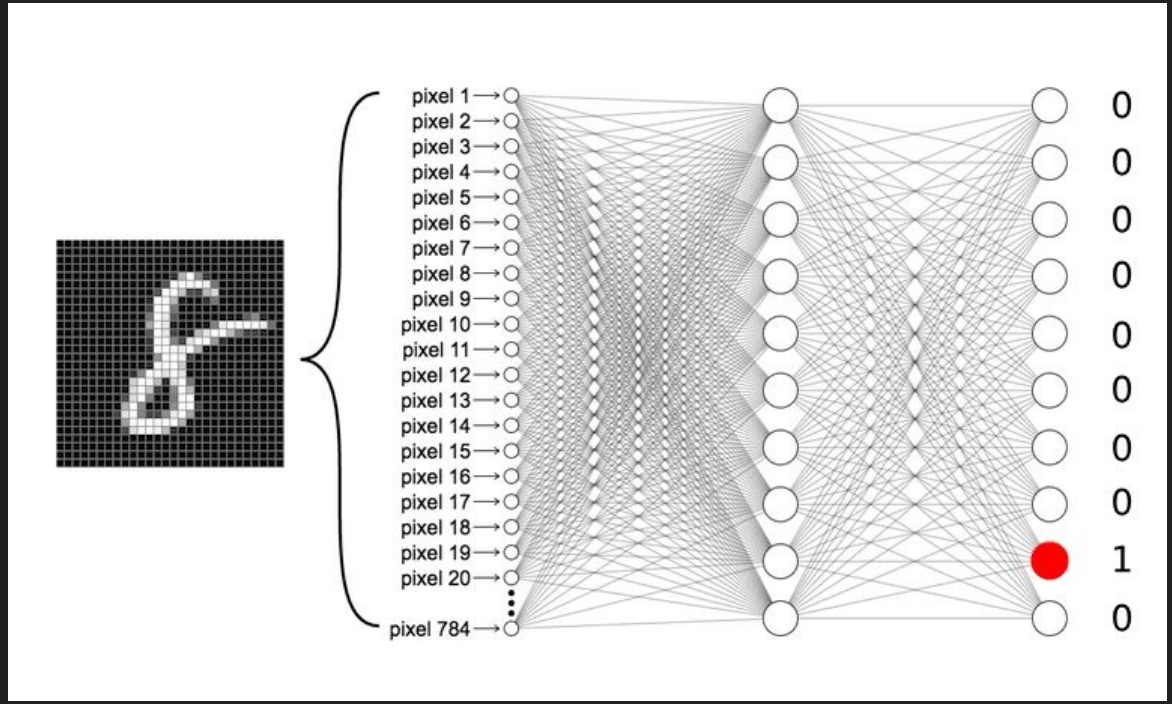
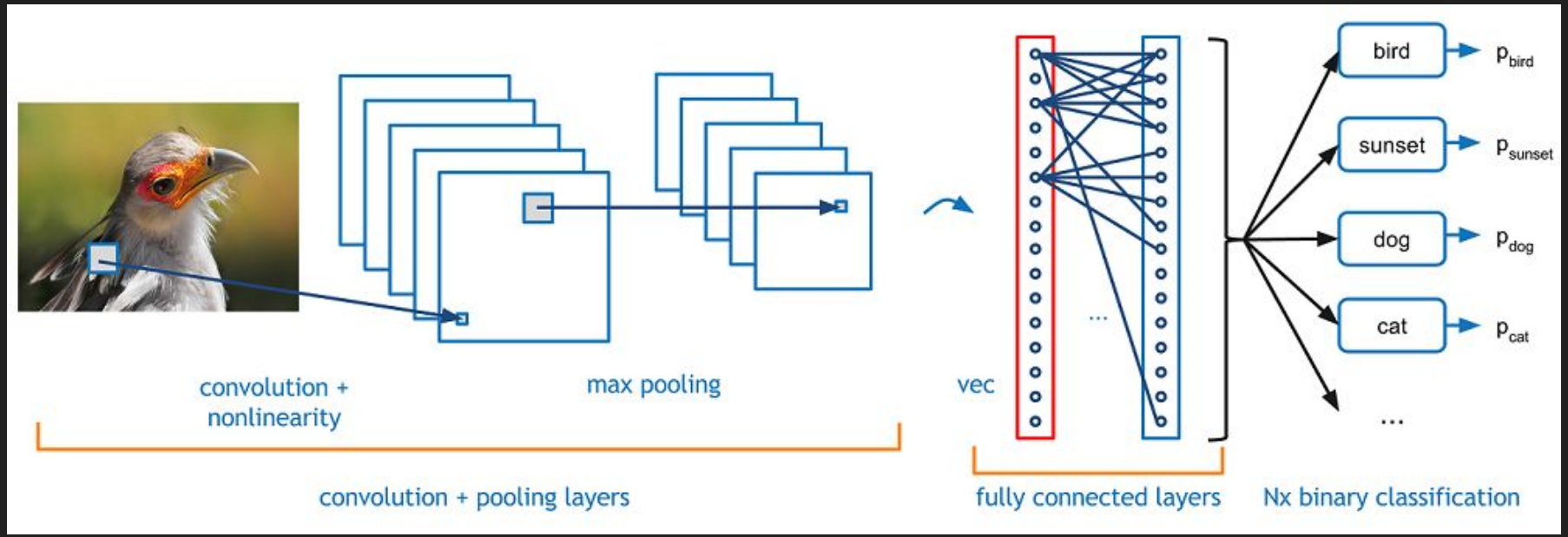
Intro to CNNs

Last Week - Fully Connected Neural Network



Convolutional Neural Network



What is convolution?

Convolution - How a function g() changes the shape of another function f()

Applications of Convolution:

- Image processing - edge detection, blurring, instagram filters
- Audio - Simulating acoustics, electronic music
- Signal processing - high pass filter, low pass filter

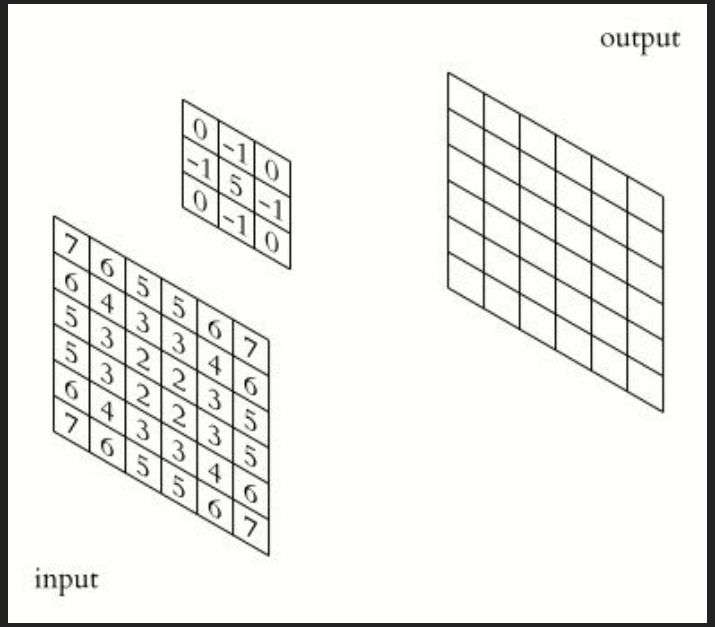
Convolution in image processing

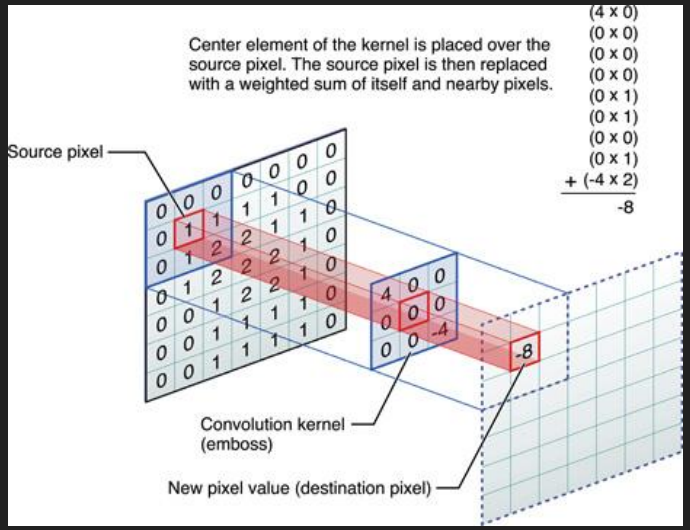
$$(f*g)(c) = \sum f(a) \cdot g(b)$$

f() - image

g() - kernel (usually nxn matrix)

Kernel slides over the image and computes new pixel value as a sum/avg of element-wise multiplication





Examples of Convolution in image processing

Image kernels
http://setosa.io/ev/image-kernels/

Some other kernel examples

1	1	1
1	1	1
1	1	1

Unweighted 3x3 smoothing kernel

0	1	0
1	4	1
0	1	0

Weighted 3x3 smoothing kernel with Gaussian blur

0	-1	0
-1	5	-1
0	-1	0

Kernel to make image sharper

-1	-1	-1
-1	9	-1
-1	-1	-1

Intensified sharper image

Gaussian Blur

Sharpened image

Padding and Striding

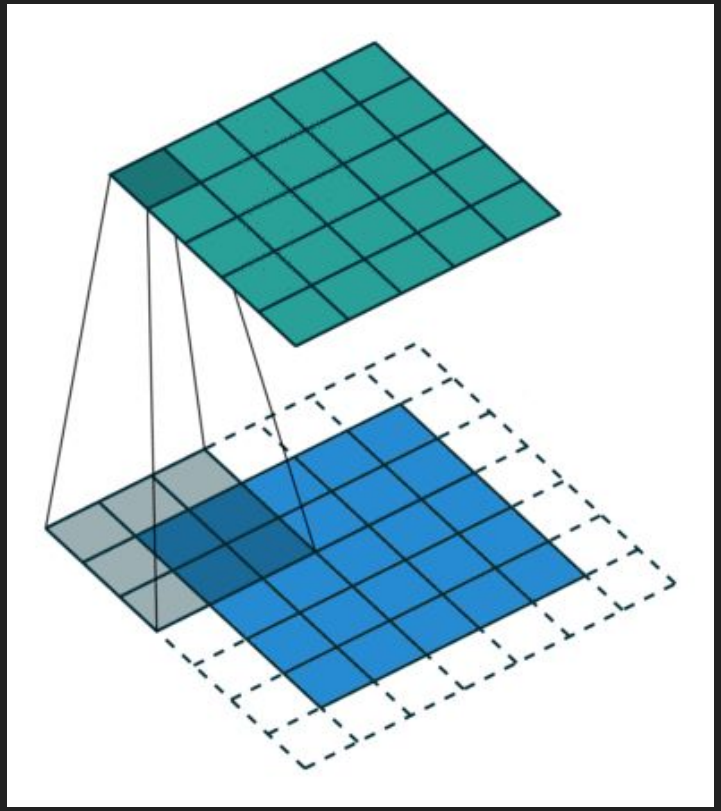
3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

5x5

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

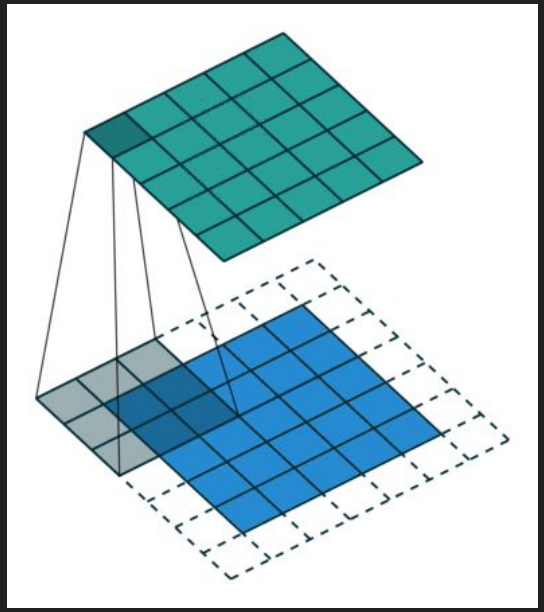
3x3

Normal Convolution

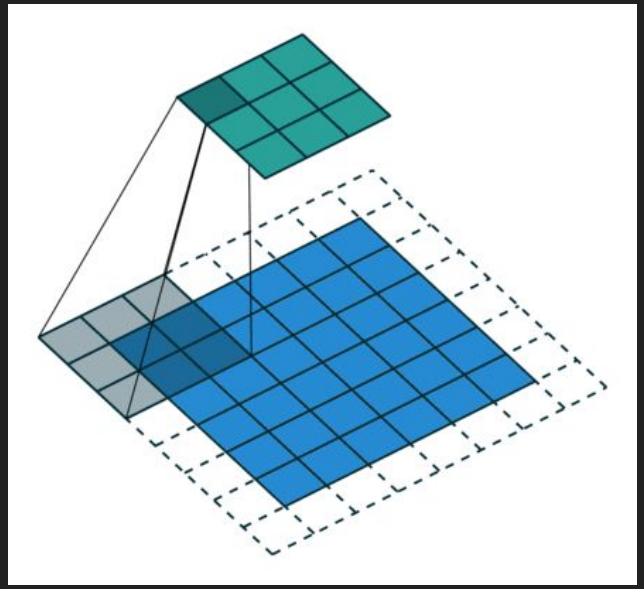


Convolution with padding

Padding and Striding

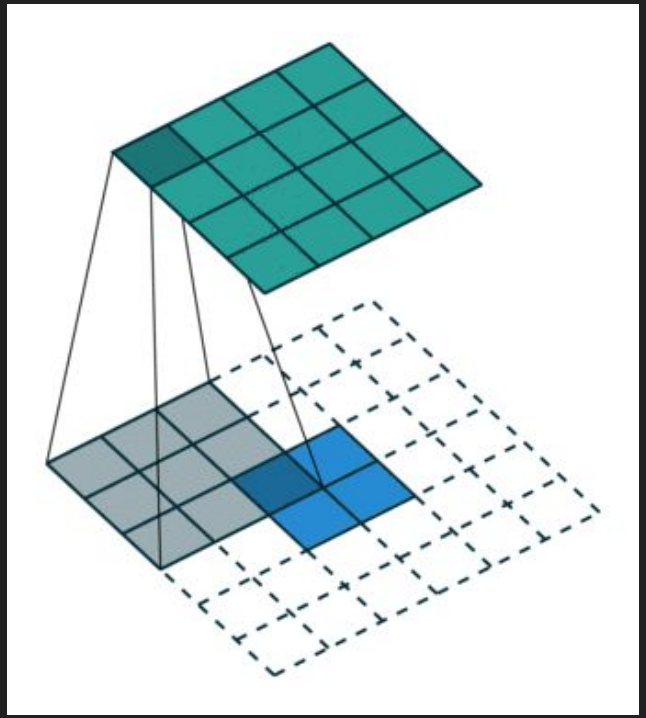


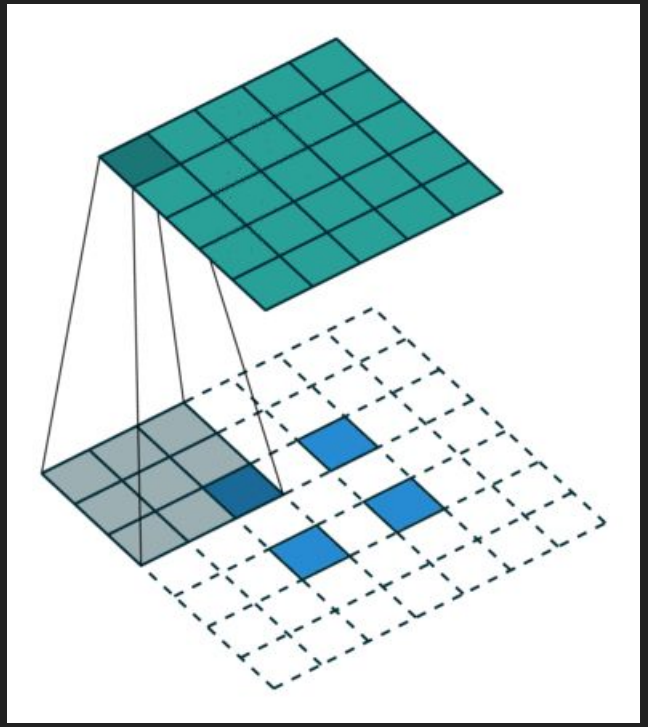
Padding = true
Stride = 1
Input = 5x5
Output = 5x5



Padding = true
Stride = 2
Input = 5x5
Output = 3x3

Deconvolution/Upsampling





Classifying MNIST using image kernels

*

1	1	1
0	0	0
0	0	0

→ True

*

0	0	1
0	1	0
1	0	0

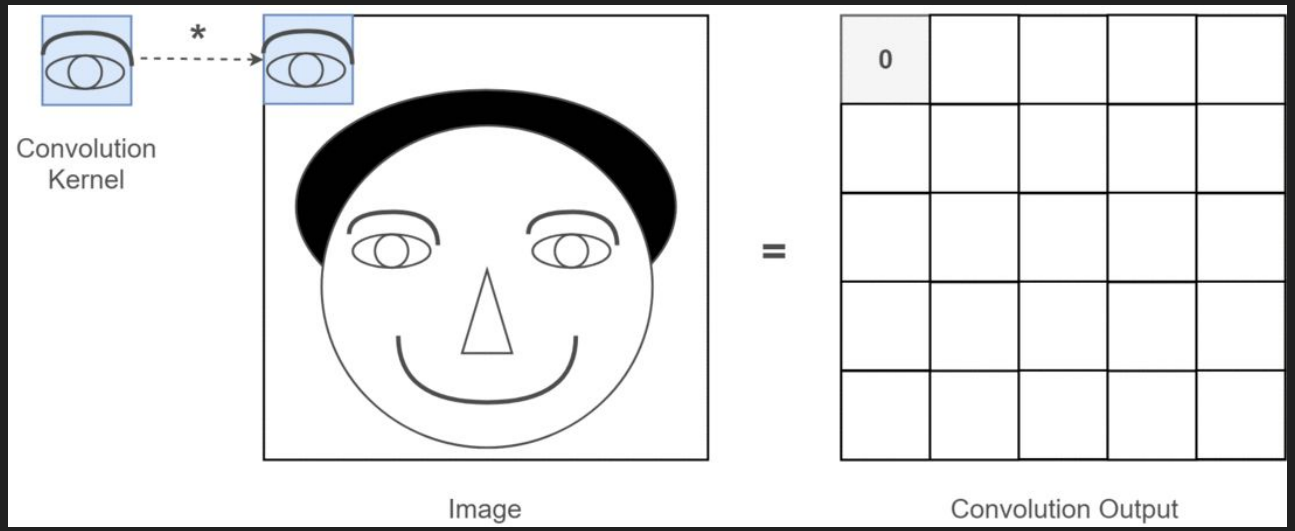
→ True

*

0	1	0
1	0	0
0	1	0

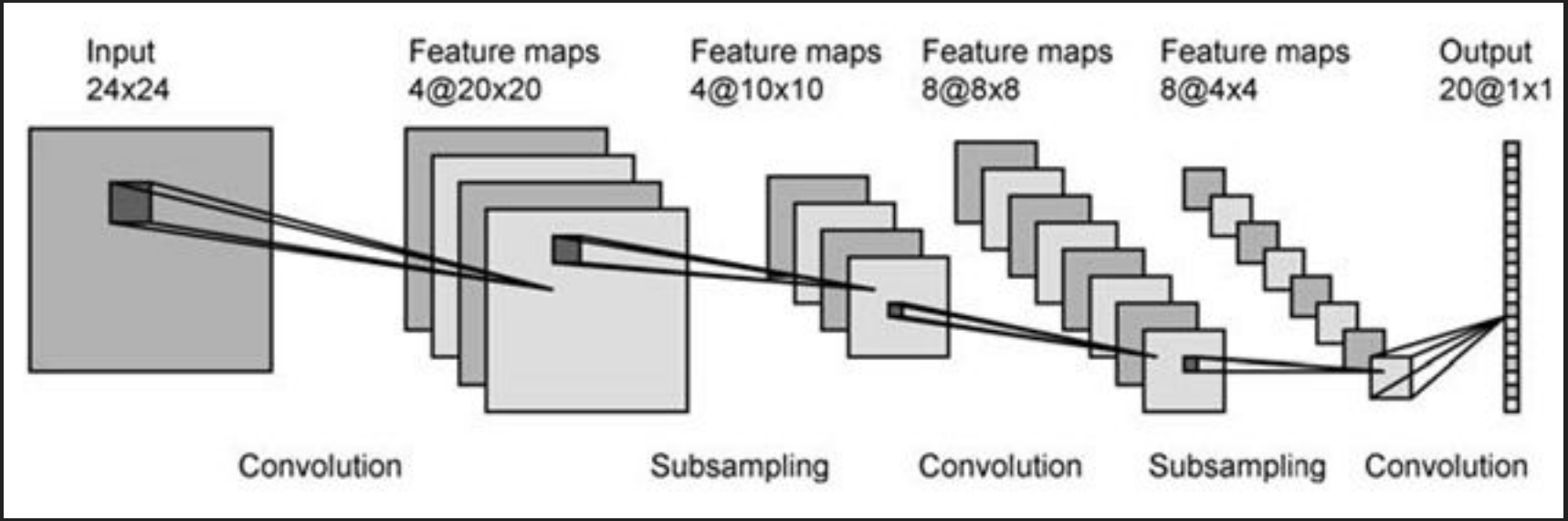
→ False

Classifying a face using image kernels



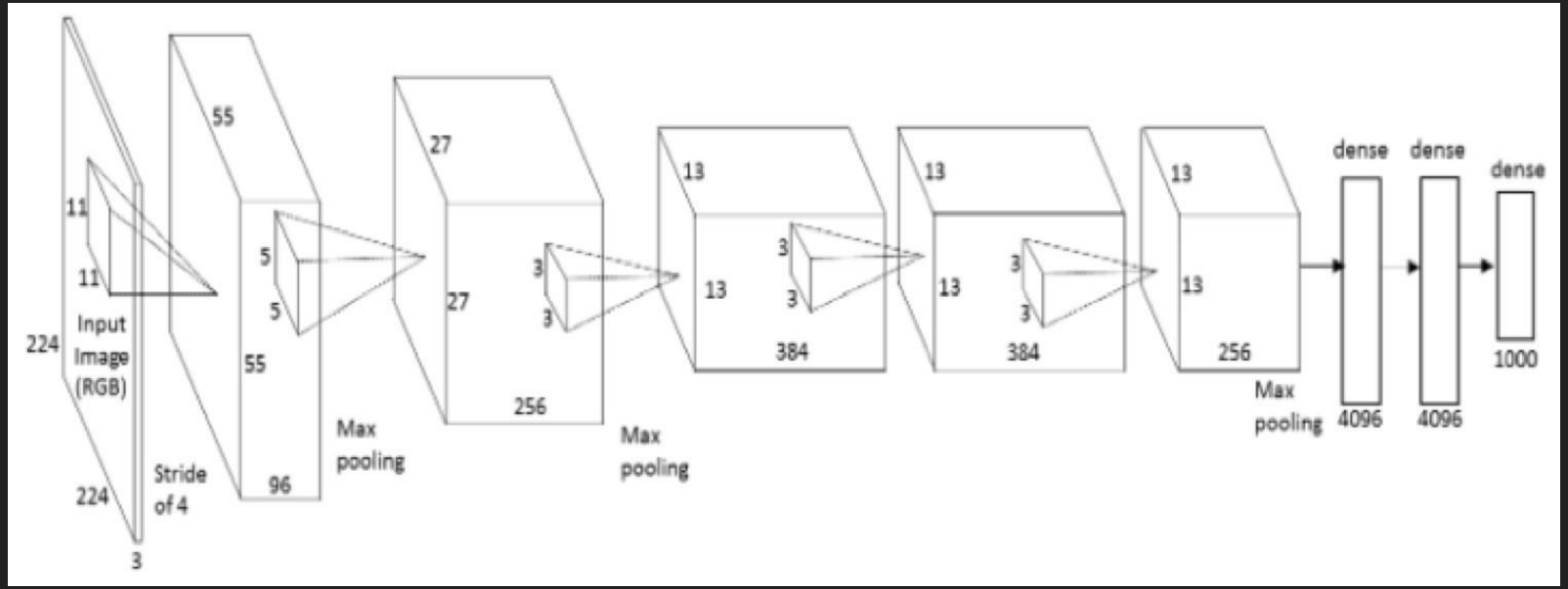
Very difficult to hand code all the image kernels for a face classifier

Convolutional Neural Network



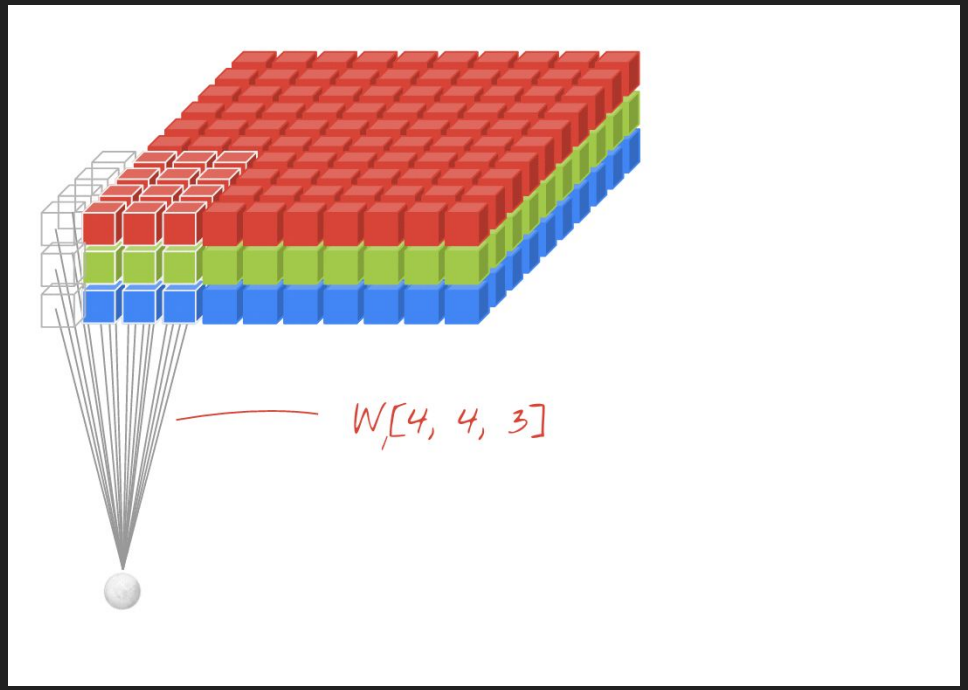
Kernels/Filters are learnable parameters

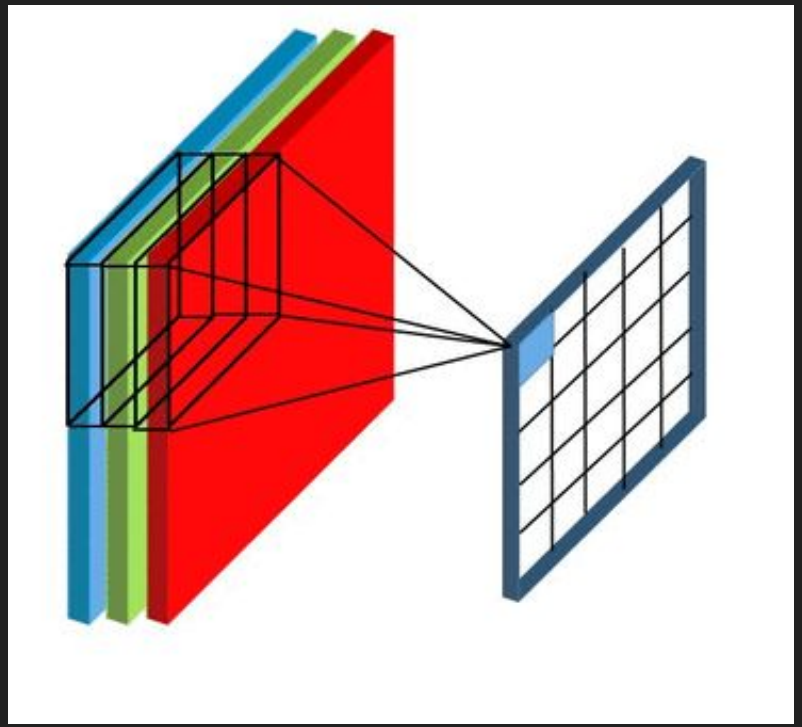
Convolutional Neural Network



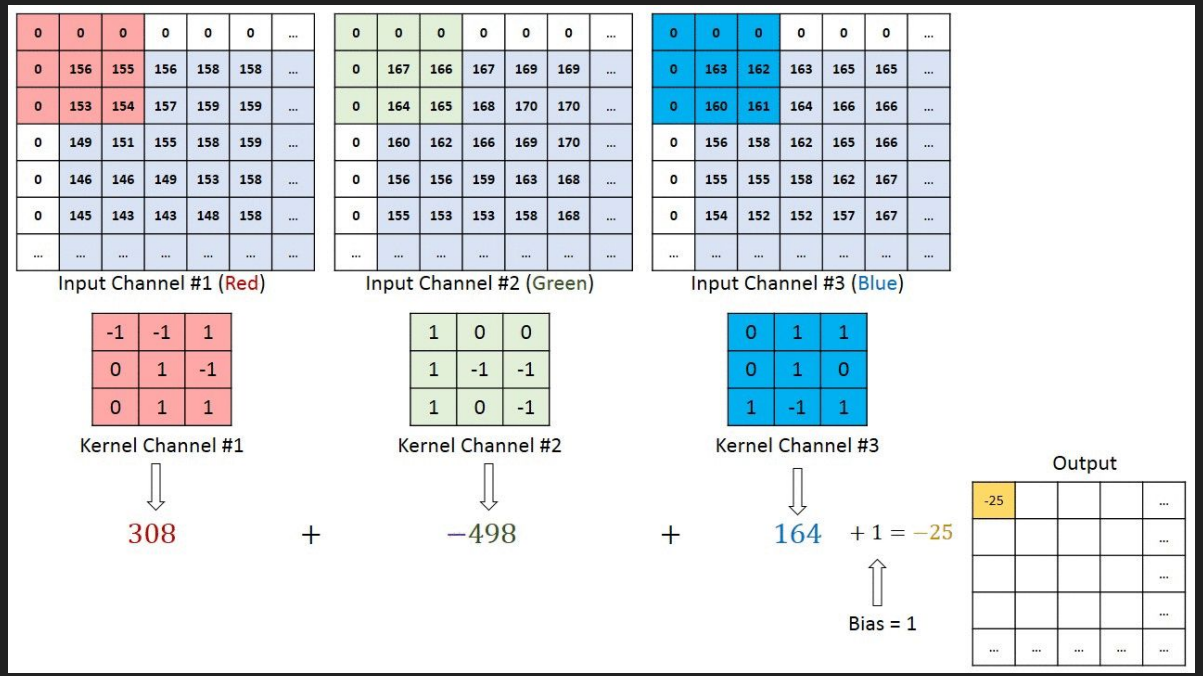
Kernels/Filters are learnable parameters

CNN with RGB input

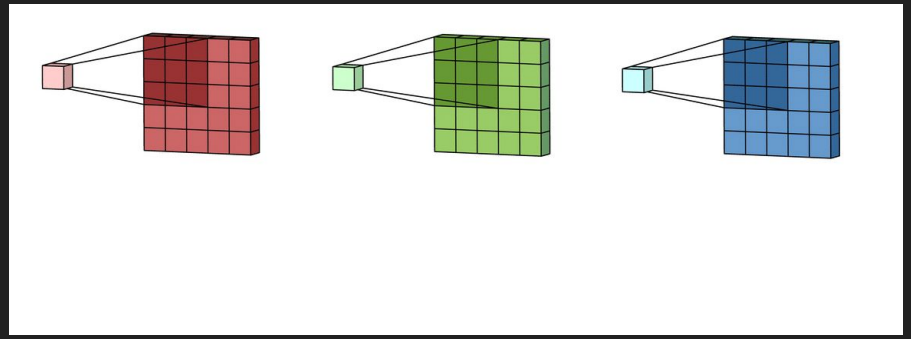


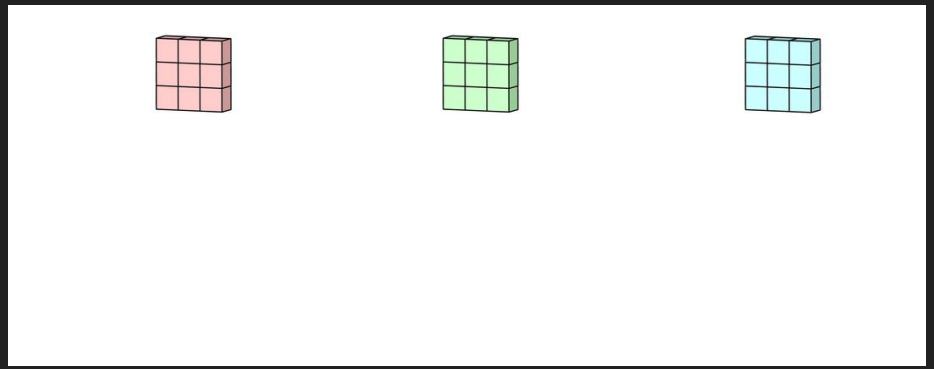


CNN with RGB input

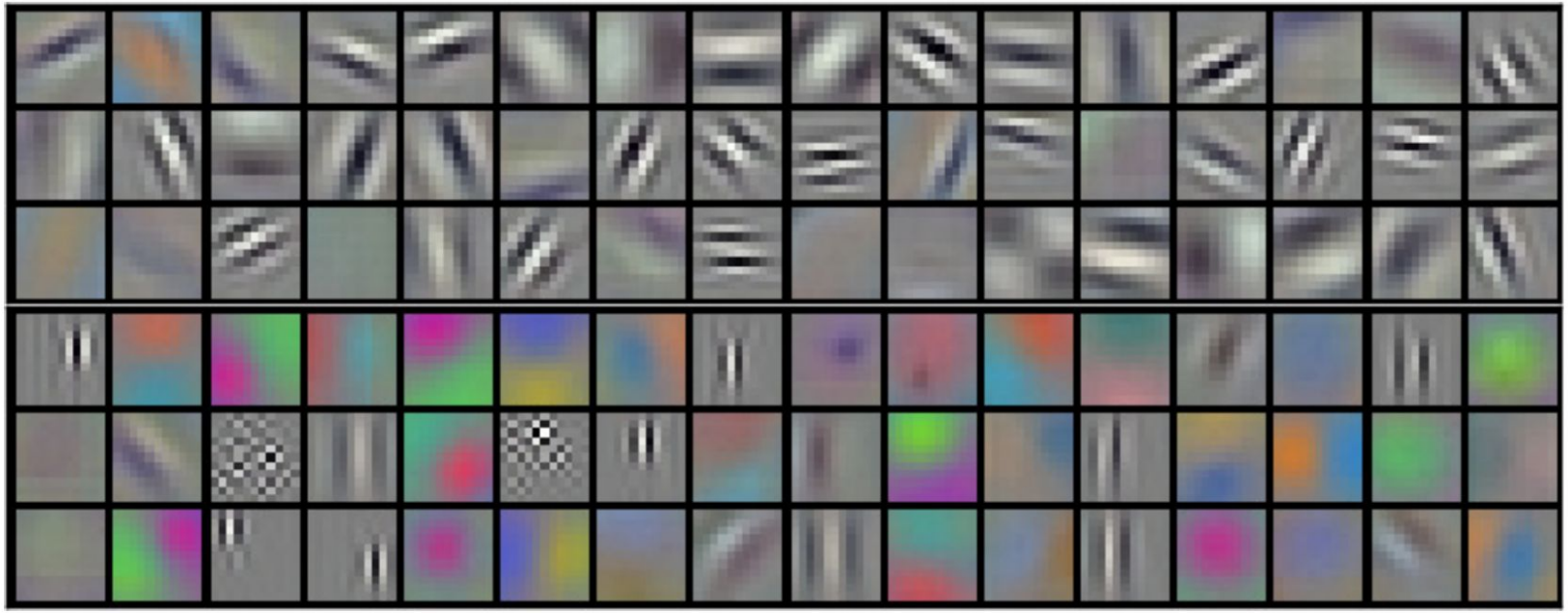


CNN with RGB input



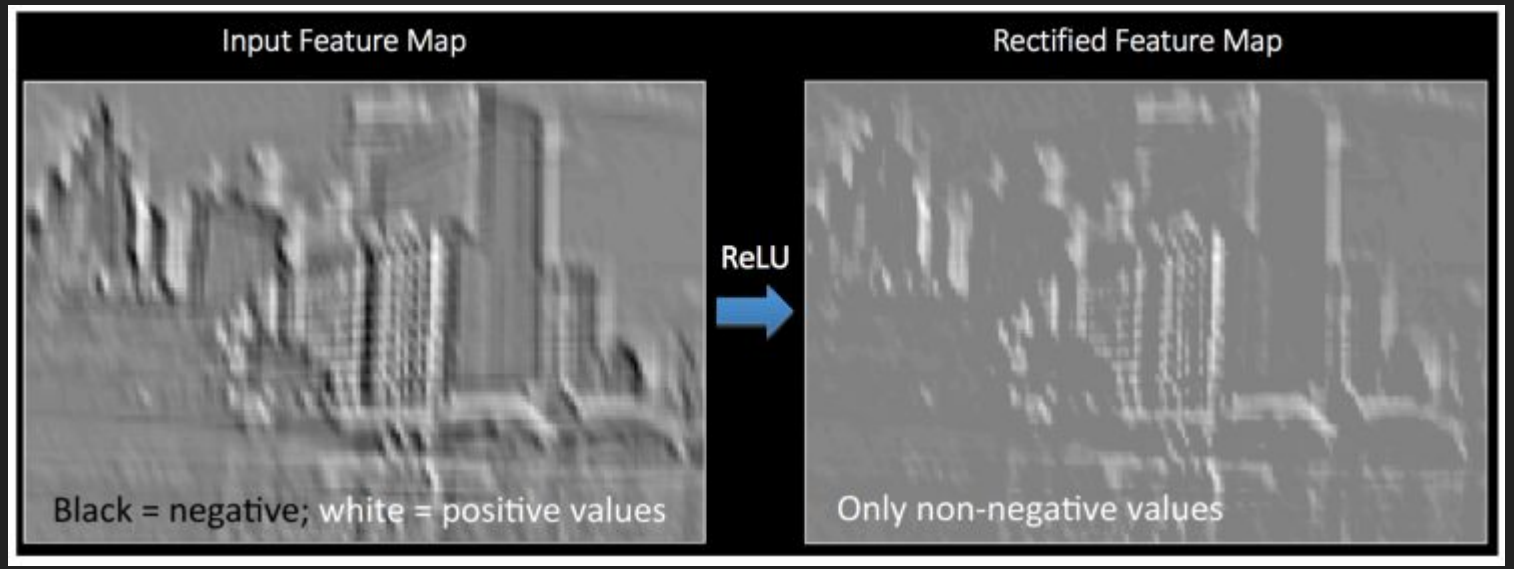


Learned Kernels/Filters

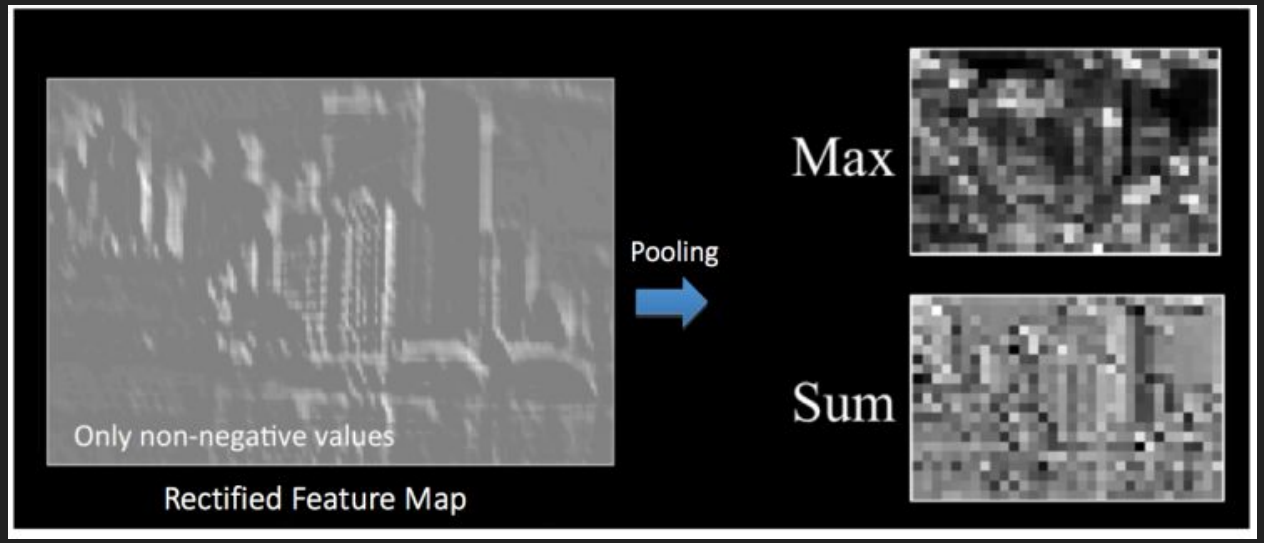


Feature map

Activation function



Pooling (optional)

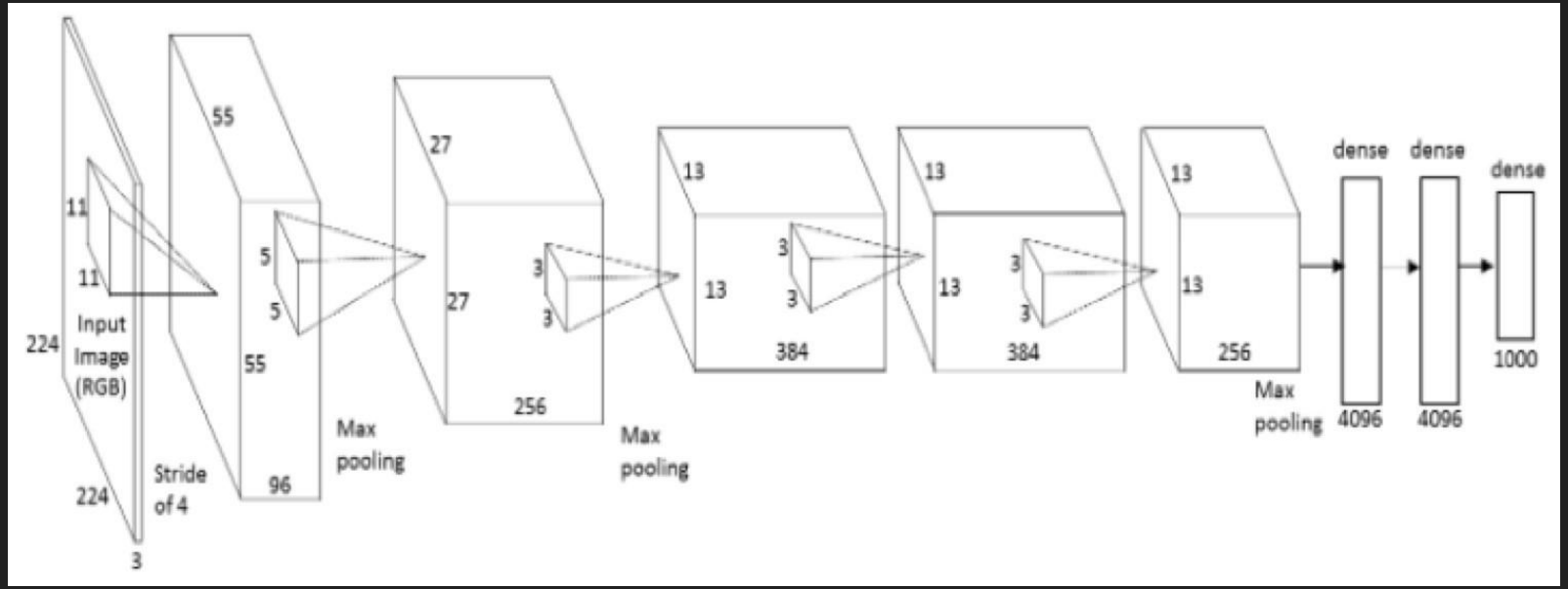


Pooling is good for extracting the most important features

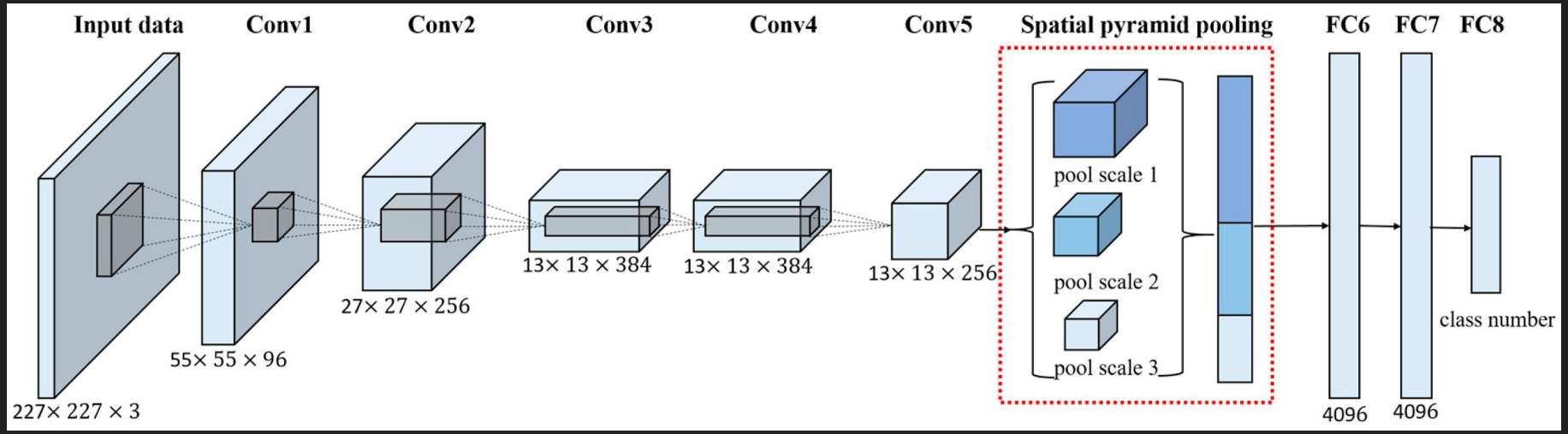
Reduces the no. of parameters(weights) in the next layer

Another alternative is stride = 2 or 3

Convolutional Neural Network



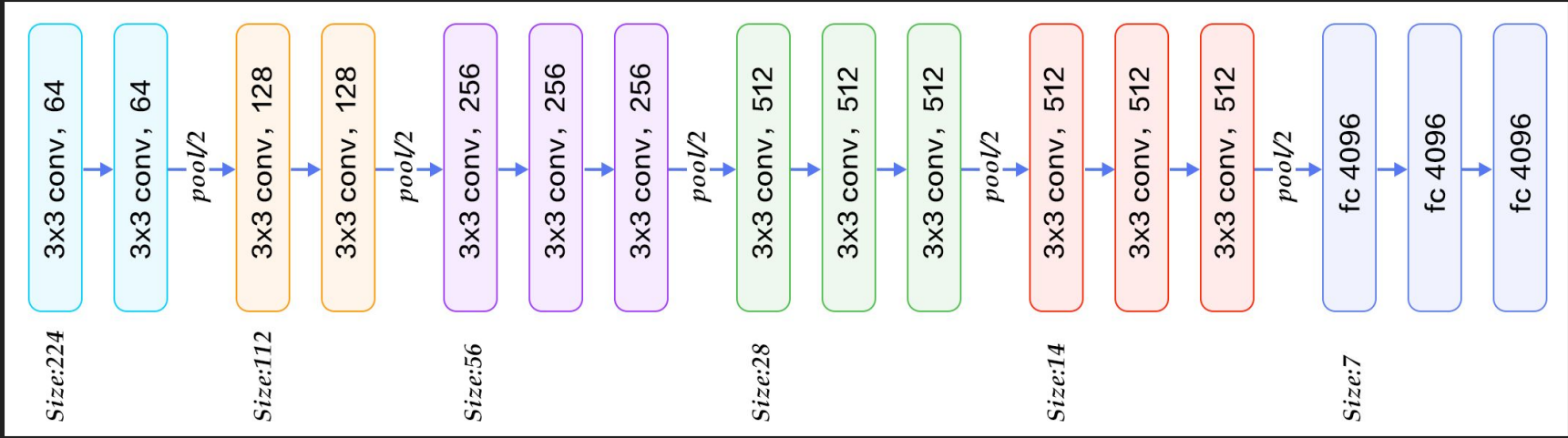
CNN for classification



Alexnet 2012

8 layers

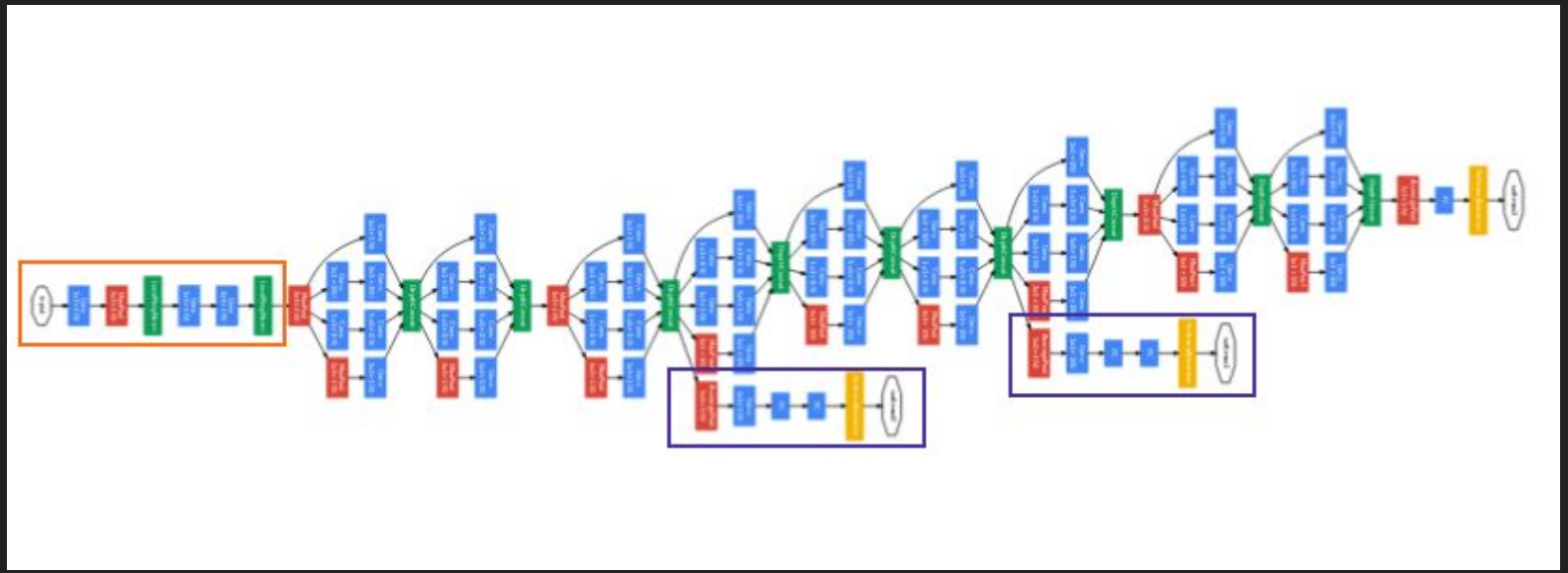
CNN for classification



VGG-16 2014

16 layers

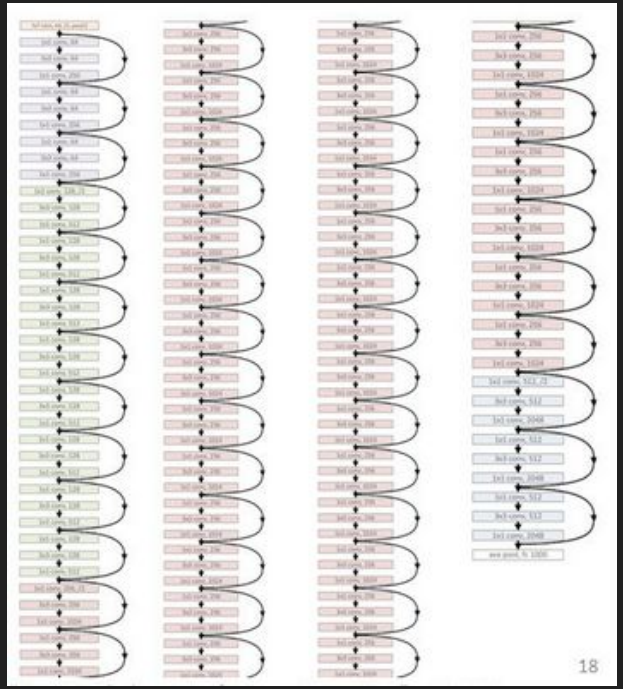
CNN for classification



Google inception v3 2015

48 layers

CNN for classification

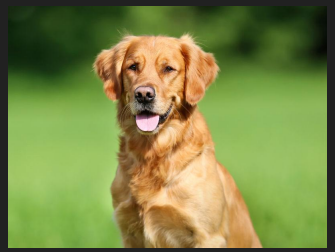


Microsoft Resnet-50/101 2015

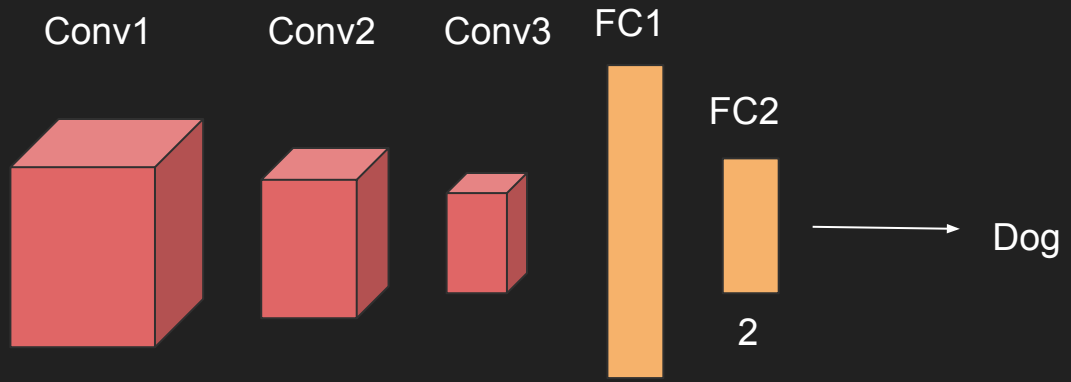
50 -152 layers

State of the art image classifier

Our simple image classifier



64 x 64 x 3



16 kernels

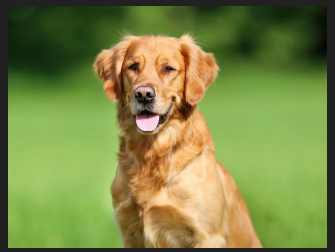
16

16

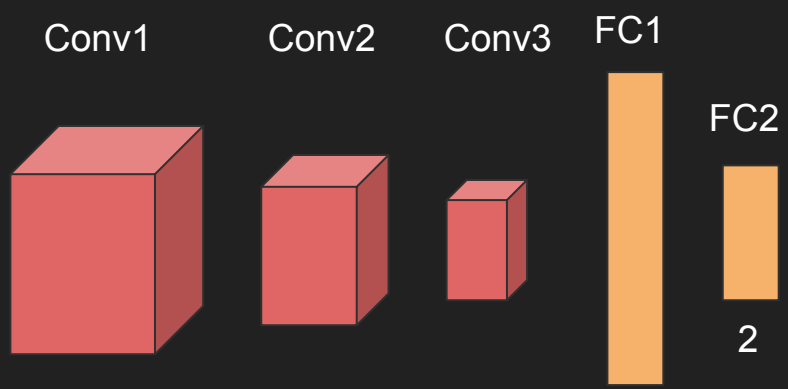
32 neurons

5 Layers and not enough depth (but CPU friendly)

Our simple image classifier



64 x 64 x 3



16 kernels

16

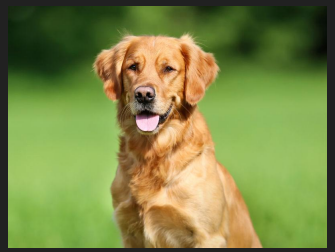
16

32 neurons

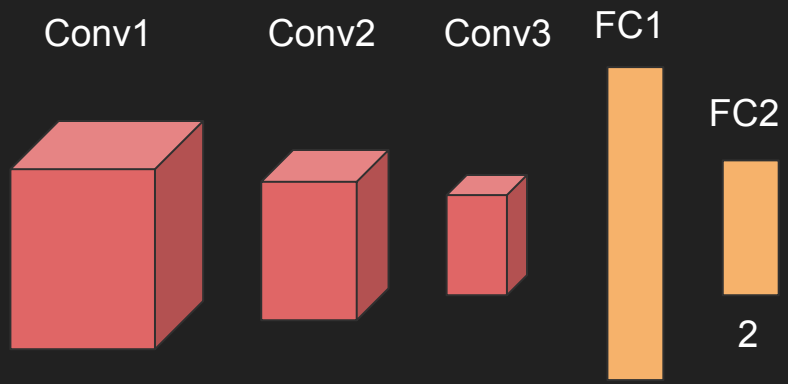
Iteration 0

Output	Ground truth	
0.75	0	Cat
0.25	1	Dog

Our simple image classifier



64 x 64 x 3



16 kernels

16

16

32 neurons

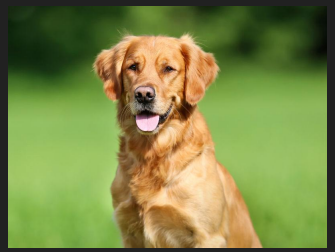
Iteration 0

Output	Ground truth	
0.75	0	Cat
0.25	1	Dog

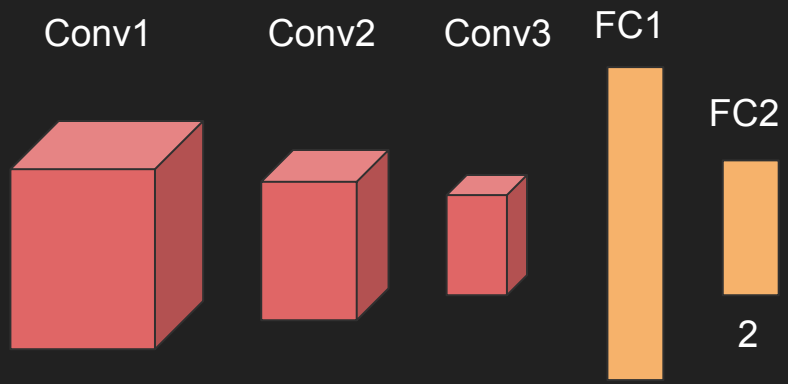
min(distance(output ,gt))

Cross entropy is a distance function(kind of) for probability distributions

Our simple image classifier



64 x 64 x 3



16 kernels

16

16

32 neurons

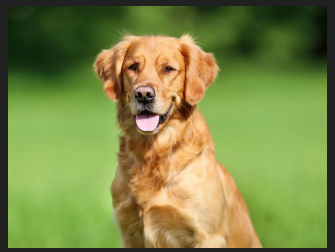
Iteration 500

Output	Ground truth	
0.40	0	Cat
0.60	1	Dog

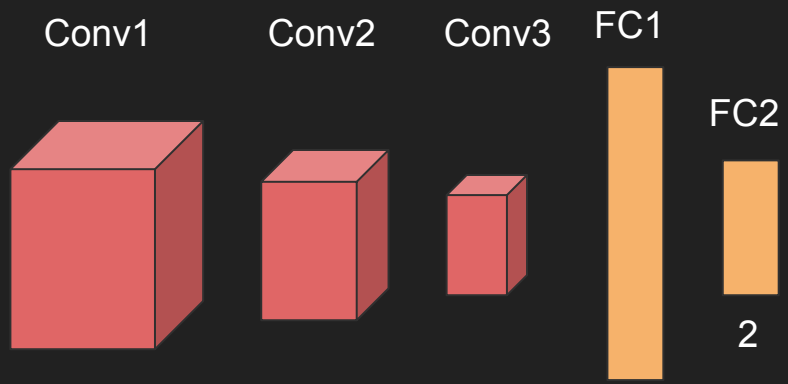
min(distance(output ,gt))

Cross entropy is a distance function(kind of) for probability distributions

Our simple image classifier



64 x 64 x 3



16 kernels

16

16

32 neurons

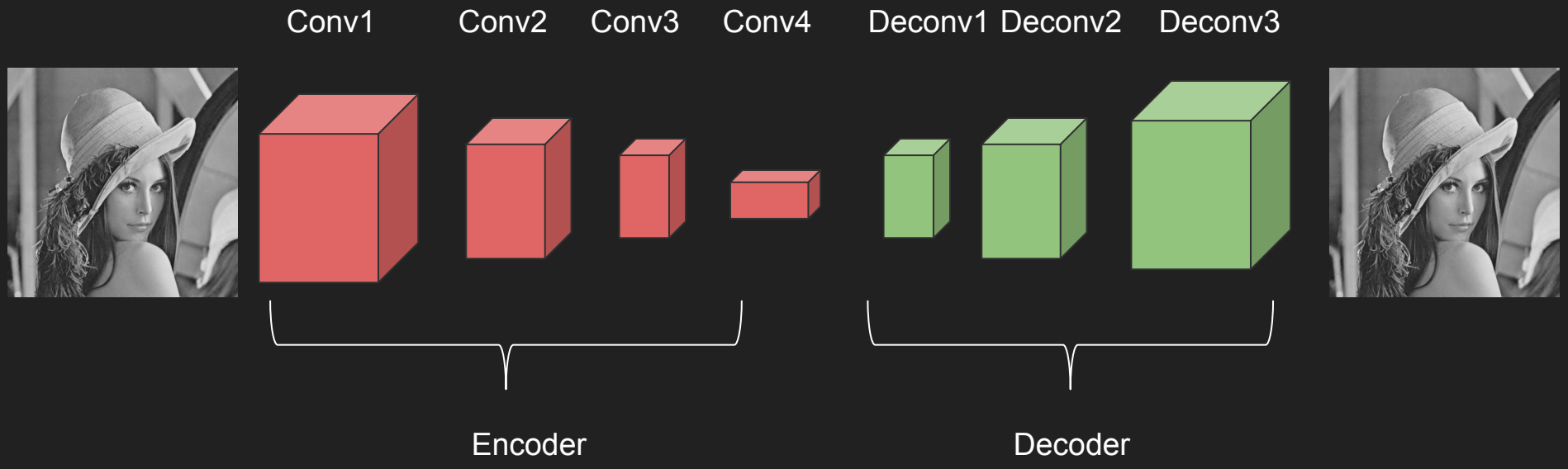
Iteration 10000

Output	Ground truth	
0.05	0	Cat
0.95	1	Dog

min(distance(output ,gt))

Cross entropy is a distance function(kind of) for probability distributions

Generative CNN - Autoencoder



Convolution + Bottleneck extracts the most significant features from the input to reconstruct the output

Generative CNN - Autoencoder

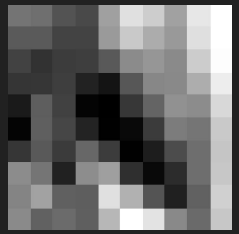
Conv1 Conv2 Conv3 Conv4 Deconv1 Deconv2 Deconv3

Encoder

Decoder

Iteration 0

Output



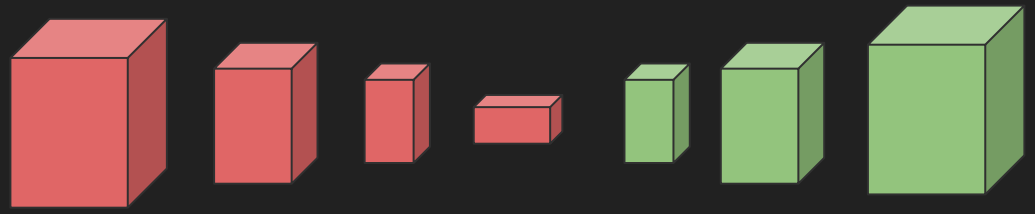
Ground truth

min(distance(output ,gt))

L1 and L2 norms are used for computing pixel distance

Generative CNN - Autoencoder

Conv1 Conv2 Conv3 Conv4 Deconv1 Deconv2 Deconv3



Encoder

Decoder

Iteration 10000

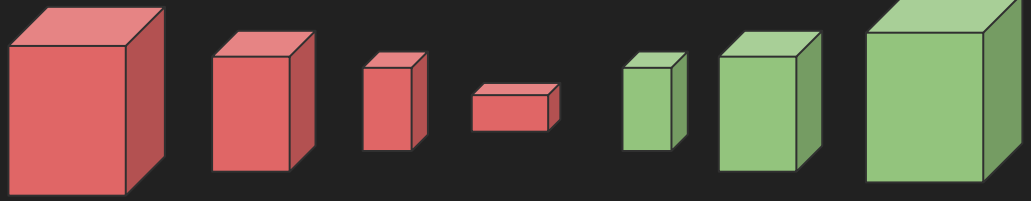
Output

Ground truth

min(distance(output ,gt))

L1 and L2 norms are used for computing pixel distance

Generative CNN - Autoencoder

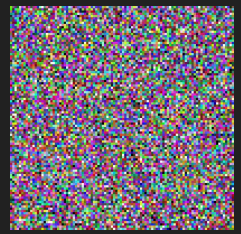


Encoder

Decoder

Iteration 0

Output

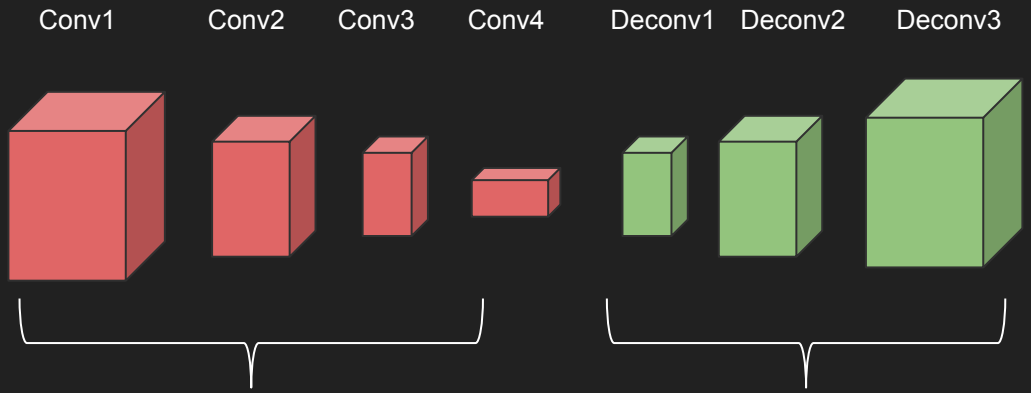


Ground truth

min(distance(output ,gt))

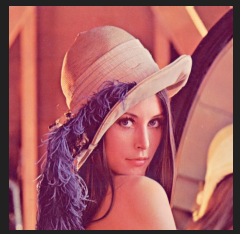
L1 and L2 norms are used for computing pixel distance

Generative CNN - Autoencoder

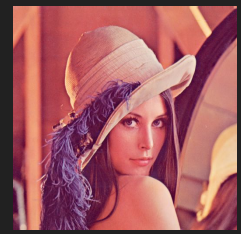


Iteration 10000

Output



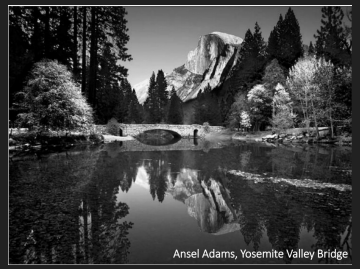
Ground truth

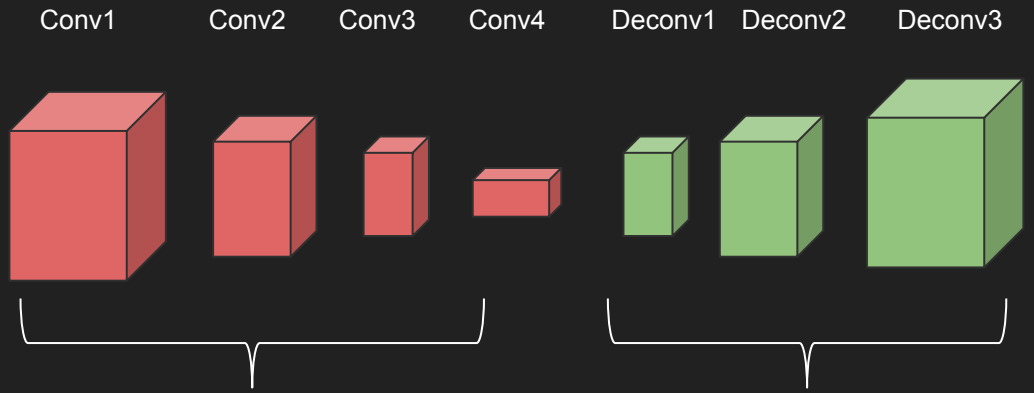


min(distance(output ,gt))

L1 and L2 norms are used for computing pixel distance

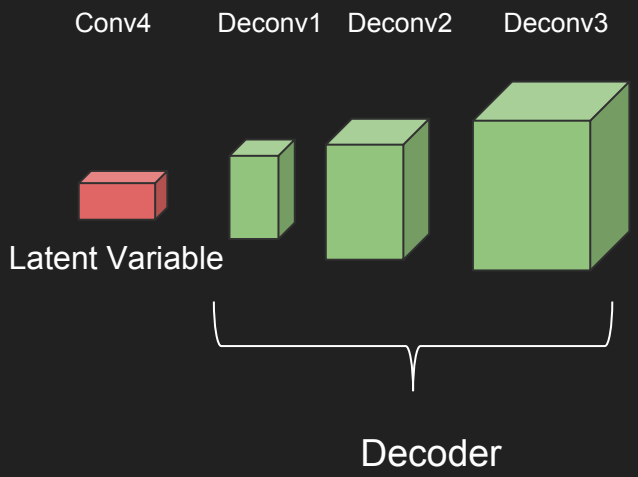
Generative CNN - Autoencoder

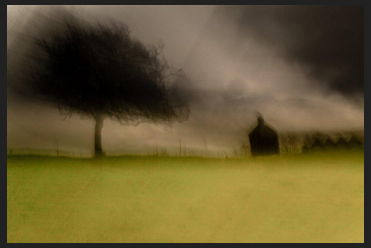


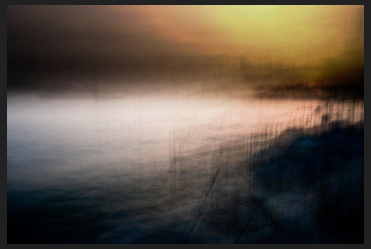


Dataset not used in training

Generative CNN - Autoencoder

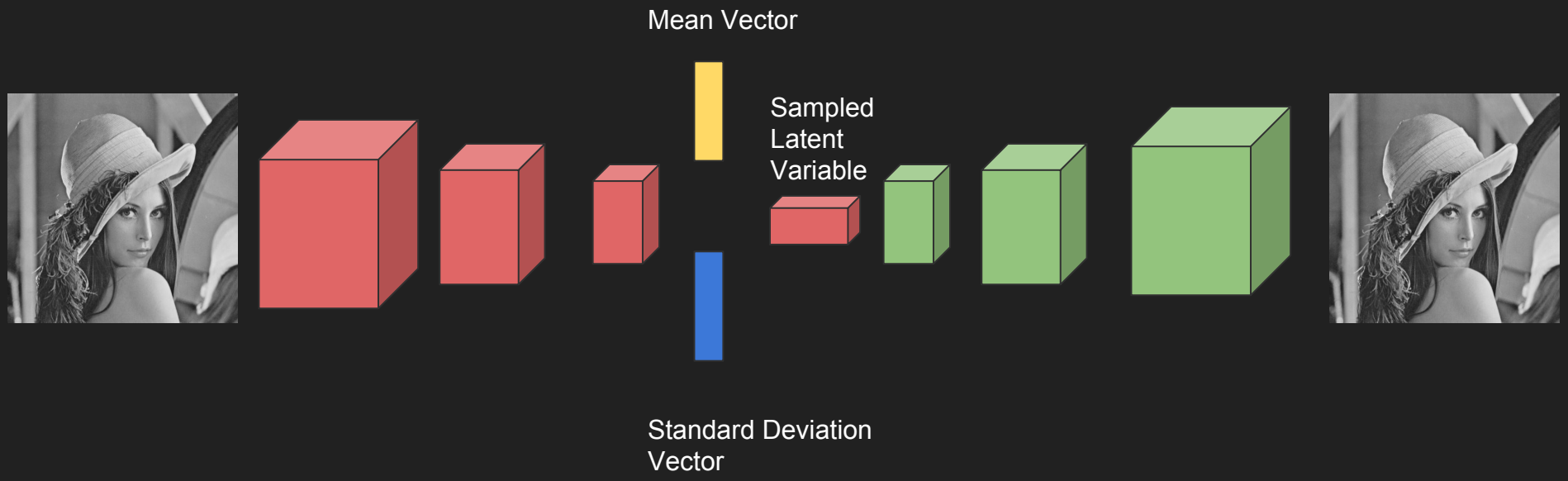






Changing the latent variable randomly will generate new images

Generative CNN - Variational Autoencoder



Generative CNN - Generative Adversarial Network

GANS