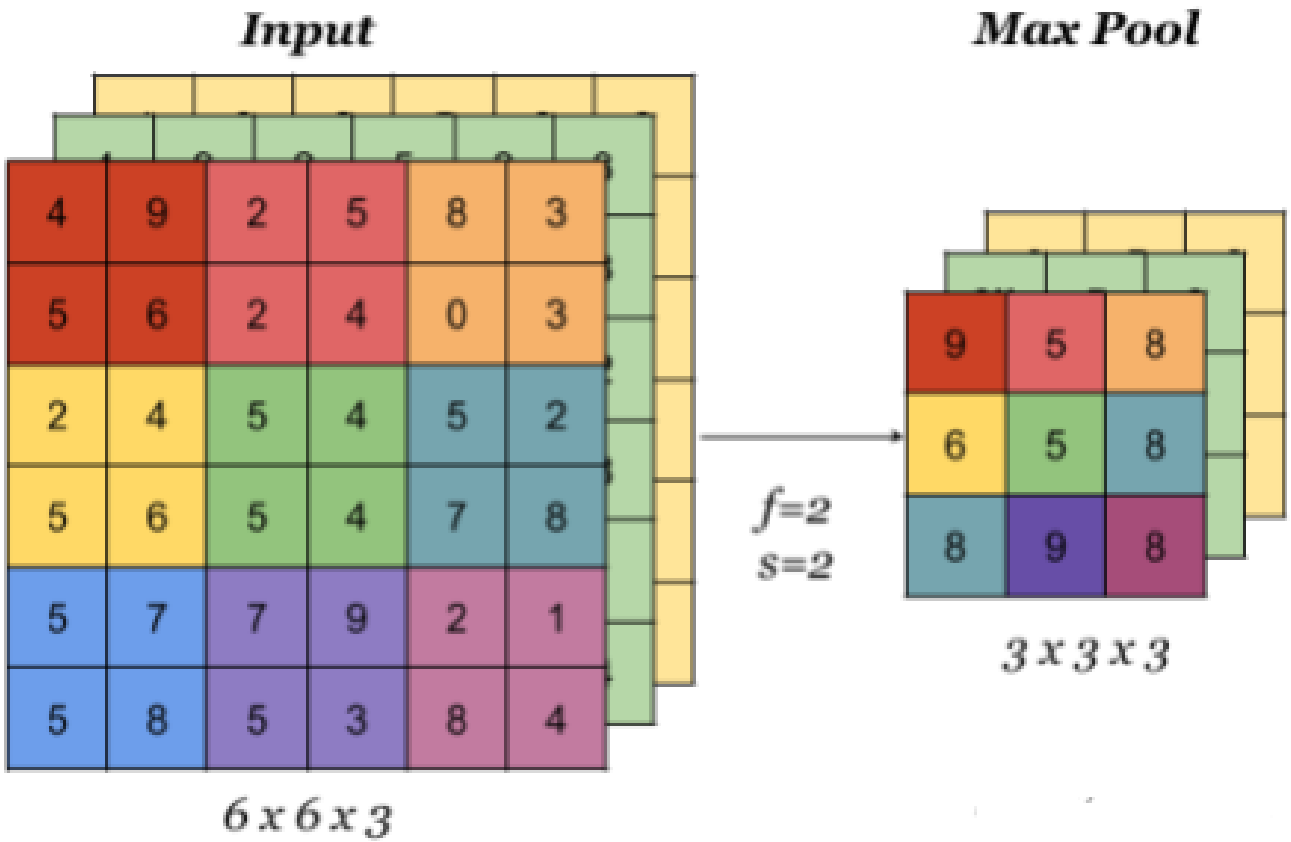
- It reduces the size of the representations, thereby accelerating computation.
- It enhances the robustness of the features that the model detects.

Common types of pooling are Max Pooling and Average Pooling:

- Max Pooling, which is currently more prevalent, effectively captures the most salient information from the input data.
- Average Pooling, though less commonly used, calculates the average of the input data.



Max pooling


Input
Max Pool
4 9 2 5 8 3
5 6 2 4 0 3
2 4 5 4 5 2
5 6 5 4 7 8
5 7 7 9 2 1
5 8 5 3 8 4
6 x 6 x 3
f=2
s=2
9 5 8
6 5 8
8 9 8
3 x 3 x 3

Sparse Connectivity

- **Highlight one input unit, x_3, and the output units in s that are affected by this unit.**
  - **Top:** When s is formed by convolution with a kernel of width 3, only three outputs are affected by x_3.
  - **Bottom:** When s is formed by matrix multiplication, connectivity is no longer sparse, so all of the outputs are affected by x_3.

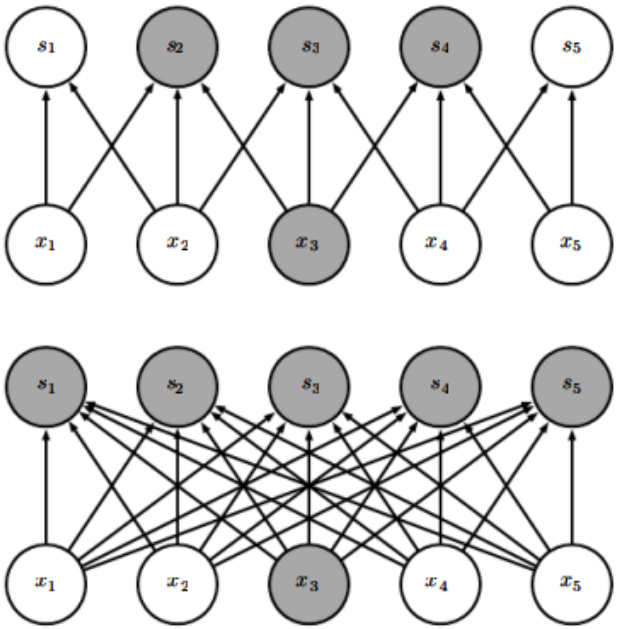


Figure: Sparse connectivity: Top diagram (convolution) and bottom diagram

Sparse Connectivity: Viewed from Above

- **Highlight one output unit, s_3, and the input units in x that affect this unit.**
  - These units are known as the receptive field of s_3.
  - **Top:** When s is formed by convolution with a kernel of width 3, only three inputs affect s_3.
  - **Bottom:** When s is formed by matrix multiplication, connectivity is no longer sparse, so all of the inputs affect s_3.

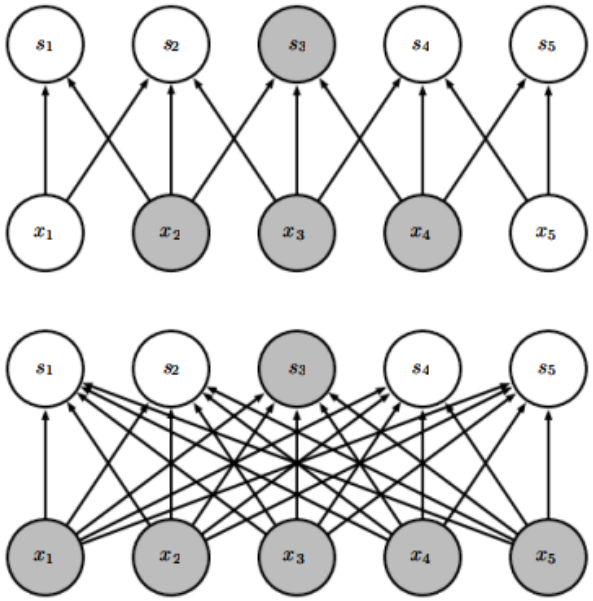


Figure: Sparse connectivity: Top diagram (convolution) and bottom diagram

Receptive Field in Convolutional Networks

- The receptive field of the units in the deeper layers of a convolutional network is larger than the receptive field of the units in the shallow layers.
- This effect increases if the network includes architectural features like strided convolution or pooling.
- Even though direct connections in a convolutional net are very sparse, units in the deeper layers can be indirectly connected to all or most of the input image.

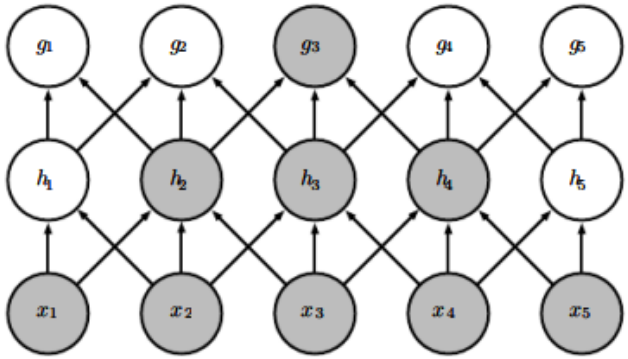


Figure: Receptive field in convolutional networks: Deeper layers vs. shallow layers.

Parameter Sharing

- Black arrows indicate the connections that use a particular parameter in two different models.
  - **Top:** The black arrows indicate uses of the central element of a 3-element kernel in a convolutional model. Due to parameter sharing, this single parameter is used at all input locations.
  - **Bottom:** The single black arrow indicates the use of the central element of the weight matrix in a fully connected model. This model has no parameter sharing, so the parameter is used only once.

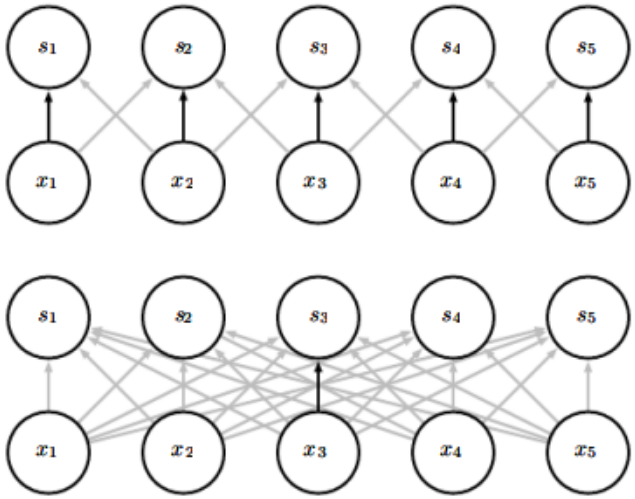
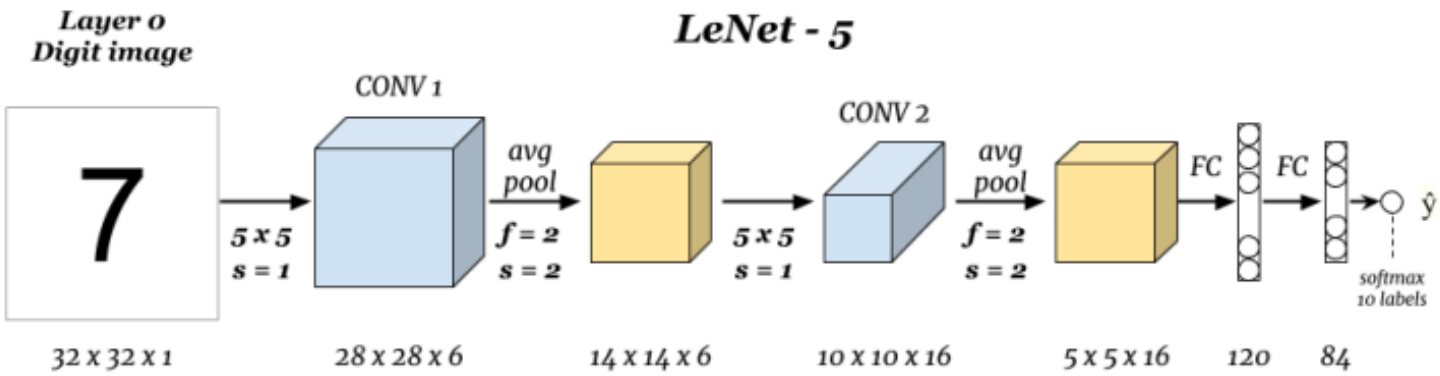


Figure: Parameter sharing in convolutional and fully connected models.

Lenet

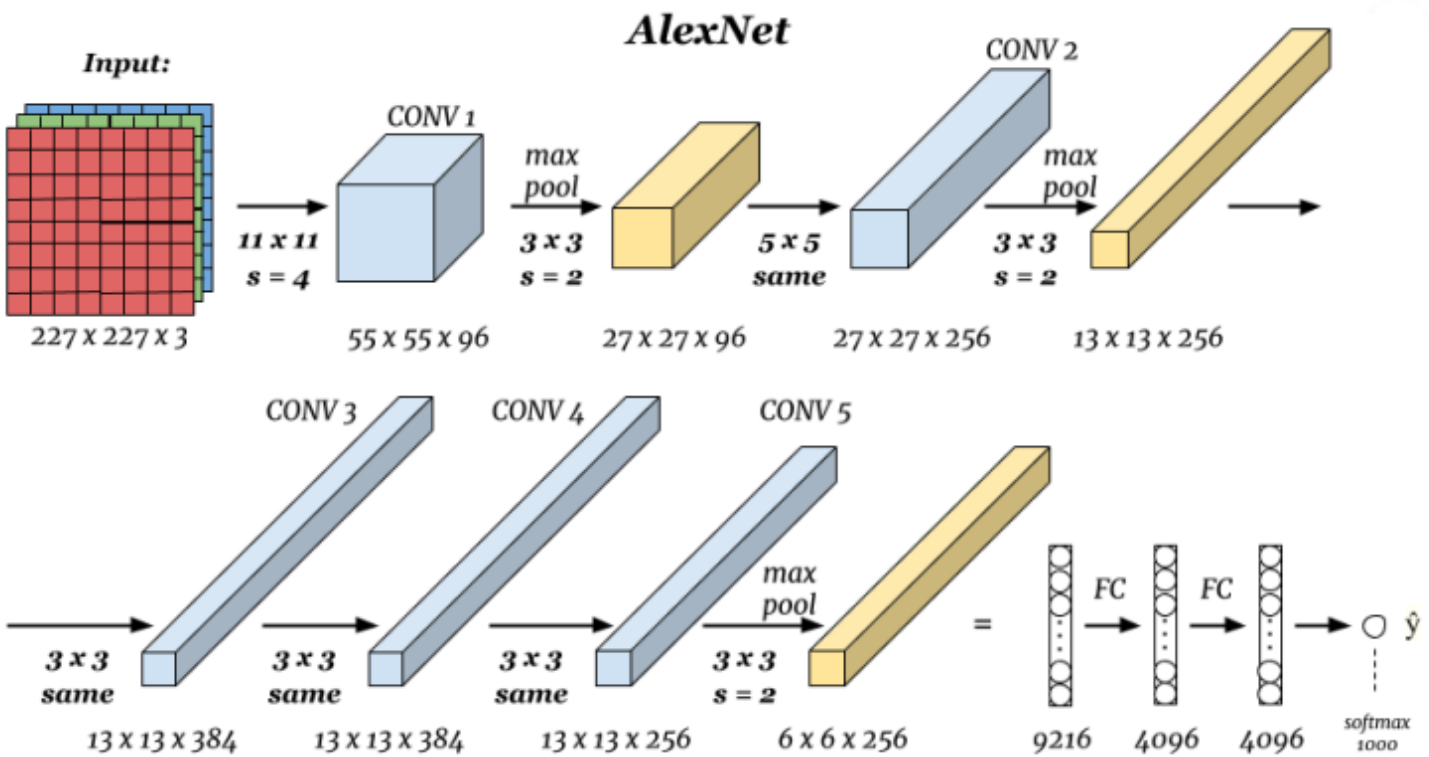
One example of classic networks is LeNet-5, from the paper *Gradient-Based Learning Applied to Document Recognition* by Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner (1998).



- Number of parameters: ~ 60 thousands.

AlexNet

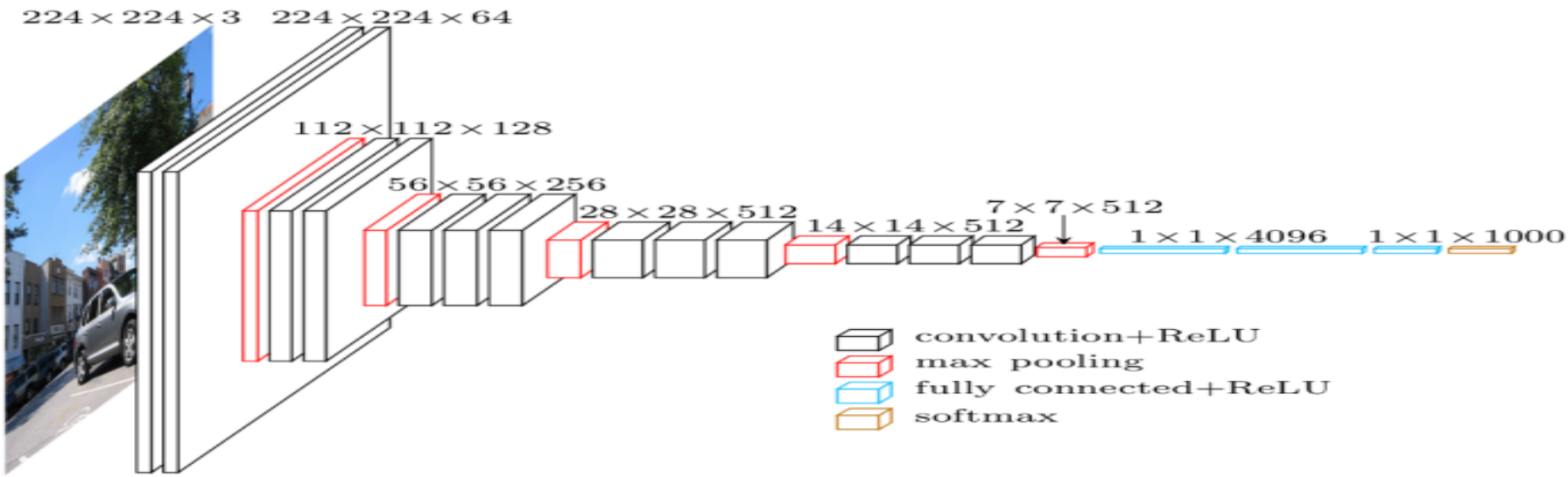
AlexNet is another classic CNN architecture from ImageNet Classification with Deep Convolutional Neural Networks paper by Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever (2012).



- Number of parameters: 60 millions.

VGG-16

VGG-16 from Very Deep Convolutional Networks for Large-Scale Image Recognition paper by Karen Simonyan and Andrew Zisserman (2014). The number 16 refers to the fact that the network has 16 trainable layers (i.e. layers that have weights).

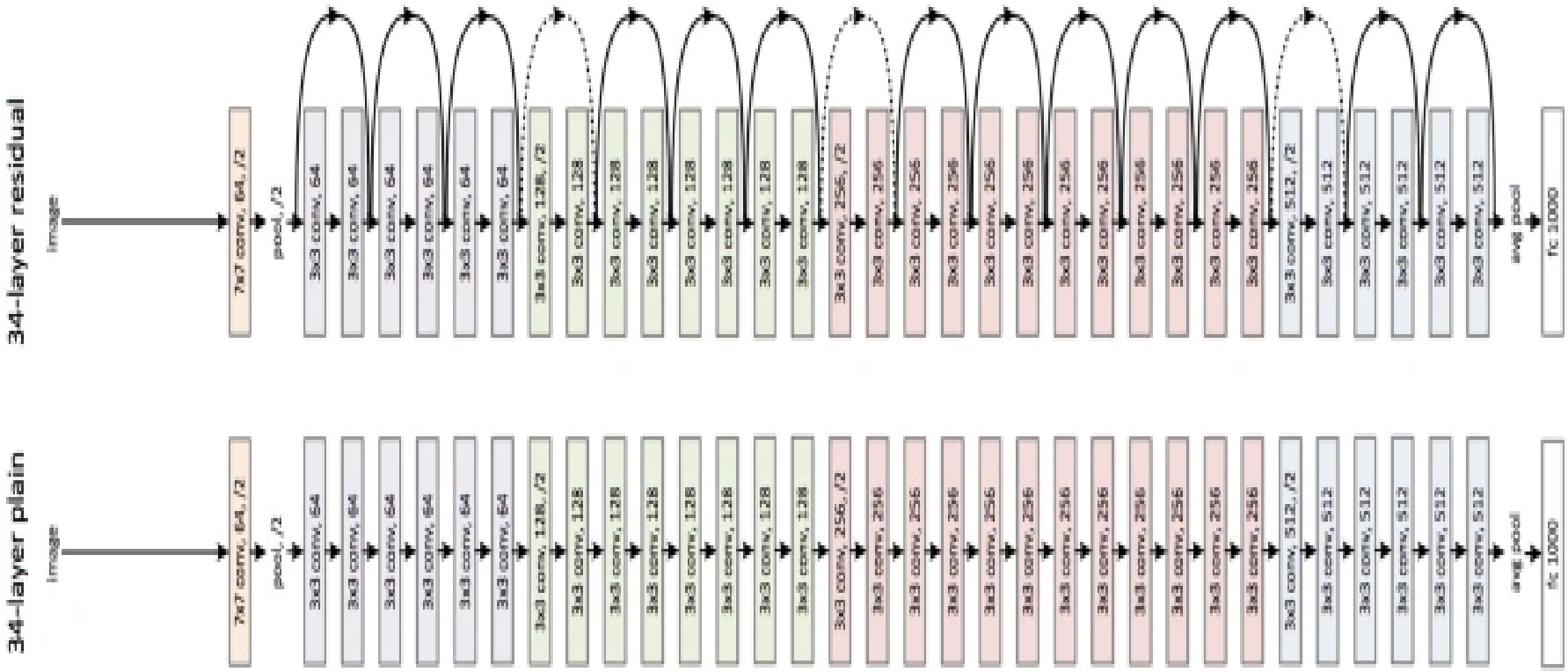


- Number of parameters: 138 millions.
- The strength is in the simplicity: the dimension is halved and the depth is increased on every step (or stack of layers)

Deep Residual Networks (ResNets)

- Introduction to the problem with deeper neural networks
  - Training difficulty and performance degradation with increasing depth
  - Exploding and vanishing gradients problem
- ResNet (Residual Network)
  - Proposed by He et al. in the paper "Deep Residual Learning for Image Recognition" (2015)
  - Solution to the training difficulties of deep networks
  - Implementation of skip connections in ResNets
- Skip Connections in ResNets
  - Description of skip connections as a key component of ResNets
  - Enable information to bypass certain layers
  - Output from one layer is fed to a deeper layer in the network
- Benefits of Skip Connections
  - Addressing the vanishing gradients problem in deep networks
  - Easier gradient flow and improved gradient propagation
  - Facilitating the training of very deep architectures
- ResNet's Impact
  - Successful training of extremely deep neural networks

VGG-16



Comparison of 34 layers ResNet with plain network

Inception

- The motivation of the Inception network is to let the network automatically determine the best filter sizes for each layer.
- Instead of manually choosing filter sizes, we provide choices and allow the network to decide what is most effective in each layer.

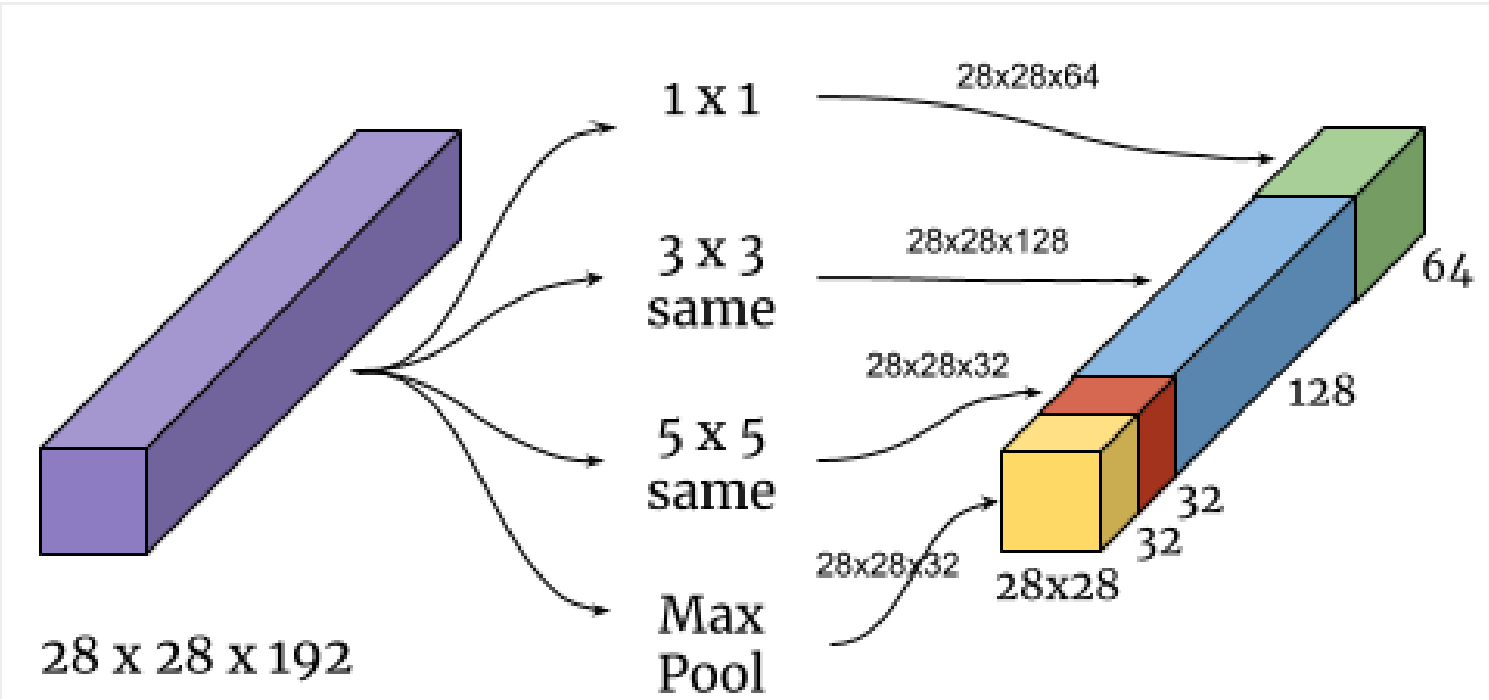


Figure: Inception module architecture

Inception Module with 1x1 Convolutions

- The original Inception module can have high computation cost due to the use of large filters, such as 5x5.
- For example, the computation cost of a single 5x5 filter in the module can be calculated as:

$$\text{Computation Cost} = (28 \times 28 \times 32) \times (5 \times 5 \times 192) \approx 120 \text{ million}$$

- To reduce the computation cost, 1x1 convolutions are introduced in the Inception module.
- Using 1x1 convolutions reduces the number of parameters and operations required for computation.
- The 1x1 convolutions act as a dimensionality reduction step, reducing the depth of the input feature maps.
- By reducing the depth, the subsequent larger convolutions (e.g., 3x3 or 5x5) operate on smaller feature maps, significantly reducing the computation cost.
- The reduction in computation can be estimated to be about 1/10 of

Inception Module with 1x1 Convolutions

- The reduction in computation can be estimated to be about 1/10 of the original cost.

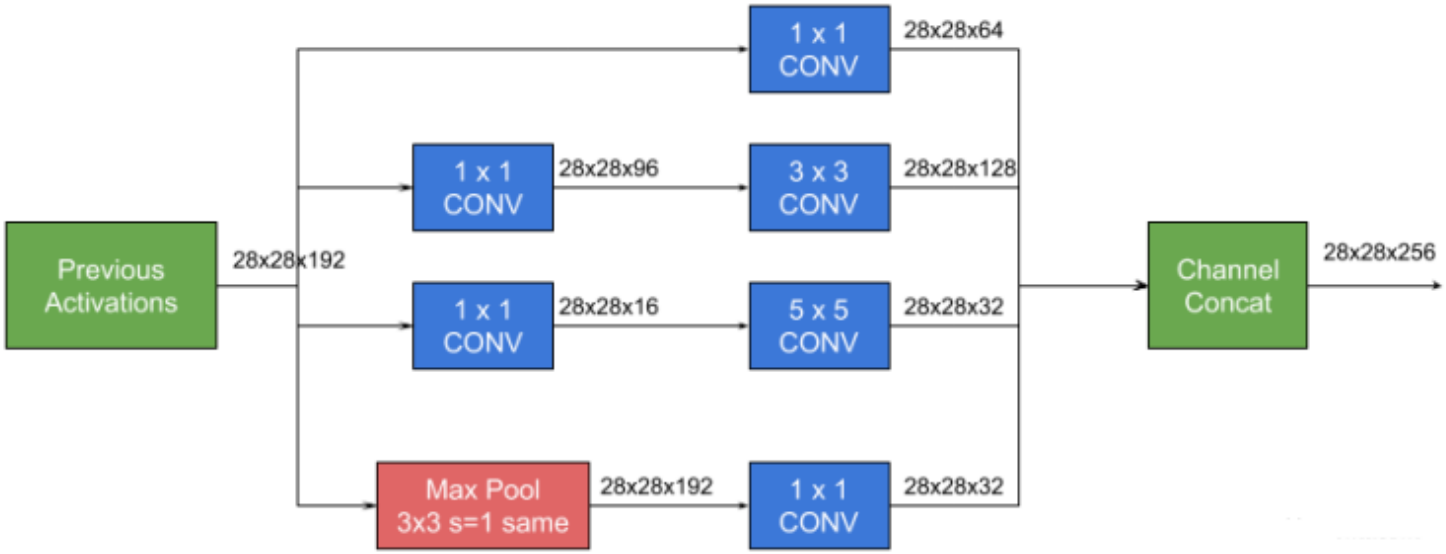


Figure: Inception module with 1x1 convolutions

GoogLeNet with Inception Modules

- GoogLeNet is an inception-based network introduced in the "Going Deeper with Convolutions" paper by Szegedy et al. (2014).
- It features a deep architecture with 9 inception modules stacked together.

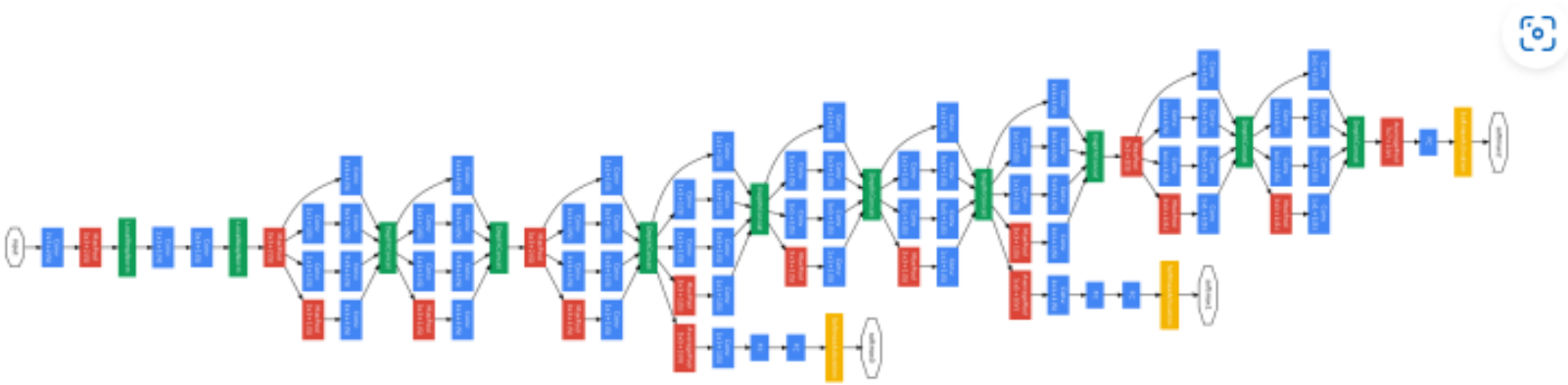


Figure: Architecture of GoogLeNet

Overview

1. Introduction
2. Architecture
3. LeNet-5
4. AlexNet
5. VGGNet
6. GoogLeNet (Inception)
7. ResNet
8. DenseNet
9. Conclusion

LeNet-5

- **Novelty:**
  - Early CNN model.
  - Introduced alternating convolutional and pooling layers.
- **Advantages:**
  - Pioneering architecture.
  - Simple design.
  - Effective for simple tasks.
- **Disadvantages:**
  - Limited scalability.
  - Overfitting on small datasets.

AlexNet

- **Novelty:**
  - Popularized ReLU activation and dropout regularization.
  - Extensive data augmentation techniques.
- **Advantages:**
  - Significant performance improvement on large-scale datasets.
  - Faster training with ReLU activation.
  - Effective dropout regularization to reduce overfitting.
- **Disadvantages:**
  - High computational cost and memory usage.
  - Increased complexity compared to earlier models.

VGGNet

- **Novelty:**
  - Utilized very small (3x3) convolution filters.
  - Demonstrated the effectiveness of increased network depth.
- **Advantages:**
  - Simple and uniform design.
  - Improved performance with increased depth.
  - Widely used for transfer learning.
- **Disadvantages:**
  - High computational expense and memory usage.
  - Large number of parameters leading to long training times.

GoogLeNet (Inception)

- **Novelty:**
  - Introduced Inception modules with parallel convolutions.
  - Efficient architecture with fewer parameters.
- **Advantages:**
  - High efficiency with fewer parameters.
  - Effective multi-scale feature extraction.
  - Reduced overfitting with 1x1 convolutions for dimensionality reduction.
- **Disadvantages:**
  - Complex design and implementation.
  - Requires careful tuning of various components.

ResNet

- **Novelty:**
  - Introduced residual connections (skip connections).
  - Enabled training of very deep networks.
- **Advantages:**
  - Allows for very deep networks, capturing complex features.
  - Mitigates vanishing gradient problem.
  - Outstanding performance on various benchmarks.
- **Disadvantages:**
  - Increased complexity in model design and implementation.
  - High computational requirements.

DenseNet

- **Novelty:**
  - Introduced dense connections between layers.
  - Improved gradient flow and feature reuse.
- **Advantages:**
  - Efficient gradient flow for stable and efficient training.
  - High parameter efficiency with fewer parameters.
  - Improved learning and generalization through feature reuse.
- **Disadvantages:**
  - Increased memory consumption during training.
  - Complex dense block structure.

Conclusion

- Summarize the key contributions of each architecture.
- Highlight the importance of understanding the advantages and disadvantages.
- Emphasize the role of novelty in advancing CNN research.