Introduction to Computer Vision and Convolutional Neural Networks

Wing Yue Geoffrey Louie and Suraj Goyal, HRI ECE-5900, Fall 2019

Traditional CV vs Deep Learning

Traditional Computer Vision

Hand crafted Features

HOG (Histogram of Gradient)
SIFT (Scale-Invariant Feature Transform)
SURF (Speeded-Up Robust Features)
BRIEF (Binary Robust Independent Elementary Features)

Simple Classifier

CAT

Deep Learning

Learned Features

Classifier

CAT

Introduction to CNN

Convolutional neural networks

- Also known as CNN or ConvNet are deep artificial neural networks
- Easier to train with fewer connections while results are similar to a standard neural network
- Used primarily to classify images, cluster them by similarity and perform object recognition within a scene or a frame.
- Common applications are facial recognition, traffic signs, tumor detection etc.
- Therefore, used in self driving cars, robotics, medical diagnosis, security etc.

Introduction to CNN – Convolution?

Convolution

- Element-wise multiplication of two matrices and adding all the elements together

Handcrafted 3X3 Element Filters

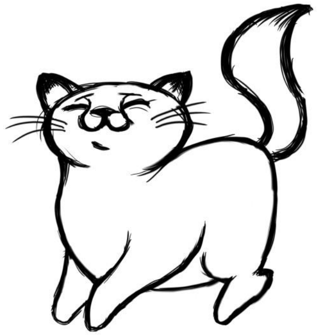
- $\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ *-> Detects vertical lines*

- $\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ -> Detects horizontal lines

Challenges to CV

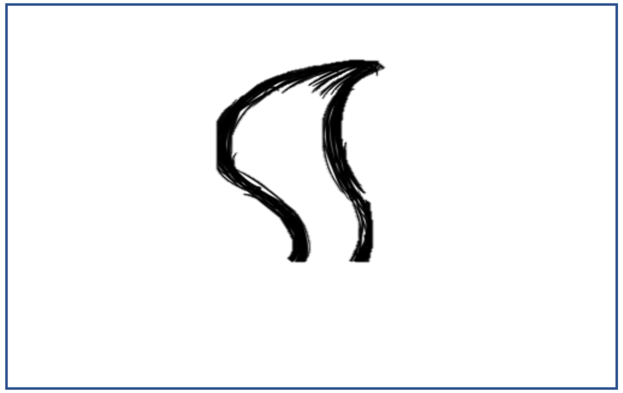
- A given object may be seen from any orientation, in any lighting conditions, with any type of occlusion from other objects, and so on. A true vision system must be able to “see” in any of an infinite number of scenes and still extract something meaningful.

CNN High Level Intuition



- Lets say you are fascinated with curved tails and you want to detect all the specific curved Cat tails in the world.
- Create a filter and call it the curved tail detector
- Therefore, we take a 5x5 filter ignoring the depth for simplicity

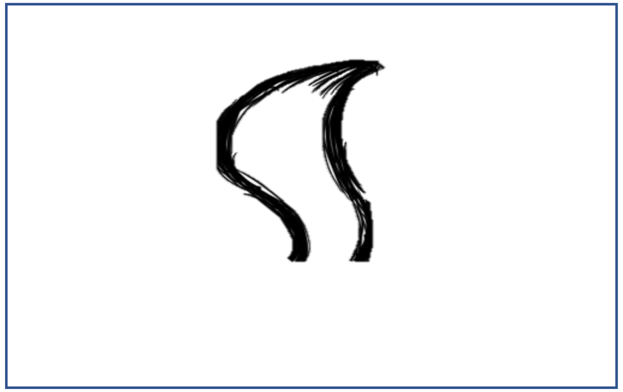
0	0	0	**0**	**24**
0	0	0	32	0
0	0	0	25	0
0	0	0	0	24
0	0	0	0	0



CNN High Level Intuition

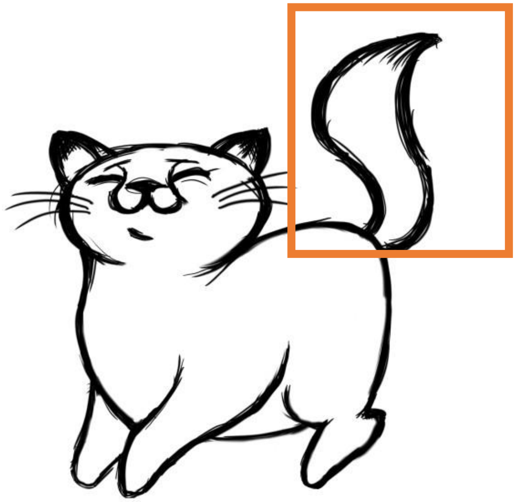
- Comparing the filter with the image for classification

0	0	0	**0**	**24**
0	0	0	32	0
0	0	0	25	0
0	0	0	0	24
0	0	0	0	0



$\star$

0	0	0	0	40	0
0	15	1	50	0	0
1	2	4	40	0	0
8	0	0	1	44	0
0	10	0	0	0	0
0	0	12	14	1	0
0	7	4	1	0	0

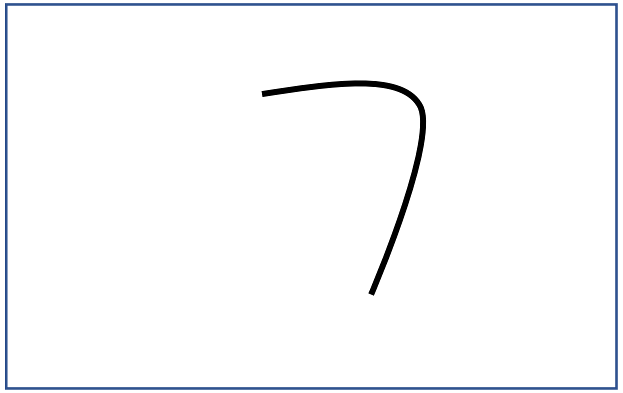


<u>Example not valid after the first conv layer or even the first iteration</u>

CNN High Level Intuition

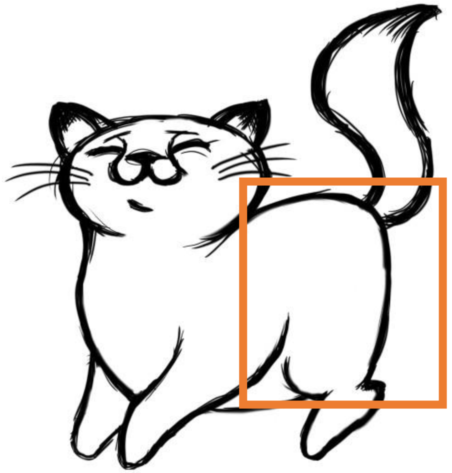
- Comparing the filter with the image for classification

0	0	0	**0**	**0**
0	0	20	32	0
0	0	0	25	0
0	0	0	10	0
0	0	0	0	0



$\ast$

0	0	0	0	40	0
0	15	1	50	0	0
1	2	4	40	0	0
8	0	0	1	44	0
0	10	0	0	0	0
0	0	12	14	1	0
0	7	4	1	0	0



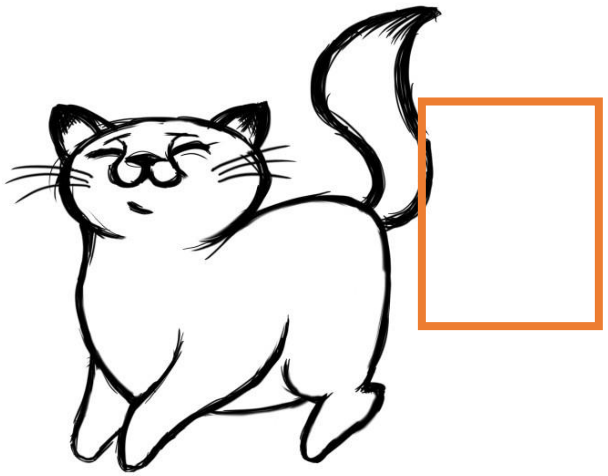
CNN High Level Intuition

- Comparing the filter with the image for classification

0	0	0	**0**	**40**
0	0	0	0	20
0	0	0	0	40
0	0	0	0	20
0	0	0	0	10

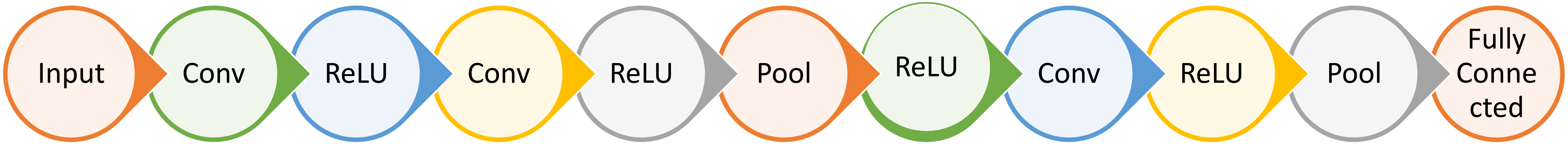
$\star$

0	0	0	0	40	0
0	15	1	50	0	0
1	2	4	40	0	0
8	0	0	1	44	0
0	10	0	0	0	0
0	0	12	14	1	0
0	7	4	1	0	0



<u>High mismatch with filter weights results in low value of the product or zero.</u>

CNN Classic Architecture



CNN High Level Intuition

- Filters in the first conv layer are designed to detect low level features such as curves and edges.
- To predict whether an image belongs to a certain class, we need to be able to recognize higher level features such as tails or paws or whiskers.
- For an input of a 32x32x3 image the output of the network after the first conv layer would be a 28 x 28 x 3 volume using three 5 x 5 x 3 filters -> Assuming default stride and padding

CNN High Level Intuition

- For the next conv layer, the output of the first conv layer becomes the input of the 2nd conv layer. Here instead of original image as an input, we have activation map(s) from the first layer.
- Each layer of the input is basically describing the locations of low level features in the original image. Filters on top of that (pass it through the 2nd conv layer), the output will be activations that represent higher level features.
- As you go through the network and go deeper, you get activation maps that represent more and more complex features.

CNN High Level Intuition

- After detecting high level features, we use a fully connected layer at the end.
- It takes an input volume, which is the output is of the preceding layer conv or ReLU or pool layer and outputs an N dimensional vector where N is the number of classes that the program has to choose from. Then it determines which features align with a particular class.
- For example, if we were classifying Cats based on Ears, Nose, Eyes, Tail and Whiskers, N would be 5. Each number in this N dimensional vector represents the probability of a certain class between 0 and 1.).
- 1 tail and 4 legs will have higher co-relation for a Cat vs a bird.

CNN High Level Intuition

- Remember backpropagation, that is how the model is able to adjust its weights or the filter values.
- This happens through multiple iterations of a forward pass, loss function, backward pass, and weight update to reduce the loss
- 1st step is a training image passed through the whole network.
- For this the network assigns a random weight or filter value to trigger an output from the network.

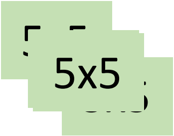
CNN High Level Intuition

- But after the network output it is compared against the predicted value to calculate the loss function.
- After this a backward pass is performed through the network.
- This is to determine which weights contributed most to the loss and finding ways to adjust them so that the loss decreases.
- Finally weights are updated finishing a single training step/ iteration.

Introduction to CNN

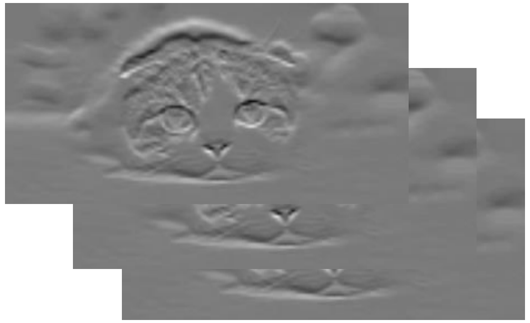
Input 32x32x3

RGB Split



5x5x3, 3 conv filters

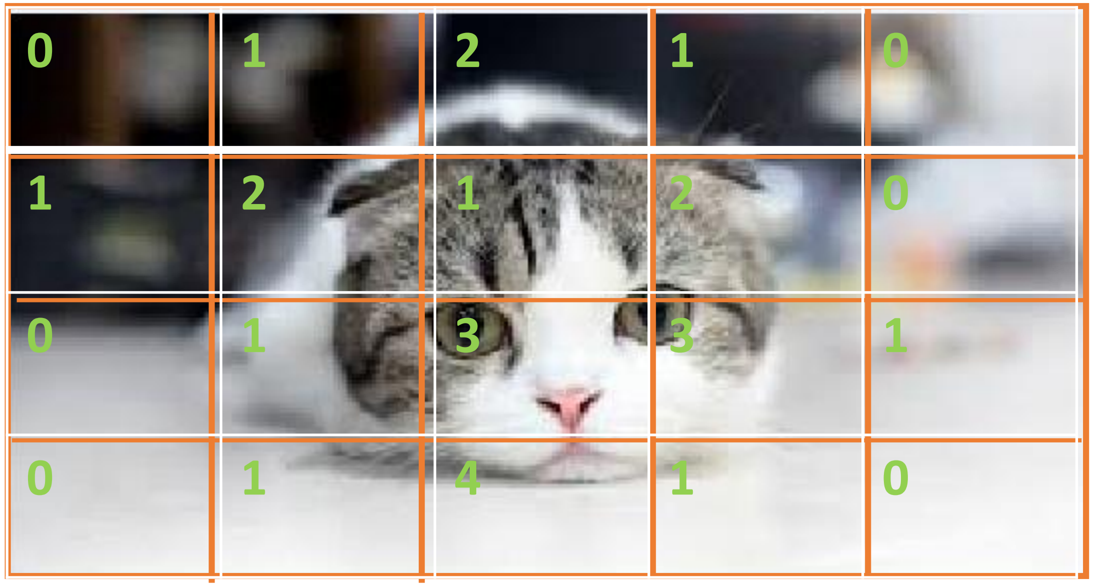
28x28x3



P(Bird)	0.01
P(Cat)	0.80
P(Dog)	0.04
P(Panda)	0.14
P(Fish)	0.01

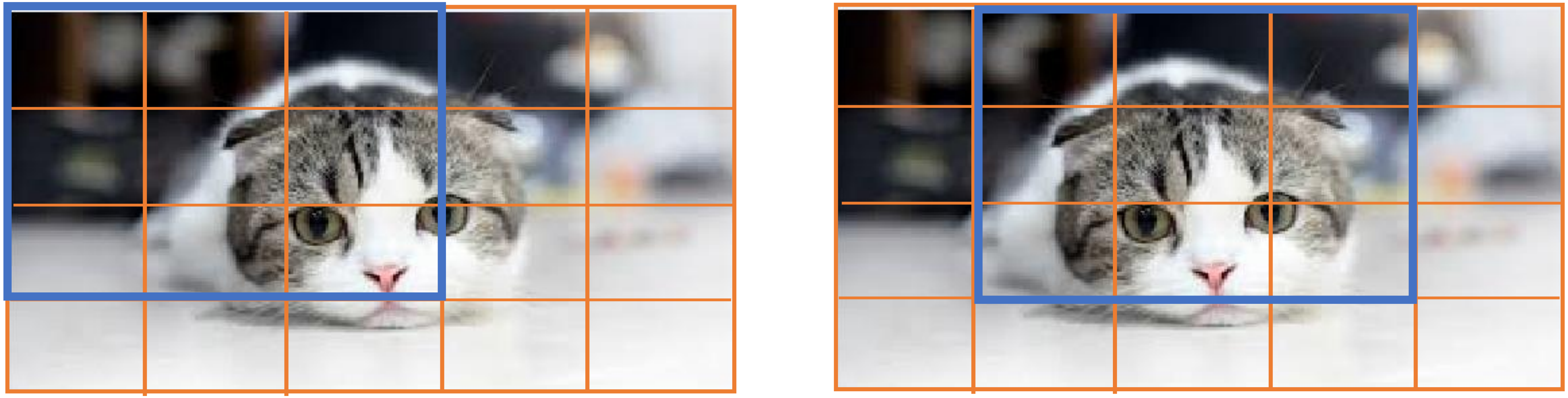
Softmax Output

Cat Image

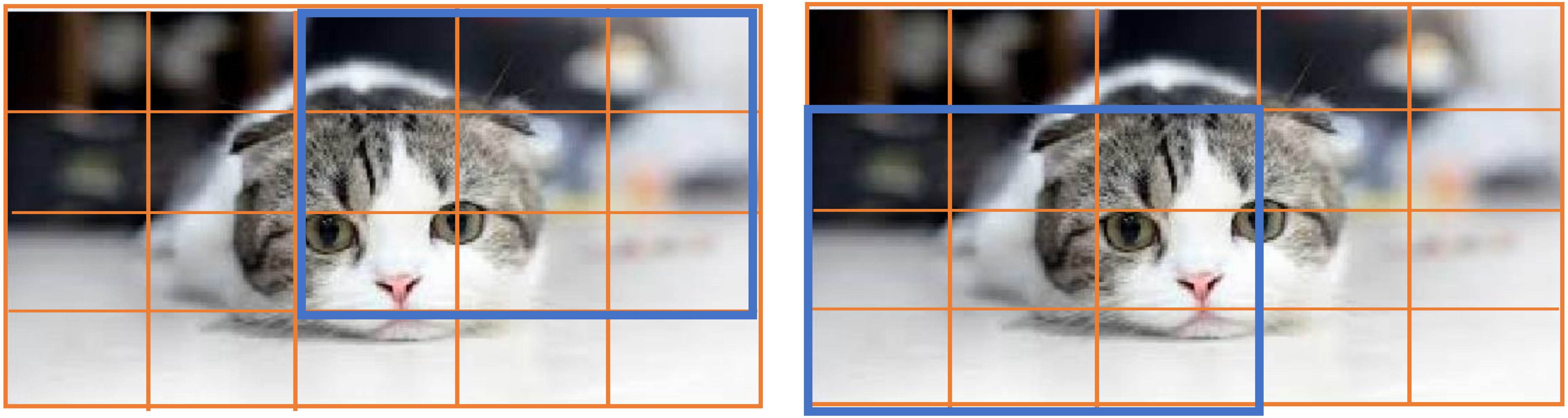


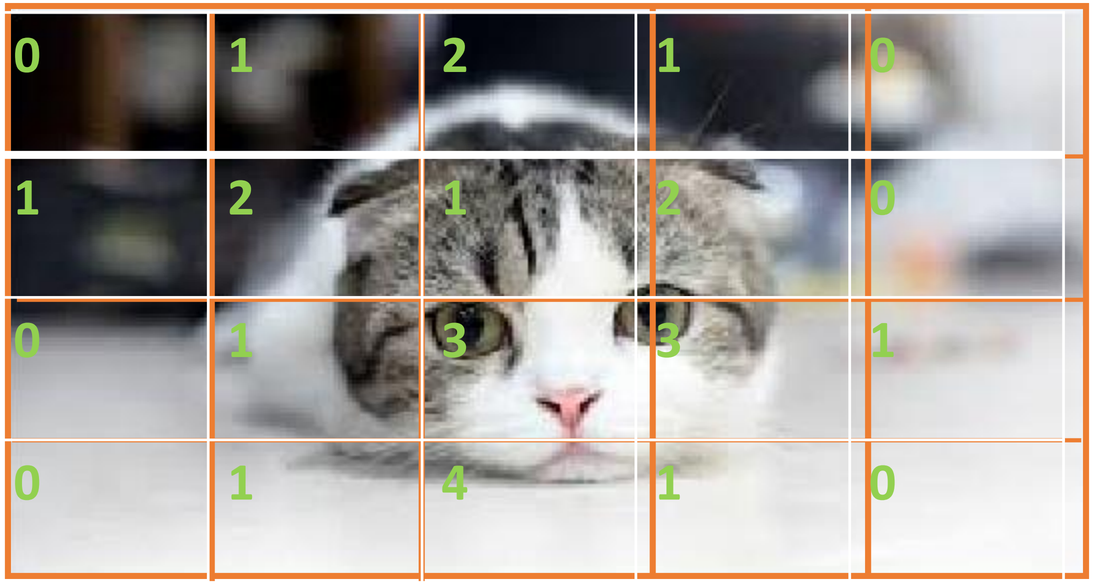
Cat Image – Converted to Pixels

0	**1**	**2**
1	2	1
0	1	3



Convolution Operation with stride 1



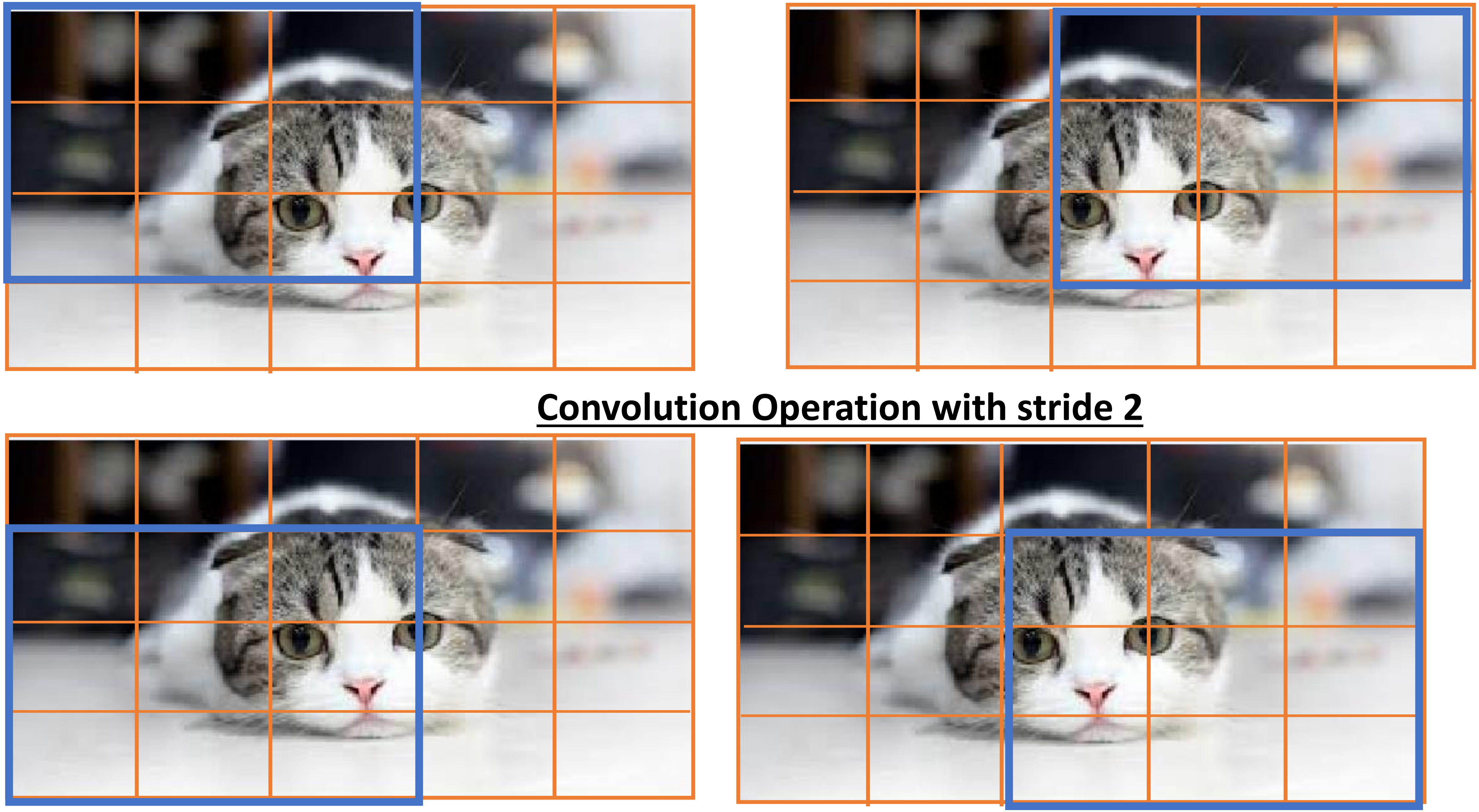


Cat Image – Converted to Pixels

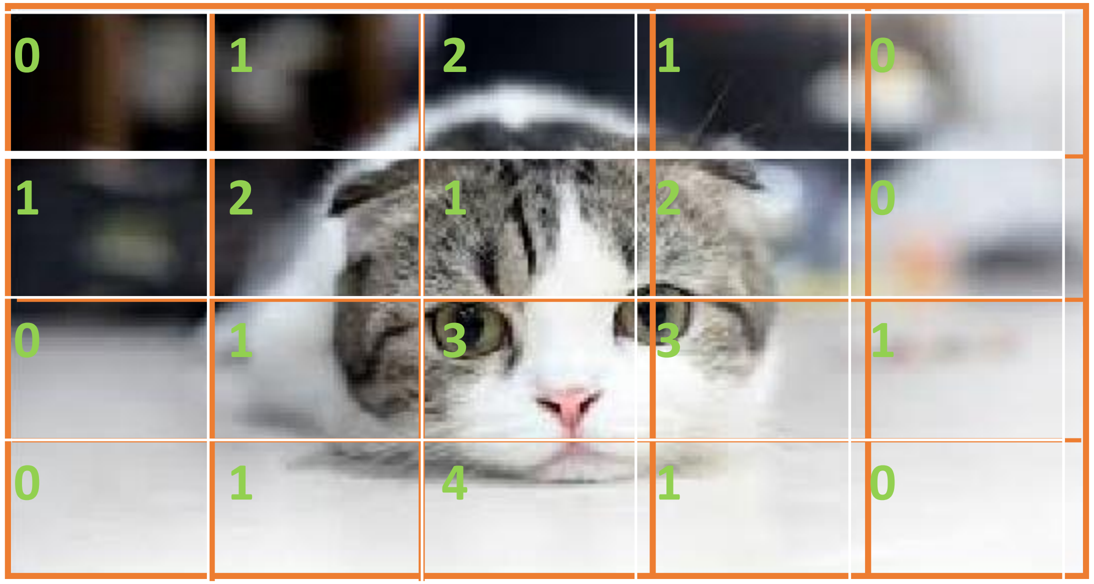
21	**22**	**12**
22	22	13

0	**1**	**2**
1	2	1
0	1	3

Kernel 3x3



Convolution Operation with stride 2



Cat Image – Converted to Pixels

?	**?**
?	?

0	**1**	**2**
1	2	1
0	1	3

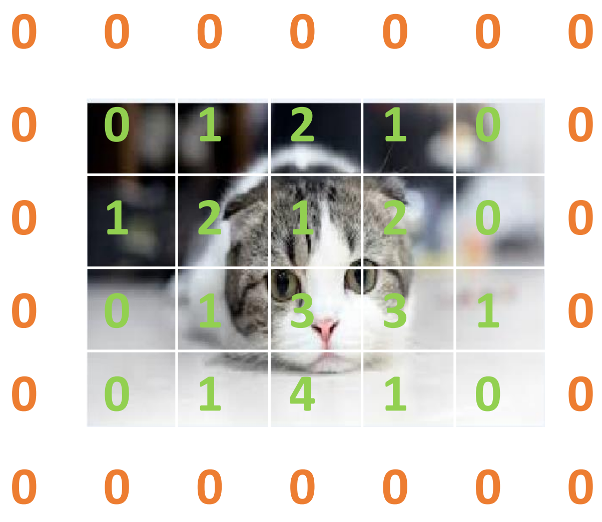
Kernel 3x3

CNN - Dimension Reduction Problem

Convolutional neural networks

- When we apply three 5 x 5 x 3 filters to a 32 x 32 x 3 input -> output is 28 x 28 x 3.
- With a deeper network the size of the volume will decrease faster which leads to a problem of losing essential information from the original input to extract low level features.
- Can we preserve the dimensionality then?

CNN - Dimension Reduction Problem



<u>Padding</u>

- Zero padding = (K-1)/2; where K is the filter size
- Output size= {(W-K+2P)/S} + 1; where

W is the input size, K is the filter size, P is the padding, and S is the stride

Calculate the zero padding and filter size to maintain input and output size (32x32x3) to be the same for stride 1?

ConvNet Essential Components/ Layers

Input Layer

- Holds raw pixels of the image
- [W, H, Channel], where image of Width W, Height H and 3 RGB color channels

Conv Layer

- Computes neuron outputs
- Dot Product is calculated between weights and region connected to input volume
- [W, H, N], where N defines the no. of filters

Activation Function

- RELU
- Results in same volume as the Conv Layer[W,H,N]

Pool Layer

- Downsampling and result may look like [W/2,H/2, N]
- Average Pooling: Calculate the average value for each patch on the feature map.
- Max Pooling: Calculate the maximum value for each patch of the feature map.
- Reduces computational complexity

Fully Connected Layer

- Connects every neuron in one layer to every neuron in another layer
- Results in [1,1,C] where C is the class score among categories of a dataset

Popular CNN architectures

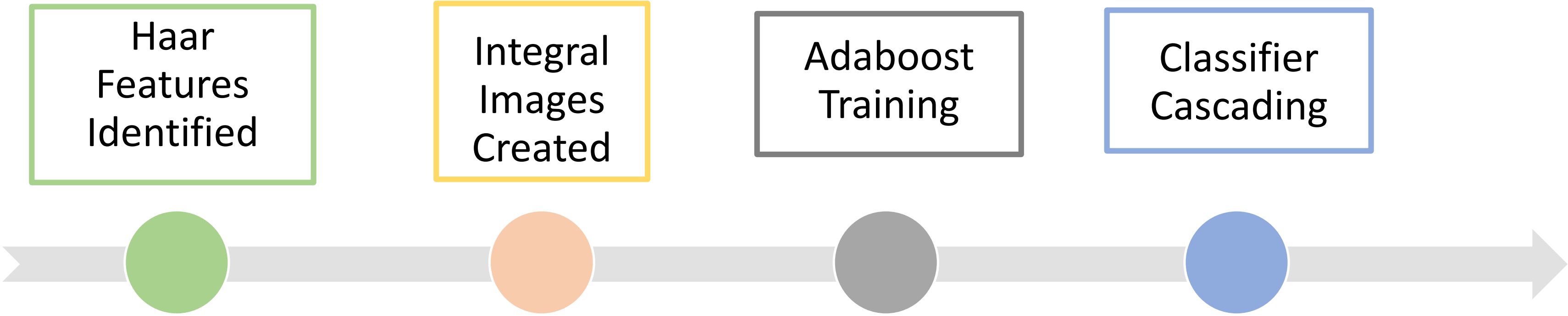
Convolutional neural networks

- AlexNet
- GoogleNet
- ResNet
- VGG

Haar Cascades

Haar Cascade - Machine learning object detection algorithm to identify objects in an image or video. It is based on the concept of features proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.

In this a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.



Haar Cascades

Initial step is to provide the algorithm face images and non-face images to train the classifier.

Next step is to collect the Haar Features. These are similar to CNN kernels or weight filters used to detect features without the training aspect. A Haar feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums.

-1	**-1**	**5**
-1	-1	**5**
-1	-1	**5**

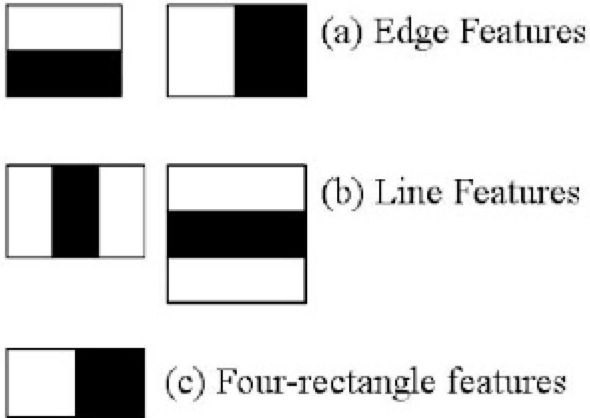
5	**5**	**5**
-1	-1	**-1**
-1	-1	**-1**

-1	**-1**	**-1**
5	**5**	5
-1	-1	**-1**

-1	**5**	**-**	**-1**	**-1**	**-1**
-1	**5**	-	**5**	**5**	5
-1	5	**-**	-1	-1	**-1**

-1	**5**	**-1**
-1	**5**	-1
-1	5	**-1**

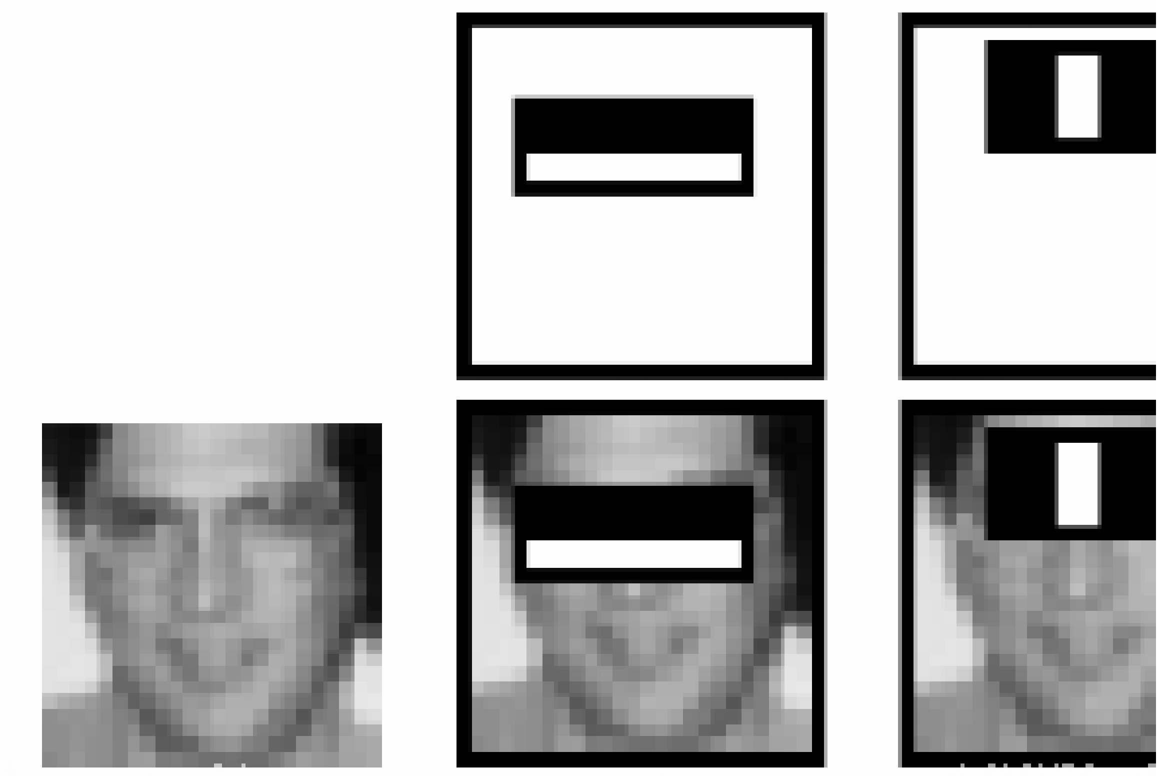
[Reference: http://www.willberger.org/cascade-haar-explained/](http://www.willberger.org/cascade-haar-explained/)



Haar Cascades

Top row shows two good features.

- First feature seems to focus on the region of the eyes as the region is often darker than of the nose and cheeks.
- Second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant.



[Reference: http://www.willberger.org/cascade-haar-explained/](http://www.willberger.org/cascade-haar-explained/)