SURGICAL VIDEO ANALYSIS

VL SAHITHI
IMT2020047

DATASET

Original image
Instrument annotations
Task of instruments
Target organ

WHAT CAN WE DO?

predict the phase
surgeon skills

TRIPLET ANNOTATIONS

INSTRUMENT ANNOTATIONS

VERB ANNOTATIONS

TARGET ANNOTATIONS

DEEP NN

A deep neural network (DNN) is a type of neural network that consists of multiple layers of interconnected neurons, allowing it to learn complex and hierarchical representations of data. Each layer in a DNN consists of a set of neurons that are connected to the neurons in the previous and subsequent layers.

DNN -> CNN

While DNN uses many fully-connected layers, CNN contains mostly convolutional layers.

In a Convolutional Neural Network (CNN), the input data is typically represented as a three-dimensional tensor,

Dimensions:
width
depth
height

CNN

CONVOLUTIONAL LAYERS: A convolutional layer is the main building block of a CNN. It contains a set of filters (or kernels), parameters of which are to be learned throughout the training. The size of the filters is usually smaller than the actual image. Each filter convolves with the image and creates an feature map.

POOLING LAYERS: Pooling layers are used to reduce the spatial dimensions of the feature maps produced by the convolutional layers.

FUlIY CONNECTED LAYERS: These fully connected layers perform classification based on the features extracted by the previous convolutional and pooling layers.

CNN

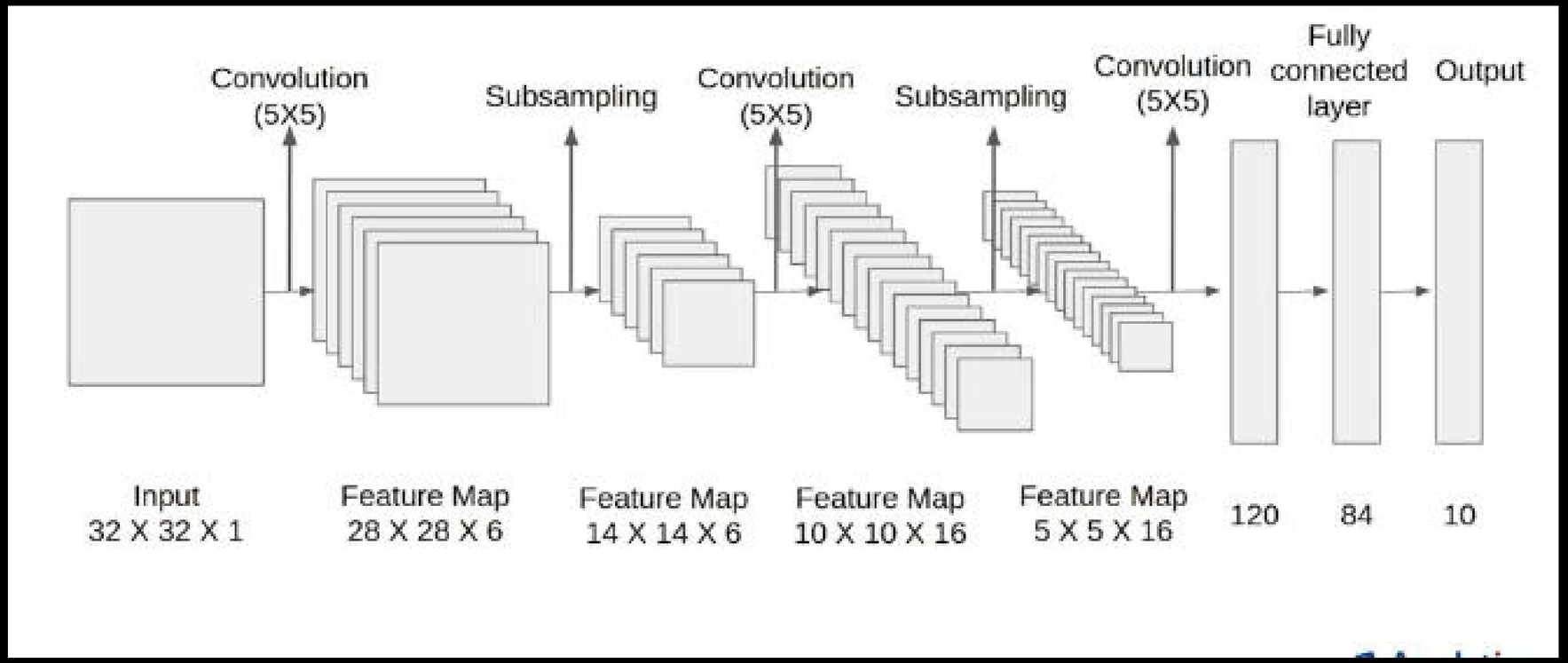
Stride - filter is moved across the image from left to right with a one-pixel column change

Padding - Without padding, the output feature maps are smaller than the input data, which can result in a loss of information at the edges of the input data.

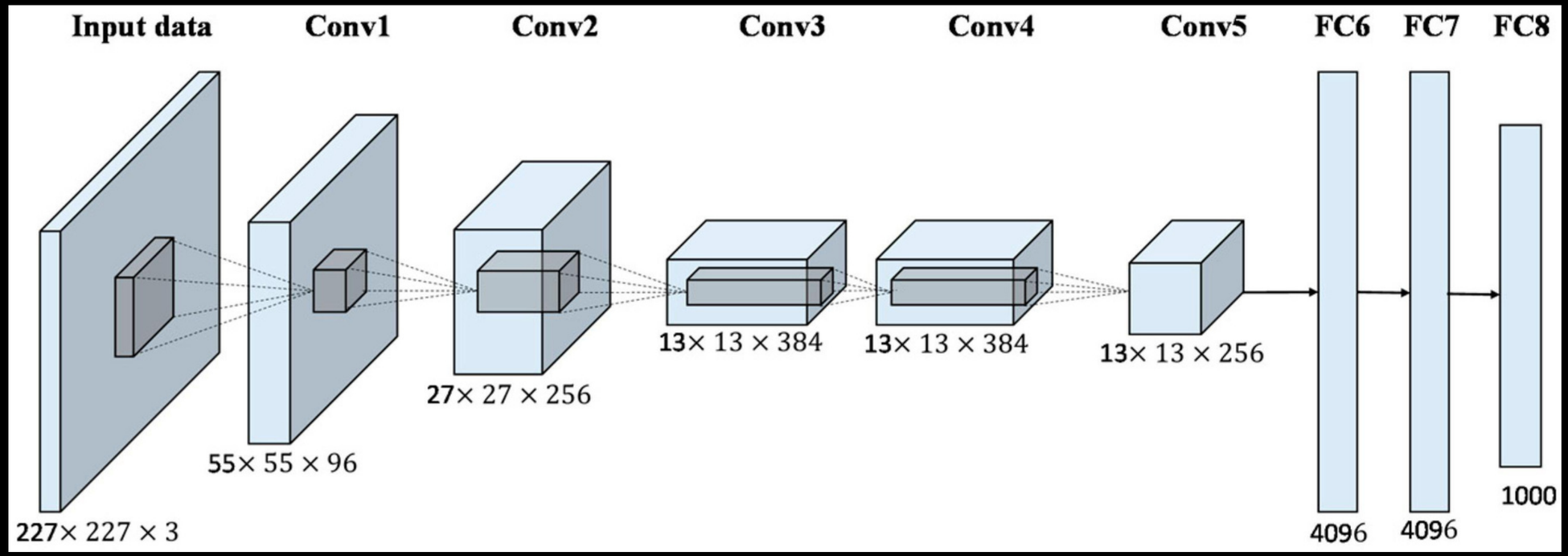
Pooling - Pooling is a common operation in Convolutional Neural Networks (CNNs) that is used to reduce the spatial size of the input data and extract the most important features.

Max pooling
Average pooling

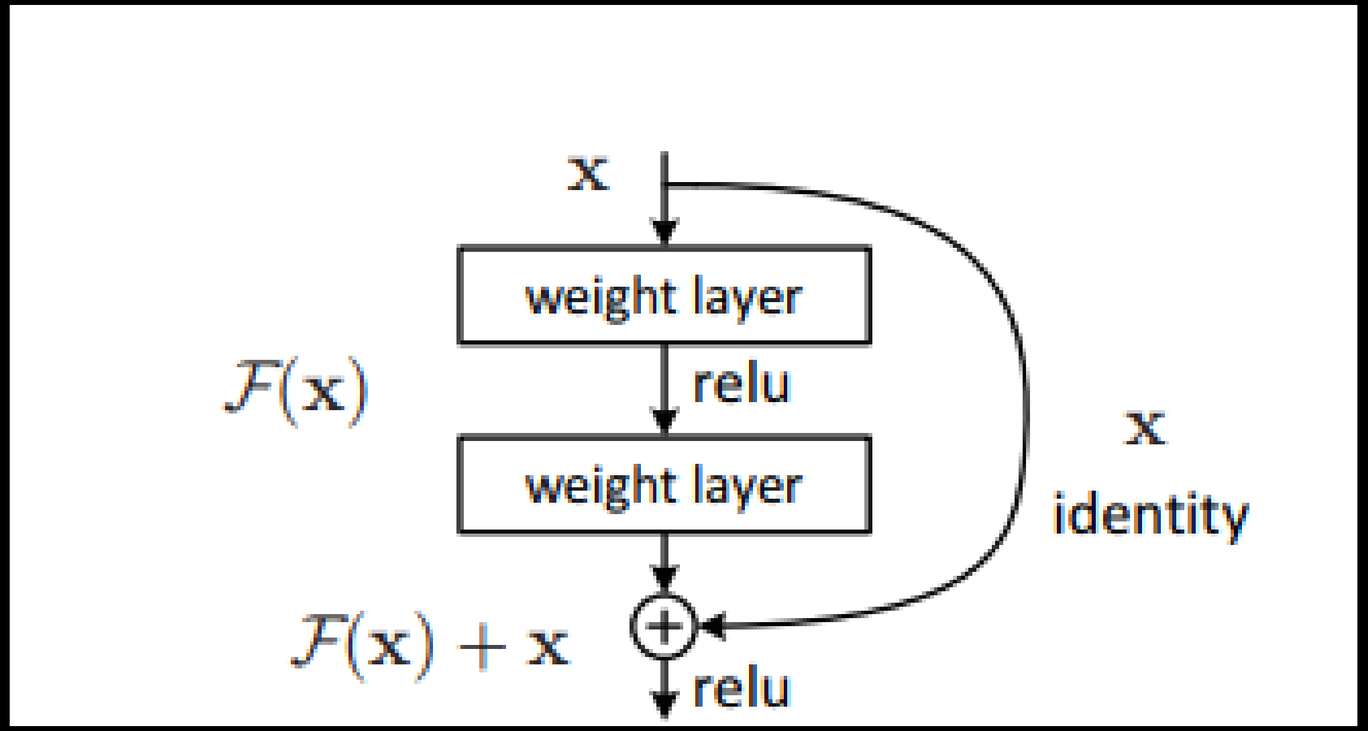
LENET5


Convolution
(5X5)
Subsampling
Convolution
(5X5)
Subsampling
Convolution
(5X5)
Fully
connected
layer
Output
Input
32 X 32 X 1
Feature Map
28 X 28 X 6
Feature Map
14 X 14 X 6
Feature Map
10 X 10 X 16
Feature Map
5 X 5 X 16
120
84
10

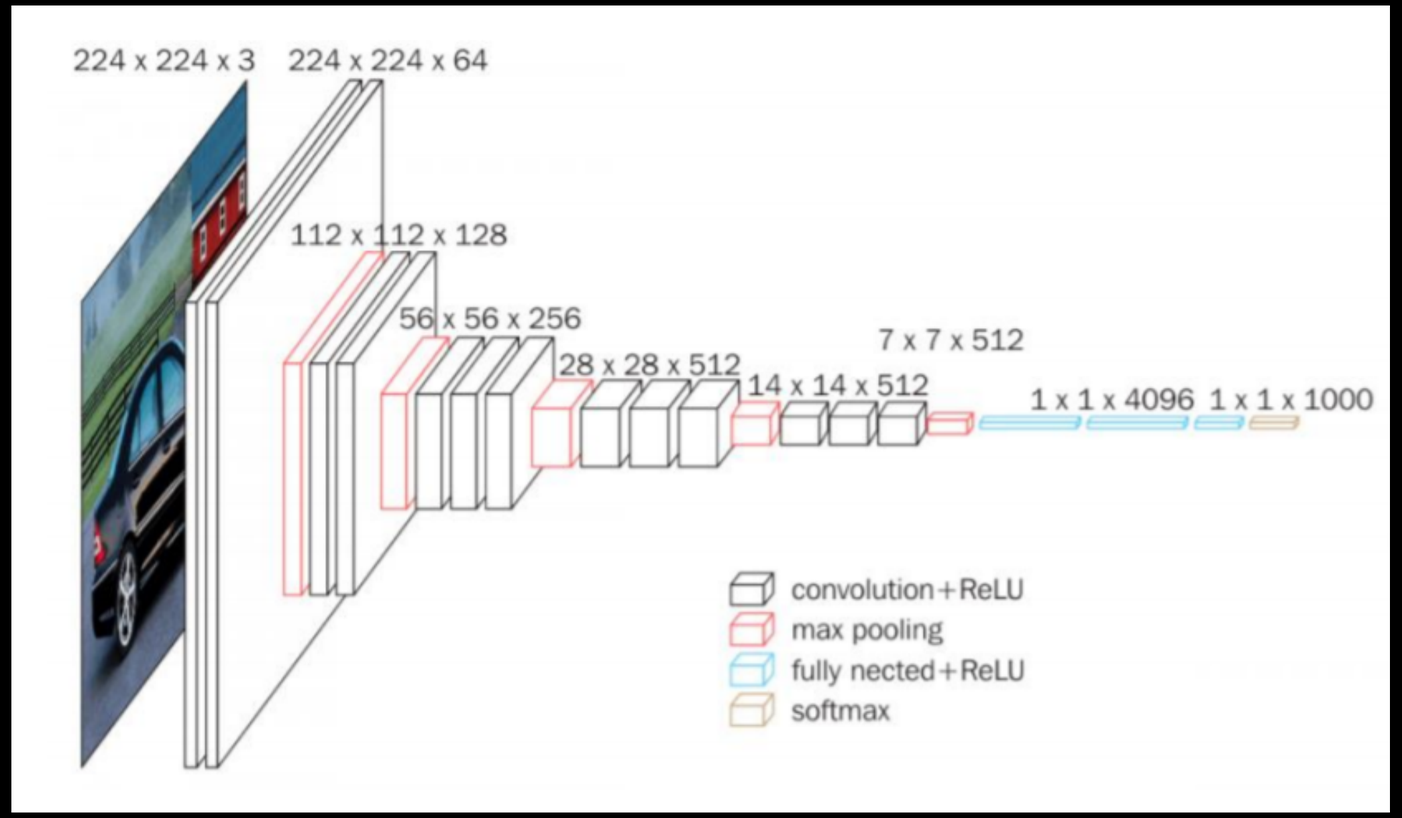
ALEXNET


Input data
Conv1
Conv2
Conv3
Conv4
Conv5
FC6
FC7
FC8
227× 227 × 3
55× 55 × 96
27× 27 × 256
13× 13 × 384
13× 13 × 384
13× 13 × 256
4096
4096
1000

RESNET


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

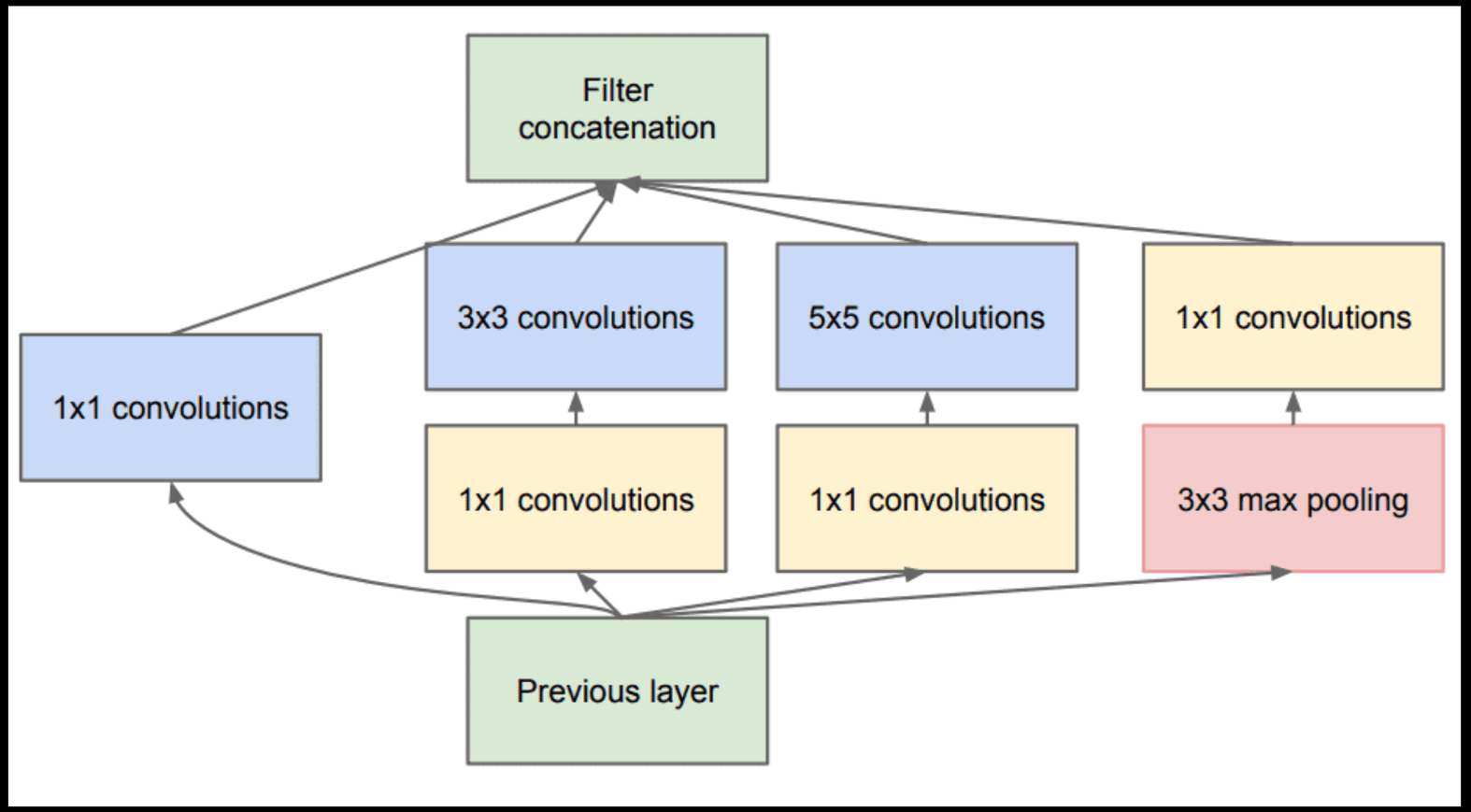
VGG


224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

VGG

The input to the convolution neural network is a fixed-size 224 × 224 RGB image.
VGG16 has a total of 16 layers that has some weights. Only Convolution and pooling layers are used.
Always uses a 3 x 3 Kernel for convolution. 20 , 2×2 size of the max pool.
To adapt VGG for object detection, we typically add additional layers to the architecture. One common approach is to add a set of layers known as the Region Proposal Network (RPN) on top of the existing VGG architecture.
The VGG network is a deep CNN with 16 or 19 layers, depending on the version used. The network consists of a series of convolutional layers, each followed by a ReLU activation function and a max pooling layer. The final layers of the network are fully connected layers that perform the classification task.
Transfer learning is used to adapt the VGG network to the surgical tool presence detection task. Specifically, the weights of the pre-trained VGG network are used as initialization for the network, and the final layer of the network is re-trained on the surgical tool dataset.
The final layer of the network is re-trained on the surgical tool dataset. During re-training, the weights of the final layer are adjusted to minimize the loss function on the training set.

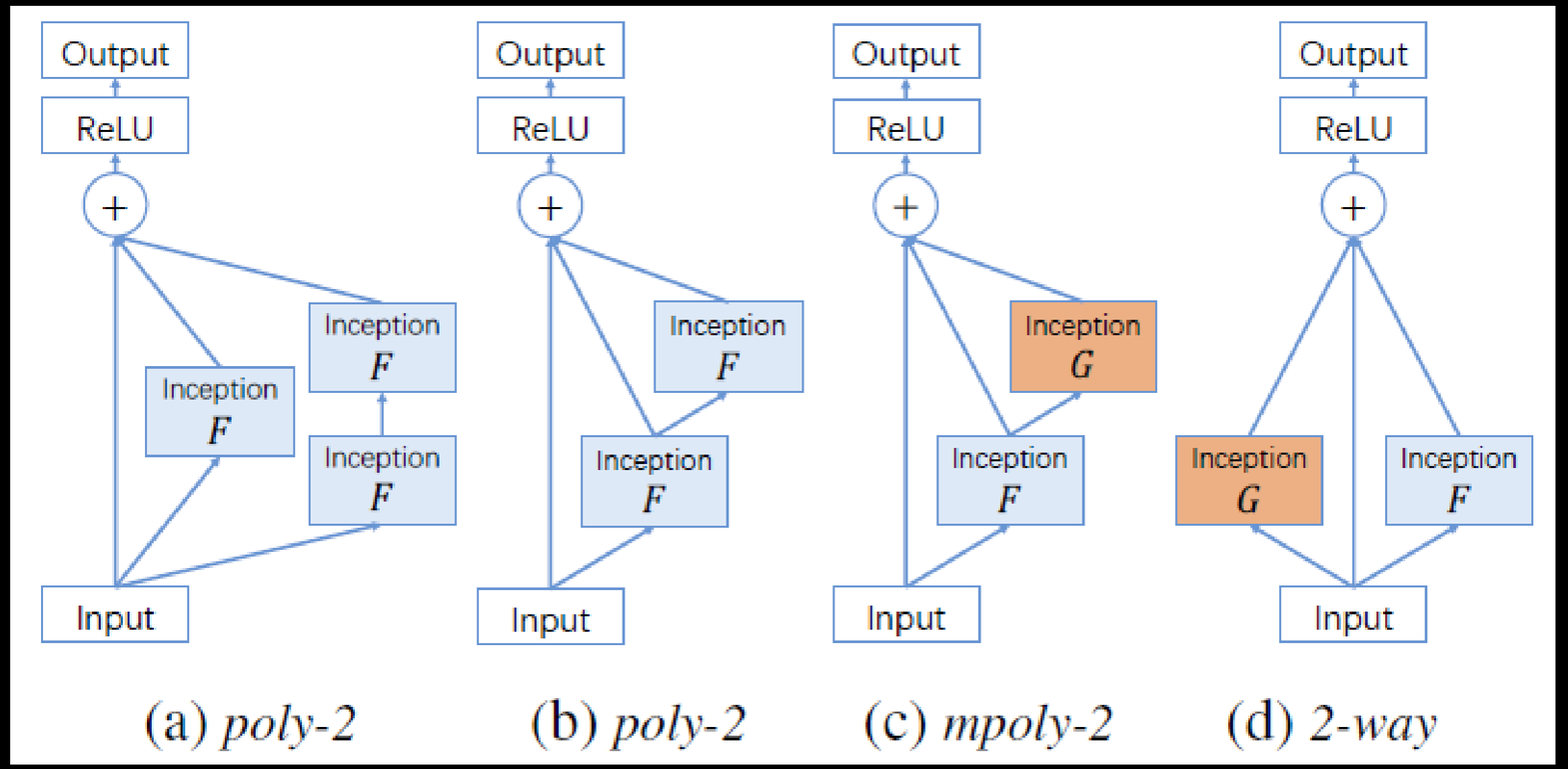
INCEPTION


Filter concatenation
3x3 convolutions
5x5 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
3x3 max pooling
Previous layer

INCEPTION

The InceptionNet design is made up of nine inception modules stacked on top of each other, with max-pooling layers between them.
It is made up of 22 layers (27 with the pooling layers). After the last inception module, it employs global average pooling.
INCEPTION MODULE:
The Inception module is designed to improve the accuracy of convolutional neural networks (CNNs) while reducing the number of parameters. It achieves this by performing multiple convolutions with different filter sizes and pooling operations on the same input layer in parallel. The outputs of these operations are then concatenated and fed into the next layer.
The core idea behind the Inception module is to capture features at multiple scales using filters of different sizes. By using filters of different sizes in parallel, the network can capture both local and global features of an image. This allows the network to learn more complex and abstract representations of the input, leading to better performance on various computer vision tasks.
The Inception module typically consists of a 1x1 convolution layer, followed by 3x3 and 5x5 convolution layers, and a max pooling layer. These operations are performed in parallel, and the resulting feature maps are concatenated along the depth dimension.

POLYNET


Output
ReLU
+
Inception F
Inception F
Inception F
Input
(a) poly-2
Output
ReLU
+
Inception F
Inception F
Input
(b) poly-2
Output
ReLU
+
Inception G
Inception F
Input
(c) mpoly-2
Output
ReLU
+
Inception G
Inception F
Input
(d) 2-way

POLYNET

The PolyNet architecture is based on the Inception architecture, which is known for its use of "Inception modules"
PolyNet traverses the whole network and explores the entire space, making decisions about weights and structure so that it may automate improvements to increase performance and functionality, with better results for the end user.
POLYINCEPTION MODULE:
A PolyInception module is similar to an Inception module, but it includes an additional "polynomial activation function" that is applied to the output of the module. This activation function is designed to increase the expressive power of the network by allowing it to learn more complex features.
In addition to the PolyInception modules, the PolyNet architecture also includes several other features that improve its efficiency and scalability. For example, it uses "multi-scale feature maps" that allow it to capture features at different levels of granularity, and it uses "grouped convolutions" that reduce the number of parameters in the network.
In a standard convolutional layer, a set of filters is applied to the entire input volume, which includes all the channels produced by the previous layer. This can lead to a large number of parameters, particularly when the input volume is high-dimensional.
Grouped convolutions address this issue by splitting the input channels into groups and applying a set of filters to each group separately. This reduces the number of parameters in the layer

WIDERESNET

Problem: *feature reuse problem, in which some feature changes or blocks may contribute relatively little to learning*.
Wide ResNet was formed to solve this issue. The major learning potential of deep residual networks, is attributable to the residual units, whereas depth has a supportung role. ResNet was made wide rather than deep to take use of the residual blocks' strength
The architecture consists of multiple residual blocks, where each block contains two or three convolutional layers followed by a batch normalization layer and a ReLU activation function. The residual connections in the blocks allow the network to learn a residual mapping between the input and output of the block.
Wider convolutional layers in a neural network refer to increasing the number of filters in a convolutional layer. In a convolutional layer, a filter is a small matrix that is applied to the input data to extract features. By increasing the number of filters in a convolutional layer, the network can learn more diverse and complex features from the input data.

ZFNET

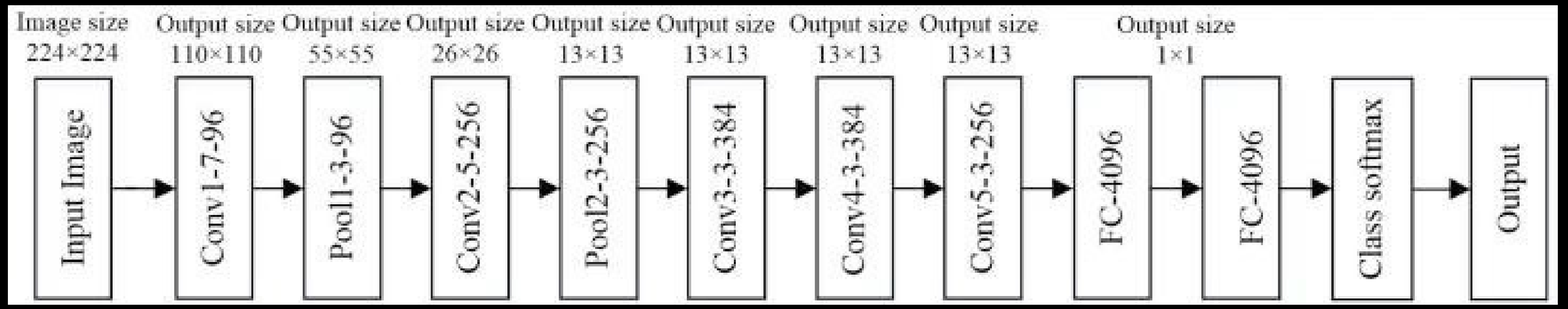


Image size
224×224
Output size
110×110
Output size
55×55
Output size
26×26
Output size
13×13
Output size
13×13
Output size
13×13
Output size
13×13
Output size
1×1
Input Image
Conv1-7-96
Pool1-3-96
Conv2-5-256
Pool2-3-256
Conv3-3-384
Conv4-3-384
Conv5-3-256
FC-4096
FC-4096
Class softmax
Output

ZFNET

The architecture of ZFNet is similar to the AlexNet architecture
It consists of 8 layers, including 5 convolutional layers, 2 fully connected layers, and a softmax output layer. The first convolutional layer has a large receptive field of 11x11 pixels, which allows it to capture global features of the input image. The subsequent layers have smaller receptive fields, allowing them to capture more local features.
In ZFNet we use a visualization technique called Deconvolutional Networks, which allows the network to produce a heat map showing which parts of the input image contributed the most to the classification decision. This visualization technique helps to understand how the network is making its decisions and can be used for debugging and improving the model.
ZFNet uses a visualization technique called Deconvolutional Networks to understand how the model is making its classification decisions by generating a heatmap that highlights the important regions of the input image that contributed to the decision.
To generate the heatmap, ZFNet modifies the CNN architecture by adding a deconvolutional layer that takes the output of the final classification layer and reverses the operation of the convolutional layers. This deconvolutional layer produces a feature map that highlights the important regions of the input image that contributed to the decision.

WORKFLOW

1 DATA PREPARATION

2 FEATURE EXTRACTION

3 MODEL ARCHITECTURE

4 TRAINING

5 EVALUATE

6 INFERENCE

Thank you