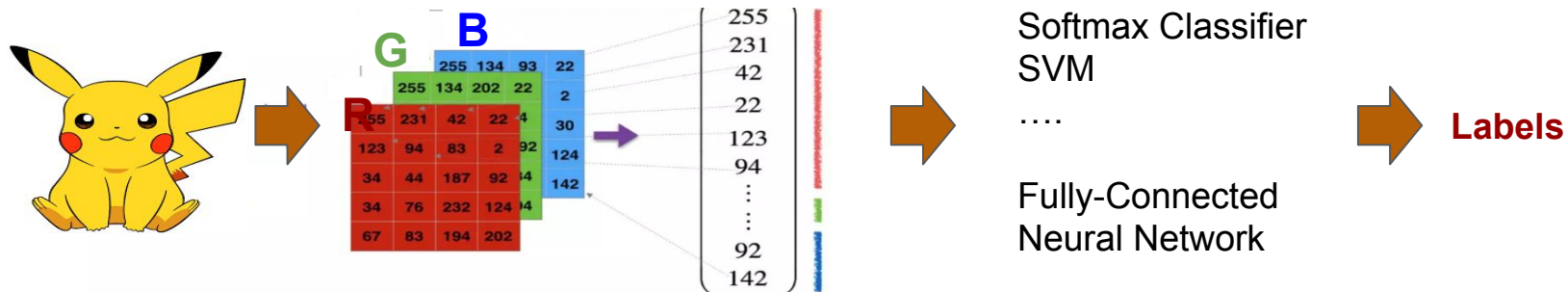


Convolutional Neural Network

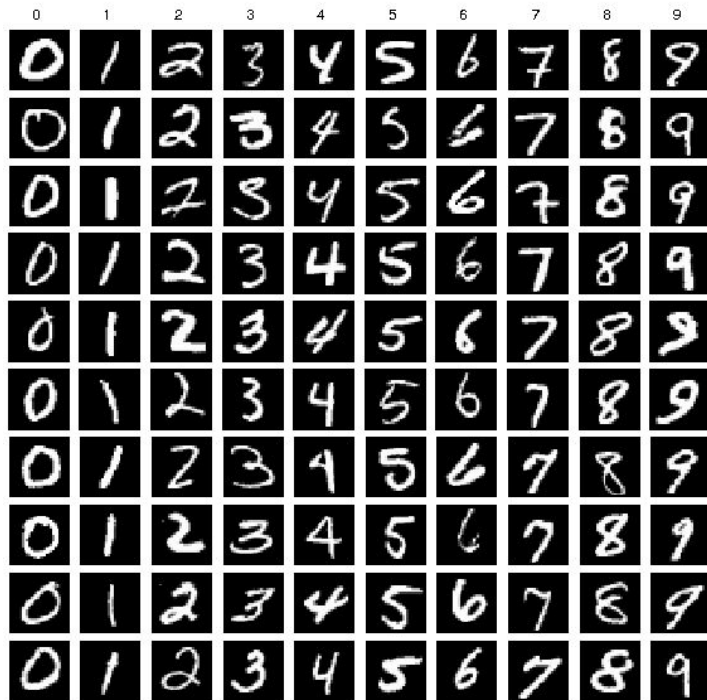
How it “understand” Image and Text

Before CNN

Computers See Image



Think about MNIST Dataset



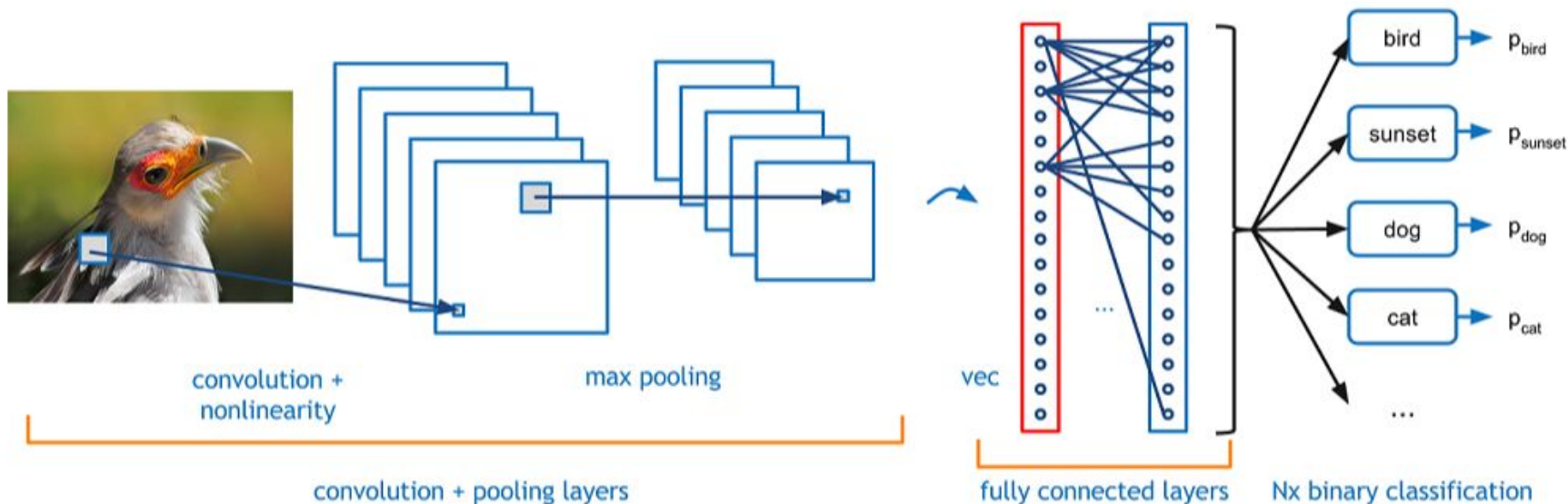
The above model requires the digit should be in the center of the image and it had to be the only thing in the image.

Intro to CNN



https://www.youtube.com/watch?v=FwFduRA_L6Q

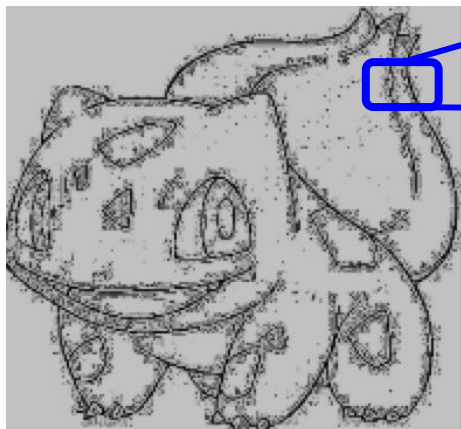
Convolutional Neural Network



**Extracting useful
features of data**

**Perform a ML task (like
classification based on the
vectorized data)**

Filter Operation



Consider
neighbor values

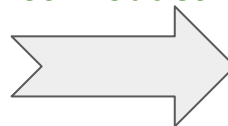
0	32	35
34	203	122
132	223	163

Current Pixel Value is 203

-1	0	-2
.5	4.5	-1.5
1.5	2	-3

Filter (3 by 3)

Dot Product



New Pixel Value

$$\begin{aligned} & (-1 * 0) + (0 * 32) + (-2 * 35) + \\ & (.5 * 34) + (4.5 * 203) + (-1.5 * 122) + \\ & (1.5 * 132) + (2 * 223) + (-3 * 163) \end{aligned}$$

Convolutional Operation

- Apply the **same** filter for every pixel in the original image
- Filter Size is the shape of the filter matrix (yellow one)

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

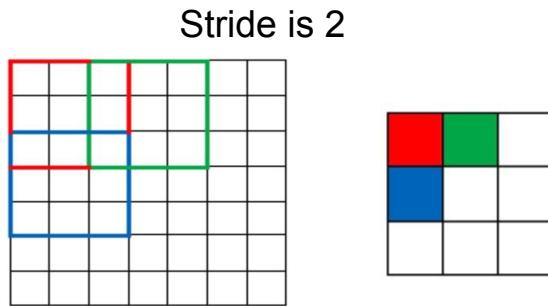
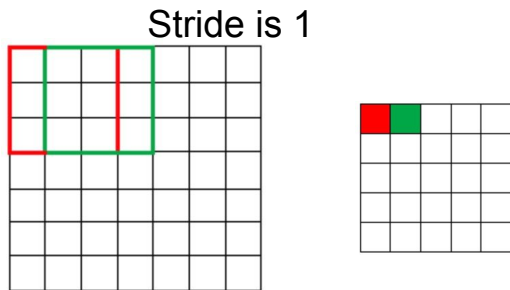
4		

Convolved
Feature

*Feature
Map*

Stride

- Controls how the filter move around the image
- It is the amount by which the filter shifts



Zero Padding

- Pads the image with zeros around the **border**
- Make the input image and feature map have the same spatial dimensions

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

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

114				

Stride: 1

Size of zero padding:

$$(k-1)/2$$

<https://stackoverflow.com/questions/52067833/how-to-plot-an-animated-matrix-in-matplotlib>

Convolutional Operation

- Filter Size: K
- Stride Size: S
- Padding Size: P

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

Input size

$$O = \frac{W - K + 2P}{S} + 1$$

Output size

Multi-Channel CNN

- A color image is a 3-D tensor
- 400 (height) 630 (width) 3 (R,G,B channels)

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...

Input Channel #3 (Blue)

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

Kernel Channel #1

308

+

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

Kernel Channel #2

-498

+

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

164

+ 1 = -25

Bias = 1

Output

-25			...
			...
			...
			...
...

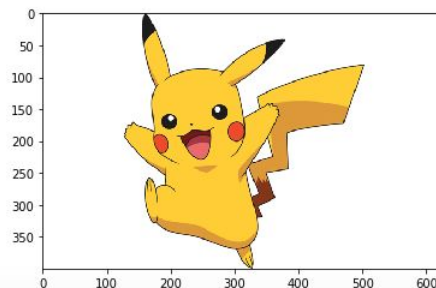
```
from matplotlib.image import imread
import numpy as np
img = imread('pikka_3.jpg')
```

```
print(img.shape)
```

```
(400, 630, 3)
```

```
plt.imshow(img, interpolation='nearest')
```

```
<matplotlib.image.AxesImage at 0x11b404278>
```



https://www.researchgate.net/post/How_will_channels_RGB_effect_convolutional_neural_network

How Filter Works



Image

-1	0	1
-2	0	2
-1	0	1



Convolved
Features

Only Keep Vertical Lines

How Filter Works



Image

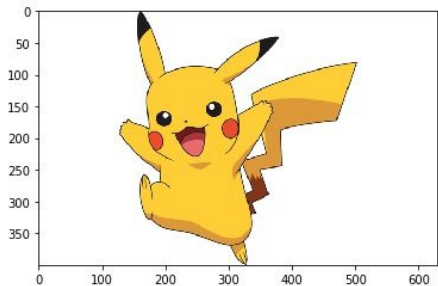
-1	-2	-1
0	0	0
-1	2	1



Convolved
Features

Only Keep Horizontal Lines

Filter comes from “Image Processing”

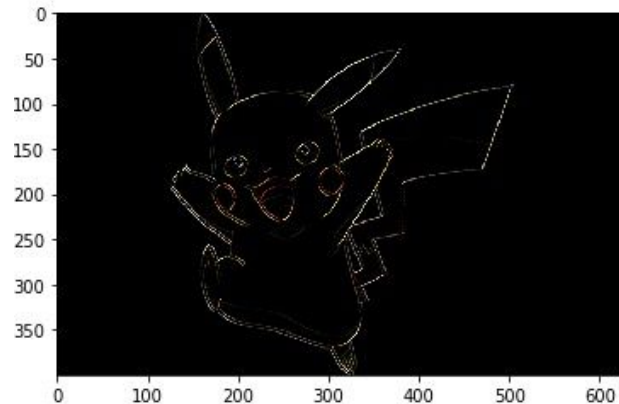


Image

```
print(kernel)
```

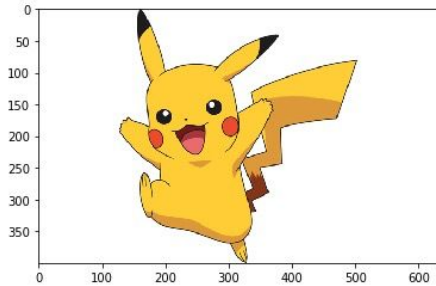
```
[[ 1  0 -1]  
 [ 0  0  0]  
 [-1  0  1]]
```

Edge
Detection



Convolved
Features

Filter comes from “Image Processing”

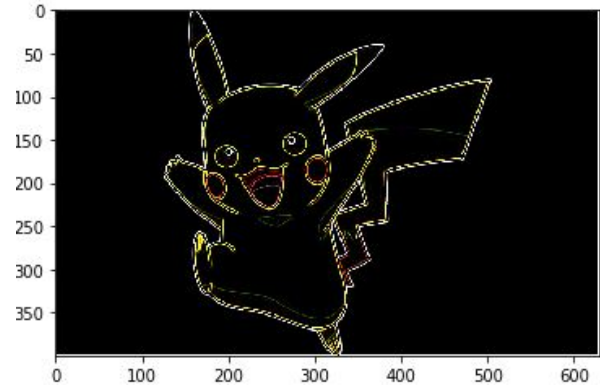


Image

```
print(kernel)
```

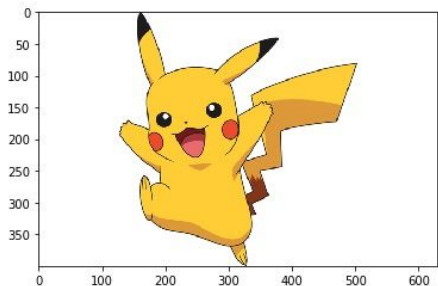
```
[[-1 -1 -1]  
 [-1 8 -1]  
 [-1 -1 -1]]
```

Edge
Detection



Convolved
Features

Filter comes from “Image Processing”

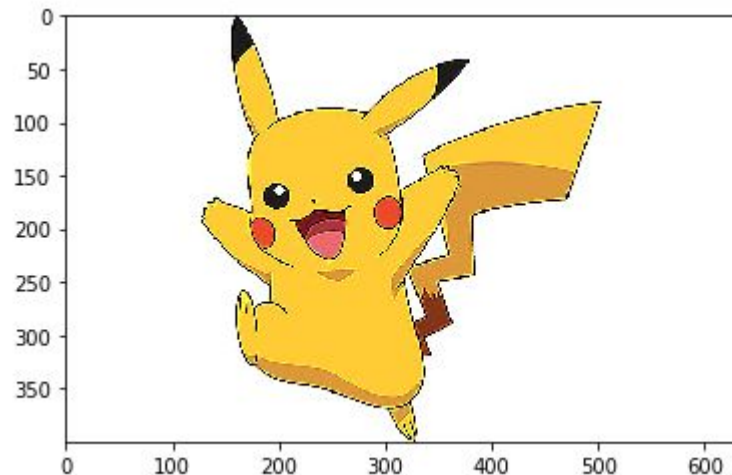


Image

```
print(kernel)
```

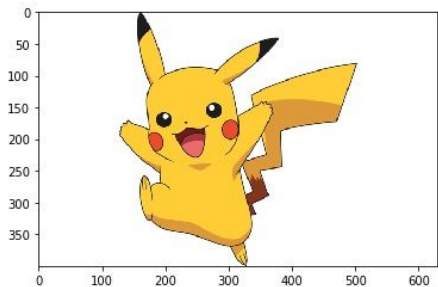
```
[[ 0 -1  0]  
 [-1  5 -1]  
 [ 0 -1  0]]
```

Sharpen



**Convolved
Features**

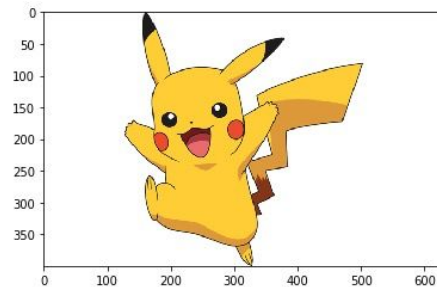
Filter comes from “Image Processing”



Image



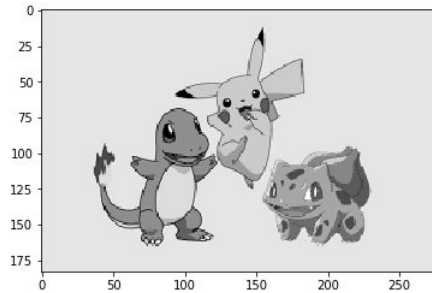
Identity



**Convolved
Features**

Non-linear Activation

- In nature, filter operation is dot product.
- In deep learning, we need to have non-linear transformation.
- Add non-linear activation



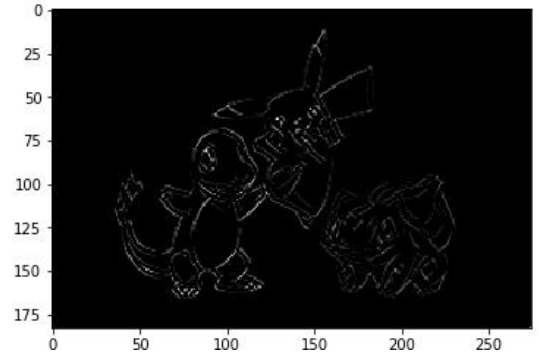
Image

```
print(kernel)
```

```
[[ 1  0 -1]  
 [ 0  0  0]  
 [-1  0  1]]
```



non-linear

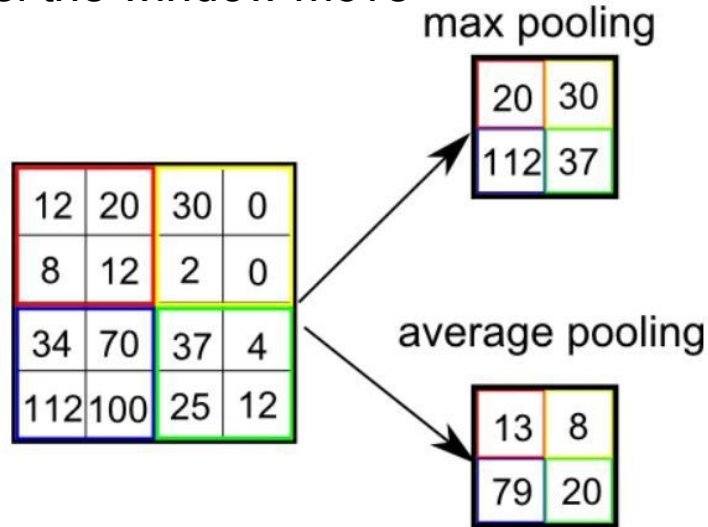


The First Task in Assignment II

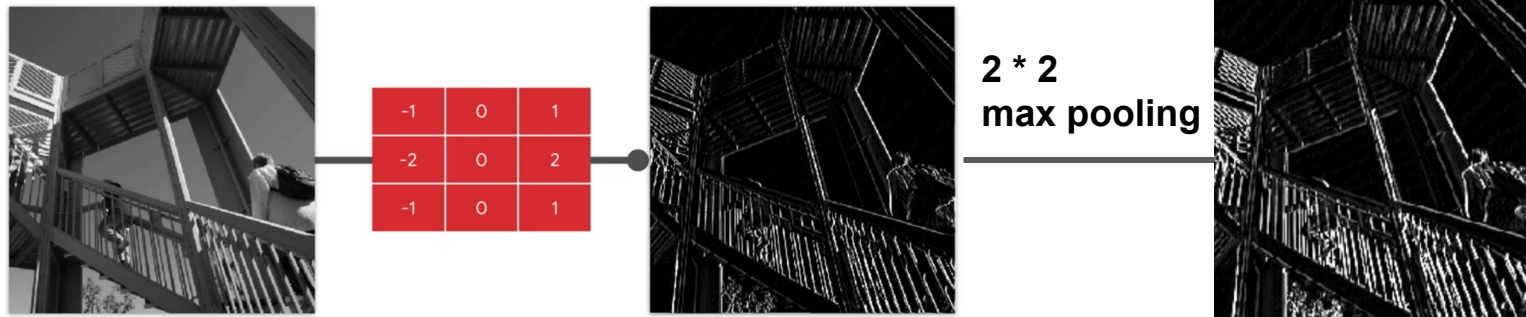
Pooling Operation

- Pooling Size: the box size. Here is 2×2
- Stride Size: how much pixel the window move

What is stride size here ?

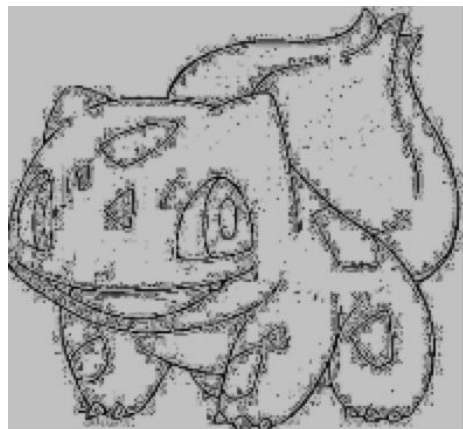


Filter then Pool



1. The size is **one quarter** the original size
2. The **vertical line** features are **enhanced**.

Conv-Pool



Conv-Pool

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}_{n \times m}$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}_{n \times m}$$

○
○
○

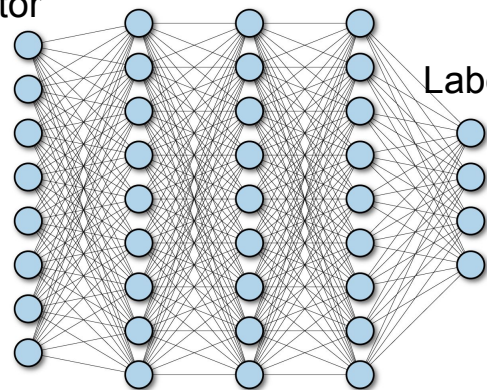
$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}_{n \times m}$$

Flatten

Concatenate

vector

Labels



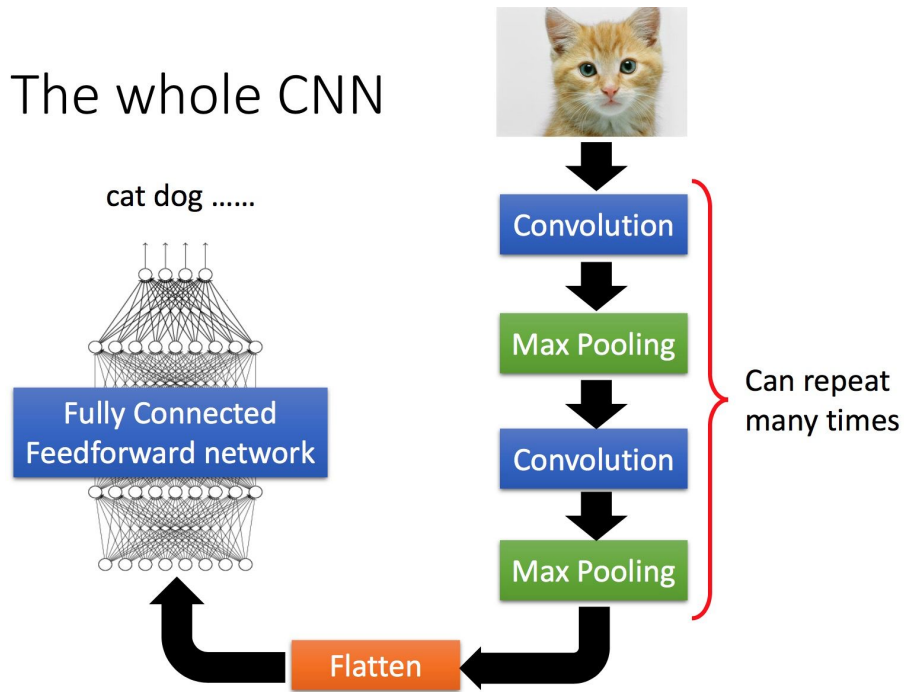
Where are these filters from?

- Filters, in nature, are model parameters, which can be **learned** by backpropagation.
- These filters weights are firstly randomly initialized, and then updated during training process.
- End-to-End optimization: Backpropagation.
- More details:
<https://towardsdatascience.com/training-a-convolutional-neural-network-from-scratch-2235c2a25754>

CNN Can be Deep

- Convolution-Pooling can be followed by another Convolution-Pooling
- At the end, after flatten operation, fully connected layers are used to map the outputs.

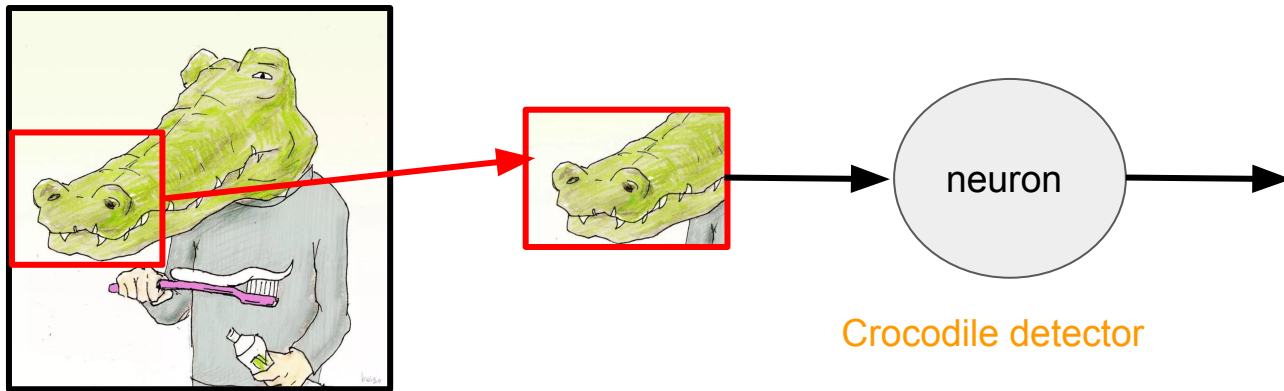
The whole CNN



Why CNN is Suitable for Images

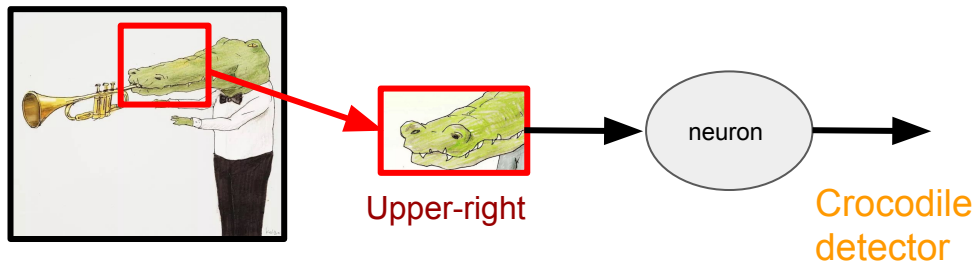
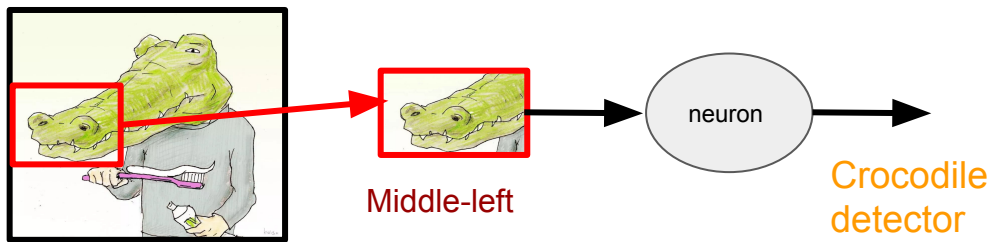
Local Features Matter

- Discriminative patterns are much smaller than the whole image
- A neuron does not have to see the whole image
- Less parameters required



Location Insensitive

- The same patterns appear in different regions
- A neuron should be location insensitive.



Subsampling Works

- Subsampling the pixels will not change the object
- We can subsample the pixels to make images smaller -> less parameters required

Crocodile



subsampling

Crocodile



Limits of CNN

CNN is different human vision

- CNN can handle translations. But they can not cope with the effects of **changing viewpoints such as rotation and scaling**
- Human is able to generalize knowledge.

neatly positioned

ImageNet

Chairs



Real world

ObjectNet

Chairs by rotation



Chairs by background



Chairs by viewpoint




Teapots



T-shirts



CNN is different human vision


$$\begin{array}{ccc} x & + .007 \times \text{sign}(\nabla_x J(\theta, x, y)) & = x + \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \\ \text{"panda"} & \text{"nematode"} & \text{"gibbon"} \\ 57.7\% \text{ confidence} & 8.2\% \text{ confidence} & 99.3\% \text{ confidence} \end{array}$$

Adversarial examples can cause neural networks to misclassify images while appearing unchanged to the human eye

Solutions

- Use 4D or 6D maps to train machine learning model
 - Too expensive
- Get huge-size training data that cover all positions of objects.
 - Data augmentation: flip the image or rotate it by some angle. Then, CNN will be trained on multiple copies of every image, each being slightly different.
 - It will never cover all of corner cases.



Enlarge your Dataset

<https://www.kdnuggets.com/2018/05/data-augmentation-deep-learning-limited-data.html>

CNN is different human vision

- CNN may get confused by seeing this bizarre teapot, since they can not understand images in terms of objects and their parts.
- Human is able to decompose an object into parts and then we can understand its nature.



CNN for Text

CNN works for Text

Images

- Local Features Matter
- Locations Insensitive
- Subsampling Works

Texts

- Key n-grams define semantics
*Pulp fiction's director is Quentin. I **am obsessed of** it.*
- Locations of key n-grams Insensitive?
*I **am obsessed of** Pulp fiction, whose director is Quentin.*
*Pulp fiction's director is Quentin. I **am obsessed of** it.*

*I owe **you** ten dollars*
***You** owe **me** ten dollars.*
- Doc. Summarization

Combinations

E.g., I hate this movie

- Compute vectors for every possible phrase
 - *I hate this movie* ----> I hate; hate this; this movie
- Compute these vectors for these phrases

Convolution Operation

Word Vectors

I
like
this
movie
very
much
!

0.6	0.5	0.2	-0.1	0.4
0.8	0.9	0.1	0.5	0.1
0.4	0.6	0.1	-0.1	0.7
...
...
...
...

0.2	0.1	0.2	0.1	0.1
0.1	0.1	0.4	0.1	0.1

Filters updated
during training

0.51

I
like
this
movie
very
much
!

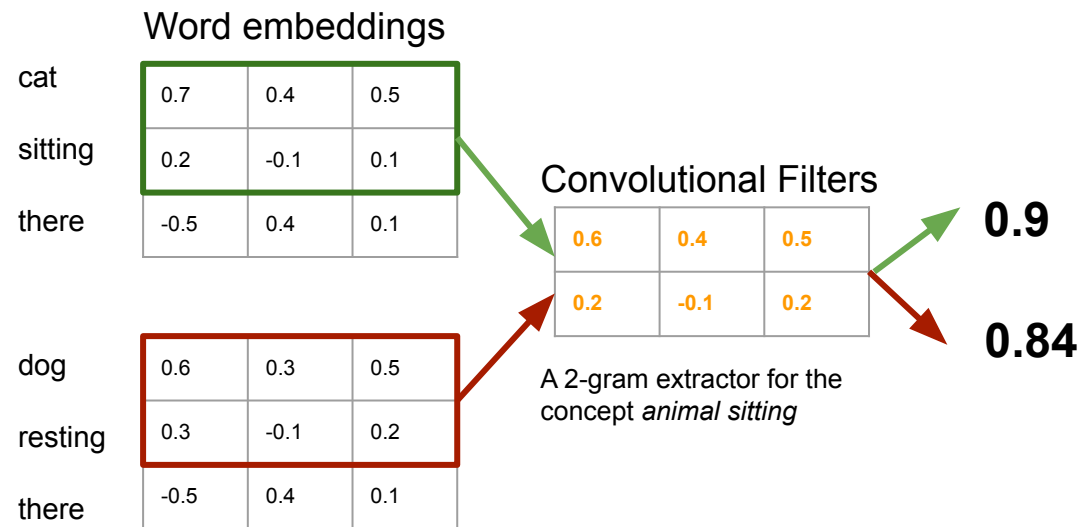
0.6	0.5	0.2	-0.1	0.4
0.8	0.9	0.1	0.5	0.1
0.4	0.6	0.1	-0.1	0.7
...
...
...
...

0.2	0.1	0.2	0.1	0.1
0.1	0.1	0.4	0.1	0.1

Feature Maps

0.51
0.53

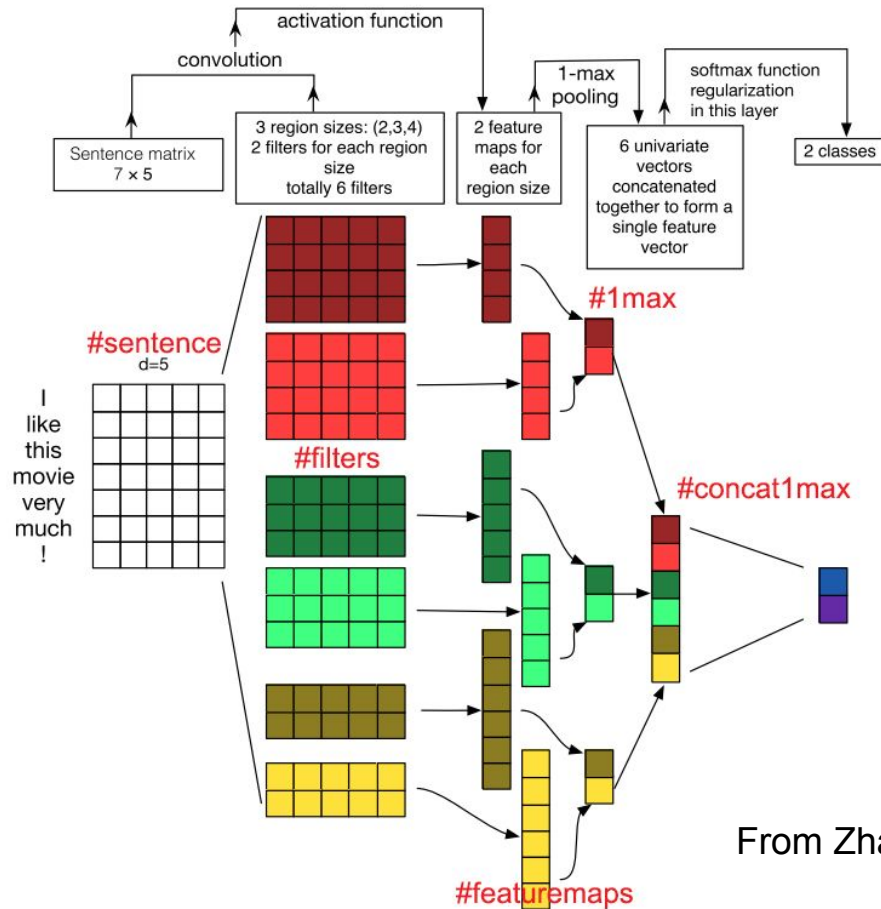
Toy Example



- This convolution provides high activations for 2-grams with certain meaning
- Can be extended to 3-grams, 4-grams, etc.
- Can have various filters, need to track many n-grams.
- They are called 1D since we only slice the windows only in one direction

Why is it better than BoW?

CNN Framework



From Zhang 2015

Multiple Channels

- Like image, CNN is applied on R-G-B channels
- For NLP, different word embeddings can be regarded as different channels

CNN for NLP

1. n-grams features are important (window size)
2. Location of **key** n-grams are trivial (pooling)
3. Stack of Convolutional layer or large window size can also capture long-range information