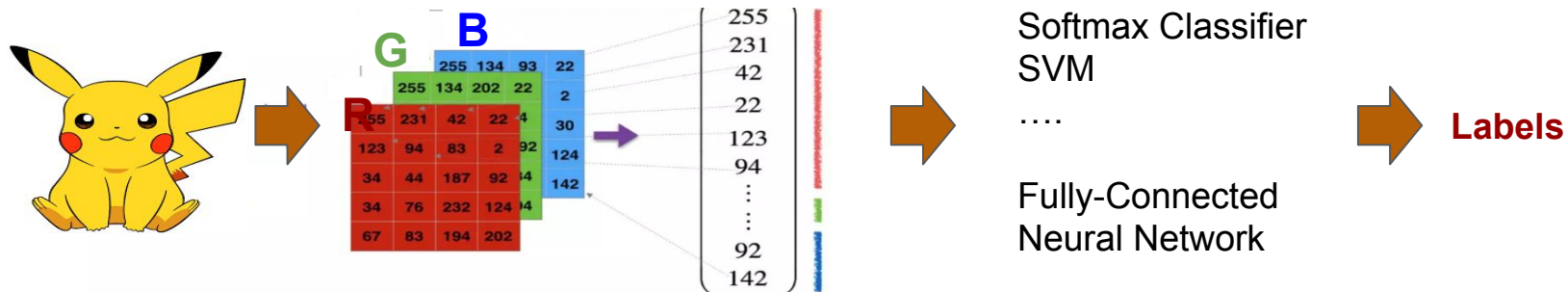


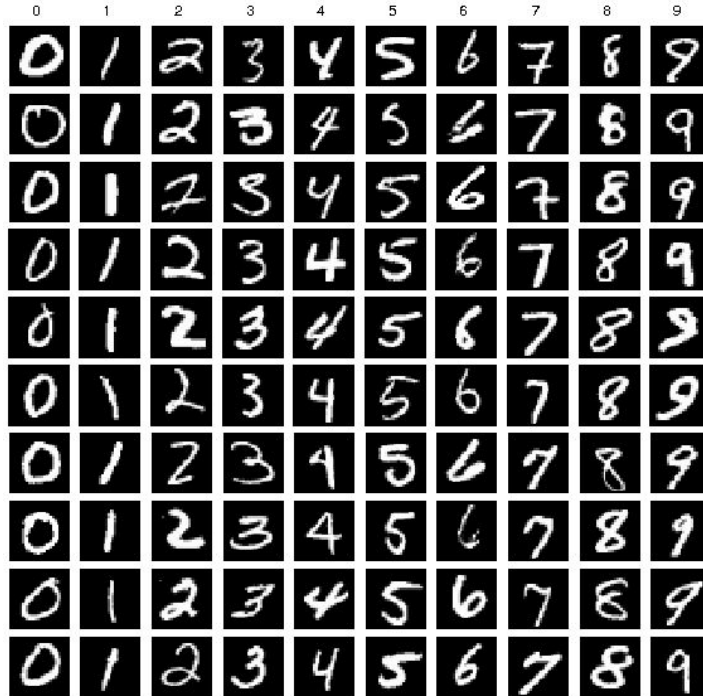
# Convolutional Neural Network

**Before CNN**

# Computers See Image



# Think about MNIST Dataset



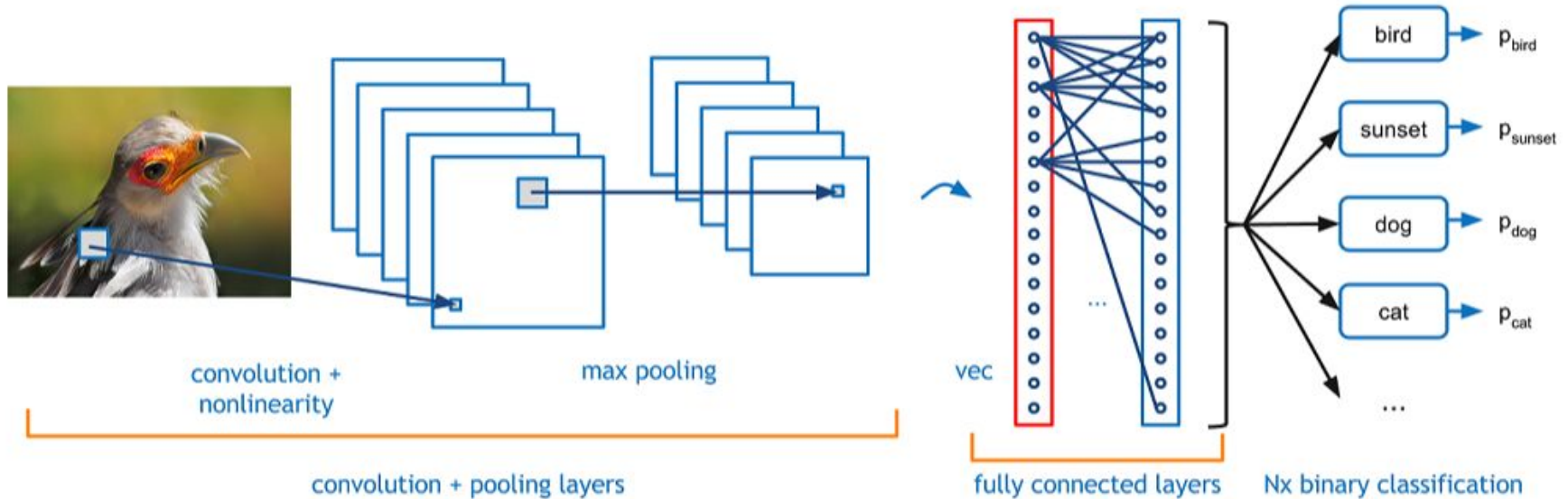
The above model requires the digit should be in the center of the image and it had to be the only thing in the image.

# Intro to CNN



[https://www.youtube.com/watch?v=FwFduRA\\_L6Q](https://www.youtube.com/watch?v=FwFduRA_L6Q)

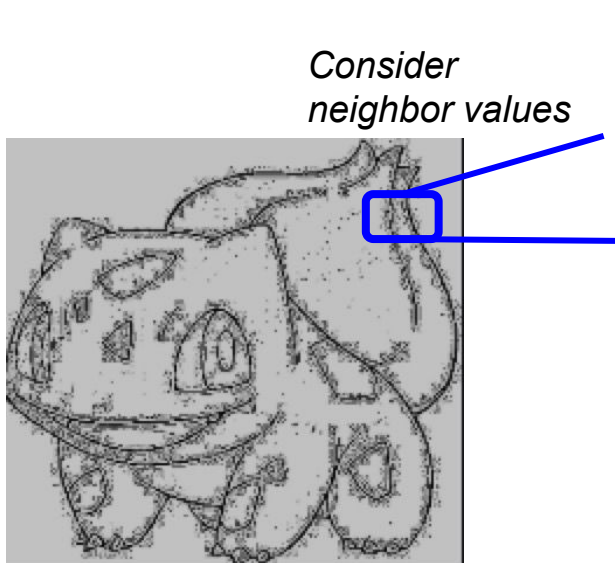
# Convolutional Neural Network



**Extracting useful  
features of data**

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

# Filter Operation



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



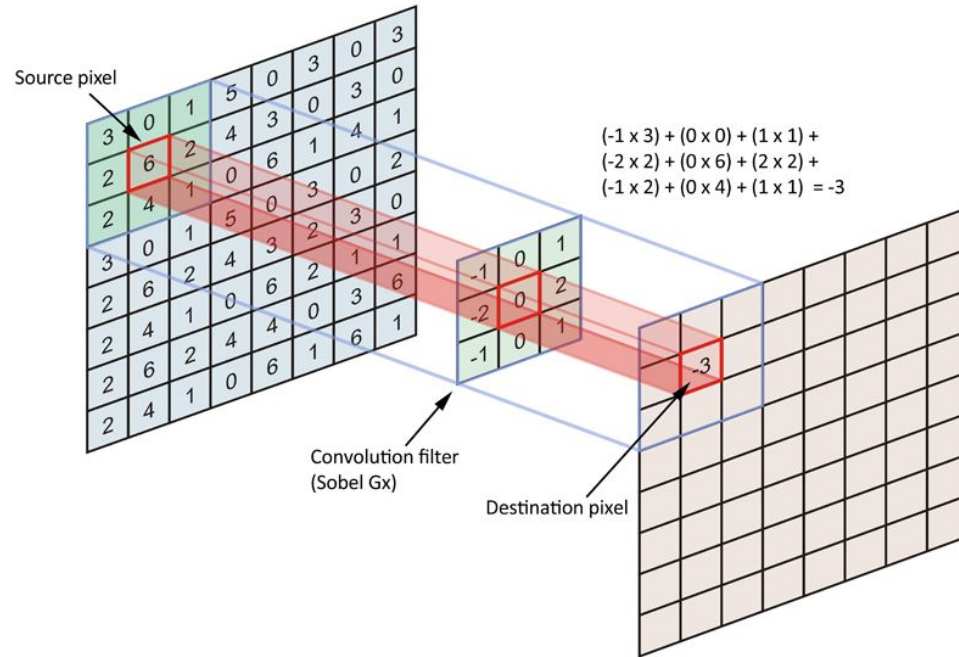
New Pixel Value

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

Filter Size



# Filter Operation



The intent of convolution is to encode source data matrix (entire image) in terms of a filter or kernel. More specifically, we are trying to encode the pixels in the **neighborhood** of **anchor/source** pixels

<https://datascience.stackexchange.com/questions/23183/why-convolutions-always-use-odd-number-s-as-filter-size>

# 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 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

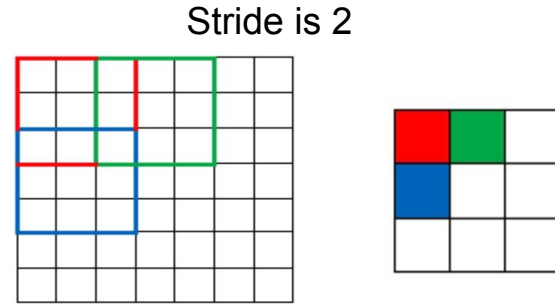
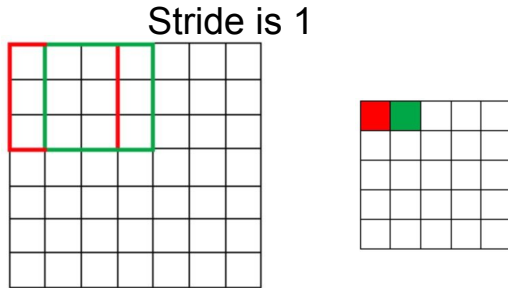
*Feature  
Map*

Check gif version here:

<https://docs.google.com/presentation/d/1V7IqLDsKXyaEwR9ZgxmlQ9ixmcT41ZGOLmJtbpgGPM/edit?usp=sharing>

# Stride Size

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



# Padding Size

- 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 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	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
0	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

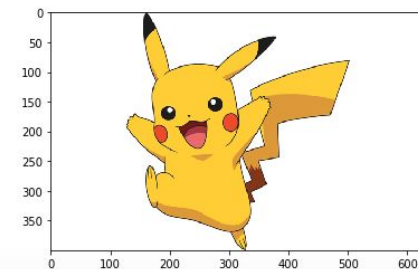
-25			...
			...
			...
			...
...	...	...	...

Output

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

print(img.shape)

plt.imshow(img, interpolation='nearest')
<matplotlib.image.AxesImage at 0x11b404278>
```



## From Keras Layers Conv2D

Input shape

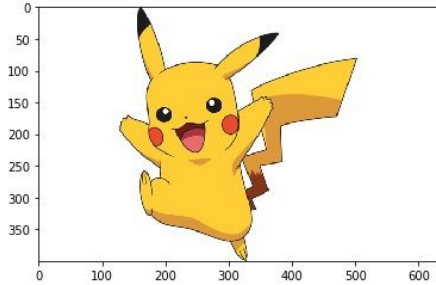
4D tensor with shape: (batch, channels, rows, cols) if data\_format is "channels\_first" or 4D tensor with shape: (batch, rows, cols, channels) if data\_format is "channels\_last".

Output shape

4D tensor with shape: (batch, filters, new\_rows, new\_cols) if data\_format is "channels\_first" or 4D tensor with shape: (batch, new\_rows, new\_cols, filters) if data\_format is "channels\_last". rows and cols values might have changed due to padding.

[https://www.researchgate.net/post/How\\_will\\_channels\\_RGB\\_effect\\_convolutional\\_neural\\_network](https://www.researchgate.net/post/How_will_channels_RGB_effect_convolutional_neural_network)

# Filter comes from “Image Processing”

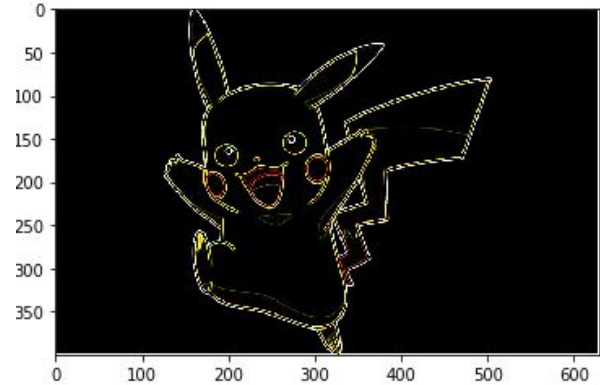


Image

```
print(kernel)
```

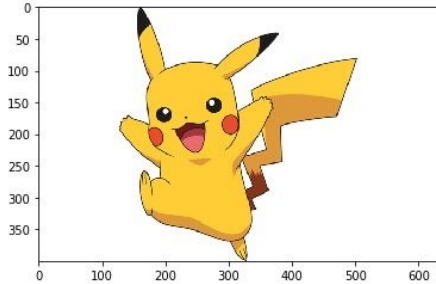
```
[[ -1  -1  -1]
 [ -1   8  -1]
 [ -1  -1  -1]]
```

Edge  
Detection



Convolved  
Features

# Filter comes from “Image Processing”

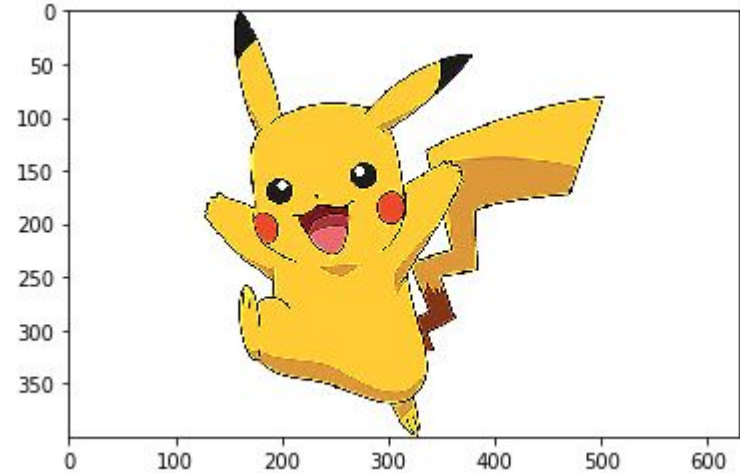


**Image**

```
print(kernel)
```

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

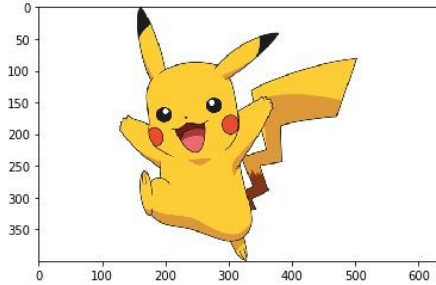
**Sharpen**



**Convolved  
Features**



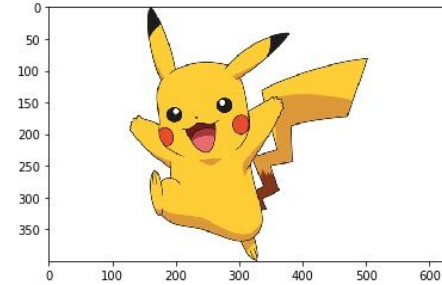
# Filter comes from “Image Processing”



**Image**



**Identity**



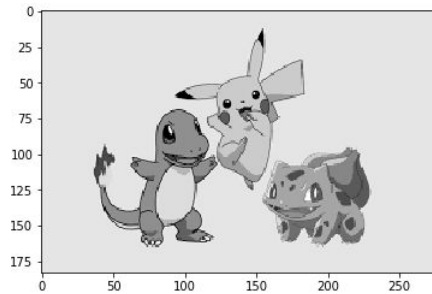
**Convolved  
Features**

# Where are these filters from?

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

# Non-linear Activation

- Filter operation is dot product (linear computation).
- In deep learning, we need to have non-linear transformations.
- Add non-linear activation



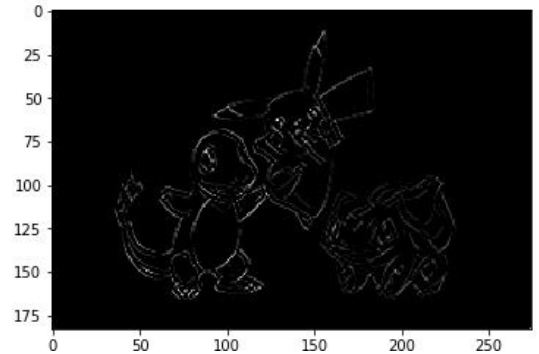
Image

```
print(kernel)
```

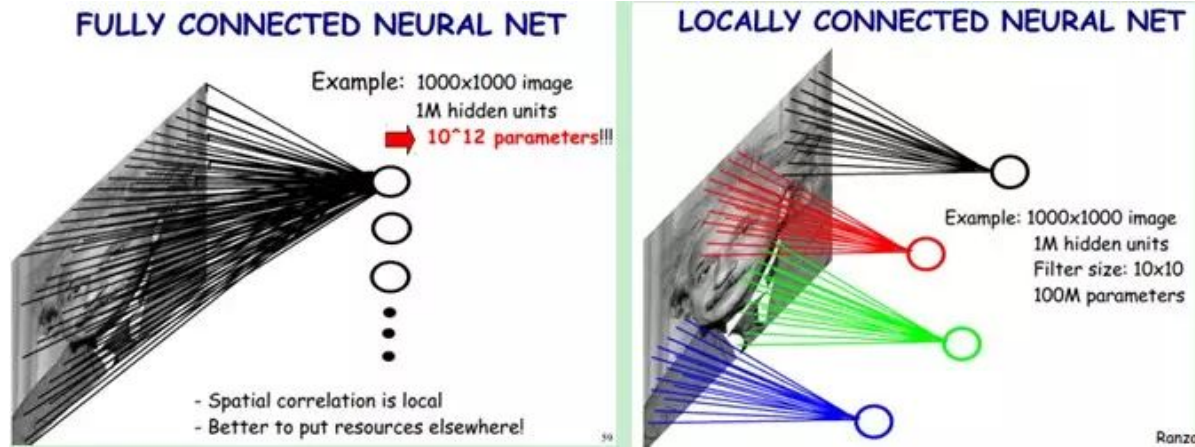
```
[[ 1  0 -1]  
 [ 0  0  0]  
 [-1  0  1]]
```



non-linear



# Locally Connected

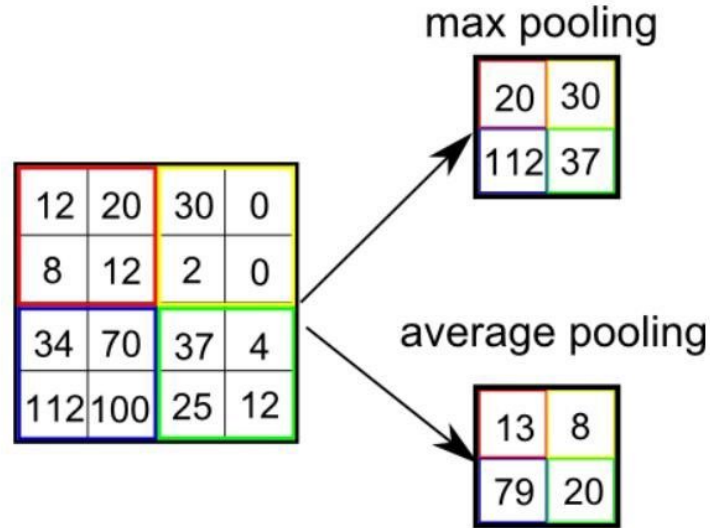


<https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/>

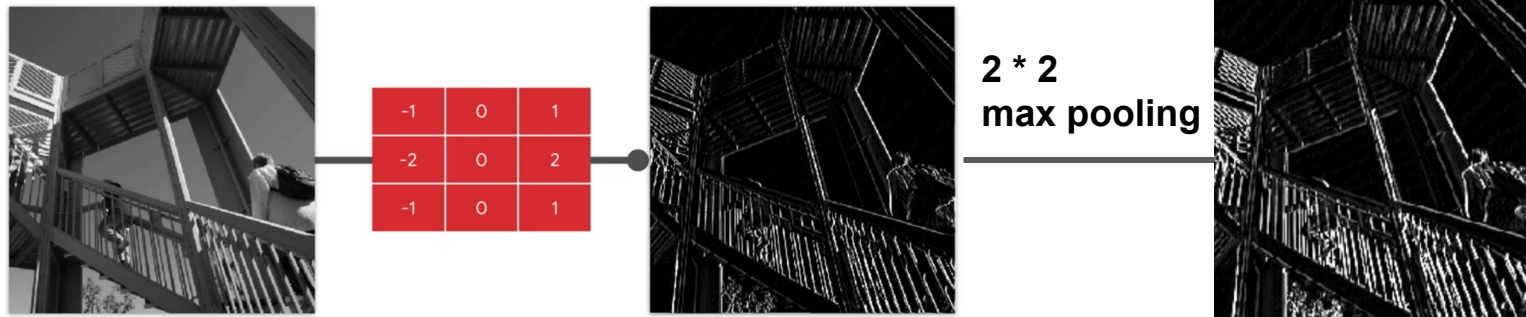
# Pooling Operation

- Pooling Size: the box size. Here is  $2 * 2$
- Stride Size: how much pixel the window move
- Reduce the dimensionality

What is stride size here ?

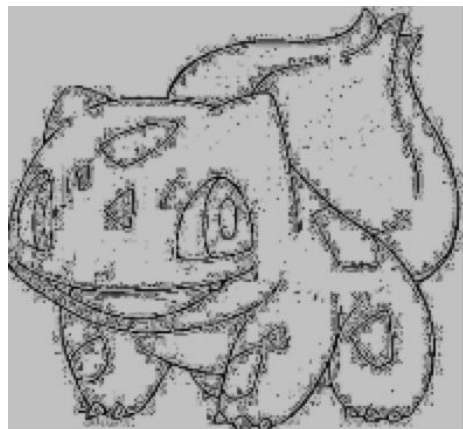


# Filter then Pool



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

# Conv-Pool



Conv-  
Pool

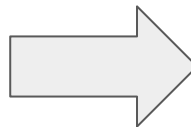
$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}_{n \times m}$$

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

○  
○  
○

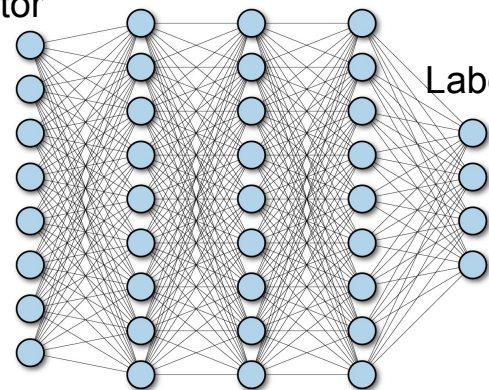
$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}_{n \times m}$$

Flatten



Concatenate

vector

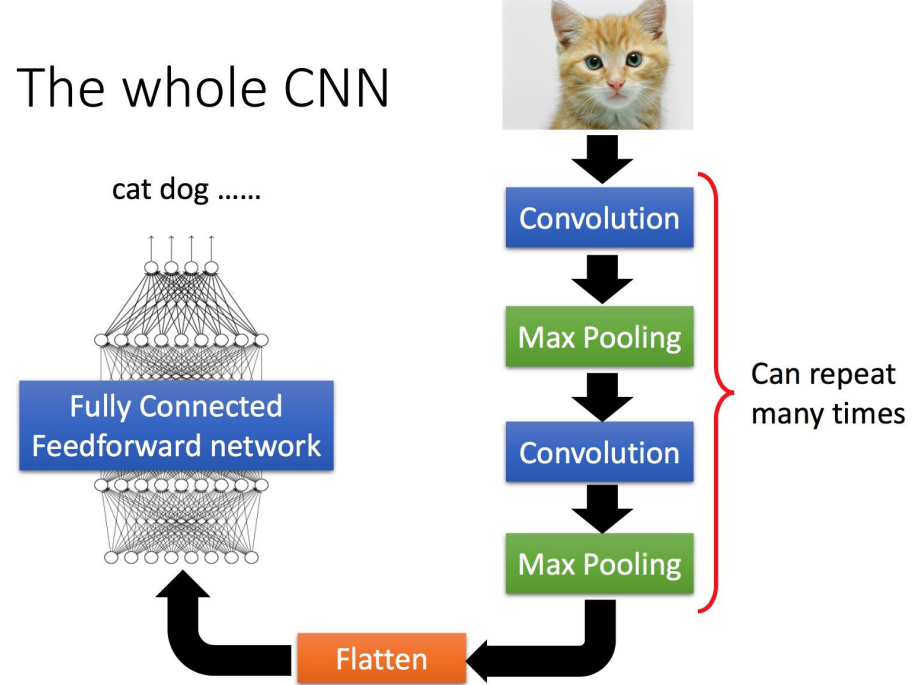


Labels

# 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

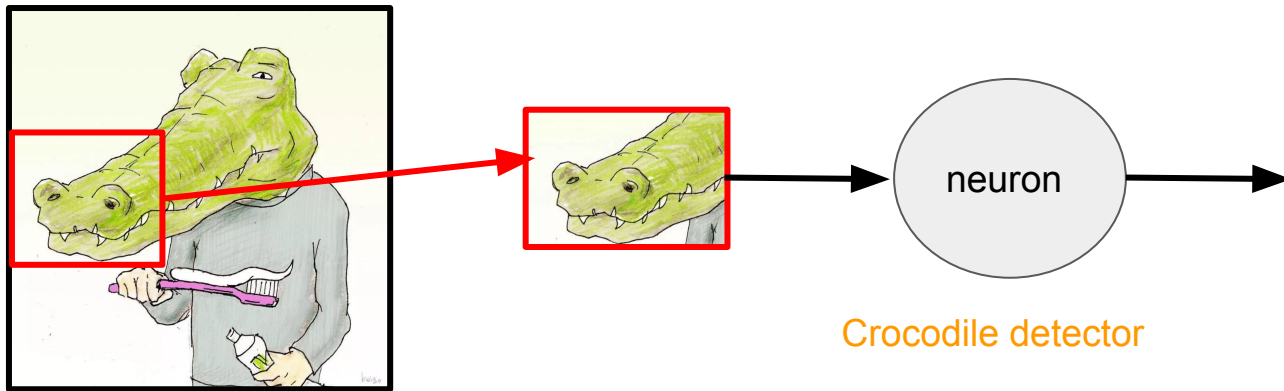




# Why CNN is Suitable for Images

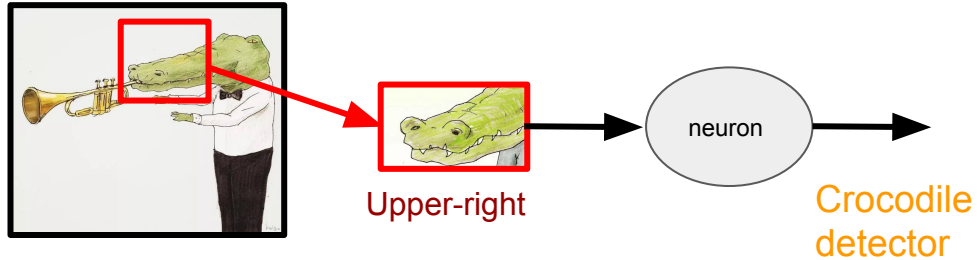
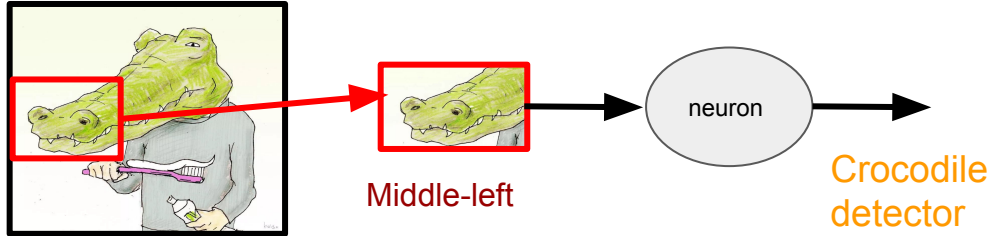
# Local Features Matter

- 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



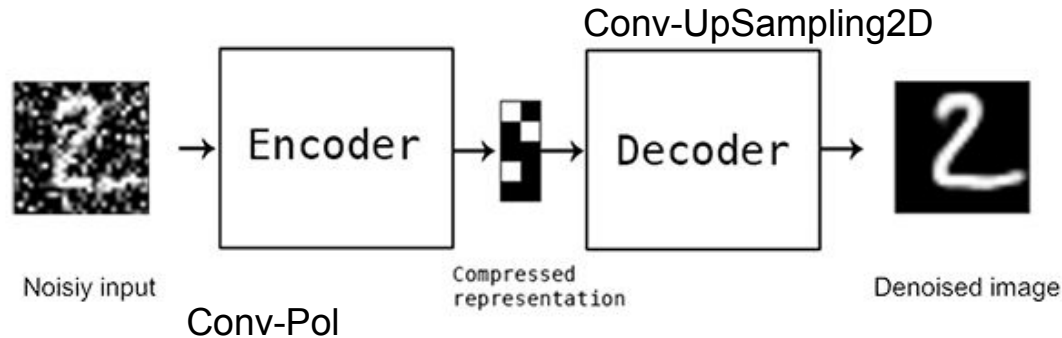
**subsampling**

Crocodile



# Applications

- Image Recognition
- Object Detection
- Image Denoising



<https://blog.keras.io/building-autoencoders-in-keras.html>

<https://www.kaggle.com/michalbrezk/denoise-images-using-autoencoders-tf-keras>

**Can CNN classify digimon  
and pokemon?**

# Case Study



<https://medium.com/@DataStevenson/teaching-a-computer-to-classify-anime-8c77bc89b881>

# Task Definition

Training Data



Digimon



Pokemon

Testing Data



Digimon or Pokemon?



# Build CNN Model

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense

model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(150, 150, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten()) # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid', name='preds'))

model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

The implementation and dataset could be found on LuminUS.

```
Epoch 1/3
8/8 [=====] - 12s 2s/step - loss: 2.7443 - accuracy: 0.7675 - val_
loss: 0.0834 - val_accuracy: 0.9922
Epoch 2/3
8/8 [=====] - 12s 2s/step - loss: 0.0560 - accuracy: 0.9835 - val_
loss: 0.0692 - val_accuracy: 0.9961
Epoch 3/3
8/8 [=====] - 12s 1s/step - loss: 0.0559 - accuracy: 0.9856 - val_
loss: 0.0684 - val_accuracy: 0.9961
```

Only after three epochs, the testing/val accuracy was easily over 99%. **Amazing!**

# Limitations 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



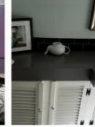
Chairs by viewpoint



Teapots



T-shirts



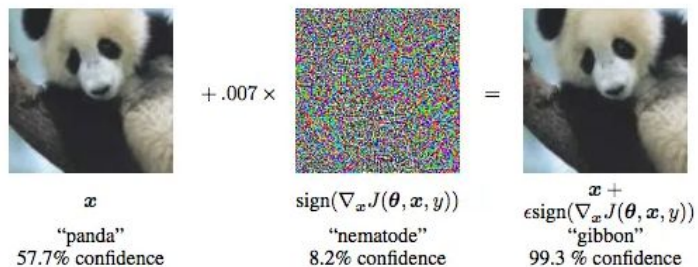
From: objectnet.dev

# 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 is different human vision



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



Granny Smith	85.6%
iPod	0.4%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.1%



Granny Smith	0.1%
iPod	99.7%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.0%

<https://www.theverge.com/2021/3/8/22319173/openai-machine-vision-adversarial-typographic-attack-clip-multimodal-neuron>

# CNN for Structured Data

# Default of Credit Card Clients Dataset

- Static Features
- Dynamic Features

**Task:** Predict the probability of credit default based on credit card owner's payment status, balance and payment history (for the past 6 months from the predicted period)

## Content

There are 25 variables:

- ID: ID of each client
- LIMIT\_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit)
- SEX: Gender (1=male, 2=female)
- EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
- MARRIAGE: Marital status (1=married, 2=single, 3=others)
- AGE: Age in years
- PAY\_0: Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, ..., 8=payment delay for eight months, 9=payment delay for nine months and above)
- PAY\_2: Repayment status in August, 2005 (scale same as above)
- PAY\_3: Repayment status in July, 2005 (scale same as above)
- PAY\_4: Repayment status in June, 2005 (scale same as above)
- PAY\_5: Repayment status in May, 2005 (scale same as above)
- PAY\_6: Repayment status in April, 2005 (scale same as above)
- BILL\_AMT1: Amount of bill statement in September, 2005 (NT dollar)
- BILL\_AMT2: Amount of bill statement in August, 2005 (NT dollar)
- BILL\_AMT3: Amount of bill statement in July, 2005 (NT dollar)
- BILL\_AMT4: Amount of bill statement in June, 2005 (NT dollar)
- BILL\_AMT5: Amount of bill statement in May, 2005 (NT dollar)
- BILL\_AMT6: Amount of bill statement in April, 2005 (NT dollar)
- PAY\_AMT1: Amount of previous payment in September, 2005 (NT dollar)
- PAY\_AMT2: Amount of previous payment in August, 2005 (NT dollar)
- PAY\_AMT3: Amount of previous payment in July, 2005 (NT dollar)
- PAY\_AMT4: Amount of previous payment in June, 2005 (NT dollar)
- PAY\_AMT5: Amount of previous payment in May, 2005 (NT dollar)
- PAY\_AMT6: Amount of previous payment in April, 2005 (NT dollar)
- default.payment.next.month: Default payment (1=yes, 0=no)

<https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset>

# Feature Engineering

- Extract as much information as possible from the available datasets, especially dynamic features.
- Given the past 6 months bill payments (a sequence of 6 numbers):
  - The averaged bill payment
  - The difference between two consecutive payments
  - .....

## Content

There are 25 variables:

- ID: ID of each client
- LIMIT\_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit)
- SEX: Gender (1=male, 2=female)
- EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
- MARRIAGE: Marital status (1=married, 2=single, 3=others)
- AGE: Age in years
- PAY\_0: Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, ... 8=payment delay for eight months, 9=payment delay for nine months and above)
- PAY\_2: Repayment status in August, 2005 (scale same as above)
- PAY\_3: Repayment status in July, 2005 (scale same as above)
- PAY\_4: Repayment status in June, 2005 (scale same as above)
- PAY\_5: Repayment status in May, 2005 (scale same as above)
- PAY\_6: Repayment status in April, 2005 (scale same as above)
- BILL\_AMT1: Amount of bill statement in September, 2005 (NT dollar)
- BILL\_AMT2: Amount of bill statement in August, 2005 (NT dollar)
- BILL\_AMT3: Amount of bill statement in July, 2005 (NT dollar)
- BILL\_AMT4: Amount of bill statement in June, 2005 (NT dollar)
- BILL\_AMT5: Amount of bill statement in May, 2005 (NT dollar)
- BILL\_AMT6: Amount of bill statement in April, 2005 (NT dollar)
- PAY\_AMT1: Amount of previous payment in September, 2005 (NT dollar)
- PAY\_AMT2: Amount of previous payment in August, 2005 (NT dollar)
- PAY\_AMT3: Amount of previous payment in July, 2005 (NT dollar)
- PAY\_AMT4: Amount of previous payment in June, 2005 (NT dollar)
- PAY\_AMT5: Amount of previous payment in May, 2005 (NT dollar)
- PAY\_AMT6: Amount of previous payment in April, 2005 (NT dollar)
- default.payment.next.month: Default payment (1=yes, 0=no)

**Design of those hand-crafted features is challenging, time-consuming, requires domain knowledge.**



# Representation of data in CNN format

Shape: 1 by 18

PAY\_0  
PAY\_2  
PAY\_3  
PAY\_4  
PAY\_5  
PAY\_6  
BILL\_AMT1  
BILL\_AMT2  
BILL\_AMT3  
BILL\_AMT4  
BILL\_AMT5  
BILL\_AMT6  
PAY\_AMT1  
PAY\_AMT2  
PAY\_AMT3  
PAY\_AMT4  
PAY\_AMT5  
PAY\_AMT6

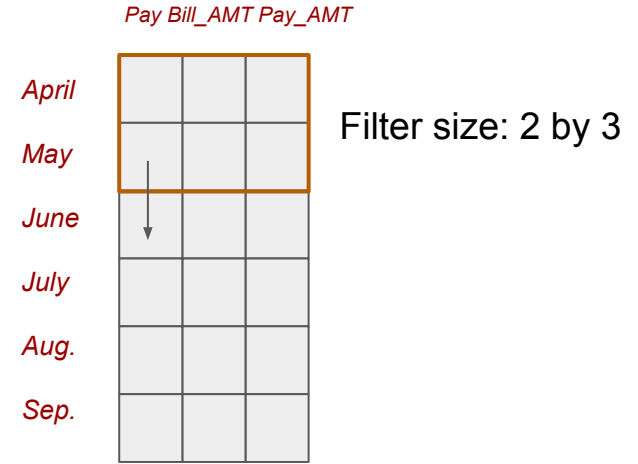
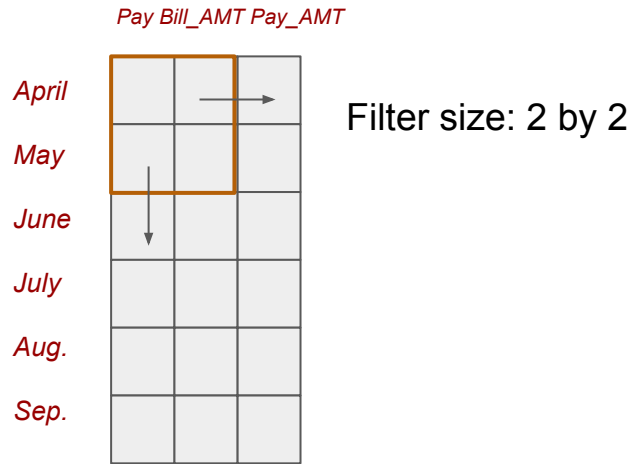


Shape: 1 by 6 by 3

	Pay	Bill_AMT	Pay_AMT
April			
May			
June			
July			
Aug.			
Sep.			

CNN can be easily applied to extract local patterns

# Convolution Operation



**Which structure is better?**

# Multiple Channels

- In computer vision, CNN is applied on R-G-B channels
- In this application, different types of credit cards or mortgage of a certain customer can be regarded as different channels



For each customer,  
the data shape: 1 by 6 by 3 by **3**

# Incorporating Static Features

- Multi-input deep learning is able to combine static and dynamic features for prediction.
- This architecture connects parts of the inputs directly to the output layer.

