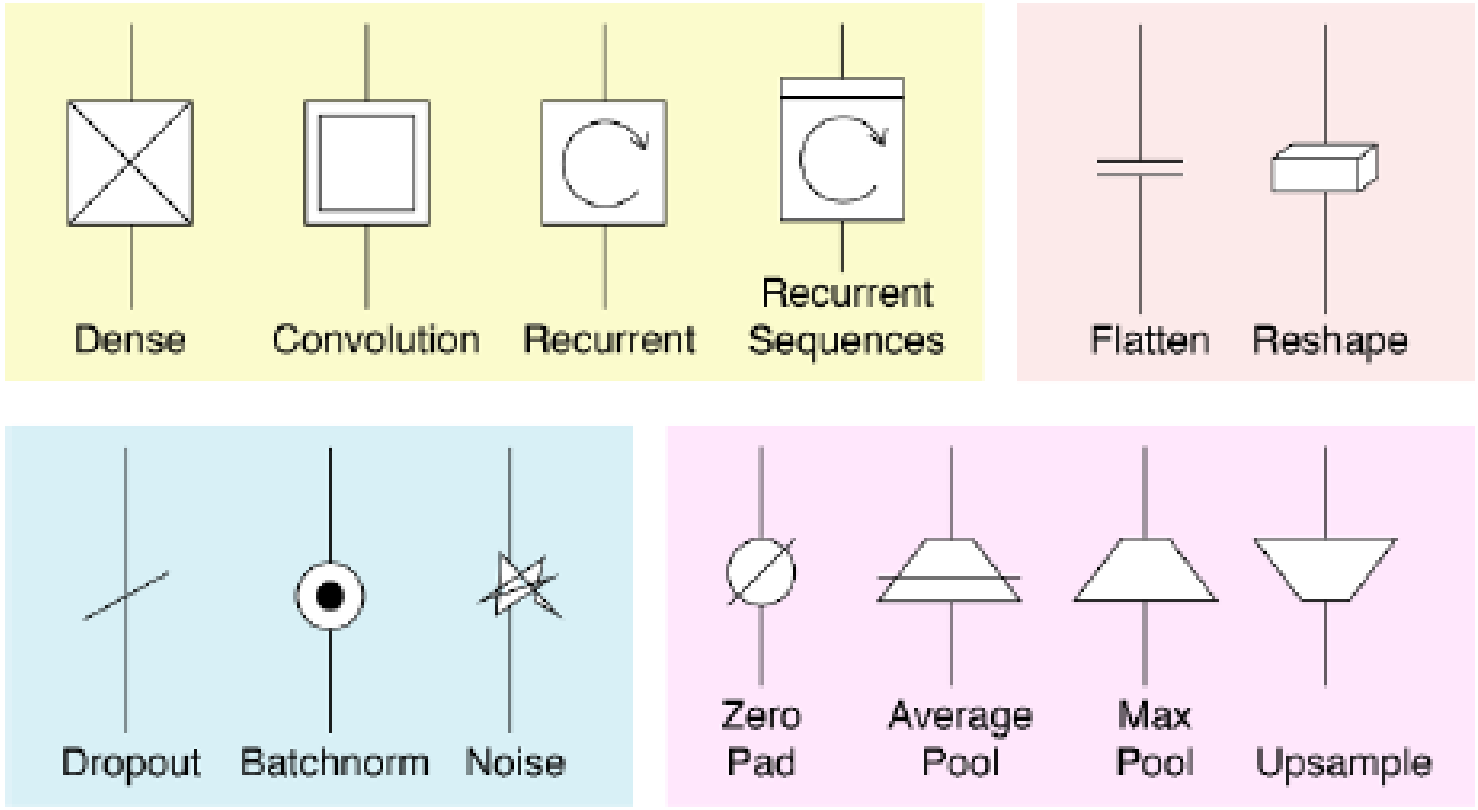
Deep Learning Models for Computer Vision

Image by kirkh.deviantart.com

Schematic Neural Network Symbols



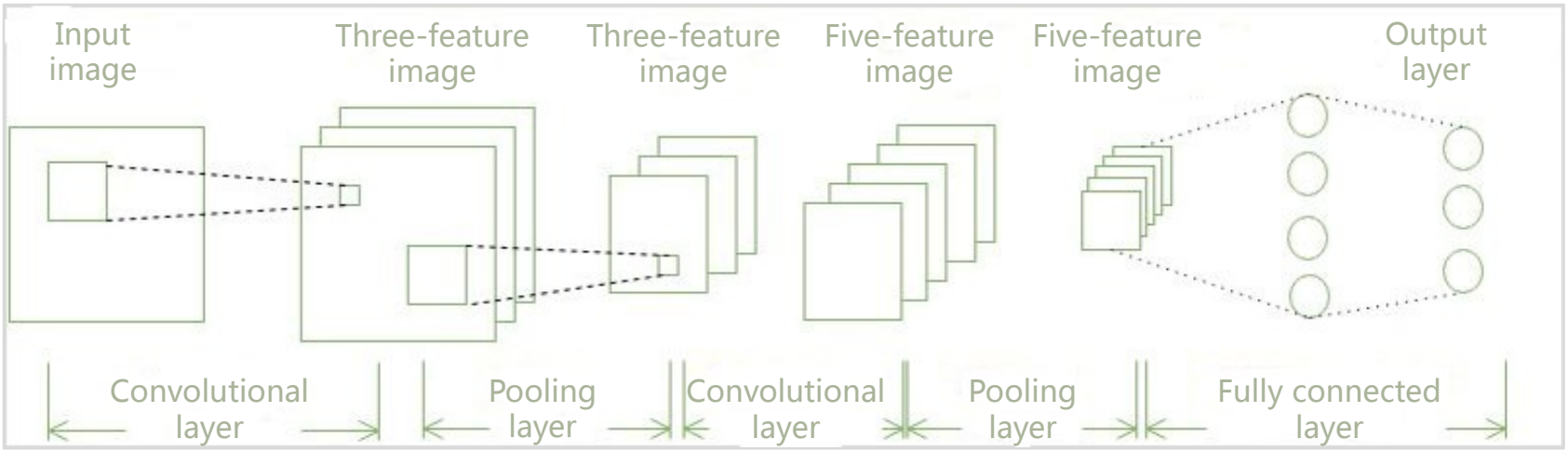
Convolutional Neural Network

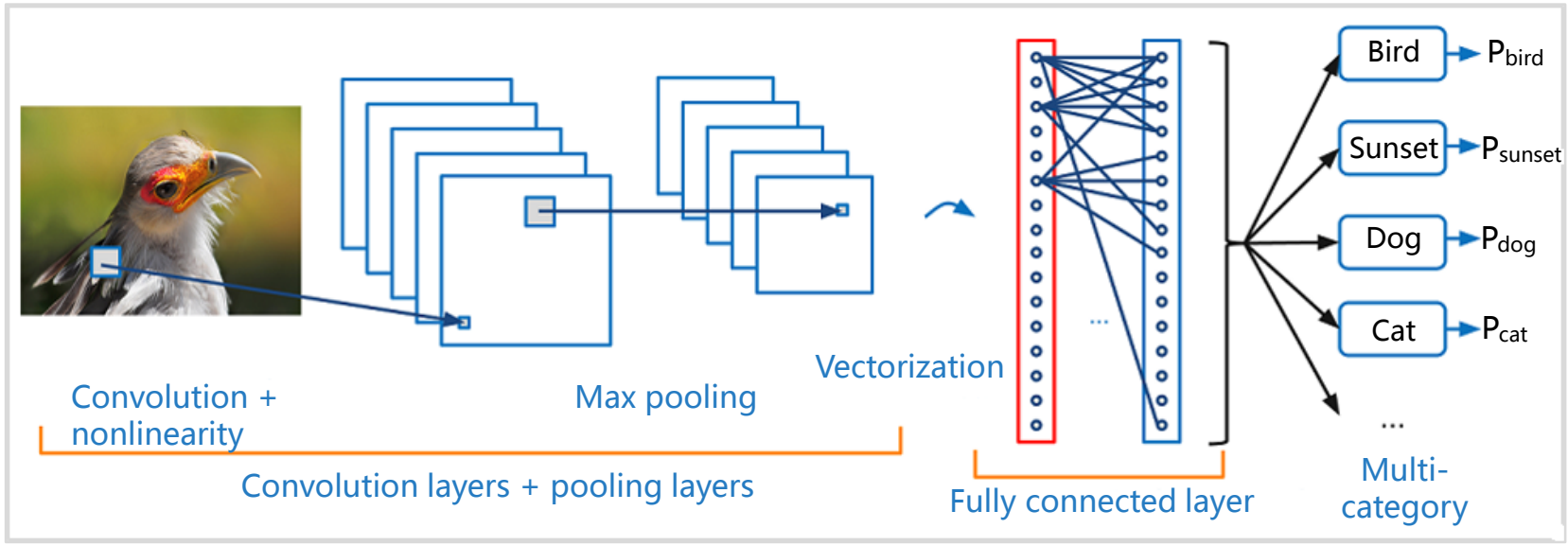
- A convolutional neural network (CNN) is a feedforward neural network. Its artificial neurons may respond to **surrounding units within the coverage range**. CNN excels at image processing. It includes a **convolutional layer**, a **pooling layer**, and a **fully connected layer**.
- In the 1960s, Hubel and Wiesel studied cats' cortex neurons used for local sensitivity and direction selection and found that their unique network structure could simplify feedback neural networks. They then proposed the CNN.
- Now, CNN has become one of the research hotspots in many scientific fields, especially in the pattern classification field. The network is widely used because it can avoid complex pre-processing of images and directly input original images.

Main Concepts of CNN

- **Local receptive field:** It is generally considered that human perception of the outside world is from local to global. **Spatial correlations among local pixels of an image are closer than those among distant pixels**. Therefore, each neuron does not need to know the global image. It only needs to know the local image. The local information is combined at a higher level to generate global information.
- **Parameter sharing:** One or more filters/kernels may be used to scan input images. Parameters carried by the filters are weights. In a layer scanned by filters, each filter uses the same parameters during weighted computation. Weight sharing means that when each filter scans an entire image, parameters of the filter are fixed.

Architecture of Convolutional Neural Network





Single-Filter Calculation

- Description of convolution calculation

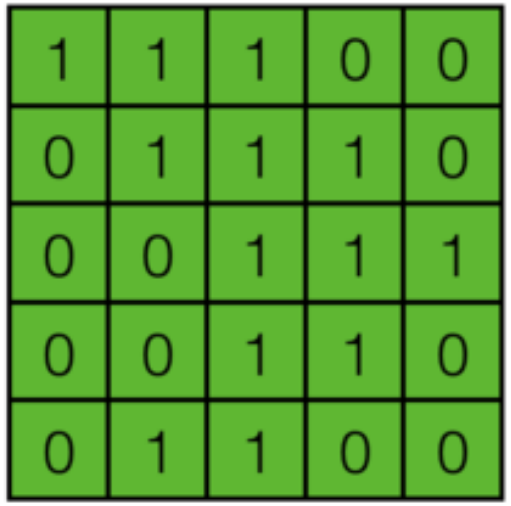
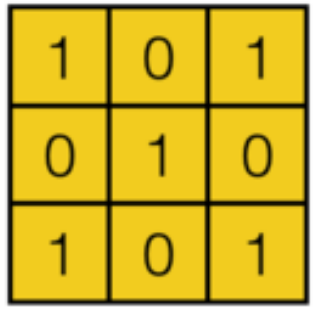
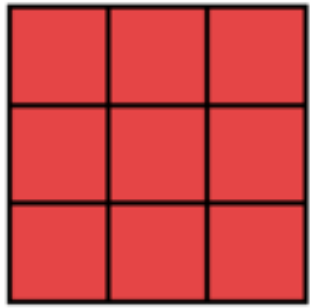


image 5*5



bias=0

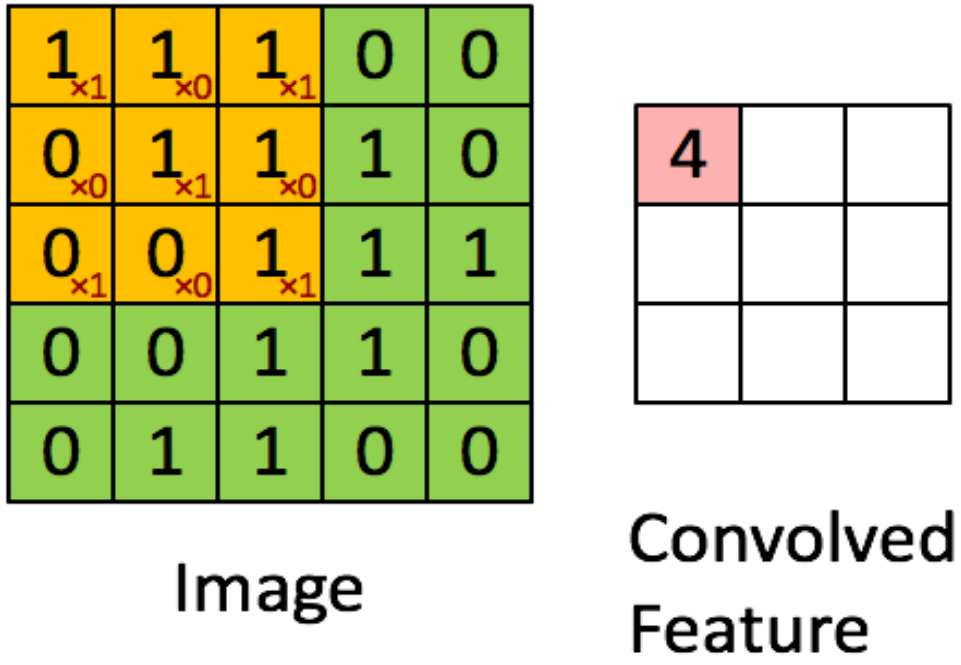
filter 3*3



feature map 3*3

Single-Filter Calculation (cont.)

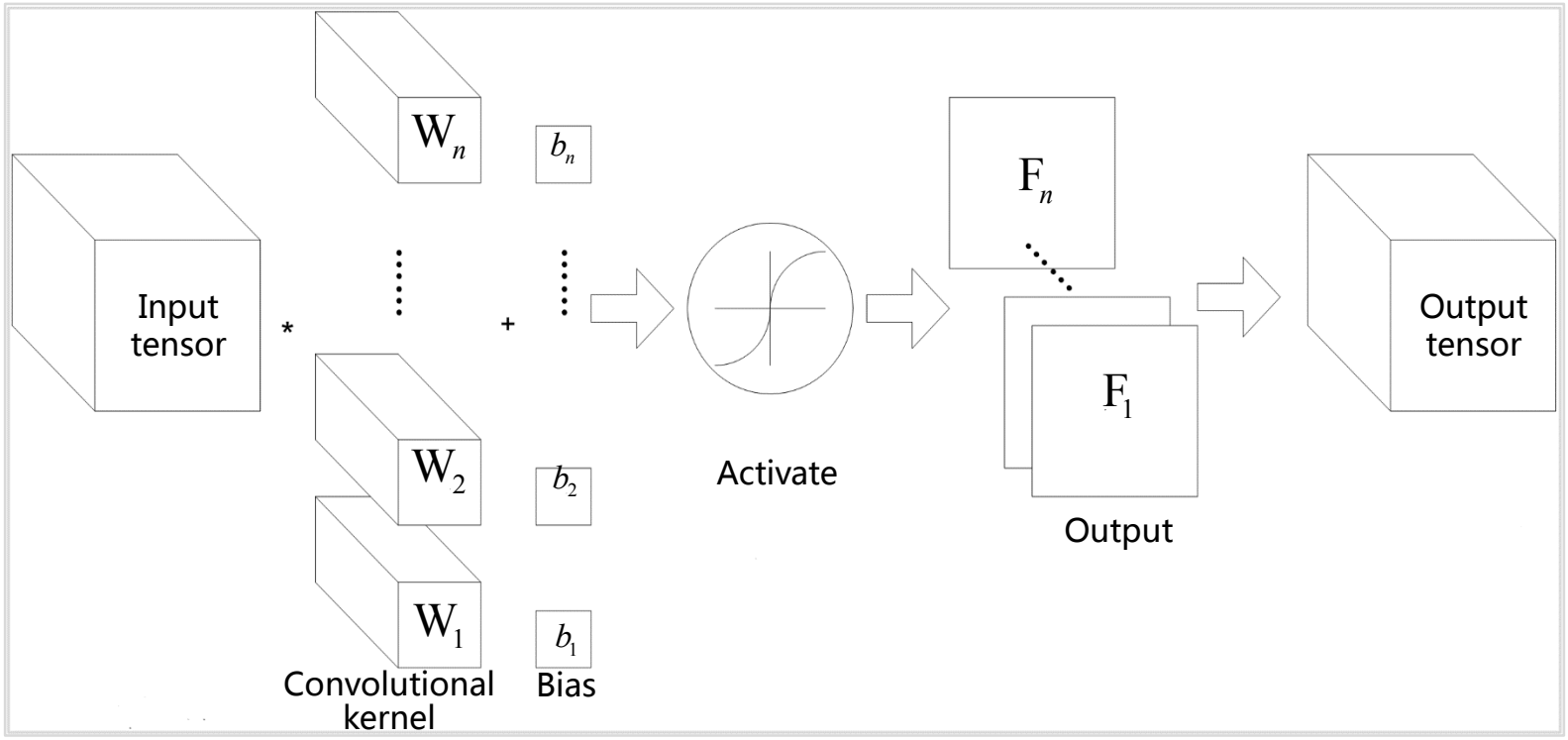
- Demonstration of the convolution calculation



Han Bingtao, 2017, Convolutional Neural Network

Convolutional Layer

- The basic architecture of a CNN is multi-channel convolution consisting of multiple single convolutions. The output of the previous layer (or the original image of the first layer) is used as the input of the current layer. It is then convolved with the filter in the layer and serves as the output of this layer. The convolution kernel of each layer is the weight to be learned. Similar to FCN, after the convolution is complete, the result should be biased and activated through activation functions before being input to the next layer.

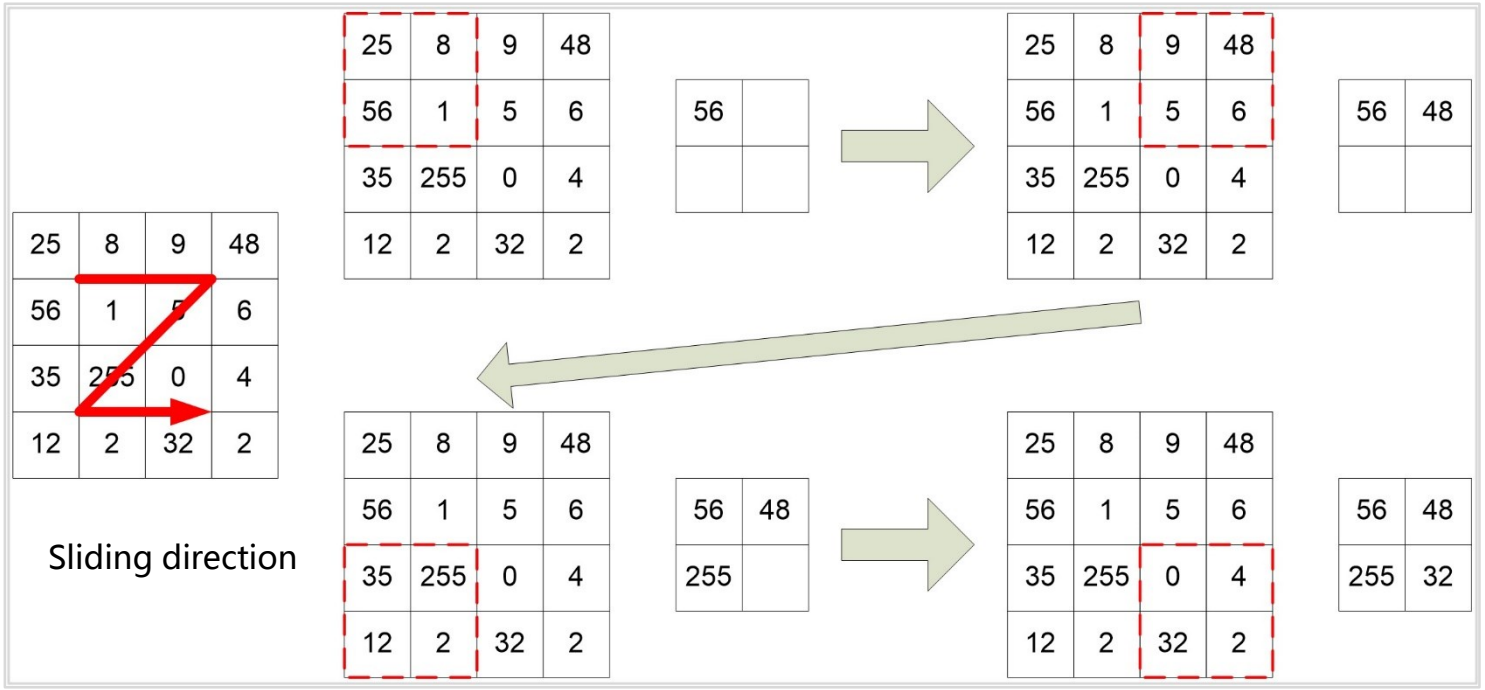


Convolutional Layer (cont.)

- Volume Size $W1 \times H1 \times D1$
- Four Hyperparameters:
  - Number of filters K
  - Filter size F
  - Stride S
  - Zero Padding P
- Output volume size $W2 \times H2 \times D2$
  - $W2 = [(W1 - F + 2P)/S] + 1$
  - $H2 = [(H1 - F + 2P)/S] + 1$
  - $D2 = K$
- Number of parameter : $F \cdot F \cdot D1$ weights per filter , total = $(F \cdot F \cdot D1) \cdot K + K$ (for bias).

Pooling Layer

- Pooling combines nearby units to reduce the size of the input on the next layer, reducing dimensions. Common pooling includes max pooling and average pooling. When max pooling is used, the maximum value in a small square area is selected as the representative of this area, while the mean value is selected as the representative when average pooling is used. The side of this small area is the pool window size. The following figure shows the max pooling operation whose pooling window size is 2.



Pooling Layer (cont.)

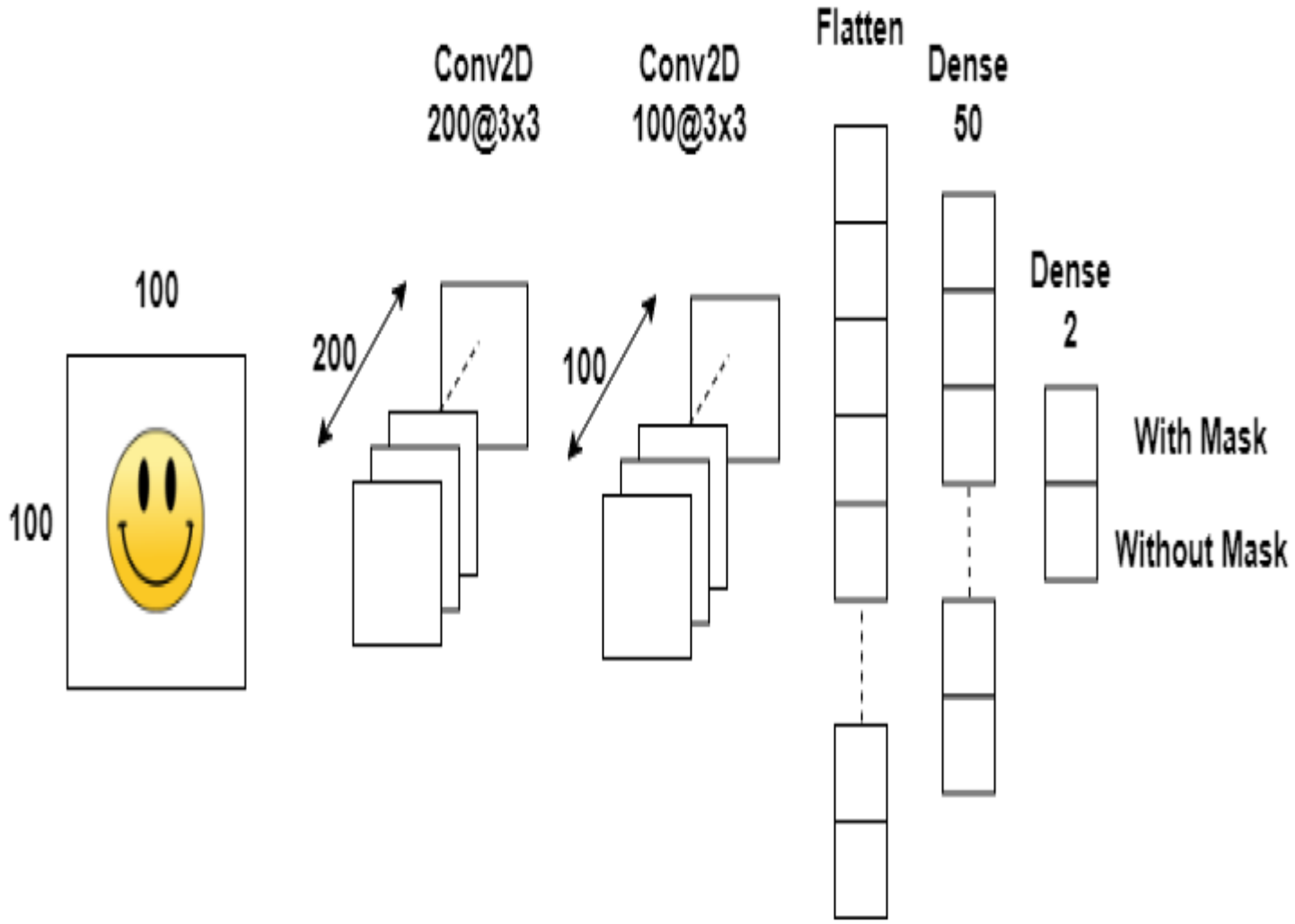
- Volume Size $W1 \times H1 \times D1$
- Two Hyperparameters:
  - Filter Size F
  - Stride S
- Output volume size $W2 \times H2 \times D2$
  - $W2 = [(W1 - F)]/S + 1$
  - $H2 = [(H1 - F)]/S + 1$
  - $D2 = D1$
- zero parameters : it computes a certain function of the input

Fully Connected Layer

- The fully connected layer is essentially a classifier. The features extracted on the convolutional layer and pooling layer are straightened and placed at the fully connected layer to output and classify results.
- Generally, the Softmax function is used as the activation function of the final fully connected output layer to combine all local features into global features and calculate the score of each type.

$$\sigma(\mathrm{z})_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

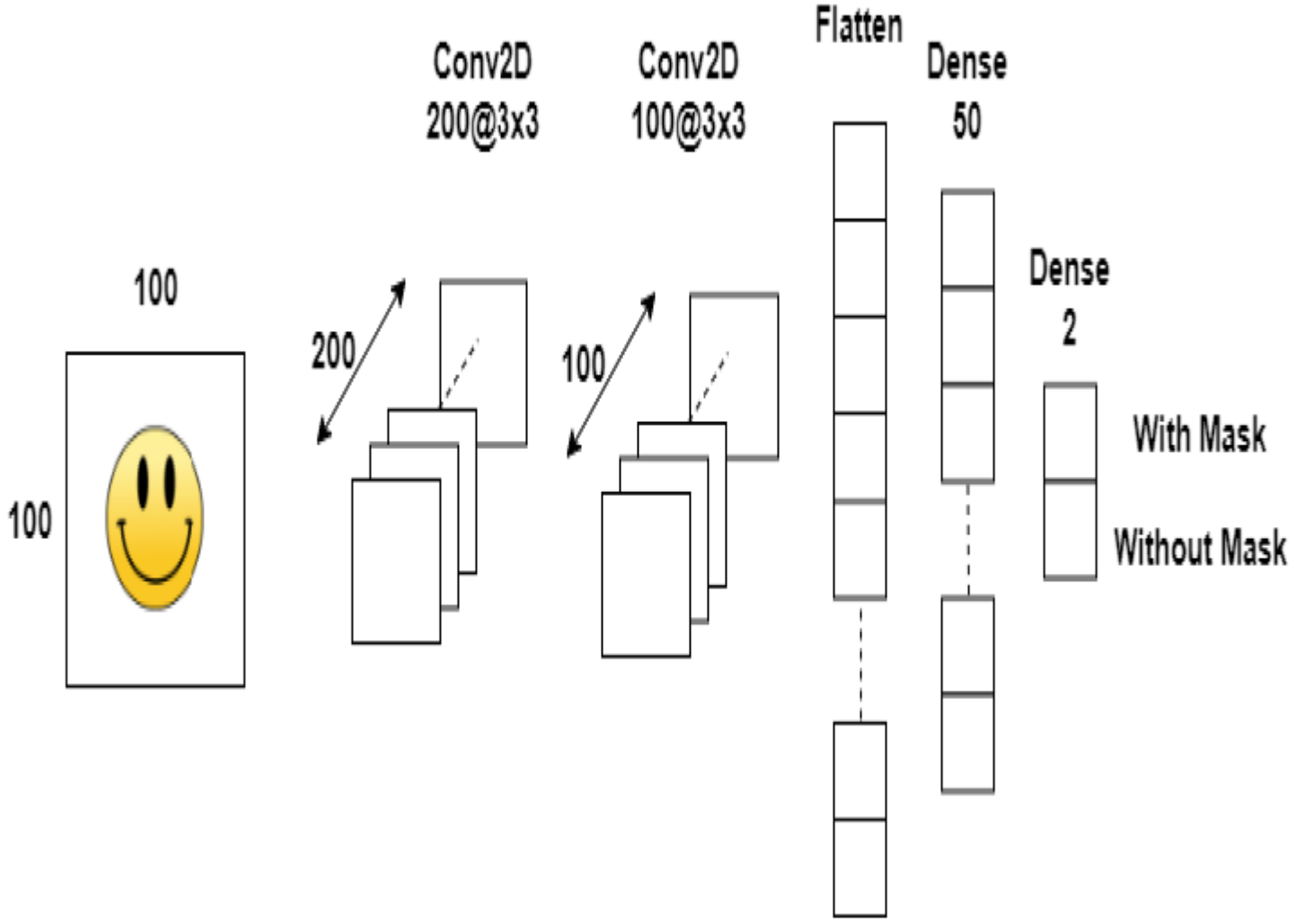
CNN Example



https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9315927

CNN Example

- For the following CNN architecture calculate the output volume size and the number of parameters for each layer
- The CNN doesn't apply Padding and Stride = 1



https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9315927

Layer 1

- Input: colored image of size 100 x 100 x 3 (the colors channel Red, Green and Blue)
  - K = 200
  - F = 3
  - P = 0
  - S = 1
- $O = (W-F+2P)/S+1 = (100 - 3 + 0)/1 + 1 = 98$
- Output volume size = 98 x 98 x 200
- No. of parameter = $F \cdot F \cdot D1$ = 3 x 3 x 3 = 27 for each filter
- Total No. of Parameter = $(F \cdot F \cdot D1) \cdot K + K = 27 \text{ X } 200 + 200 = 5600$

Layer 2

- Input: output of layer 1 with volume 98 x 98 x 200
  - K = 100
  - F = 3
  - P = 0
  - S = 1
- $O = (W-F+2P)/S+1 = (98 - 3 + 0)/1 + 1 = 96$
- Output volume size = 96 x 96 x 100
- No. of parameter = $F \cdot F \cdot D1$ = 3 x 3 x 200 = 1800 for each filter
- Total No. of Parameter = $(F \cdot F \cdot D1) \cdot K + K$ = 1800 X 100 + 100 = 180,100

Layer 3: Fully Connected Layer (Dense)

- Input: output of layer 2 with volume 96 x 96 x 100
  - Layer 3 consists of 50 Neuron
- Total No. of Parameter for Neural Network Layer = N x M + M
  = (96 x 96 x 100) x 50 + 50 = 46,080,050

Layer 4: Fully Connected Layer (Dense)

- Layer 3 consists of 50 Neuron
- Layer 4 consists of 2 Neuron
- Total No. of Parameter for Neural Network Layer = $N \times M + M$
  $= 50 \times 2 + 2 = 102$

CNN Example

- How many parameter will be needed if Fully Connected layer used instead of first convolutional layer?
- Input = 100 x 100 x 3 = 30,000
- The convolutional layer consists of 200 3x3 Filter so, 3 x 3 x 3 x 200 = 5400 neurons will be needed
- Total No. of Parameter for Neural Network Layer = N x M + M
  = 30,000 x 5400 + 5400 = 162,005,400

CNN Example

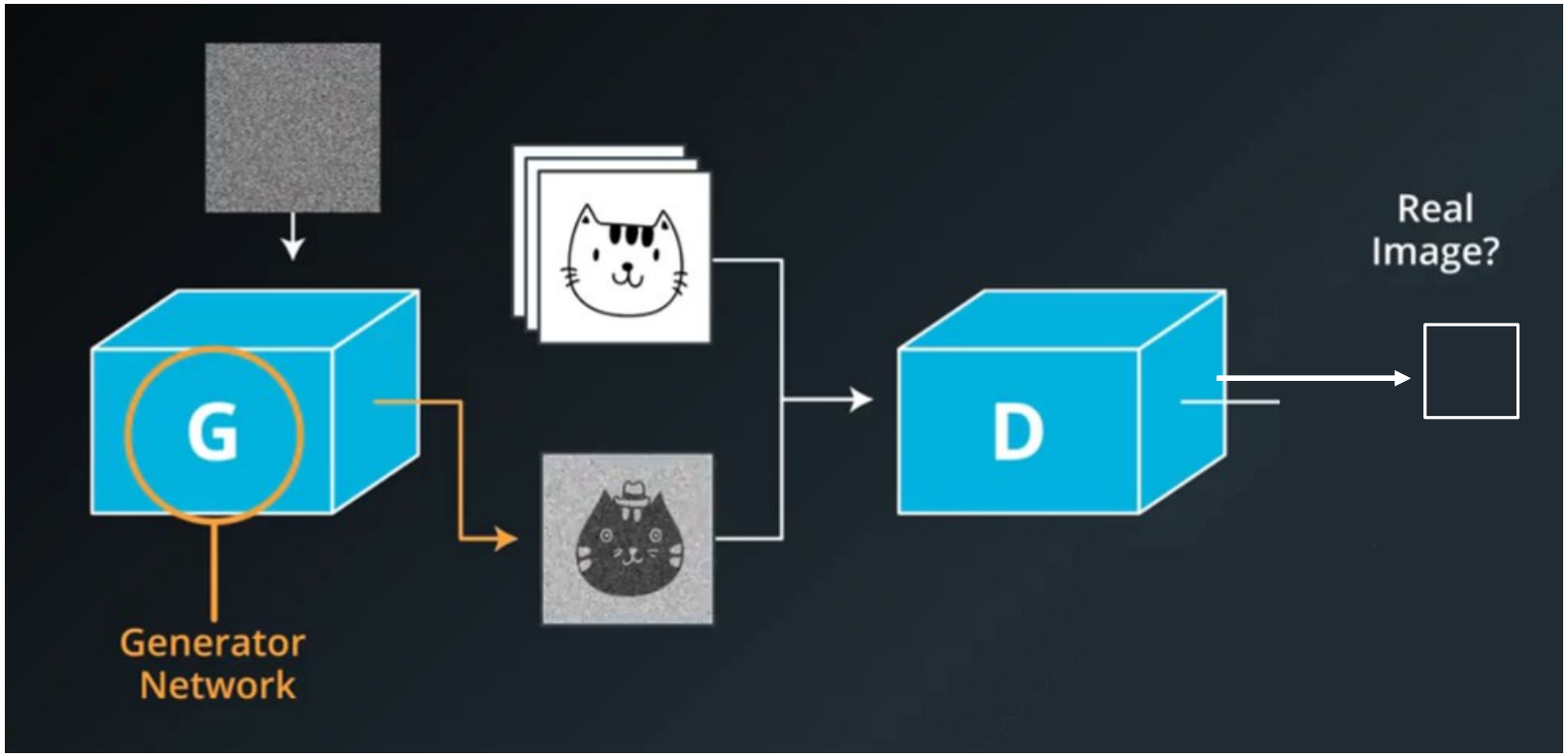
- How many zeros padding is required to keep the same input size?
- $O = (W-F+2P)/S+1$
- $100 = (100 - 3 + 2P)/1 + 1$
- $100 = 98 + 2P$
- $2 = 2P$
- $P = 1$

Generative Adversarial Network (GAN)

- Generative Adversarial Network is a **unsupervised generative framework** that trains generator G and discriminator D through the adversarial process. Through the adversarial process, the discriminator can tell whether the sample from the generator is fake or real. GAN adopts a mature BP algorithm.
- **Generator G**: The **input is noise** z, which complies with manually selected prior probability distribution, such as even distribution and Gaussian distribution. The generator adopts the network structure of the multilayer perceptron (MLP), uses maximum likelihood estimation (MLE) parameters to represent the derivable mapping G(z), and maps the input space to the sample space.
- **Discriminator D**: The **input is the real sample x and the fake sample G(z)**, which are tagged as real and fake respectively. The network of the discriminator can use the MLP carrying parameters. The output is the probability D(G(z)) that determines whether the sample is a real or fake sample.
- **GAN can be applied to scenarios such as image generation, text generation, speech enhancement, image super-resolution**.

GAN Architecture

- Generator/Discriminator

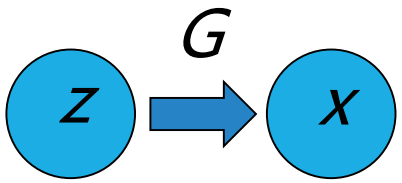


Generative Model and Discriminative Model

Generative network

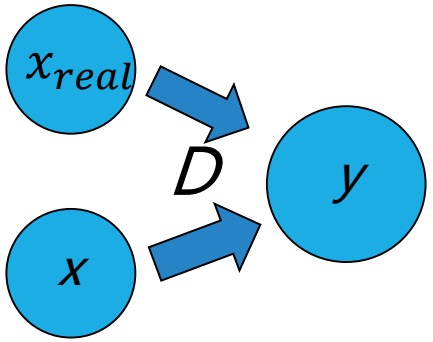
- Generates sample data
  - Input: Gaussian white noise vector z
  - Output: sample data vector x

$$x = G(z;\theta^G)$$



- Discriminator network
  - Determines whether sample data is real
    - Input: real sample data x_{real} and generated sample data $x = G(z)$
    - Output: probability that determines whether the sample is real

$$y = D(x;\theta^D)$$



Training Rules of GAN

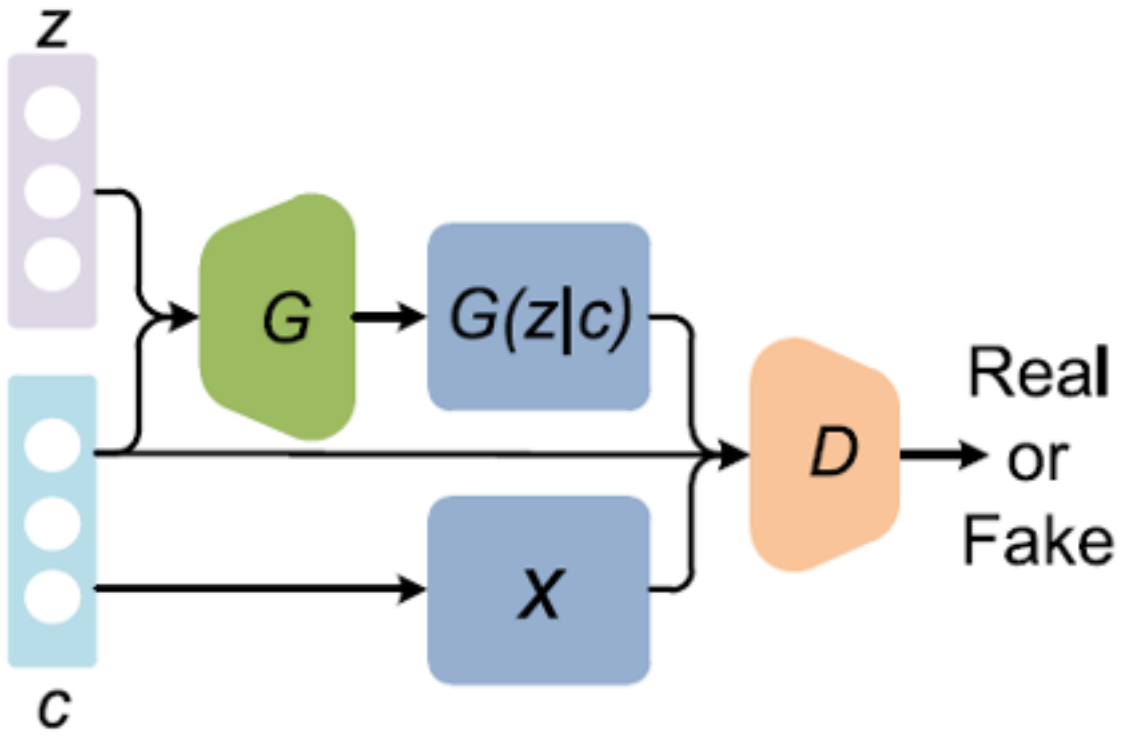
- Optimization objective:
  - Value function

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_{z(z)}}[log(1-D(G(z)))]$$

  - In the early training stage, when the outcome of G is very poor, D determines that the generated sample is fake with high confidence, because the sample is obviously different from training data. In this case, log(1-D(G(z))) is saturated (where the gradient is 0, and iteration cannot be performed). Therefore, we choose to train G only by minimizing [-log(D(G(z))].

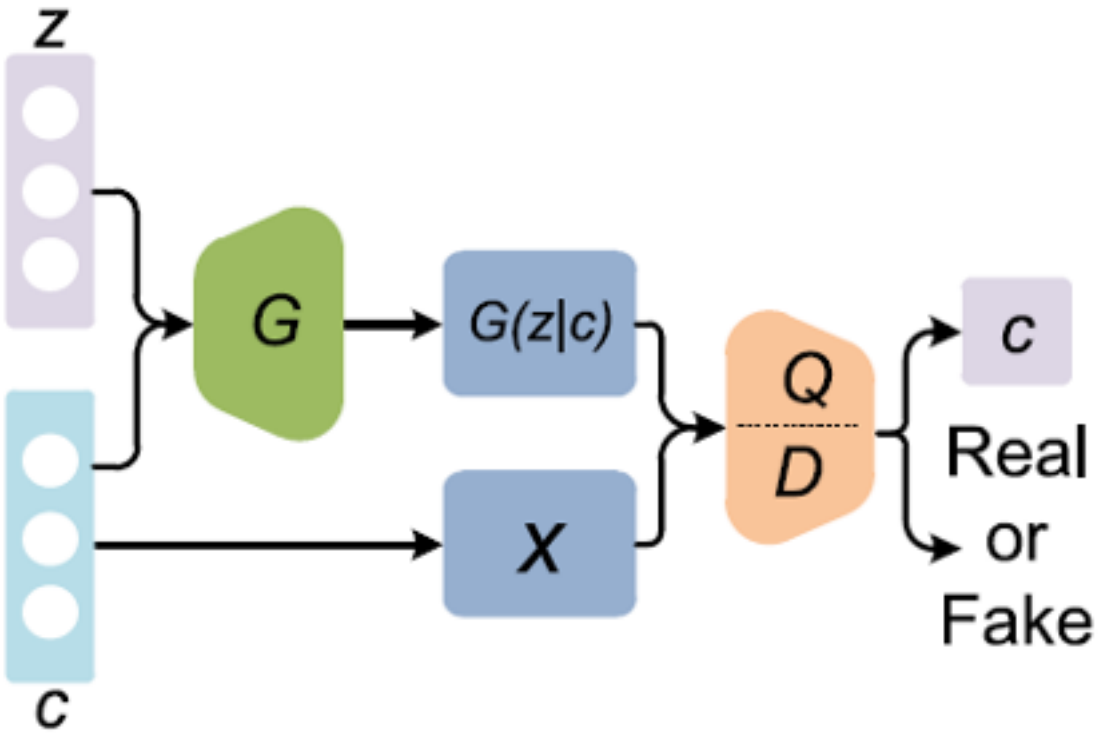
conditional GAN (cGAN)

- the network iterates (during training) over all the classes that we wish to synthesize.
- The generator G gets both a class id c and a random noise vector z as input, and the discriminator D gets the class id as well and needs to determine if its input is a real member of the given class



InfoGAN

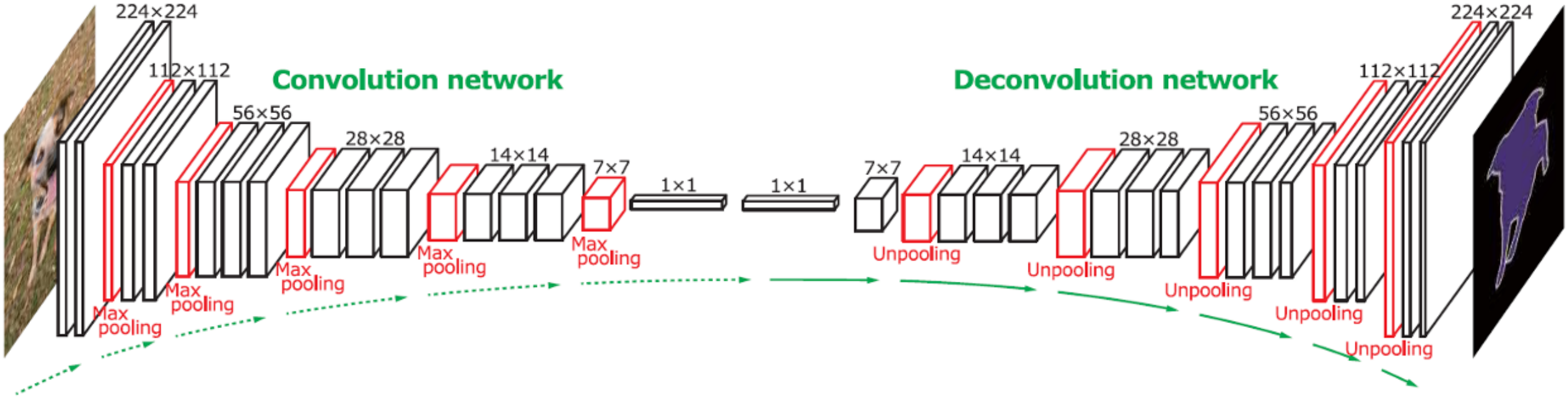
- The discriminator in an InfoGAN does not have access to the class id, but must instead infer it from the samples it is given



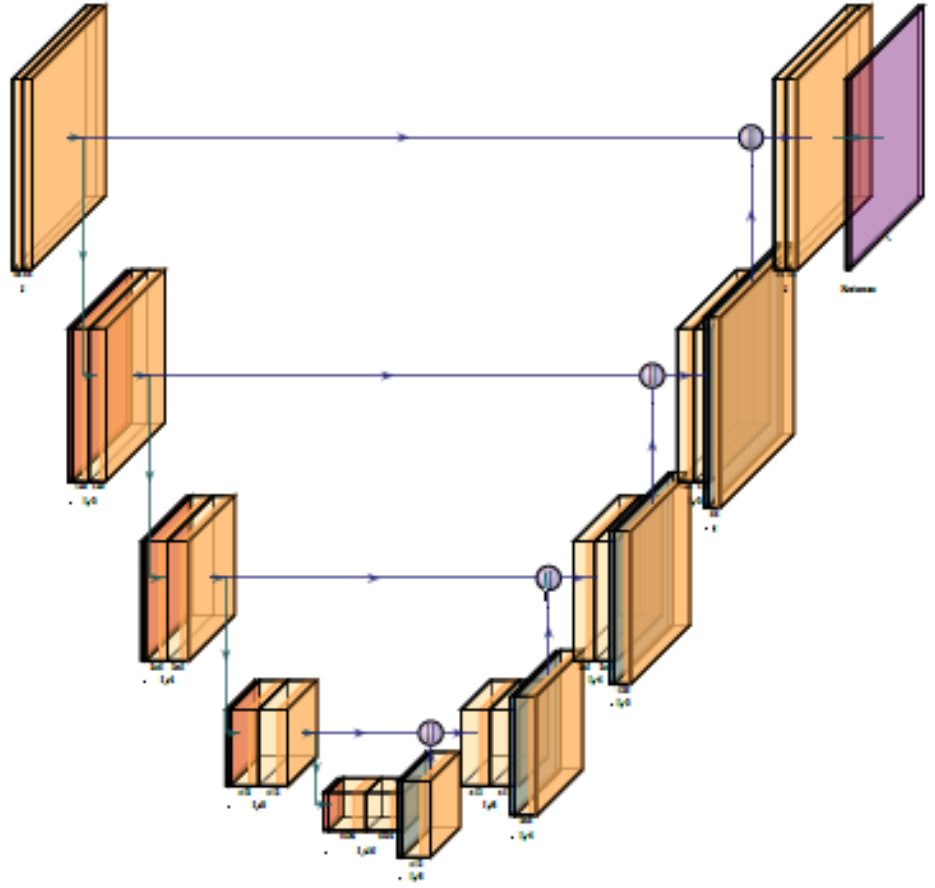
U-Nets and Feature Pyramid Networks

- Image downsampling and upsampling can be combined to achieve a variety of multi-resolution image processing tasks output to be a full-resolution image.
- Examples of such applications include
  - Pixel-wise semantic labeling
  - Image denoising and super-resolution
  - Monocular depth inference
  - Neural style transfer

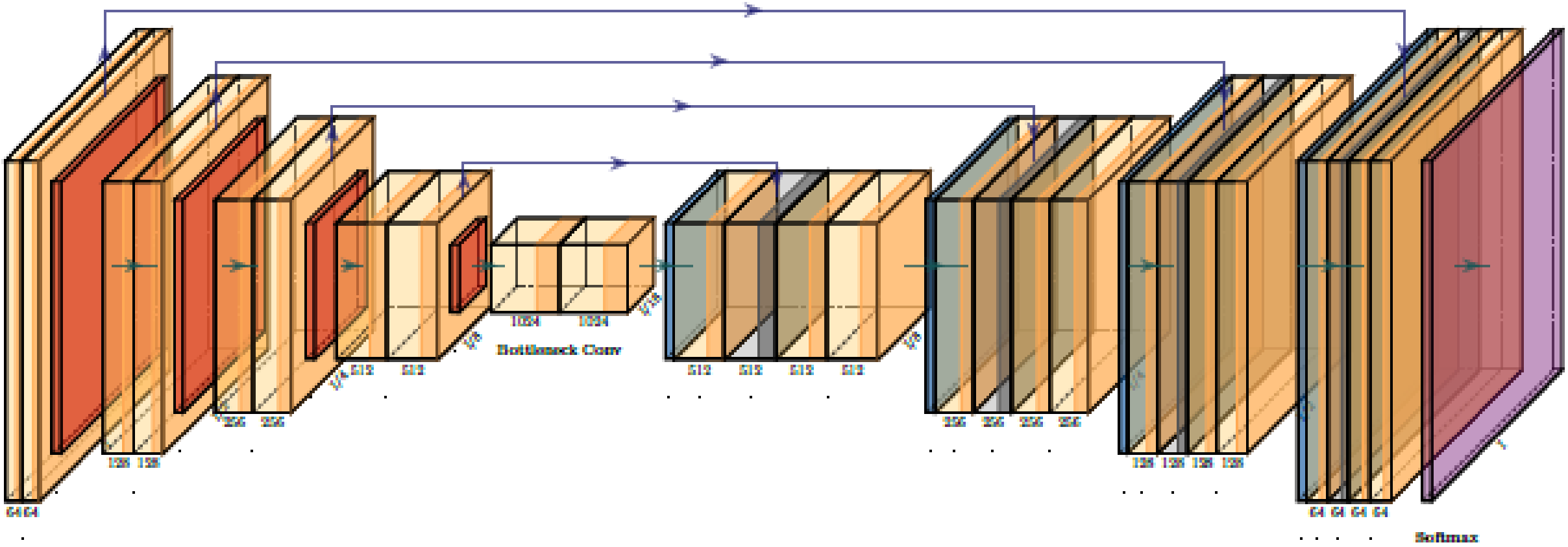
Deconvolution Network



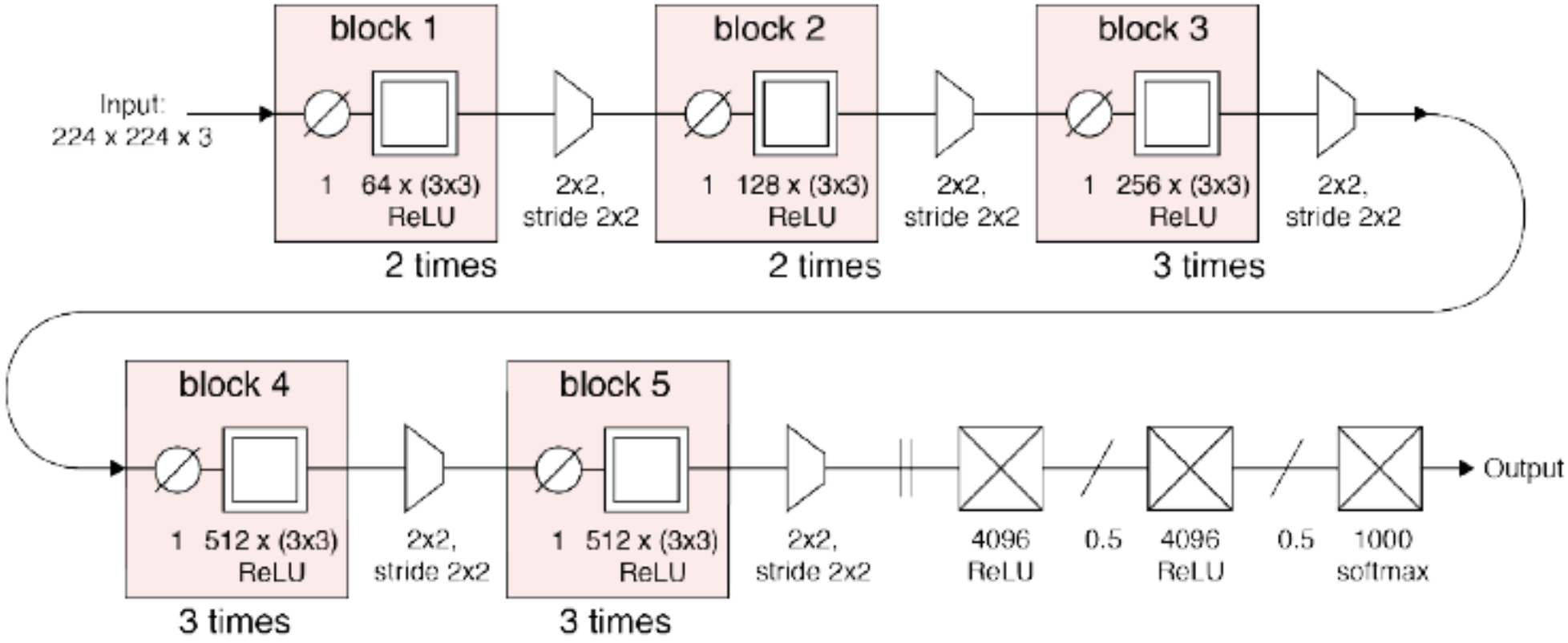
U-Nets



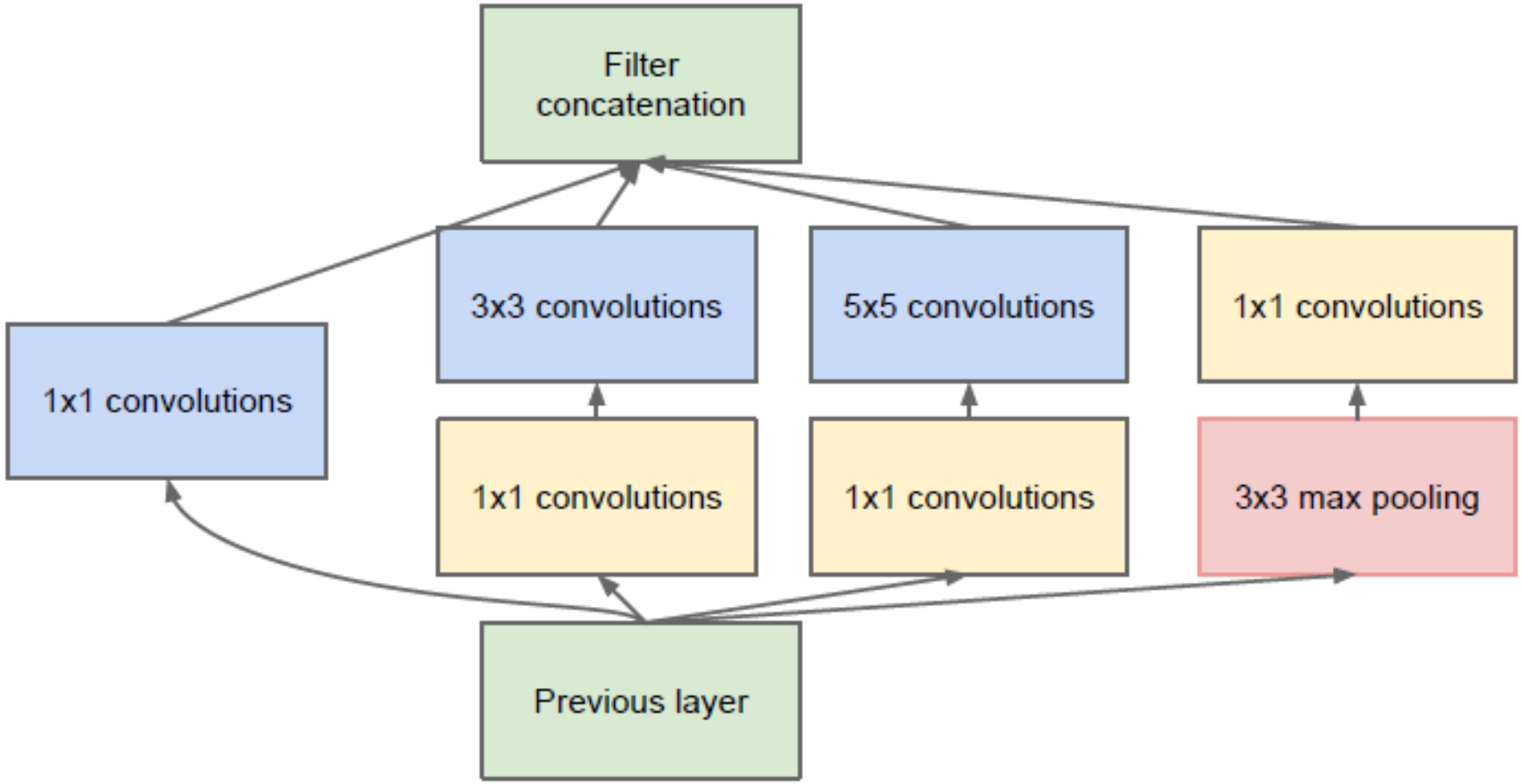
U-Nets


64 64
128 128
256 256
512 512
1024 1024
Bottleneck Conv
512 512 512 512
256 256 256 256
128 128 128 128
64 64 64 64
Softmax

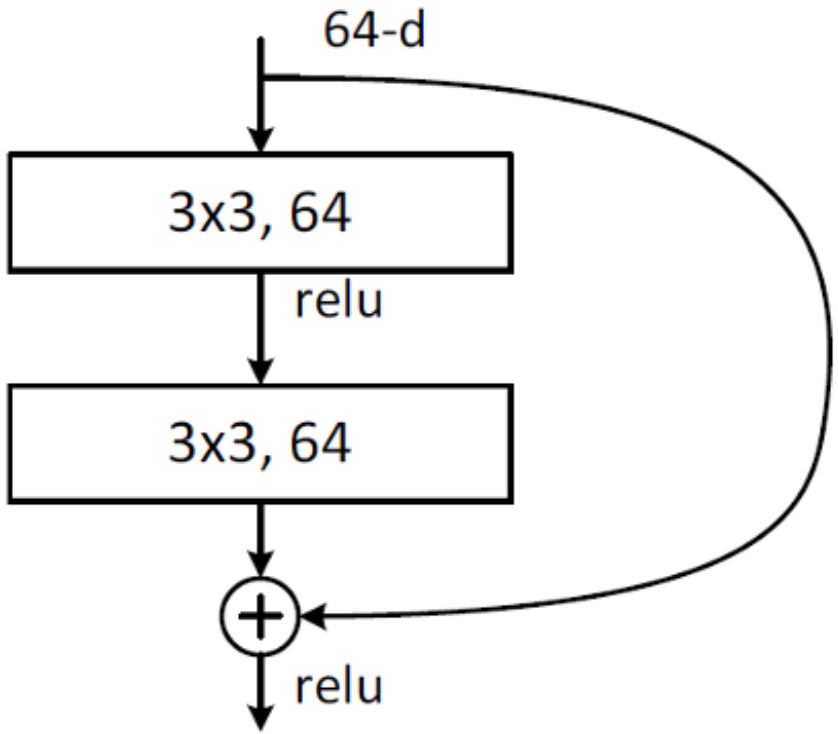
VGG16 network

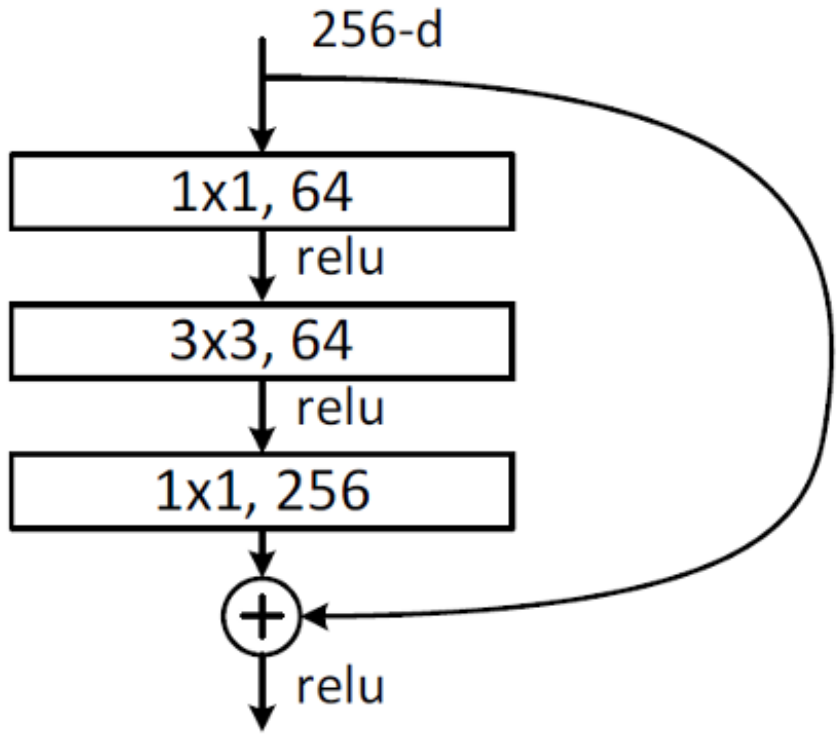


Inception Module

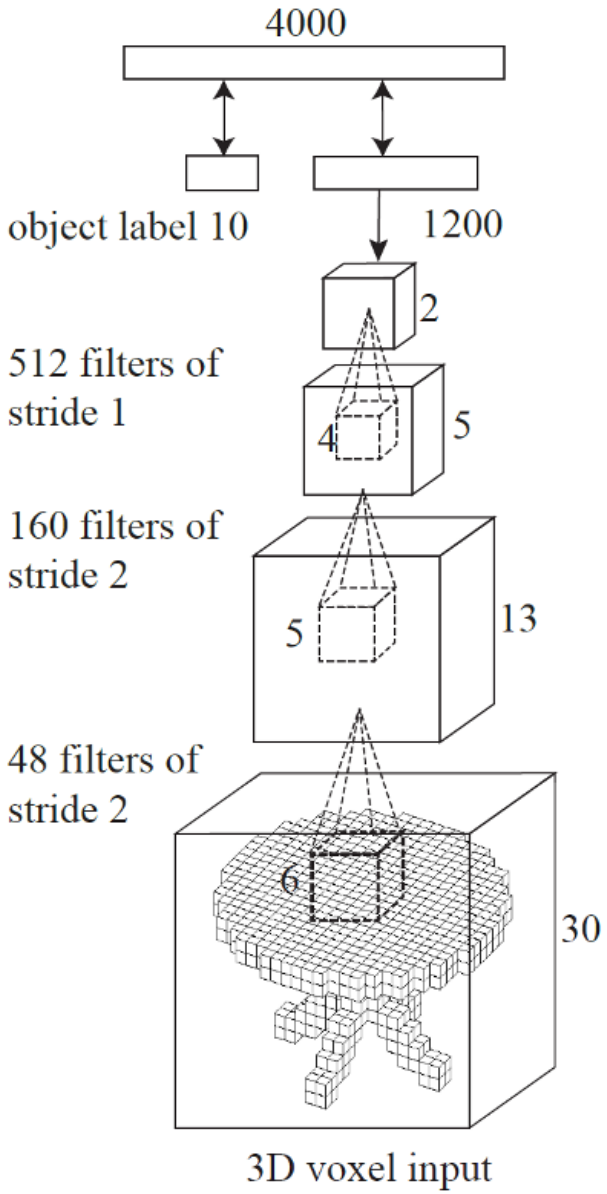


ResNet: Residual Networks

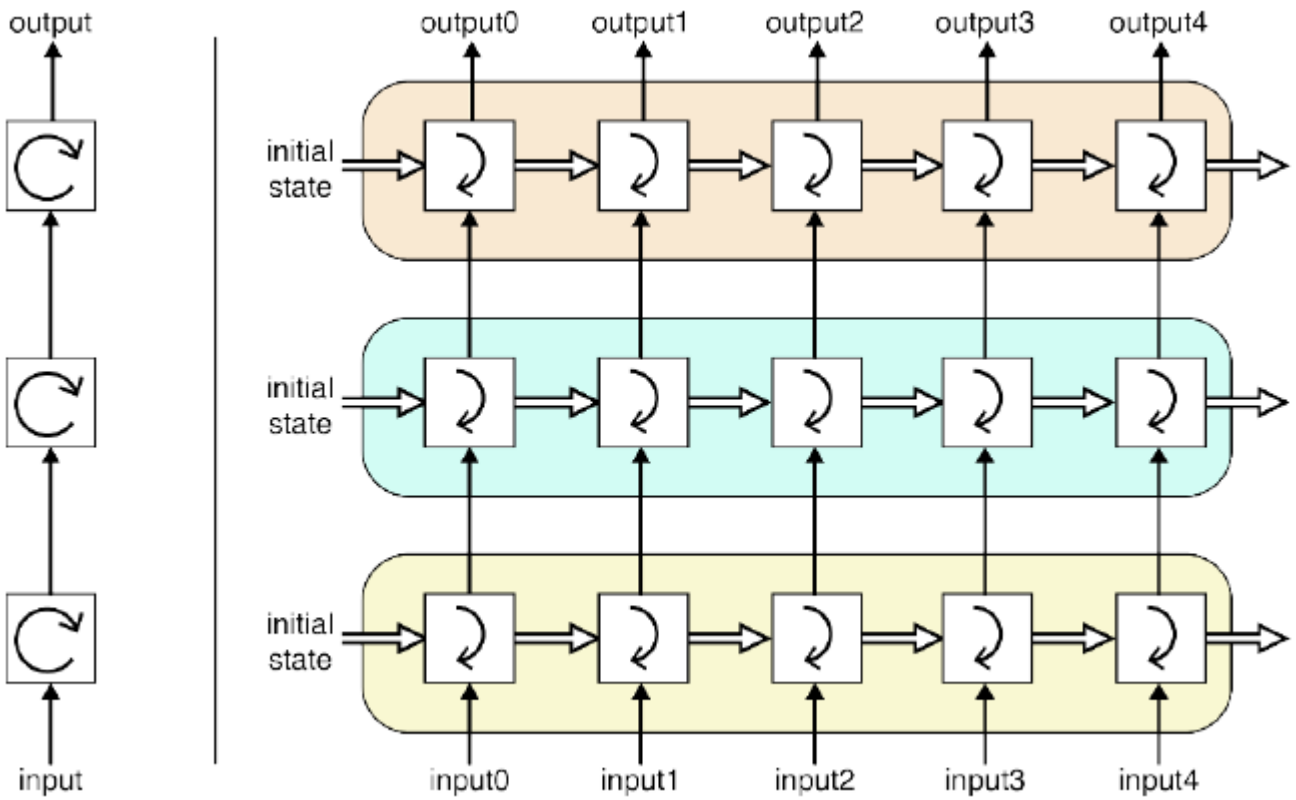




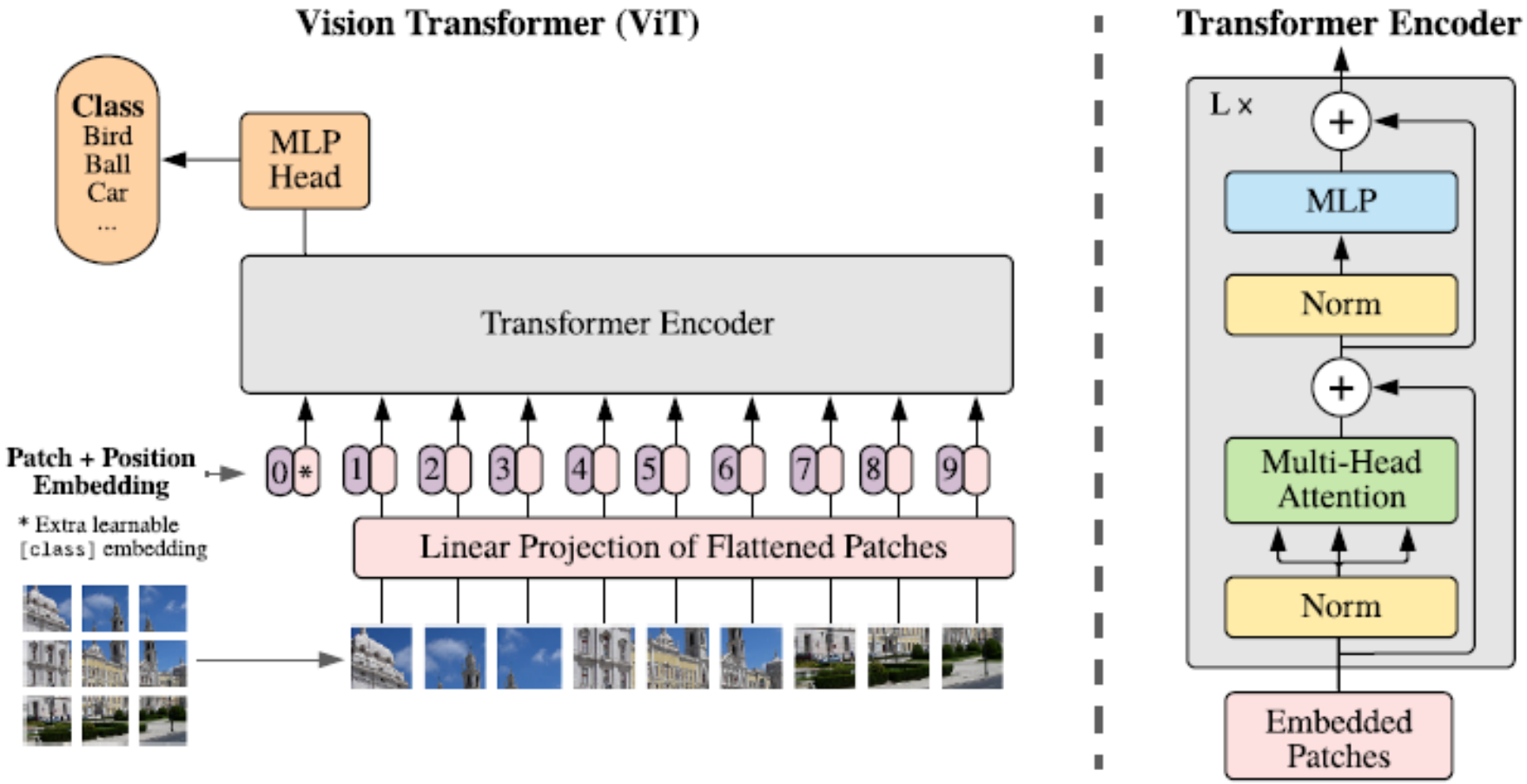
3D convolutional networks



RNN


output
input
output0
output1
output2
output3
output4
initial state
initial state
initial state
input0
input1
input2
input3
input4

Vision Transformer (ViT)



Thank You