Applied Machine Learning for Business Analytics

Lecture 6: Convolutional Neural Network

Lecturer: Zhao Rui

Logistics

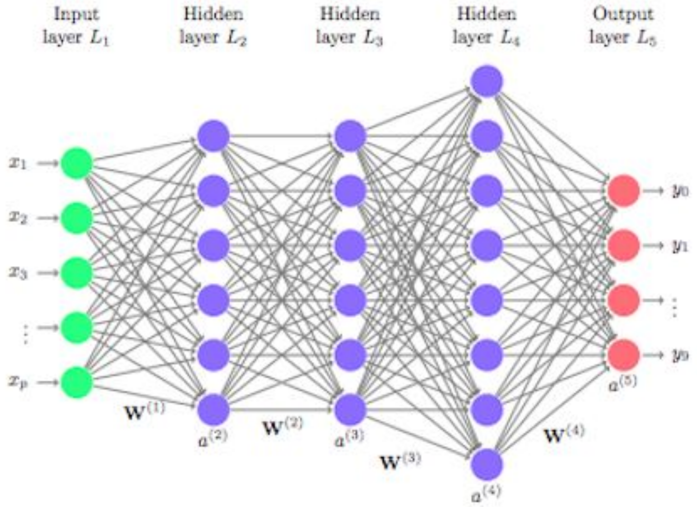
- Appreciate if you keeps video on!

Agenda

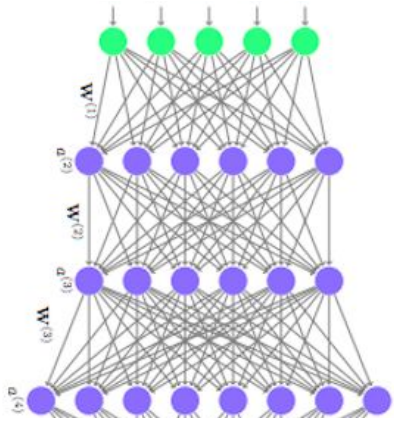
1. Representation Learning in FCNN
2. Introduction to CNN
3. Why CNN for images?
4. Limitations of CNN
5. CNN for Time-series Data

1. Representation Learning in FCNN

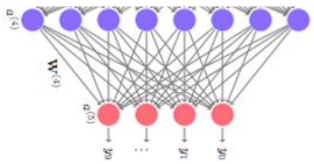
Hidden Representation in Deep Learning



Low-dim, Original Space



High-dim, **Linearly Separated** Space

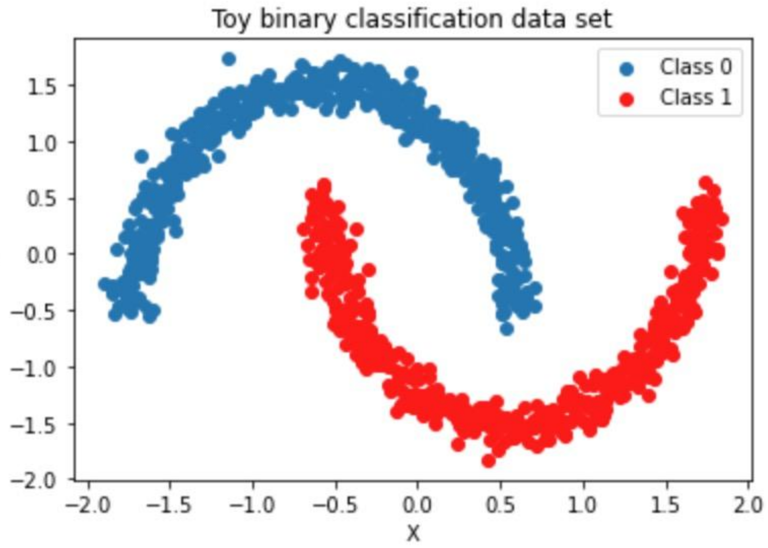


Softmax Classifier
(Linear Model)

Representation Learning

https://colab.research.google.com/drive/19p0wuleQtMTo5_FR3LDVwpCqaeoXhJOR?usp=sharing

Moons Dataset

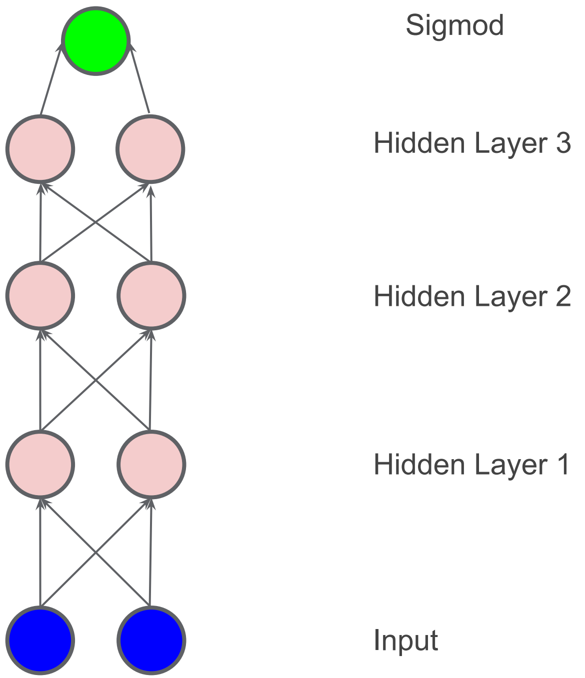



```
# fit a logistic regression model to classify this data set as a benchmark
simple_model = LogisticRegression()
simple_model.fit(X_train, Y_train)
print('Train accuracy:', simple_model.score(X_train, Y_train))
print('Test accuracy:', simple_model.score(X_test, Y_test))
```

```
Train accuracy: 0.89
Test accuracy: 0.88
```

Fully-Connected Neural Network

```
# fix a width that is suited for visualizing the output of hidden layers
H = 2
input_dim = X.shape[1]

# create sequential multi-layer perceptron
model = Sequential()

# Then, use add() to insert layers into the container
model.add(Input(shape=(input_dim,)))
model.add(Dense(H,activation='tanh'))
model.add(Dense(H, activation='tanh'))
model.add(Dense(H, activation='tanh'))
#binary classification, one output
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              metrics=['acc'])
```



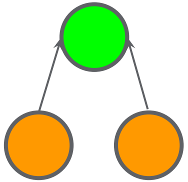
Fully-Connected Neural Network

```python
# evaluate the training and testing performance of your model
# note: you should extract check both the loss function and your evaluation metric
score = model.evaluate(X_train, Y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Train loss: 0.0007340409210883081
Train accuracy: 1.0
```

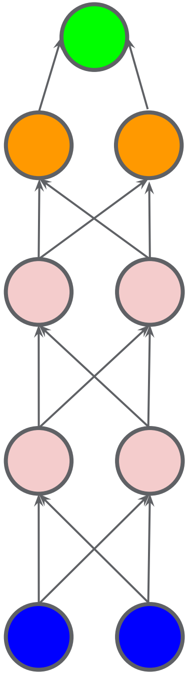
```python
score = model.evaluate(X_test, Y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.0008793871384114027
Test accuracy: 1.0
```



1. In forward computation, the output of hidden layer 3 is feed into "logistic regression" to predict labels.
2. Since the train and test accuracy are both 1, it means the hidden layer 3' output are linearly separated.

Let us visualize those outputs!



Sigmod

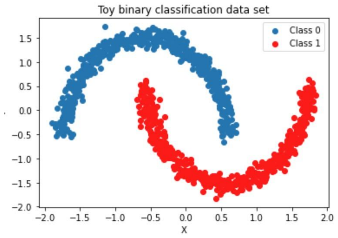
Hidden Layer 3

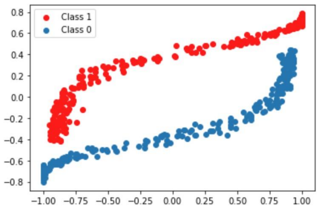
Hidden Layer 2

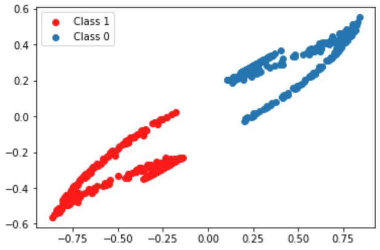
Hidden Layer 1

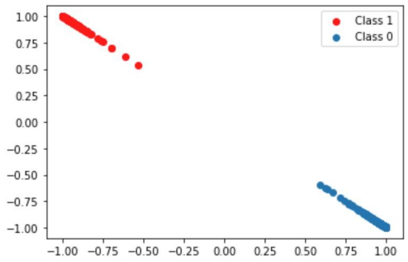
Input

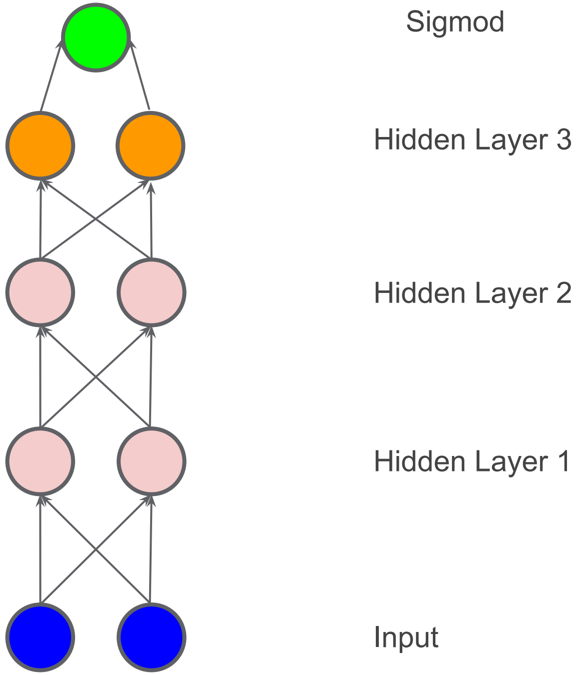
Fully-Connected Neural Network











Representation Learning in Neural Networks

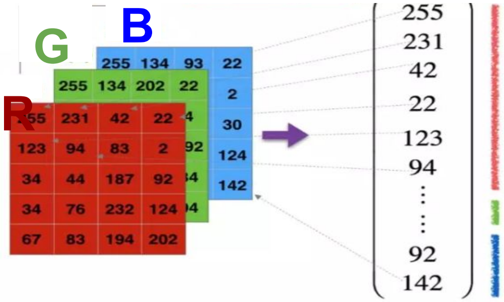
- Outputs of each hidden layer of an neural network is a non-linear transformation of the input data into a feature space. Each hidden layer should transform the input so that it is more linearly separable
- we are more interested in learning the latent representation of the data rather than perfecting our performance in a single task (such as classification).
  - We do not need to preprocess the data to add non-linear features. The neural network will learn the most suitable non-linear transformations to the input (to achieve the best classification)

2. Introduction to CNN

Image: a matrix of pixel values

- Every image can be represented as a matrix of pixel values
- The pixel value ranges from 0 to 255.
- Channel is referred to a certain component of an image
    - An image from your iphone will have three channels
    - A grayscale image has just one channel

Computers See Image

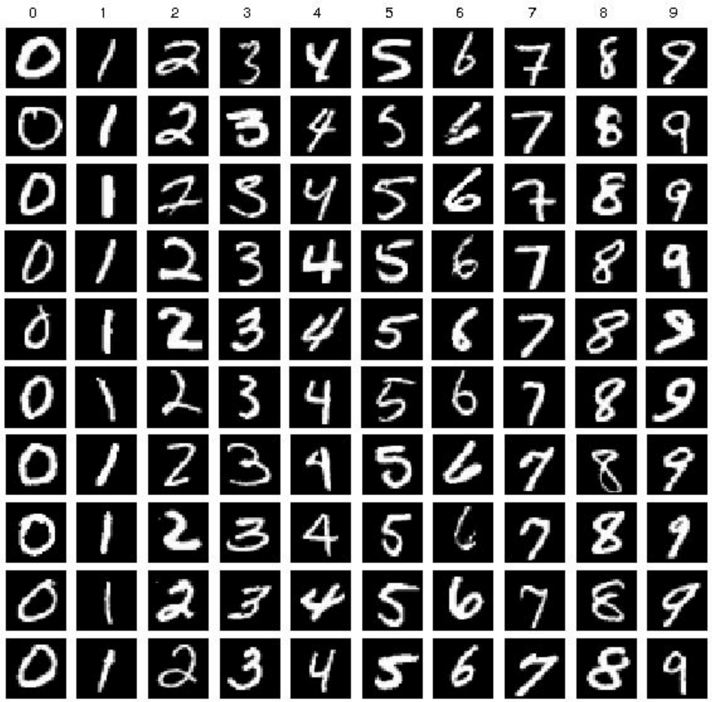


Softmax Classifier
SVM
….

Fully-Connected
Neural Network

Labels

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.

What if the digits in top-left corner

Training Data

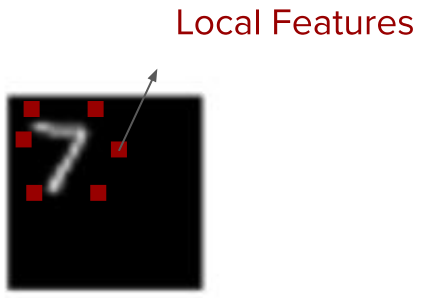
Testing Data

Limitations of Fully-connected Neural Networks

- For the grayscale image is 64 pixel by 64 pixel
- Image is represented by 64 * 64 * 1 = 4096 values
- FCNN's input size is 4096
- If the first hidden layer size is 500,
  - Number of weights in the first hidden layer is 4096*500 = 2,048,000
- The model size will explode further
  - Deep structures (many layers)
  - Color images (the input size will be 3 times)
- The concern for a huge model size:
  - Risk of Overfitting
  - Make training/deployment more time/resource consuming
  - Make learning more untraceable as dimension of search space is increased.

Limitations of Fully-connected Neural Networks

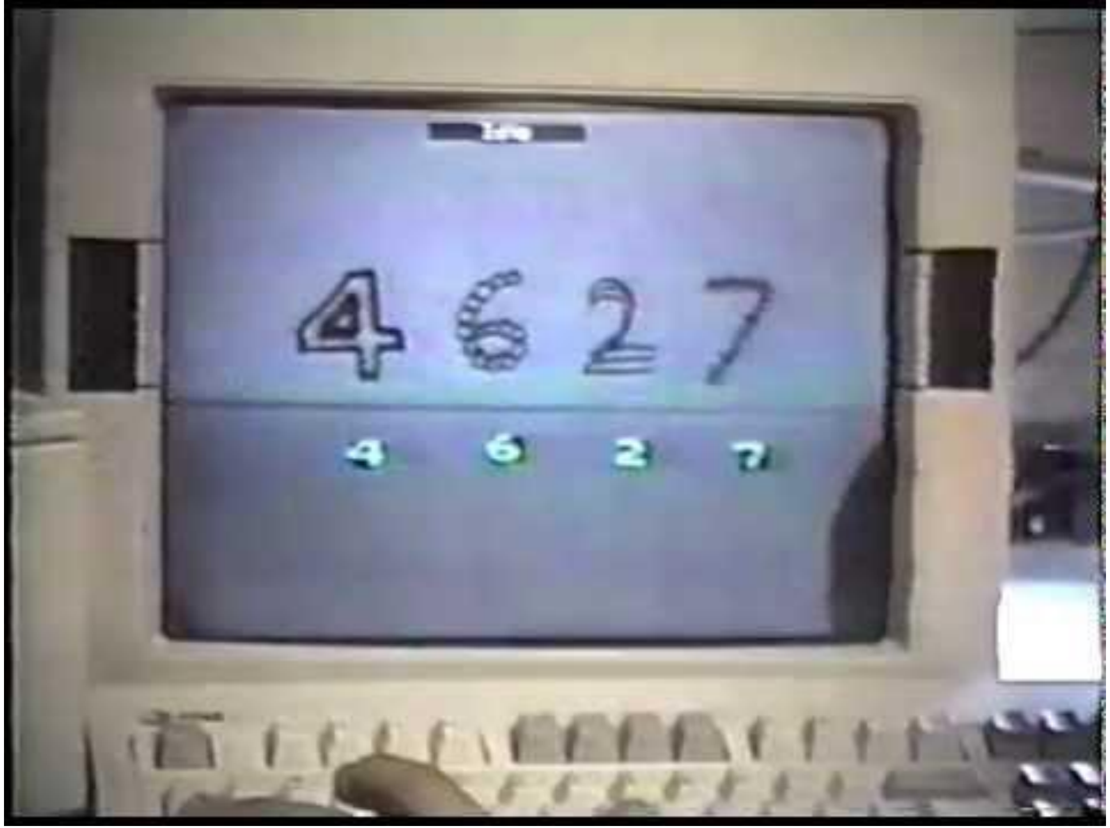
- FNN can not scale easily to computer vision (Input Size is so big-> too many weights)
- Any spatial relationship is not captured
    - 2D image is flattened to be a 1D vector.
- Global Pattern vs Local Pattern
    - In FNN, each pixel in the image is connected to the hidden neuron
    - The hidden neuron tries to learn the "global feature"



Cat vs Dog

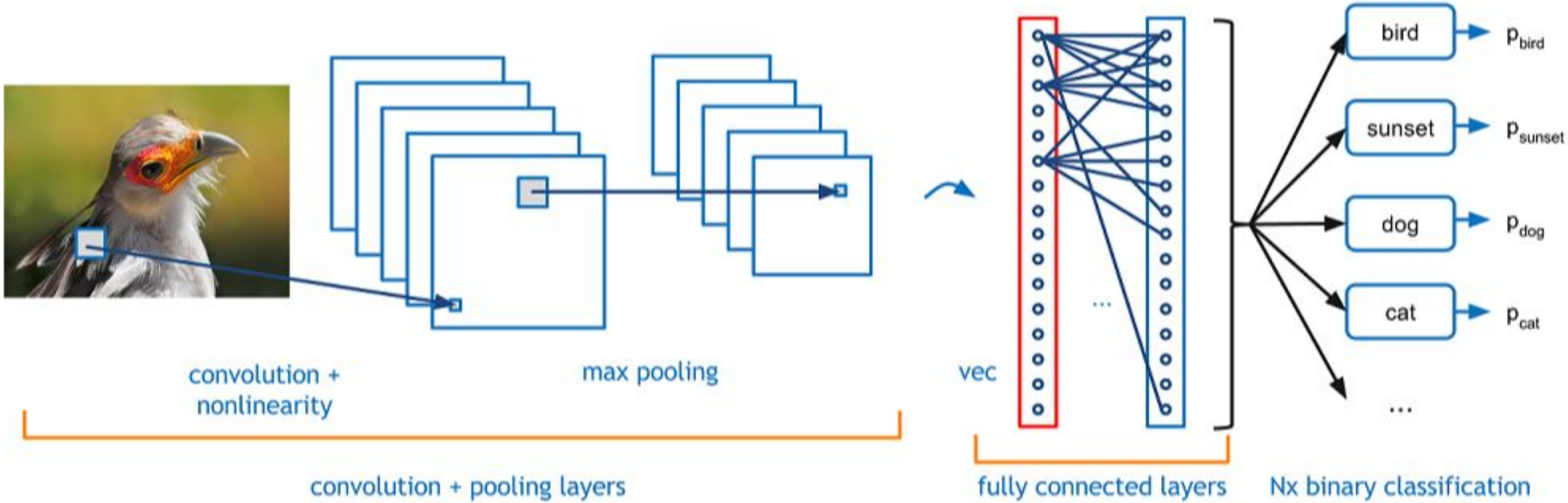
- To recognize those images, we captures the patterns
- For Cat vs Dog problems, patterns can be
  - Shapes of ears, eyes
  - Colors
  - Hairs
- Machine learning model should be trained to capture those **patterns**

In the context of CNN, patterns are also called as kernel, filters and feature extractor.



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

Convolutional Neural Network

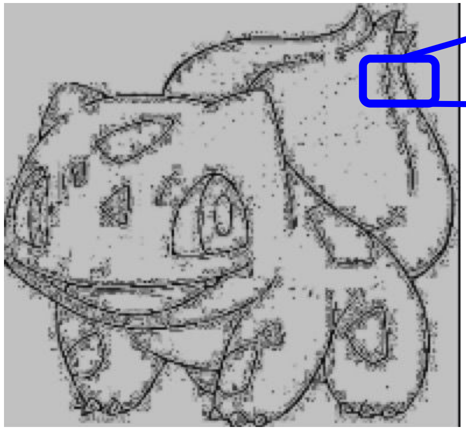


Extracting useful features of data

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

Conv-Operation

Consider neighbor values



1	0	1
0	$\boxed{1}$	0
1	0	1

Current Pixel Value is 1

Dot Product

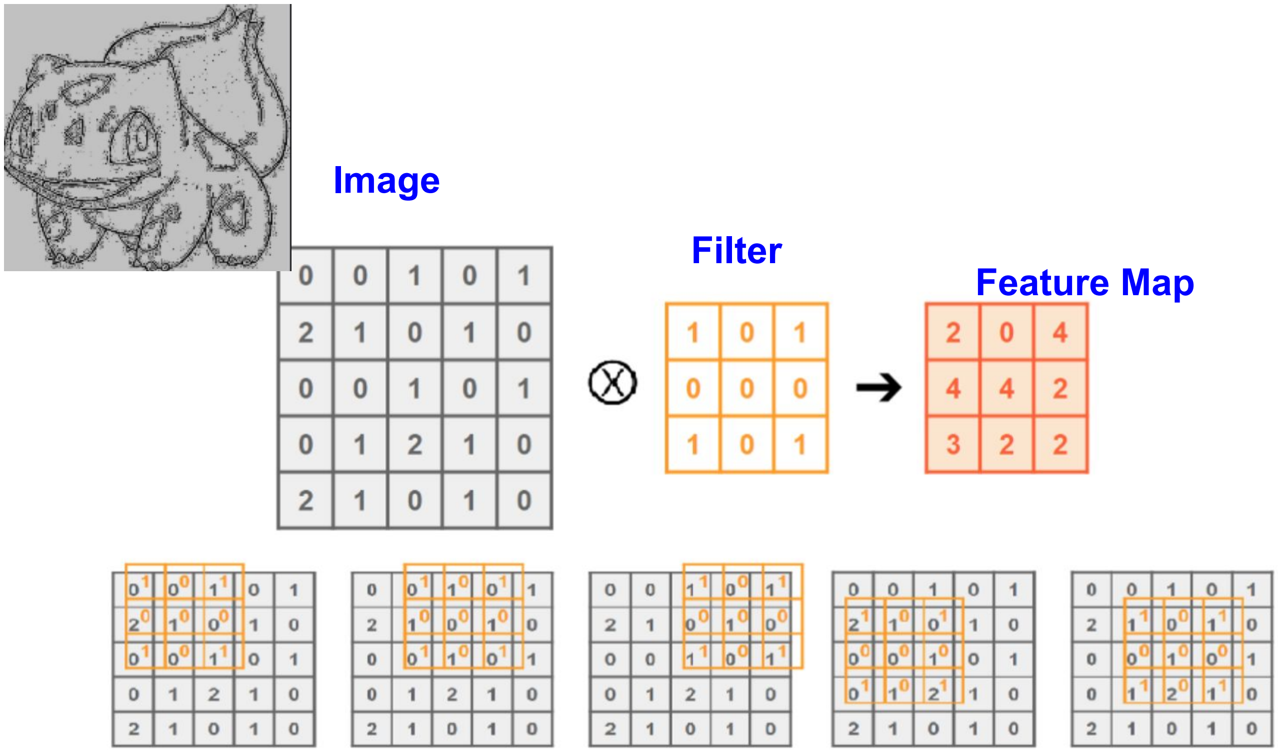
New Pixel Value
4

1	**0**	**1**
0	**0**	**0**
1	**0**	**1**

Filter (3 by 3)

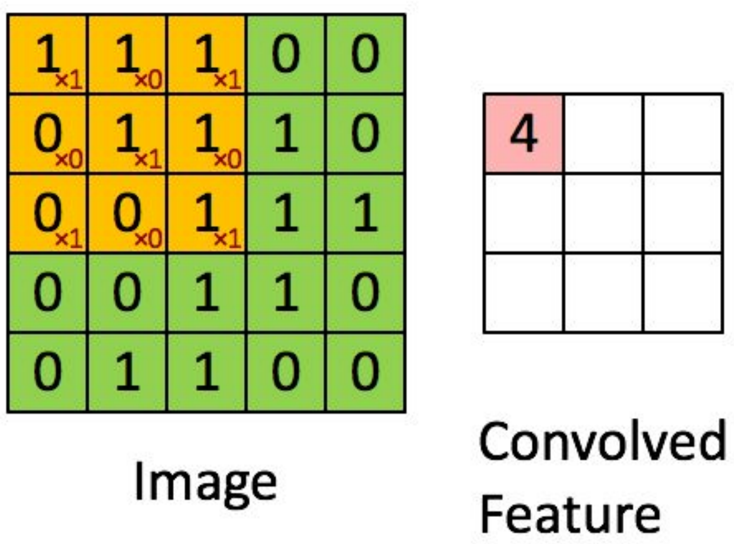
Local Patterns

Conv-Operation



Conv-Operation

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



Feature Map

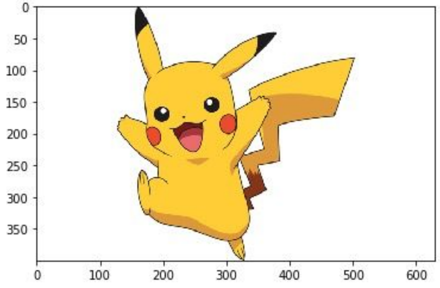
Check gif version here:
https://docs.google.com/presentation/d/1V_7IqLDsKXyaEwR9ZgxmlQ9ixmcT41ZGOLmJtbpgGPM/edit?usp=sharing

Stanford UFLDL

Conv-Operation

- Convolution is a mathematical operation on two objects to product an outcome that expresses how the shape of one is modified by the other
- In the CNN, the feature map has the information about the particular pattern corresponding to the filter

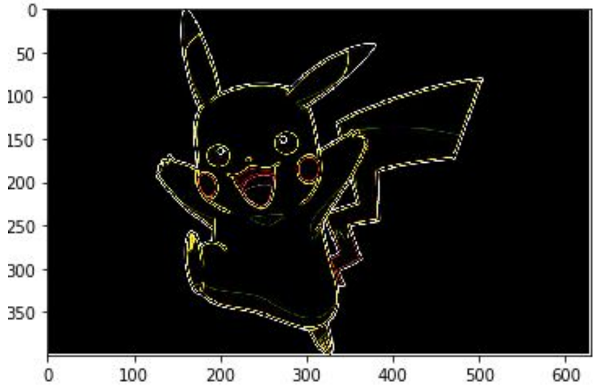
Feature Map



Image

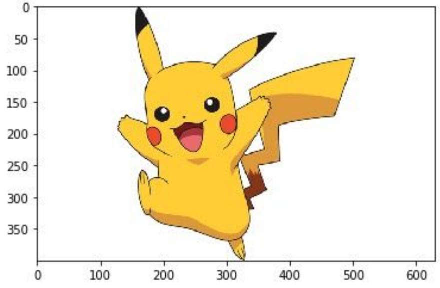
```
print(kernel)

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

Edge Detection



Feature Map

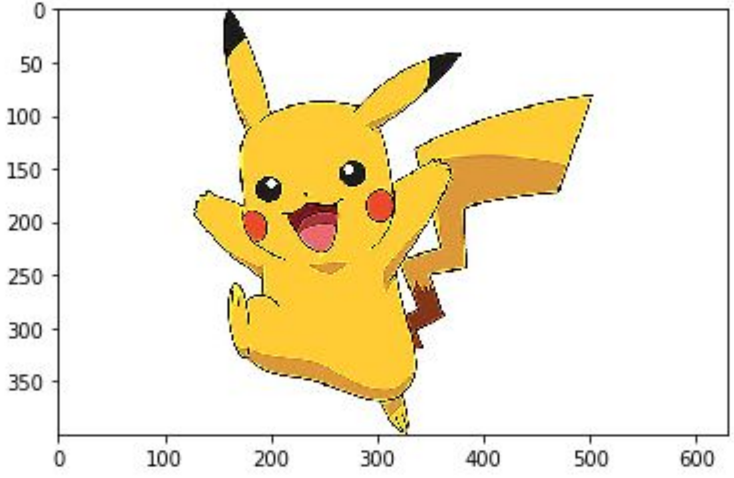
Feature Map



Image

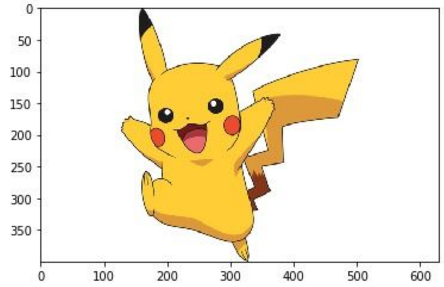
```
print(kernel)

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

Sharpen

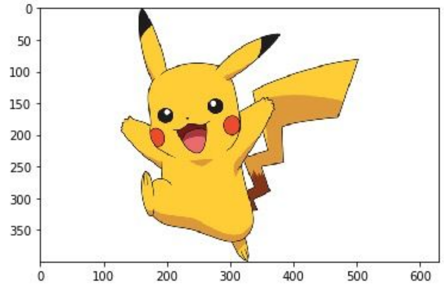


Feature Map

Feature Map



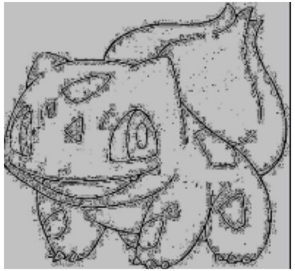
Image



Feature Map

Identity

Conv-Operation

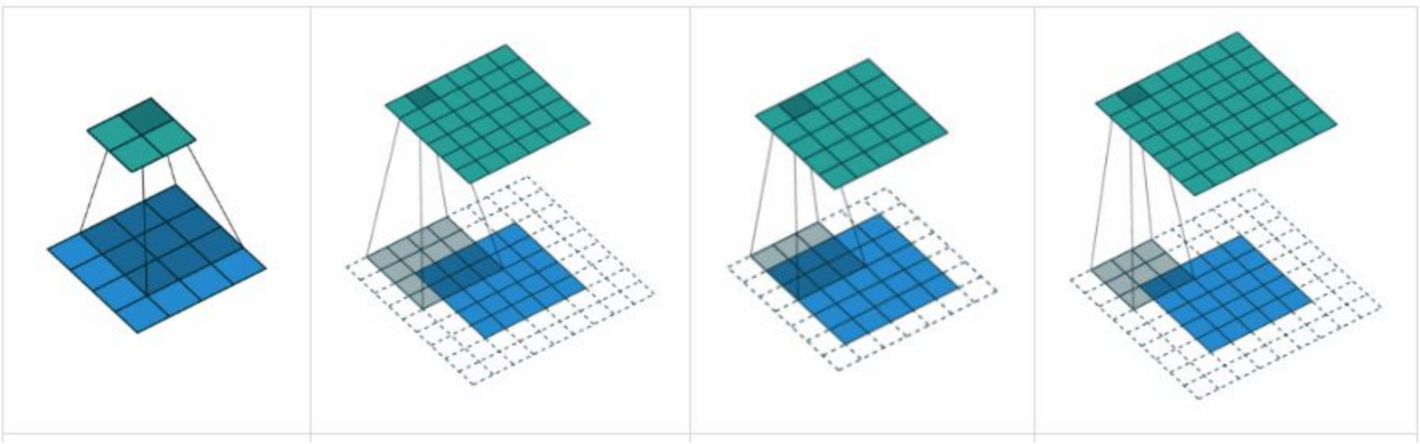


$$
\underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 2 & 1 & 0 & 1 & 0 \end{bmatrix}}_{\text{Image}} \otimes \underbrace{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}}_{\text{Filter}} \rightarrow \underbrace{\begin{bmatrix} 2 & 0 & 4 \\ 4 & 4 & 2 \\ 3 & 2 & 2 \end{bmatrix}}_{\text{Feature Map}}
$$

Those edge pixels are not captured

Padding

- Padding: give additional pixels around the boundary of the image
- Padding size: the number of additional pixels

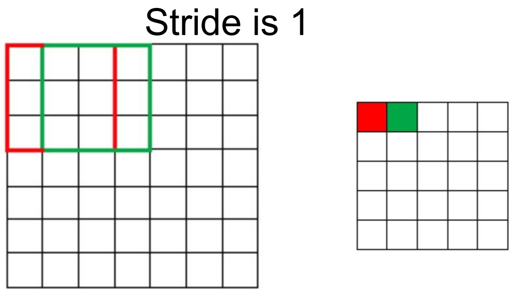


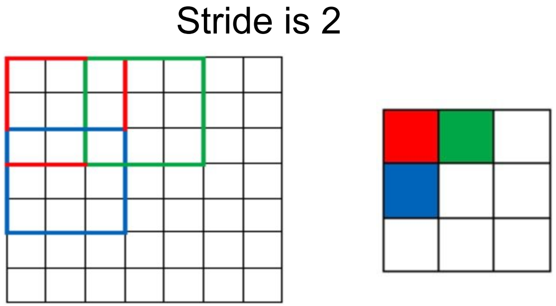
Padding Size: 0
Valid

Padding Size: 1
Same

Stride Size

- Does a filter always have to move one pixel at a time?
- Stride size is the amount by which the filter shifts





Convolutional Operation

- Three conv. Layer basic hyper-parameters:
  - Filter size: K
  - Stride size: S
  - Padding size: P
- Output Size can be decided by

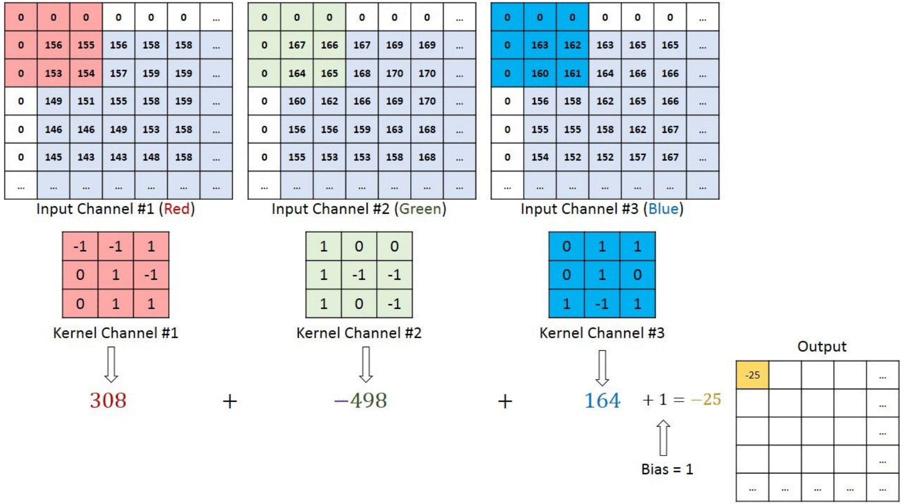
Input size

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

Output size

Multi-Channel CNN

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



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

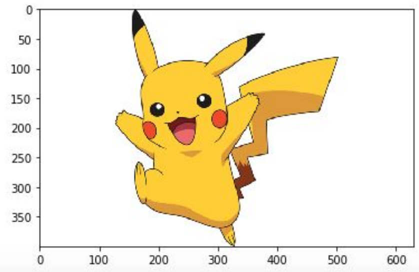
```python
from matplotlib.image import imread
import numpy as np
img = imread('pikka_3.jpg')
```

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

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

```python
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.

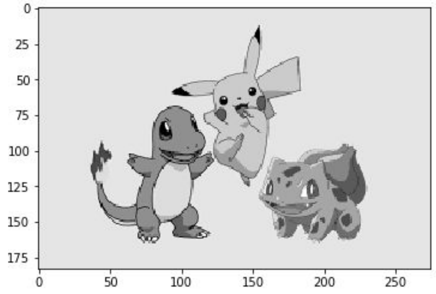
Where are these filters from?

- 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

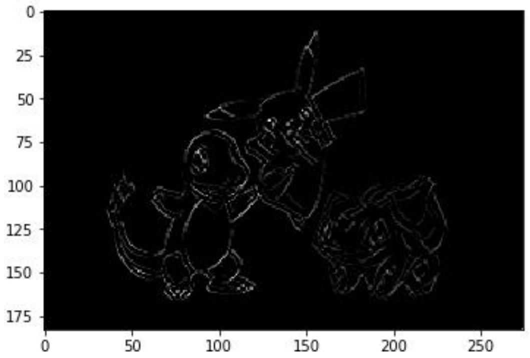
- Filter operation is dot product (linear computation)
- In deep learning, we need to have non-linear transformations
- Add non-linear activation



Image

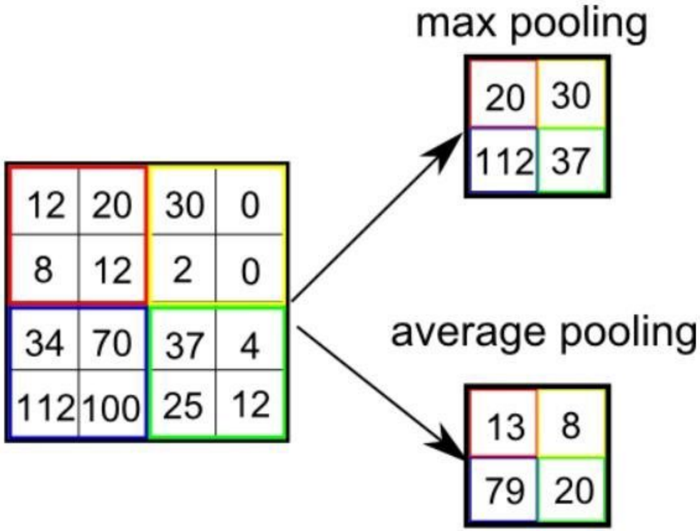
```
print(kernel)
[[ 1  0 -1]
 [ 0  0  0]
 [-1  0  1]]
```

non-linear



Pooling Operation

- Pooling Size: the box size. Here is 2 by 2
- Reduce the dimensionality
- Remove some noise
- Extract significant values



Filter then Pool

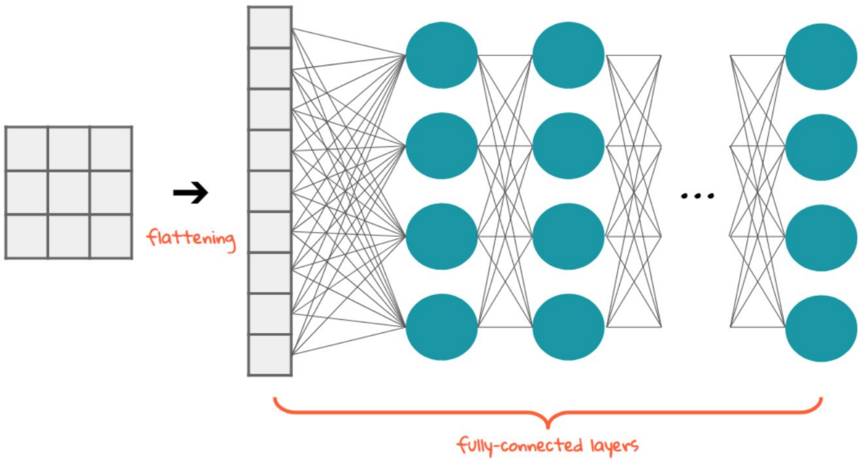
-1	0	1
-2	0	2
-1	0	1

2 * 2
max pooling

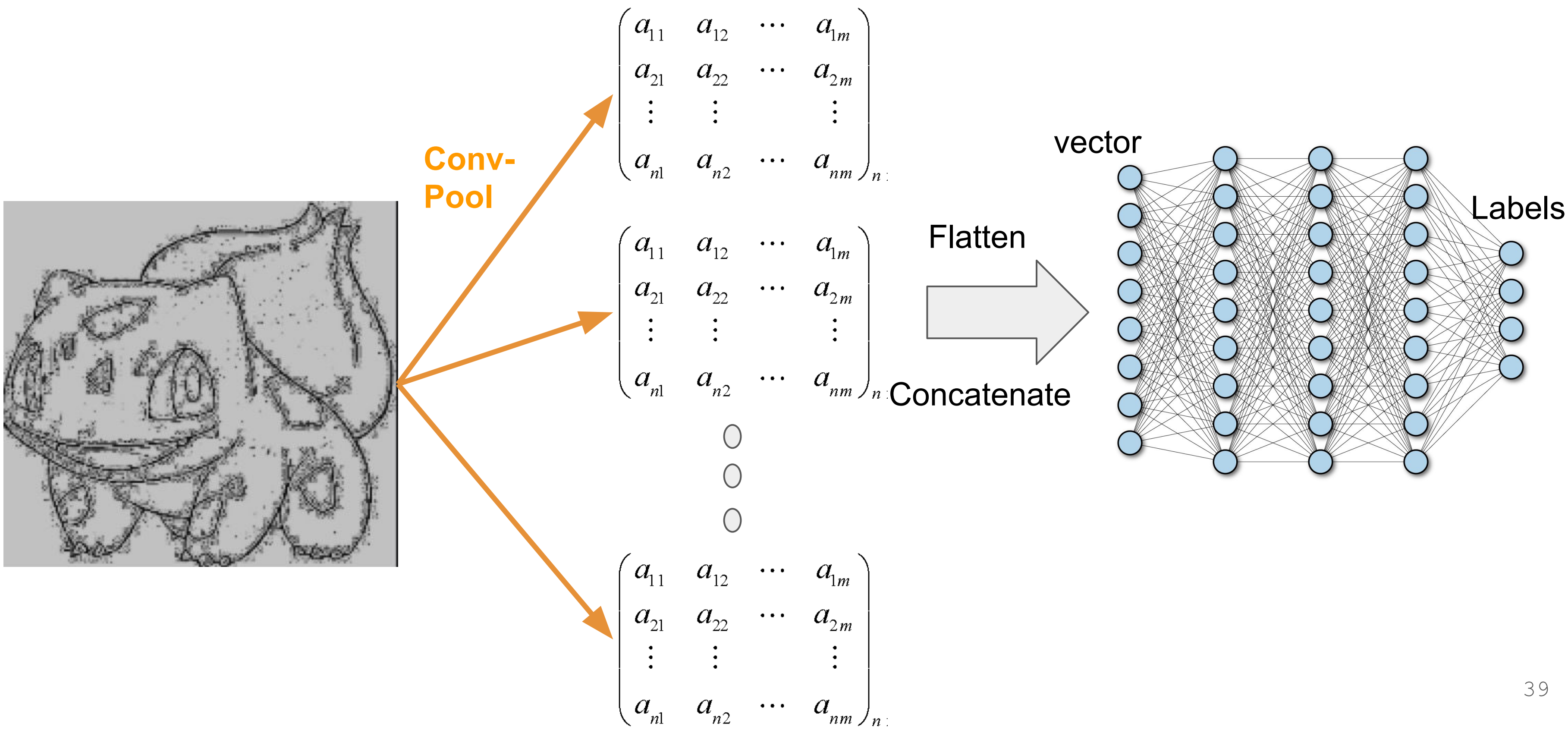
1. The size is **one quarter** the original size
2. The **vertical line** features are **enhanced**.

Flattening

- Flattening is converting the data into one-dimensional array for feeding it to the next layer.

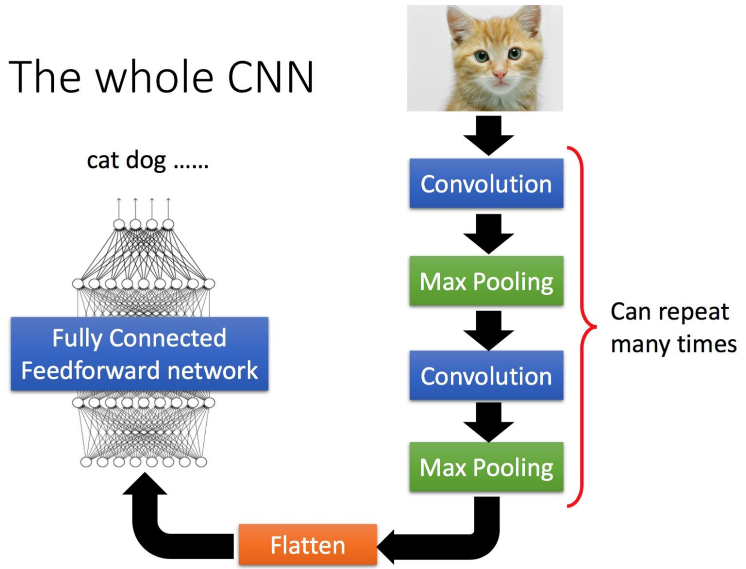


All in one shot



CNN can be Deep

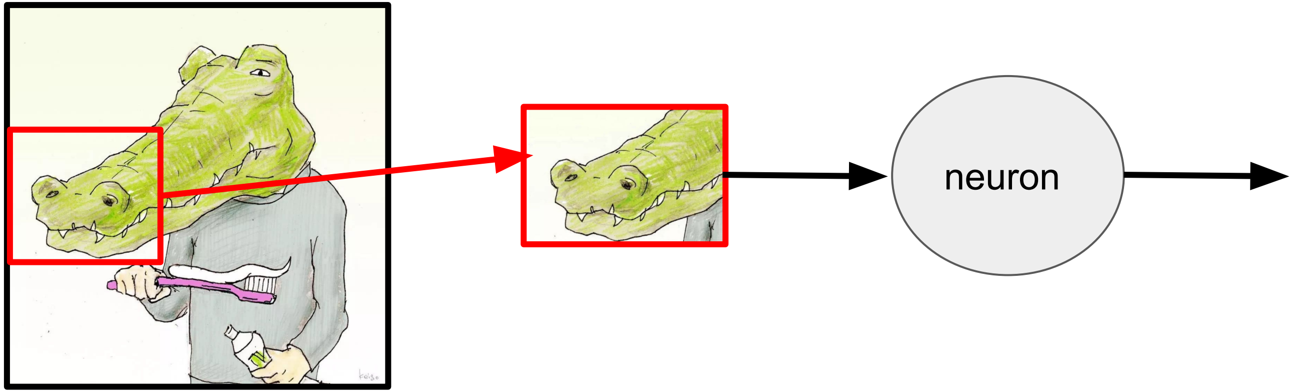
- Conv-Pool can be followed by another Conv-Pool
- At the end, after flatten operation, fully connected layers are used to map the outputs



3. Why CNN for Images

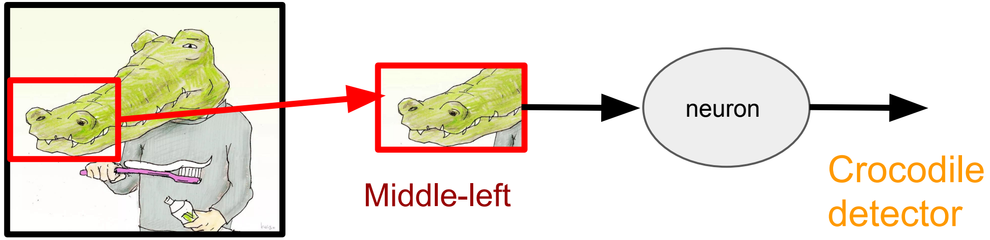
Local Features Matter

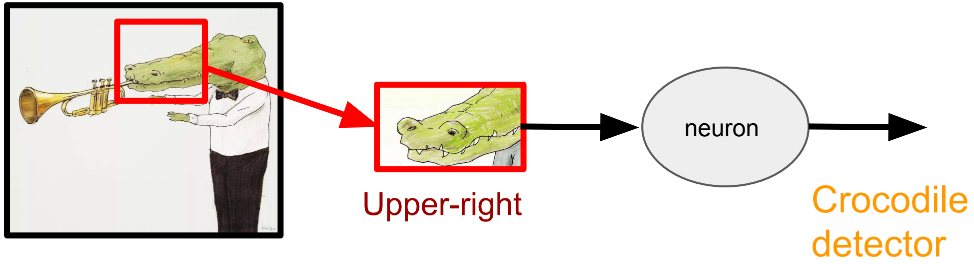
- Discriminative patterns are much smaller than the whole image
- A neuron or feature extractor does not have to see the whole image
- Less parameters required



Location Insensitive

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





Subsampling Works

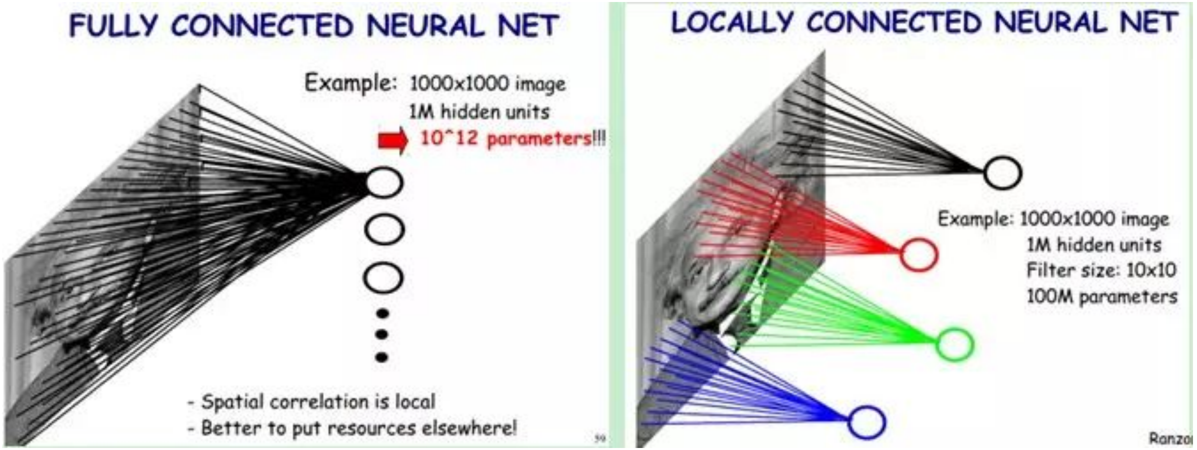
- Subsampling the pixels will not change the object
- We can subsample the pixels to make the images smaller -> less parameters required

Crocodile

subsampling

Crocodile

Locally Connected

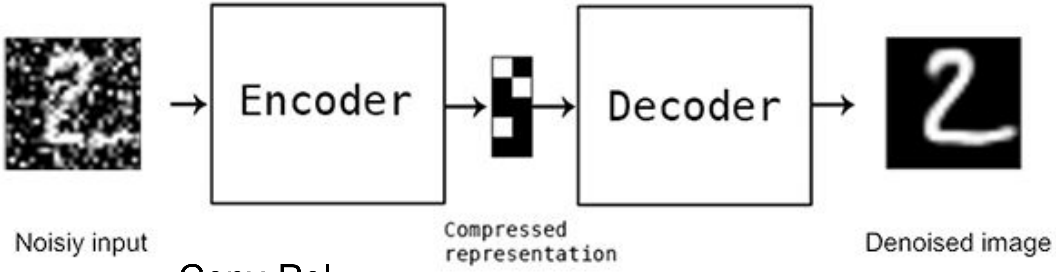


https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

Applications

- Image Recognition
- Object Detection
- Image Denoising

Conv-UpSampling2D



Conv-Pol

https://blog.keras.io/building-autoencoders-in-keras.html
https://www.kaggle.com/michalbrezk/denoise-images-using-autoencoders-tf-keras

Case Study

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

Task Definition

Training Data

Digimon

Pokemon

Testing Data

Digimon or Pokemon?

Task Definition

```
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.

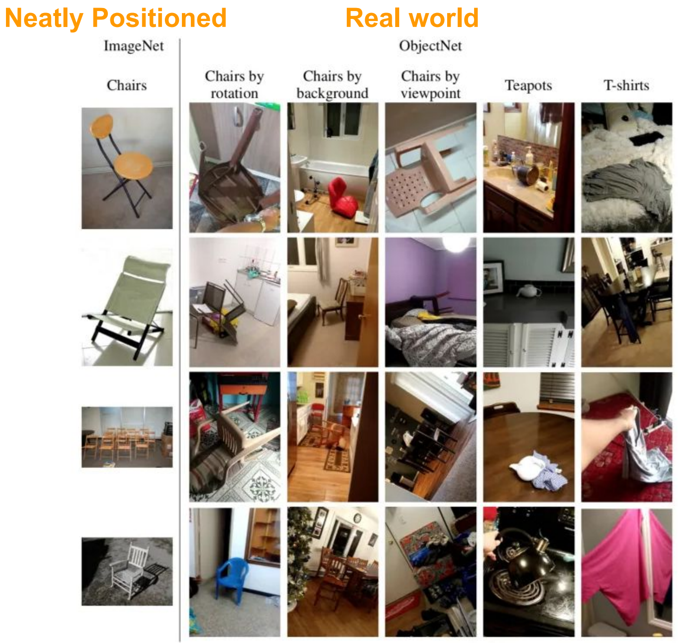
```
Epoch 1/3
8/8 [==============================] - 12s 2s/step - loss: 2.7443 - accuracy: 0.7675 - val_
loss: 0.0834 - val_accuracy: 0.9922
Epoch 2/3
8/8 [==============================] - 12s 2s/step - loss: 0.0560 - accuracy: 0.9835 - val_
loss: 0.0692 - val_accuracy: 0.9961
Epoch 3/3
8/8 [==============================]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

4. Limitations of CNN

CNN vs Human Vision

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

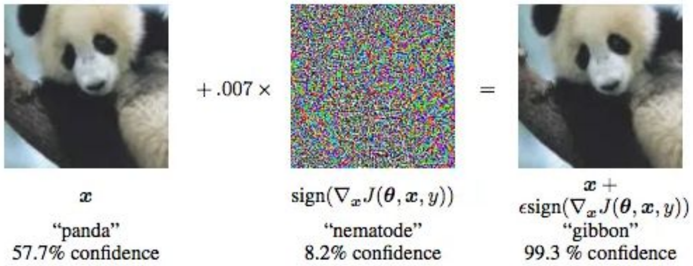


From: objectnet.dev

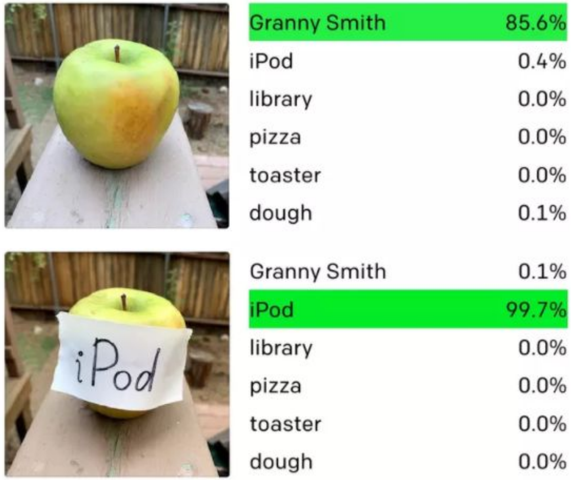
CNN vs Human Vision

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

CNN vs Human Vision



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



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

5. CNN for Time-Series Data

Credit Card Default Datasets

- Static Features
- **Time-series 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

- 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:

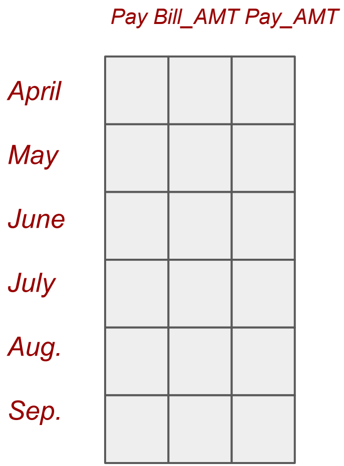
- **ID**: ID of each client
- **LIMIT_BAL**: Amount of given credit in NT dollars (includes individual and family/supplementary credit
- **SEX**: Gender (1=male, 2=female)
- **EDUCATION**: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
- **MARRIAGE**: Marital status (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)

Representation of data in CNN format

Shape: 1 by 18

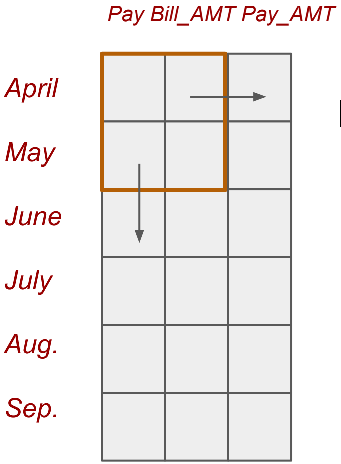
PAY_0
PAY_2
PAY_3
PAY_4
PAY_5
PAY_6
BILL_AMT1
BILL_AMT2
BILL_AMT3
BILL_AMT4
BILL_AMT5
BILL_AMT6
PAY_AMT1
PAY_AMT2
PAY_AMT3
PAY_AMT4
PAY_AMT5
PAY_AMT6

Shape: 1 by 6 by 3

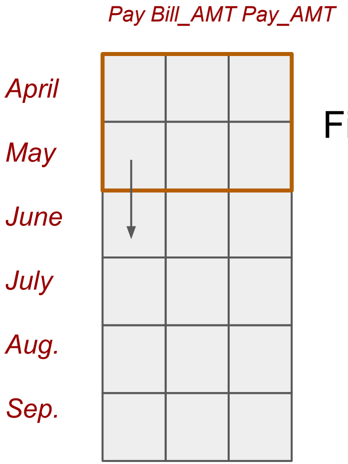


CNN can be easily applied to extract local patterns

Conv-Operation



Filter size: 2 by 2



Filter size: 2 by 3

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.

Channel 1

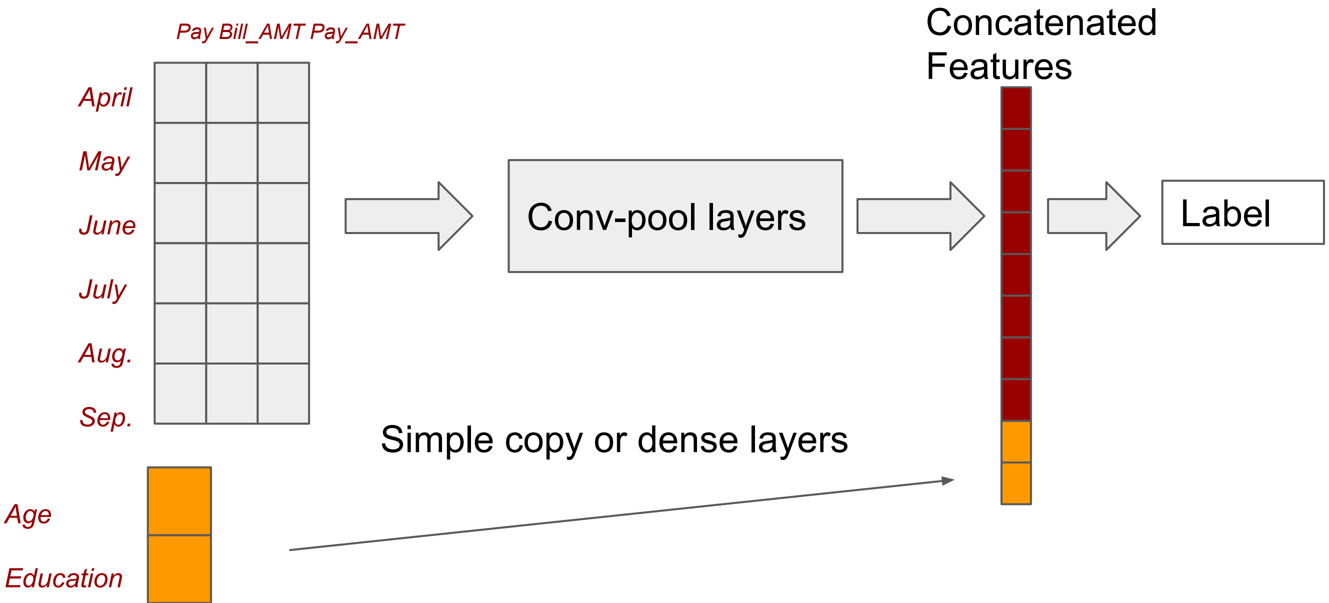
Channel 2

Channel 3

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 later



In Keras, it is easy

Manipulate complex graph topologies

Models with multiple inputs and outputs

The functional API makes it easy to manipulate multiple inputs and outputs. This cannot be handled with the `Sequential` API.

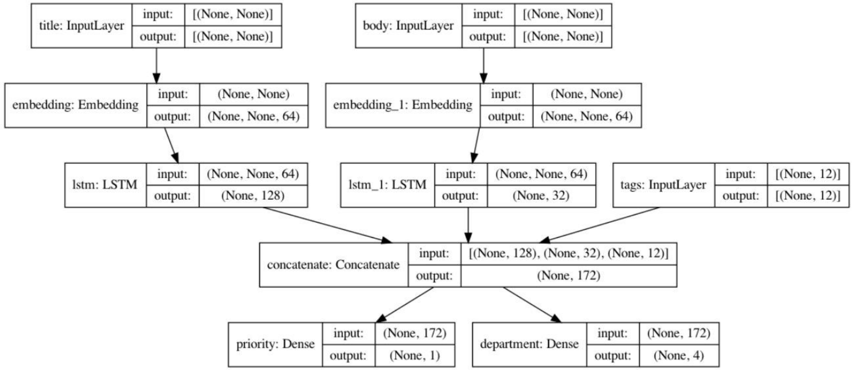
For example, if you're building a system for ranking customer issue tickets by priority and routing them to the correct department, then the model will have three inputs:

- the title of the ticket (text input),
- the text body of the ticket (text input), and
- any tags added by the user (categorical input)

This model will have two outputs:

- the priority score between 0 and 1 (scalar sigmoid output), and
- the department that should handle the ticket (softmax output over the set of departments).

You can build this model in a few lines with the functional API:



Implementation Details:
https://keras.io/guides/functional_api/