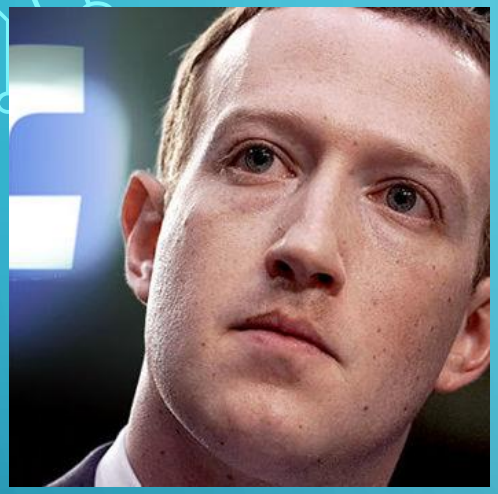
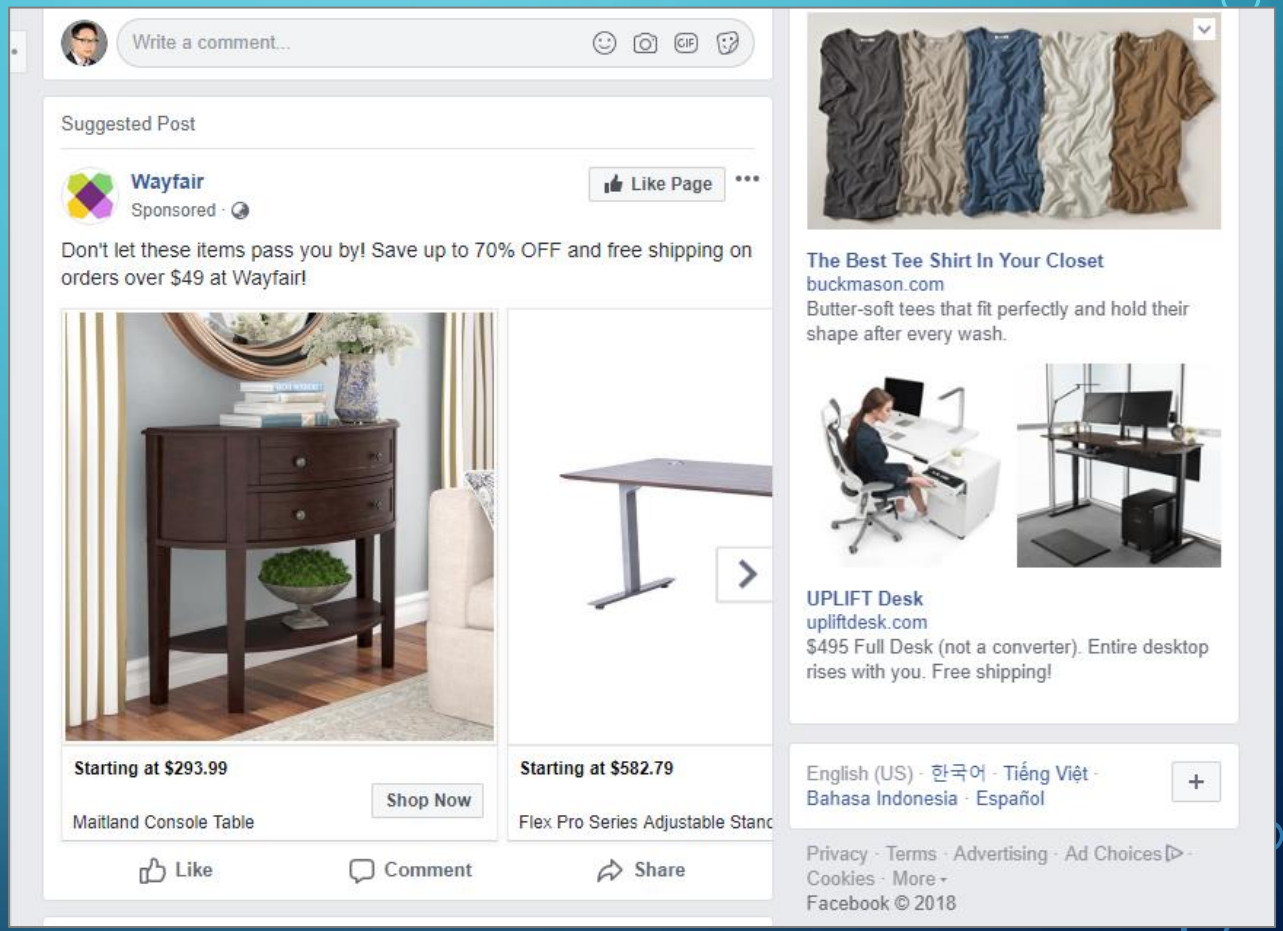
DEEP LEARNING WITH KERAS

CNN – FASHION MNIST

YANN LECUN'S JOB @ FACEBOOK





Suggested Post, Banner Ads

IMAGE RECOGNITION

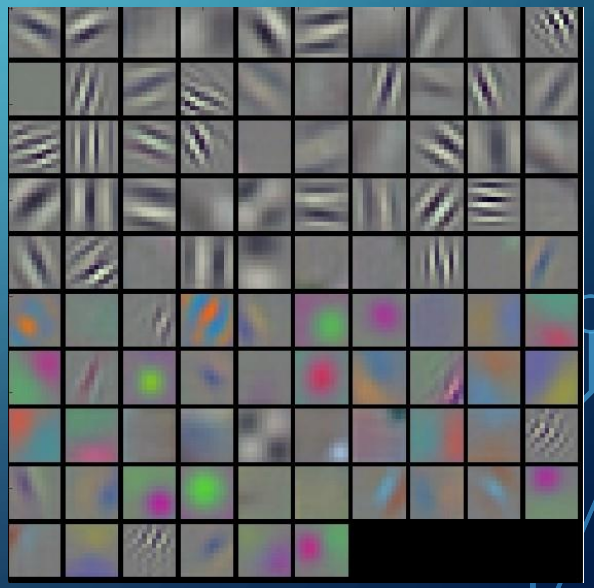
- Crucial in many ML applications

- Main challenge: # of pixels!!!!!!

CONVOLUTIONAL NEURAL NETWORK (CNN)

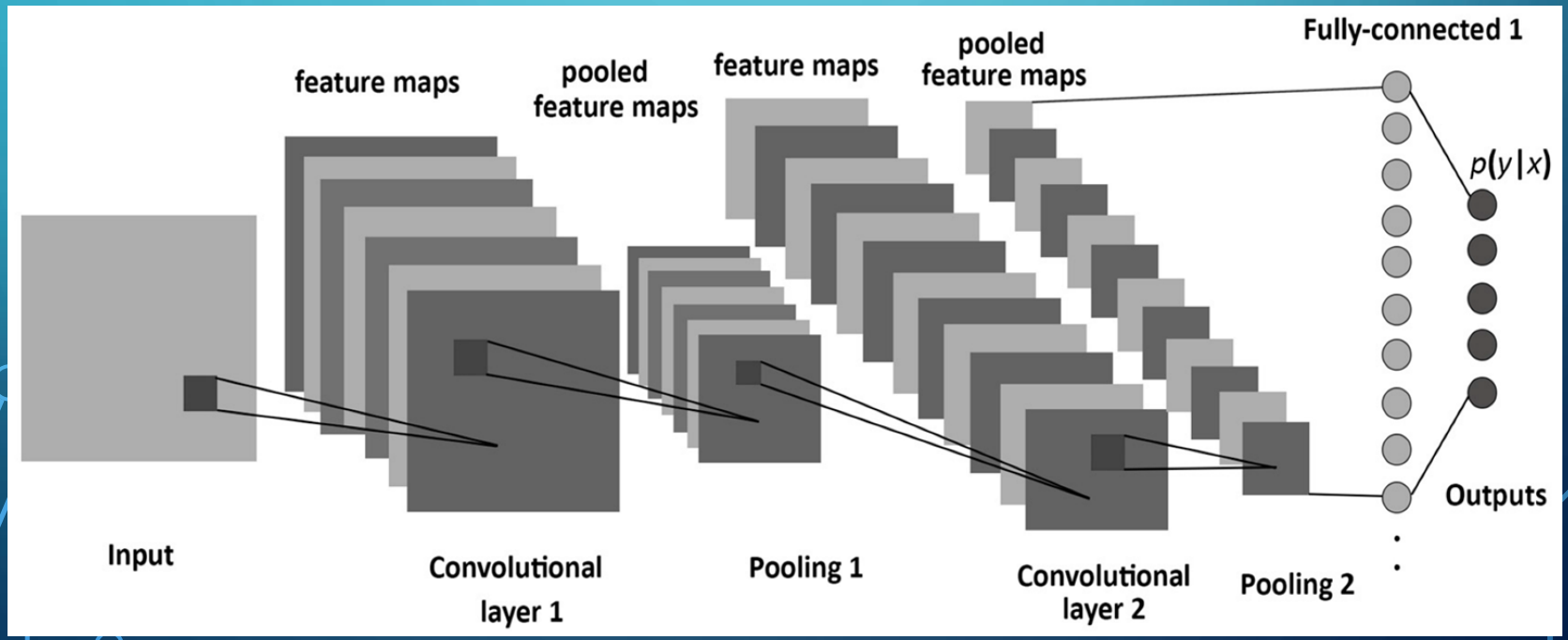
- What is Convolution?
  - 영상으로부터 특정 feature들을 추출하기 위한 필터를 구현할 때 convolution을 사용

- Why CNN?
  - Yann LeCun (1989)
  - 이미지 인식에서 MLP (Multi-Layer Perceptron)를 사용하게 되면, MLP는 모든 입력이 위치와 상관없이 동일한 수준의 중요도를 갖음
  - 이를 이용해 fully-connected neural network를 구성하게 되면 model 크기가 엄청나게 커지는 문제가 생김
  - 이에 대한 해결책으로 탄생한 것이 CNN

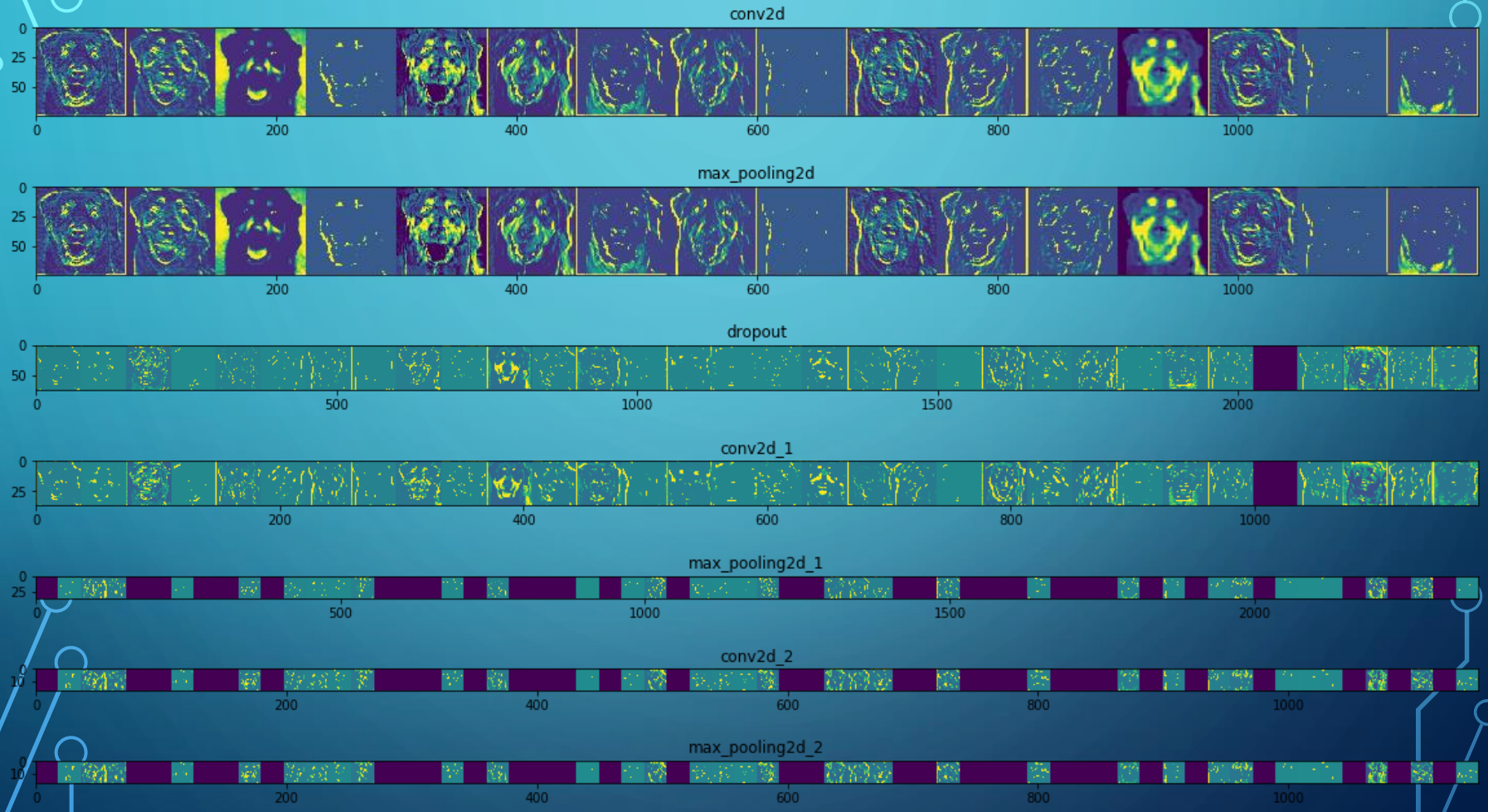


CNN DEEP LEARNING

- Feature Extraction
  - Key idea to manage complexity while maintaining accuracy
  - Features are extracted via CONV:RELU:POOL

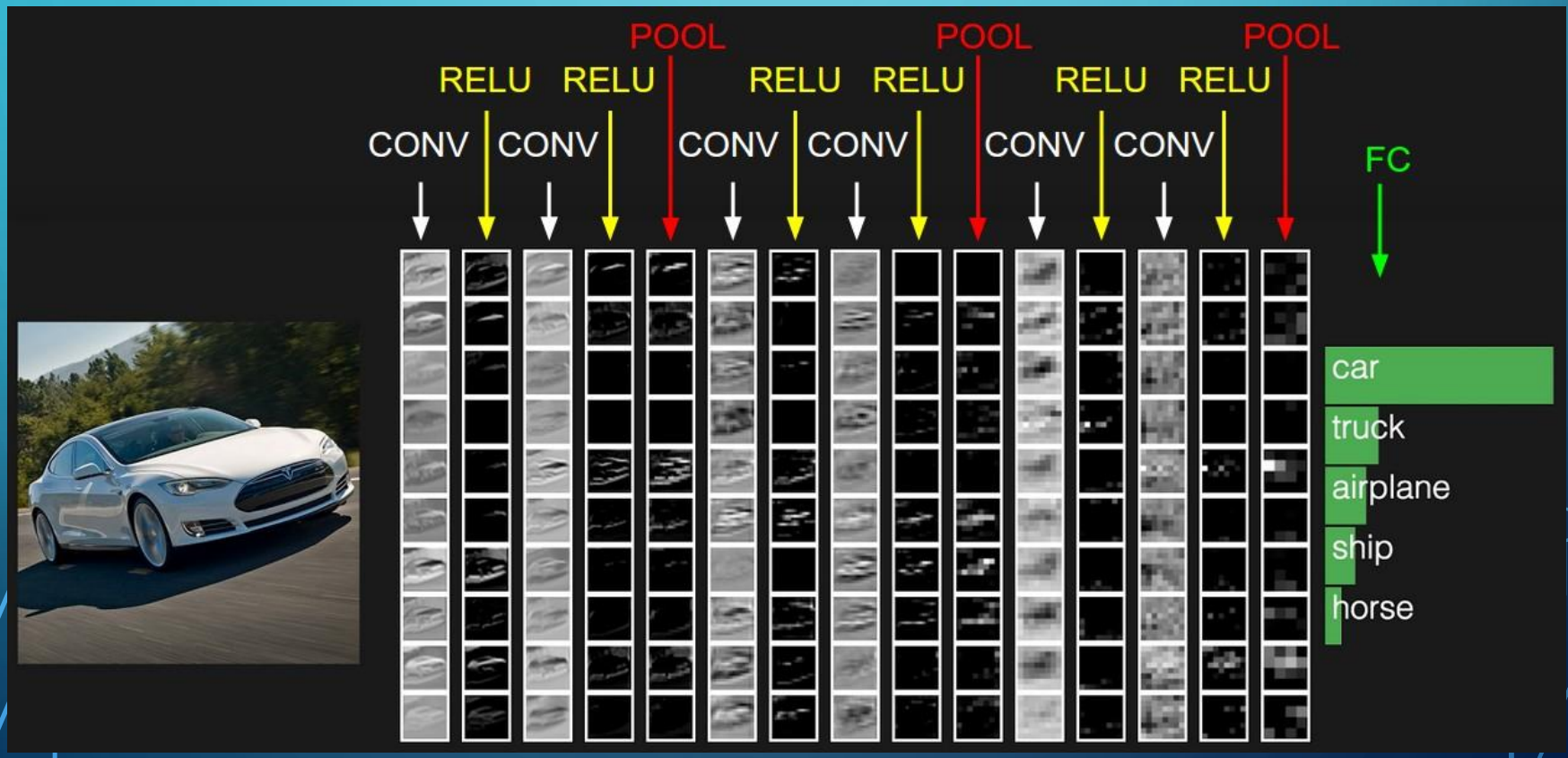


FEATURE MAP EXAMPLES



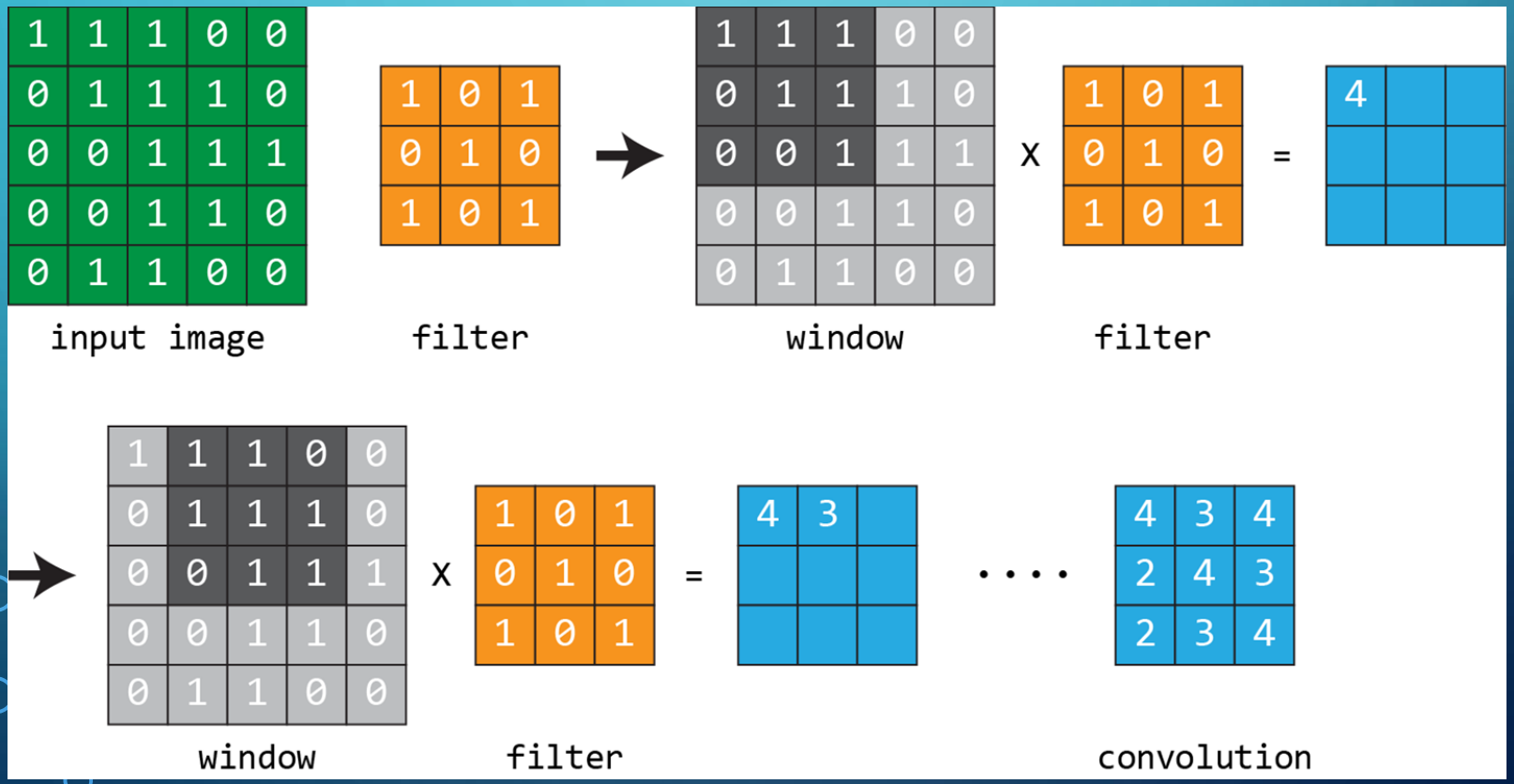
CNN DEEP LEARNING EXAMPLE

- Hugely popular in image recognition

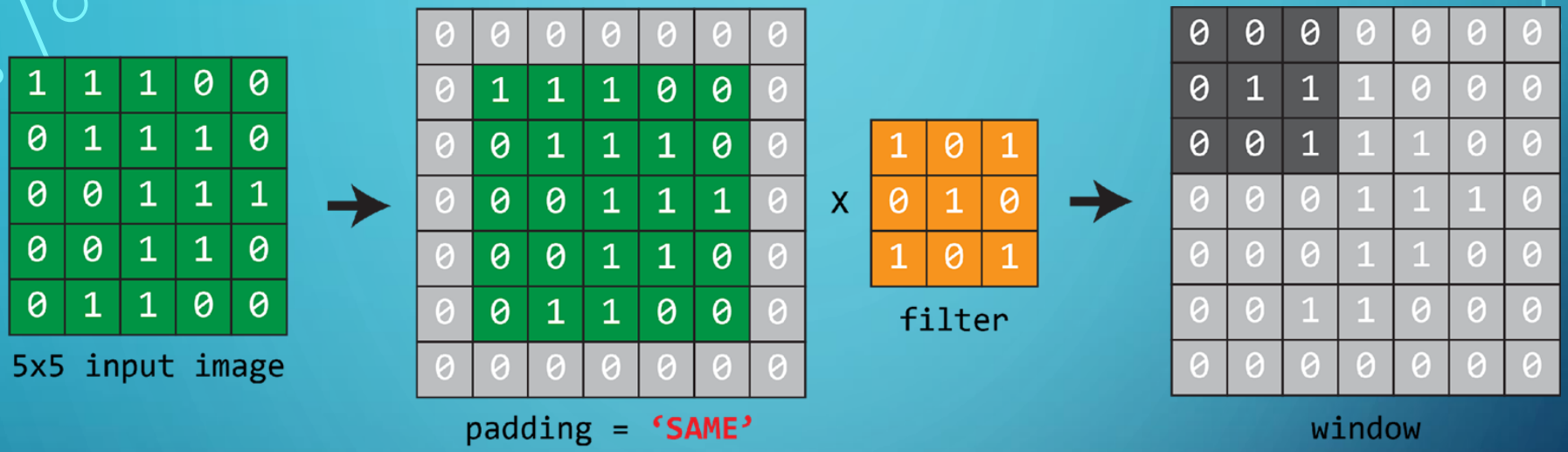


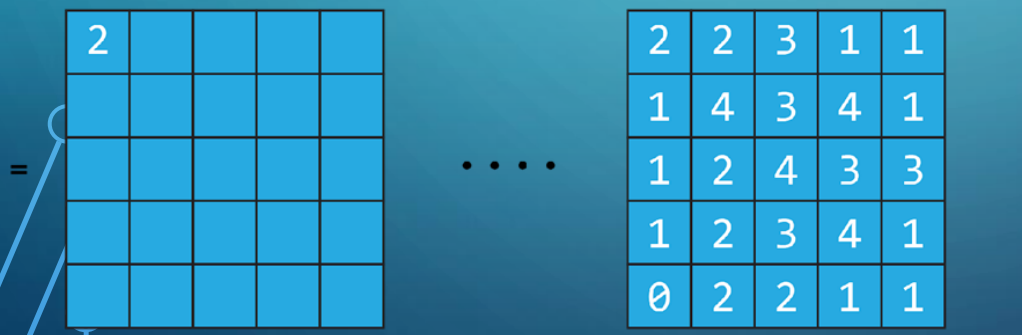
CONVOLUTION EXAMPLE

- Objective: to extract features from the input data



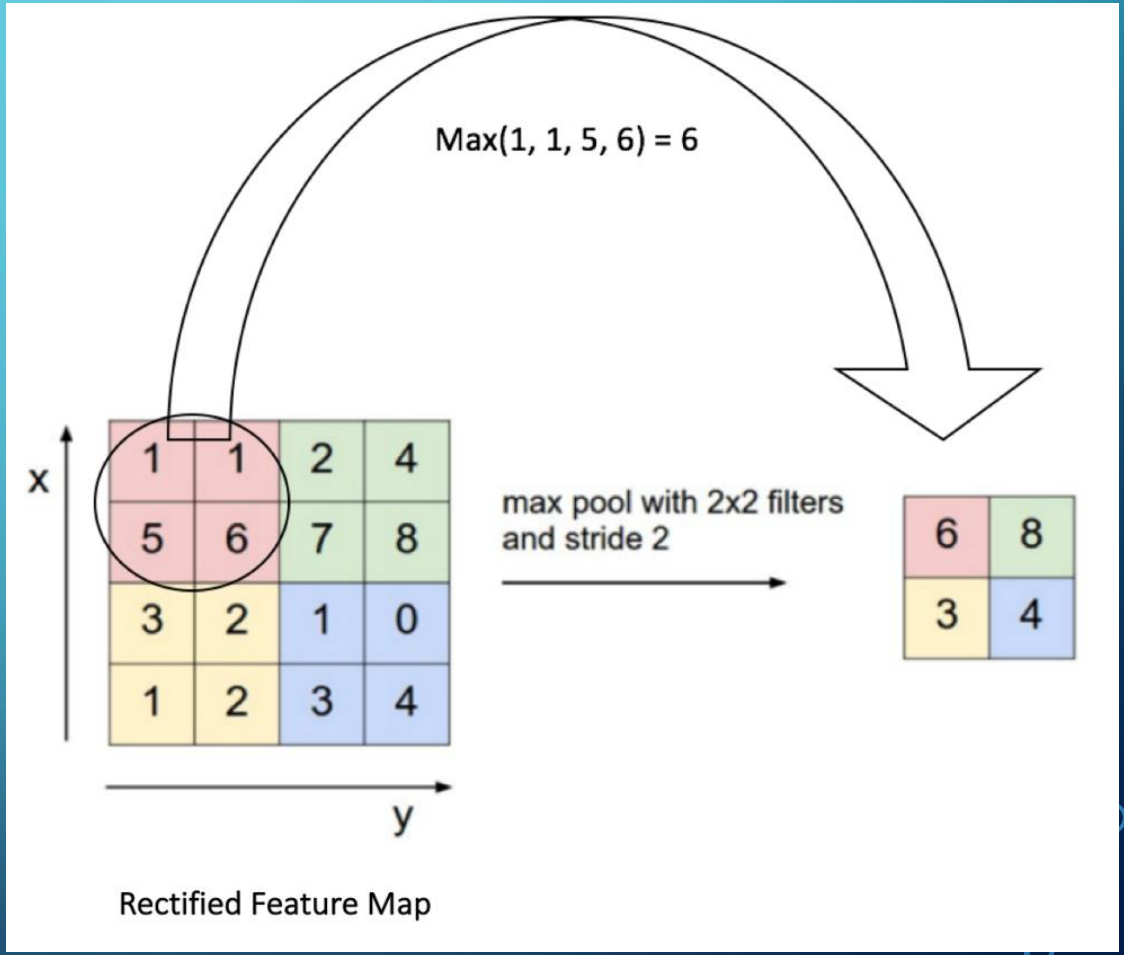
CONVOLUTION WITH PADDING





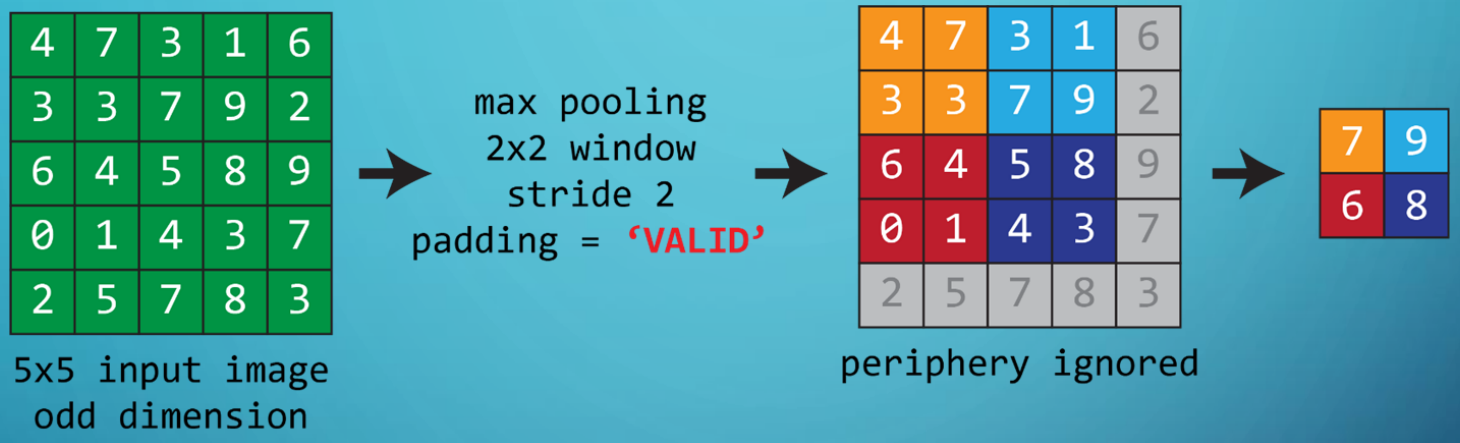
POOLING

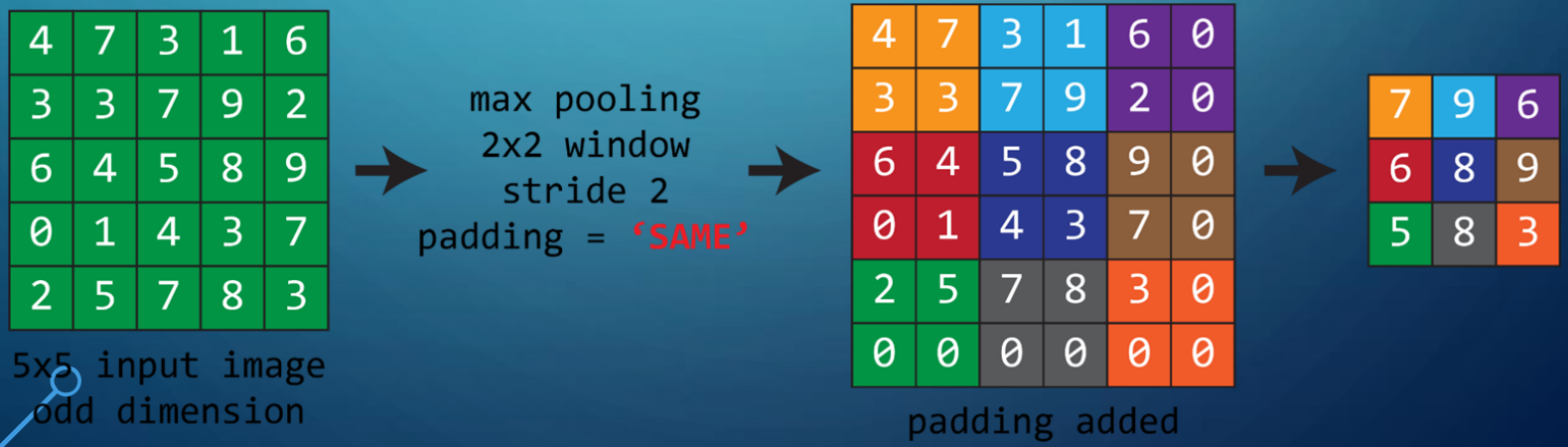
- objective
  - reduce the feature ma p dimension and keep the important informat ion
  - types: max, average, sum, etc.



POOLING WITH PADDING

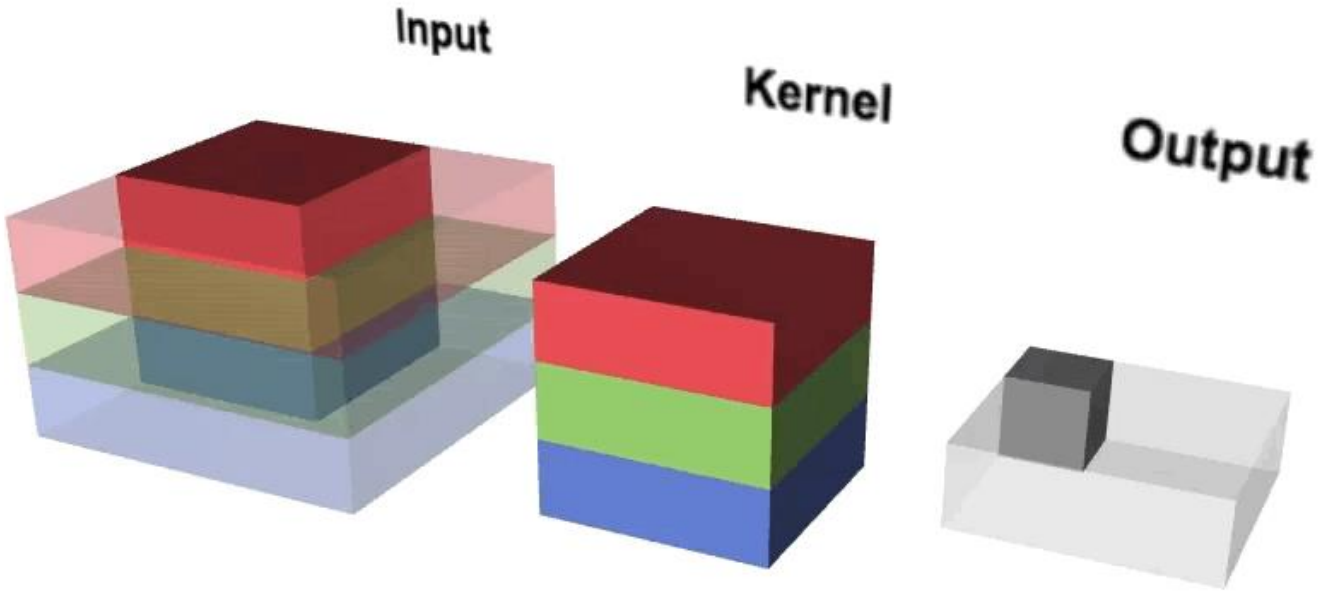
- Better handle odd dimensions





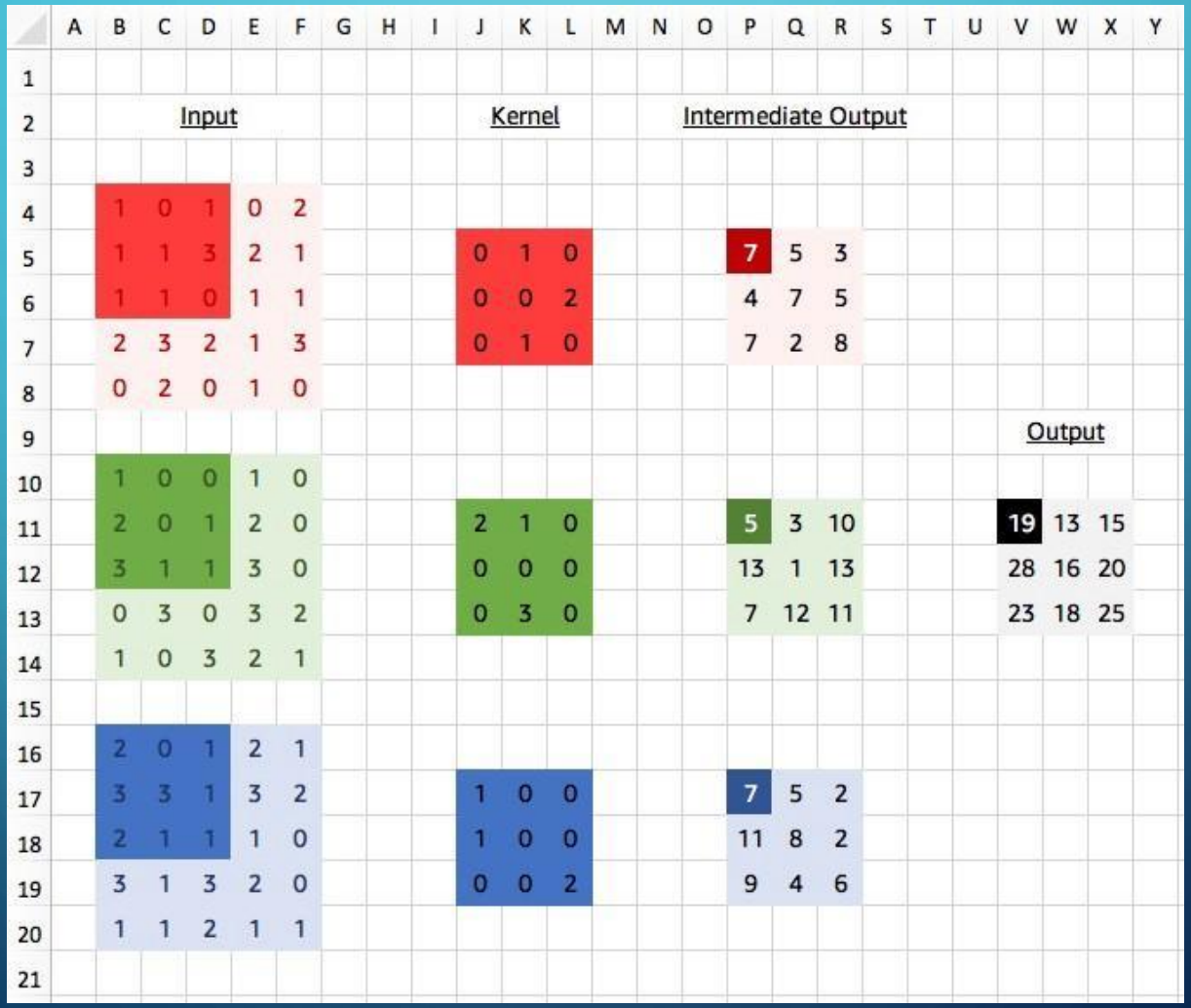
HANDLING MULTIPLE CHANNELS

- 3D filters
  - gi



https://medium.com/apache-mxnet/multi-channel-convolutions-explained-with-ms-excel-9bbf8eb77108

HANDLING MULTIPLE CHANNELS (CONT)



Input

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

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

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

Kernel

0	1	0
0	0	2
0	1	0

2	1	0
0	0	0
0	3	0

1	0	0
1	0	0
0	0	2

Intermediate Output

7	5	3
4	7	5
7	2	8

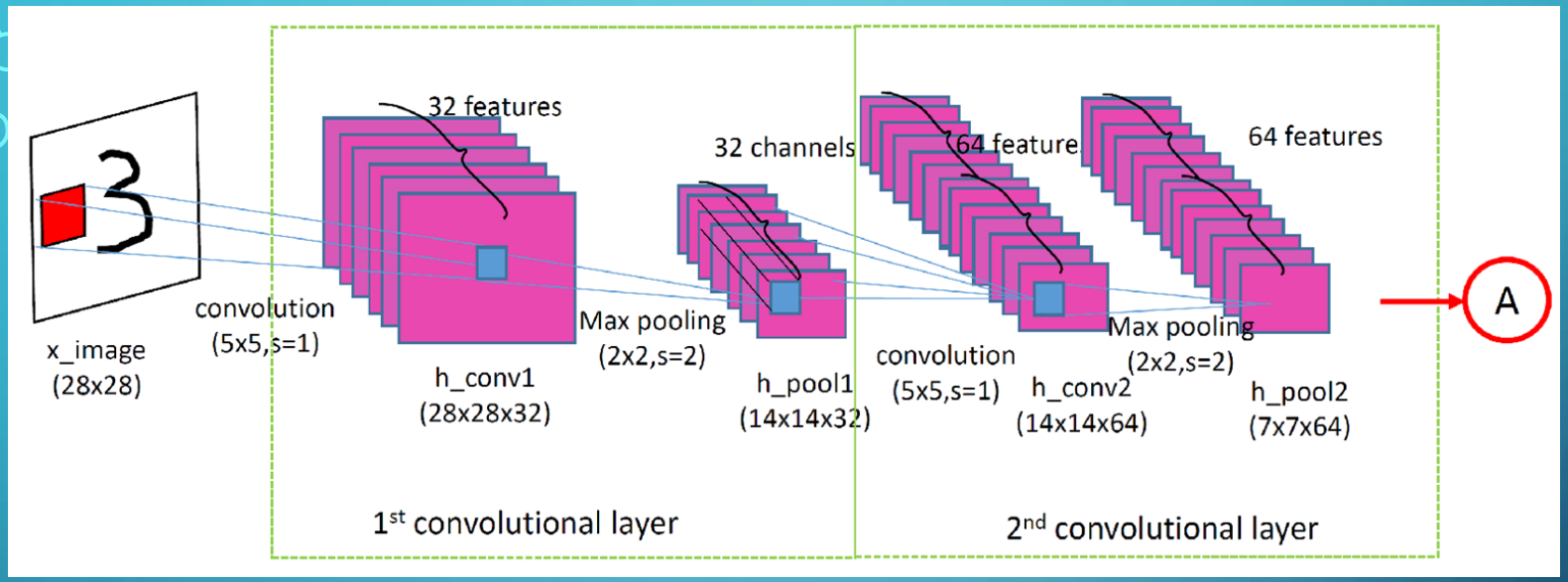
5	3	10
13	1	13
7	12	11

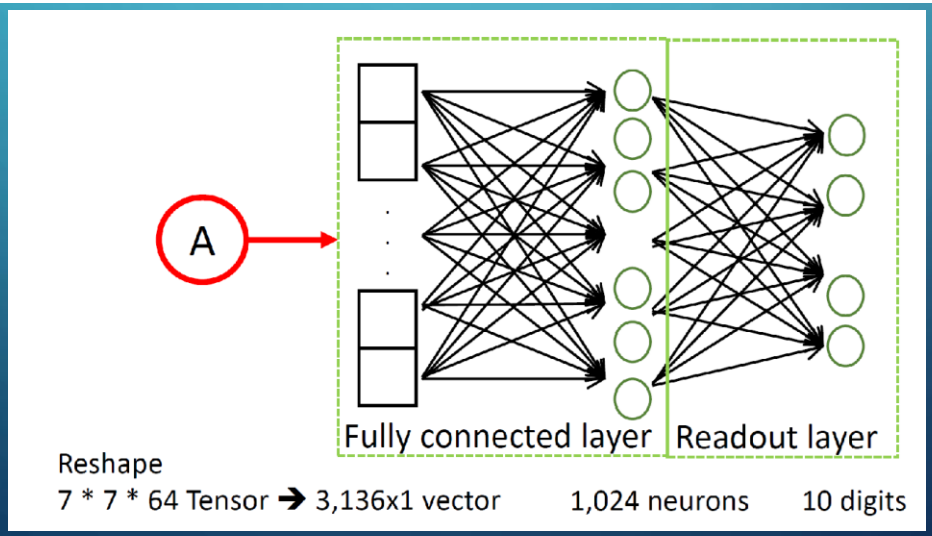
7	5	2
11	8	2
9	4	6

Output

19	13	15
28	16	20
23	18	25

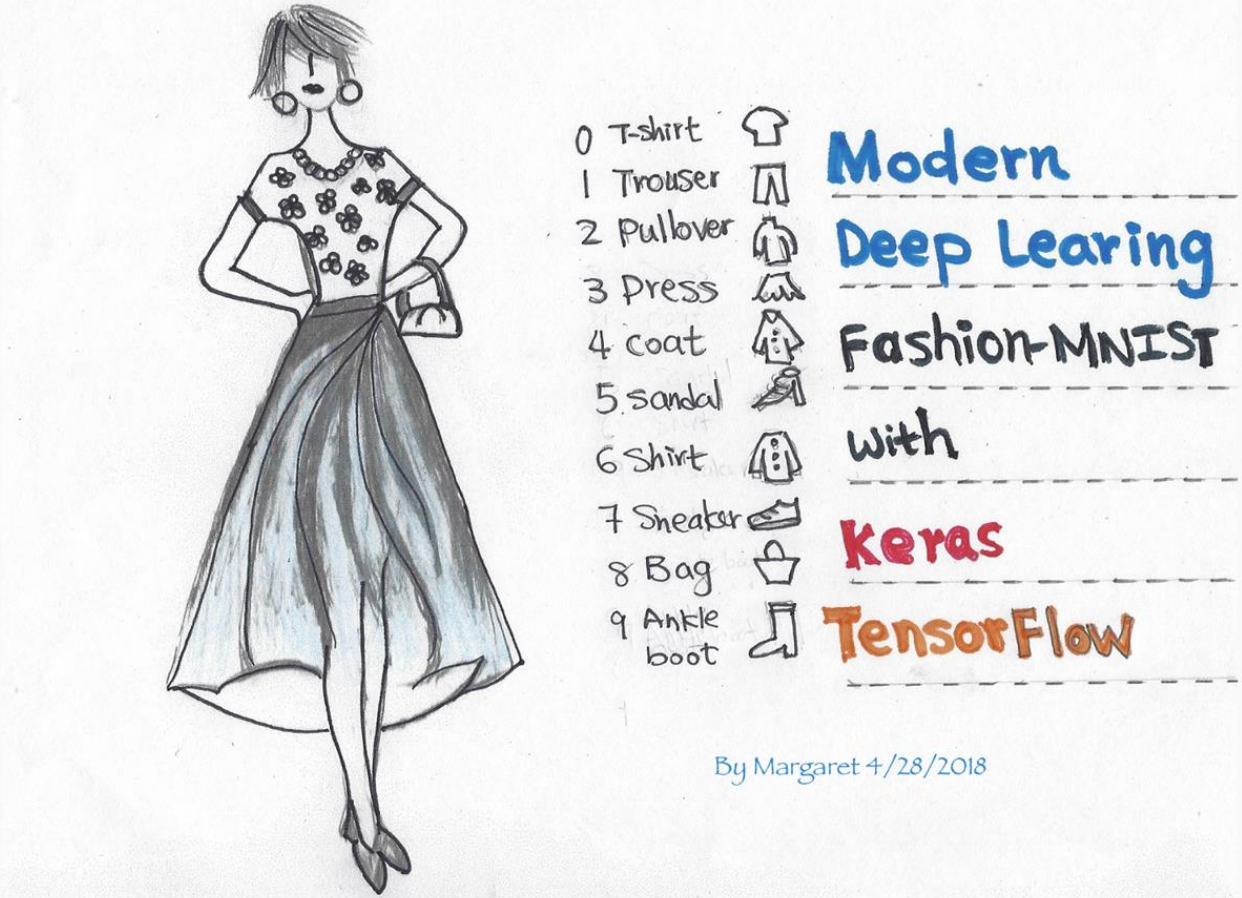
LECUN CNN ARCHITECTURE





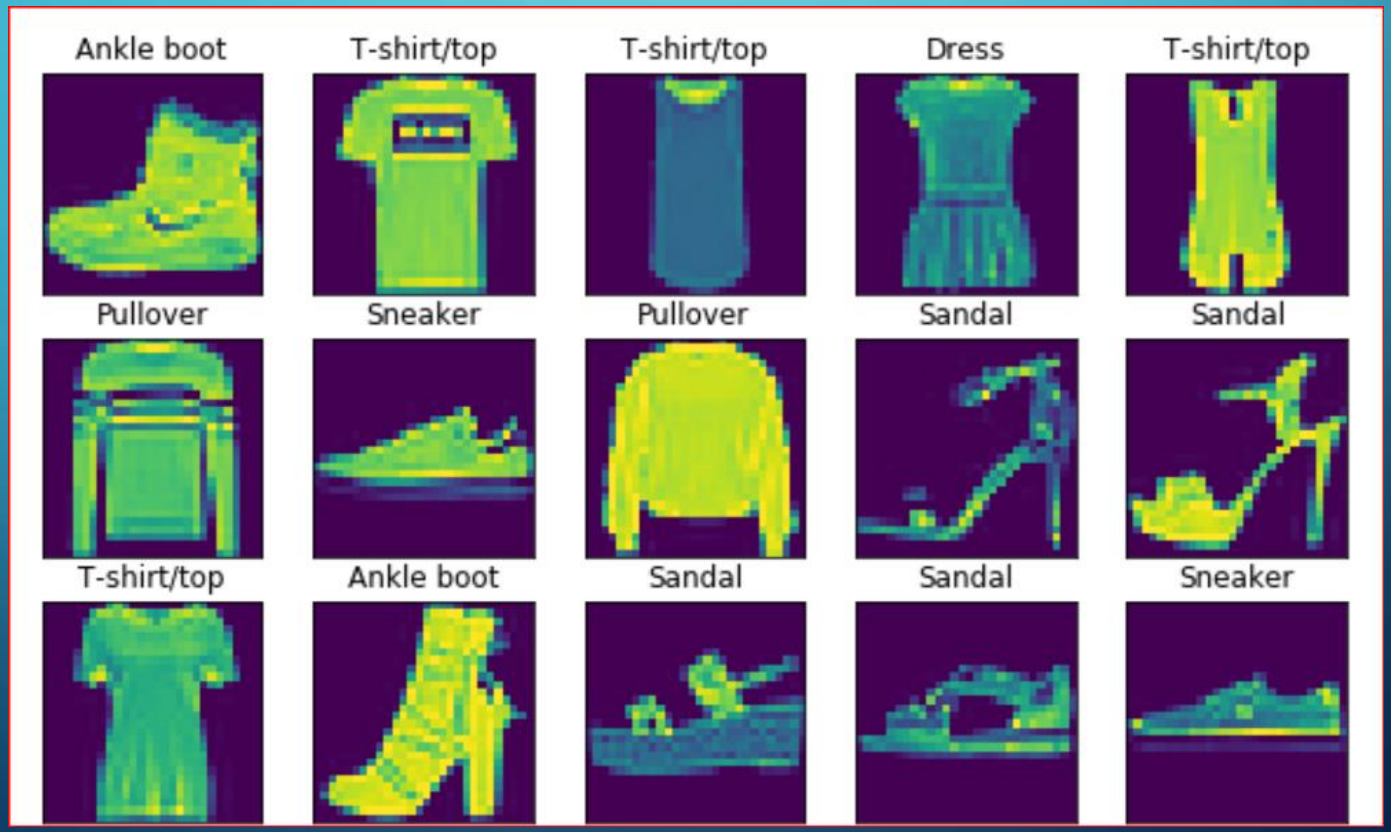
FASHION MNIST DATABASE

- 10 fashion items in gray-scale images



FASHION MNIST DATABASE (CONT)

- 70,000 low resolution image files (28x28 pixels)
  - Each pixel data is in [0, 255]



3. FASHION-ITEMS.PY (1/12)

```
# import packages
import matplotlib.pyplot as plt
import numpy as np
from keras.datasets import fashion_mnist
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Flatten, Activation
from keras.layers import Dense, MaxPool2D, Conv2D, InputLay
er
```

3. FASHION-ITEMS.PY (2/12)

```
# global constants and hyper-parameters
TOTAL_CLASS = 10
MY_SAMPLE = 5
MY_EPOCH = 10
MY_BATCH = 200
```

3. FASHION-ITEMS.PY (3/12)

```
# load DB and split into train vs. test
# fasion MNIST data is 28x28 gray-scale image
(X_train, Y_train), (X_test, Y_test) = fashion_mnist.load_d
ata()
print('\n== DB SHAPING INFO ==')
print("X train shape:", X_train.shape)
print("Y train shape:", Y_train.shape)
print("X test shape:", X_test.shape)
print("Y test shape:", Y_test.shape)
```

3. FASHION-ITEMS.PY (4/12)

```
# define labels
labels = ['t-shirt', 'trouser', 'pullover', 'dress', 'coat’
, 'sandal', 'shirt', 'sneaker', 'bag', 'ankle-boot']


# display a sample image and label
print('\n== SAMPLE DATA (RAW) ==')
sample = Y_train[MY_SAMPLE]
print("This is a", labels[sample], sample)
print(X_train[MY_SAMPLE])
plt.imshow(X_train[MY_SAMPLE])
plt.show()
```

3. FASHION-ITEMS.PY (5/12)

```
# data scaling into [0, 1]
X_train = X_train / 255.0
X_test = X_test / 255.0
```

3. FASHION-ITEMS.PY (6/12)

```
# reshaping before entering CNN
# one-hot encoding is used for the output
# we use channel-last ordering (keras default)
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)
Y_train = np_utils.to_categorical(Y_train, TOTAL_CLASS)
Y_test = np_utils.to_categorical(Y_test, TOTAL_CLASS)
```

NP_UTILS.TO_CATEGORICAL()

```
from keras.utils import np_utils
import numpy as np

my_array = [0, 2, 1, 2, 0]

# one-hot encoding
one_hot = np_utils.to_categorical(my_array, 3)
print(one_hot)
```

3. FASHION-ITEMS.PY (7/12)

```
    ###################################
    # MODEL BUILDING AND TRAINING #
    ###################################


# build a keras sequential model of our CNN
# total parameter count formula =
# (filter_height * filter_width * input_channels + 1) * #_f
ilters
model = Sequential()
model.add(InputLayer(input_shape = (28, 28, 1)))
```

3. FASHION-ITEMS.PY (8/12)

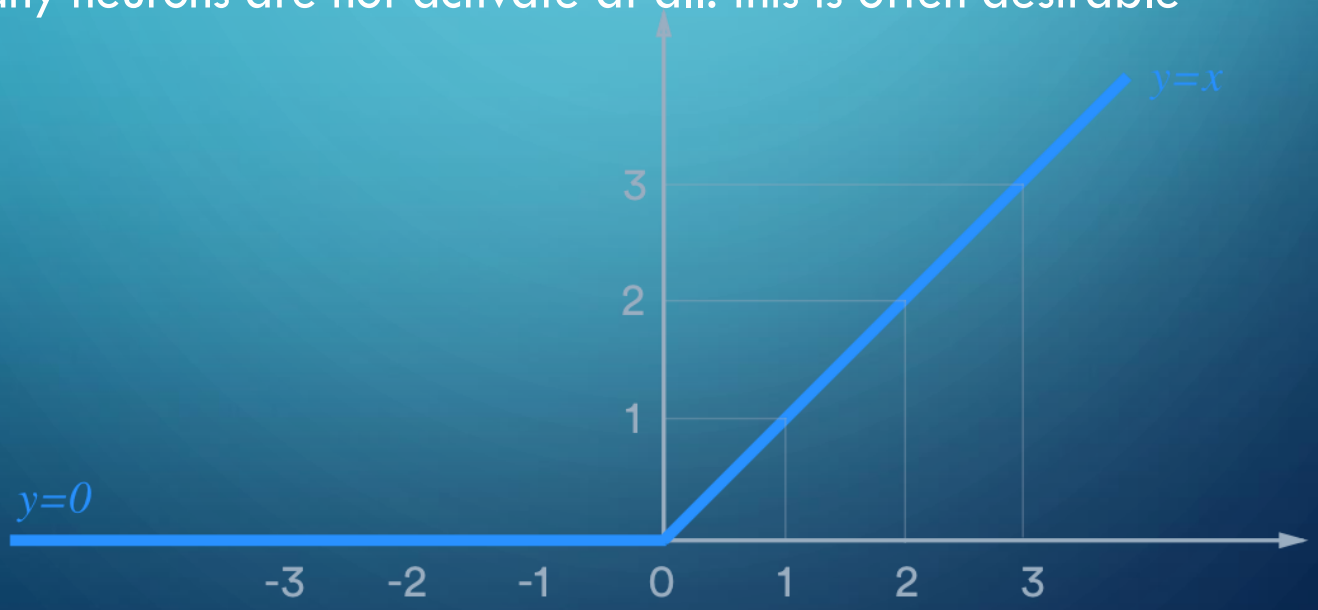
```
# no. parameters = (2 x 2 x 1 + 1) x 32
model.add(Conv2D(32, kernel_size = (2, 2), padding = 'same'
, activation = 'relu'))
model.add(MaxPool2D(padding = 'same', pool_size = (2, 2)))

# no. parameters = (2 x 2 x 32 + 1) x 64
model.add(Conv2D(64, kernel_size = (2, 2), padding = 'same'
, activation = 'relu'))
model.add(MaxPool2D(padding = 'same', pool_size = (2, 2)))
```

3. FASHION-ITEMS.PY (8/12)

```
model.add(Flatten())
model.add(Dense(128, activation = 'relu'))

model.add(Dense(TOTAL_CLASS, activation = 'softmax'))
model.summary()
```

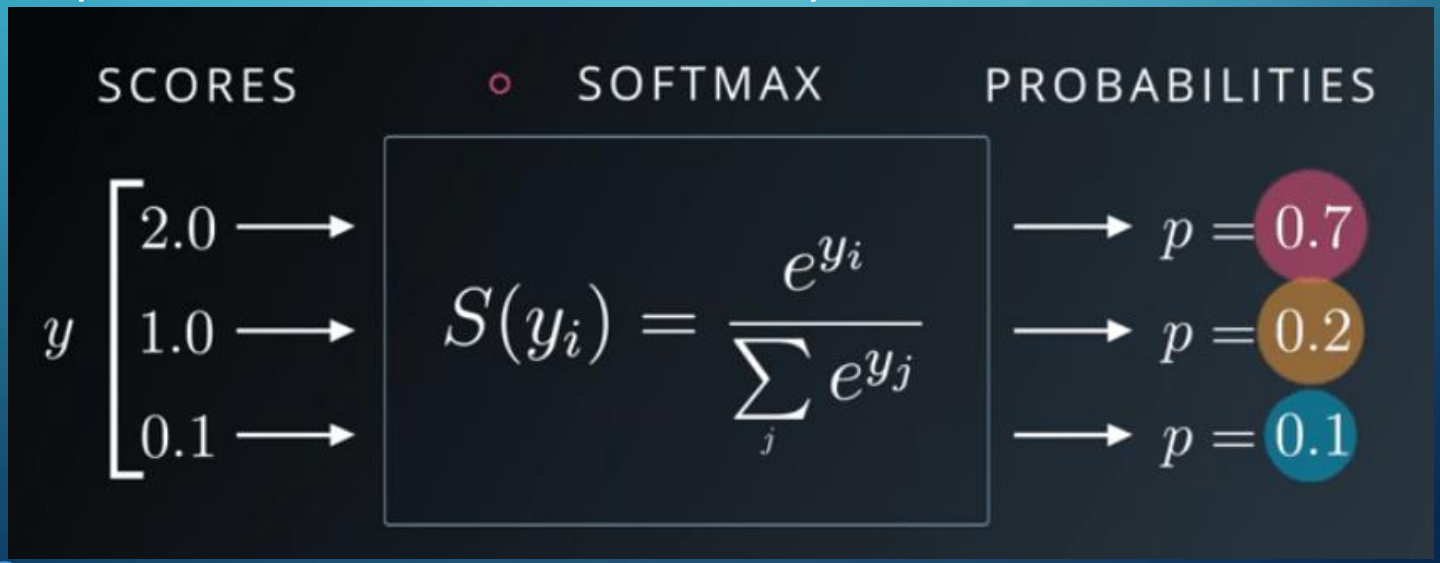
RELU ACTIVATION

- Rectified linear unit (ReLU)
  - F(x) = max(0, x)
  - Most commonly used activation function, especially in CNNs
  - No vanishing gradient problem as in sigmoid or tanh
  - Many neurons are not activate at all: this is often desirable



SOFTMAX ACTIVATION

- Useful for multi-class classification
  - Assigns probabilities to an object being one of several different things
  - Softmax gives a list of values [0, 1] that add up to 1
  - Step 1: we add up the evidence of our input being in certain classes
  - Step 2: we convert that evidence into probabilities

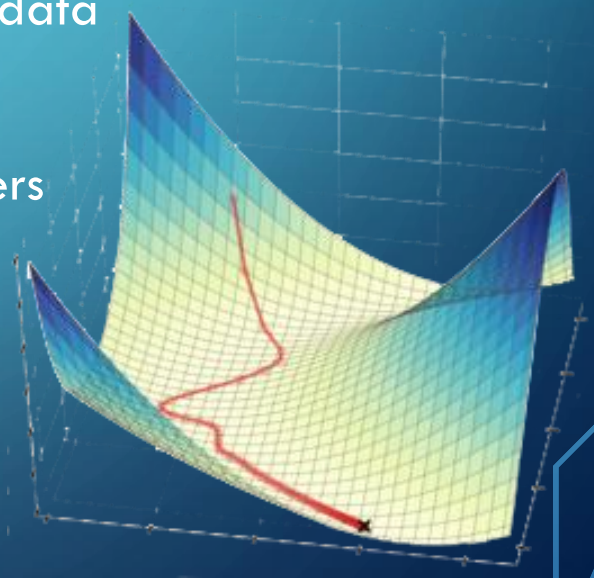


3. FASHION-ITEMS.PY (9/12)

```
# model training and saving
model.compile(optimizer = 'adam', loss = 'categorical_cross
entropy', metrics = ['accuracy'])
model.fit(X_train, Y_train, epochs = MY_EPOCH, batch_size =
MY_BATCH, verbose = 1)
model.save('after.h5')
```

ADAM OPTIMIZER

- Adam: Adaptive moment estimation
  - Keeps separate learning rates for each weight
  - Exponentially decay average of previous gradients
  - Fairly memory efficient since it doesn't keep a history of anything
  - Work well for both sparse matrices and noisy data
  - Seems promising for the stock market data.
  - Works well with little tuning of hyperparameters



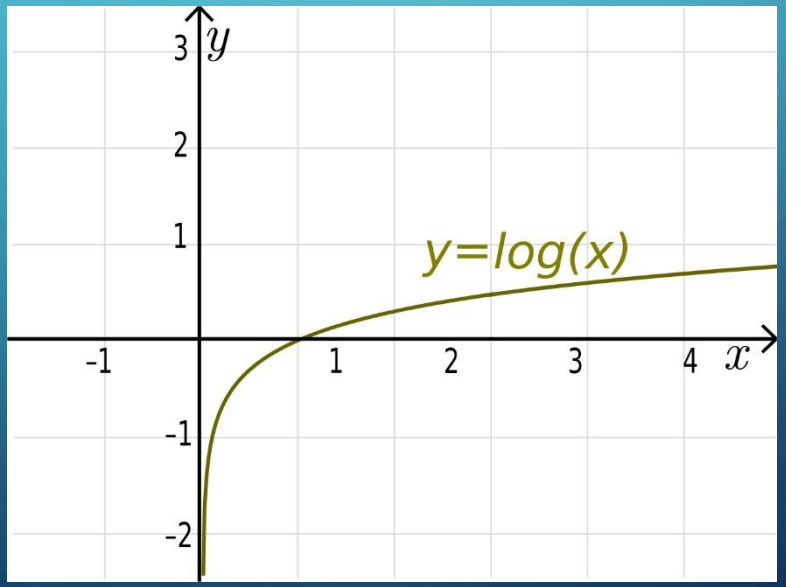
http://ruder.io/optimizing-gradient-descent/

CROSS ENTROPY LOSS FUNCTION

- **Cross entropy function**

$$H_{y'}(y) = -\sum_i y'_i \times log(y_i)$$

  - ***y*** **is prediction (softmax: [0, 1]), and** ***y'*** **is true value (one-hot: 0/1)**
  - **measures how inefficient our predictions are for describing the truth**



CROSSENTROPY

```
# illustration of cross entropy loss function
import numpy as np

j = [0.03, 0.03, 0.01, 0.9, 0.01, 0.01, 0.0025, 0.0025,
        0.0025, 0.0025]
k = [0,0,0,1,0,0,0,0,0,0]

# cross entropy calculation
log = -(np.log(j))
print(log)
prod = k * log
print(prod)
i = np.sum(prod)
print(i)
```

3. FASHION-ITEMS.PY (10/12)

```
    ######################
    # MODEL EVALUATION #
    ######################



# model evaluation
score = model.evaluate(X_test, Y_test, verbose = 1)
print('\nKeras CNN model loss = ', score[0])
print('Keras CNN model accuracy = ', score[1])
```

3. FASHION-ITEMS.PY (11/12)

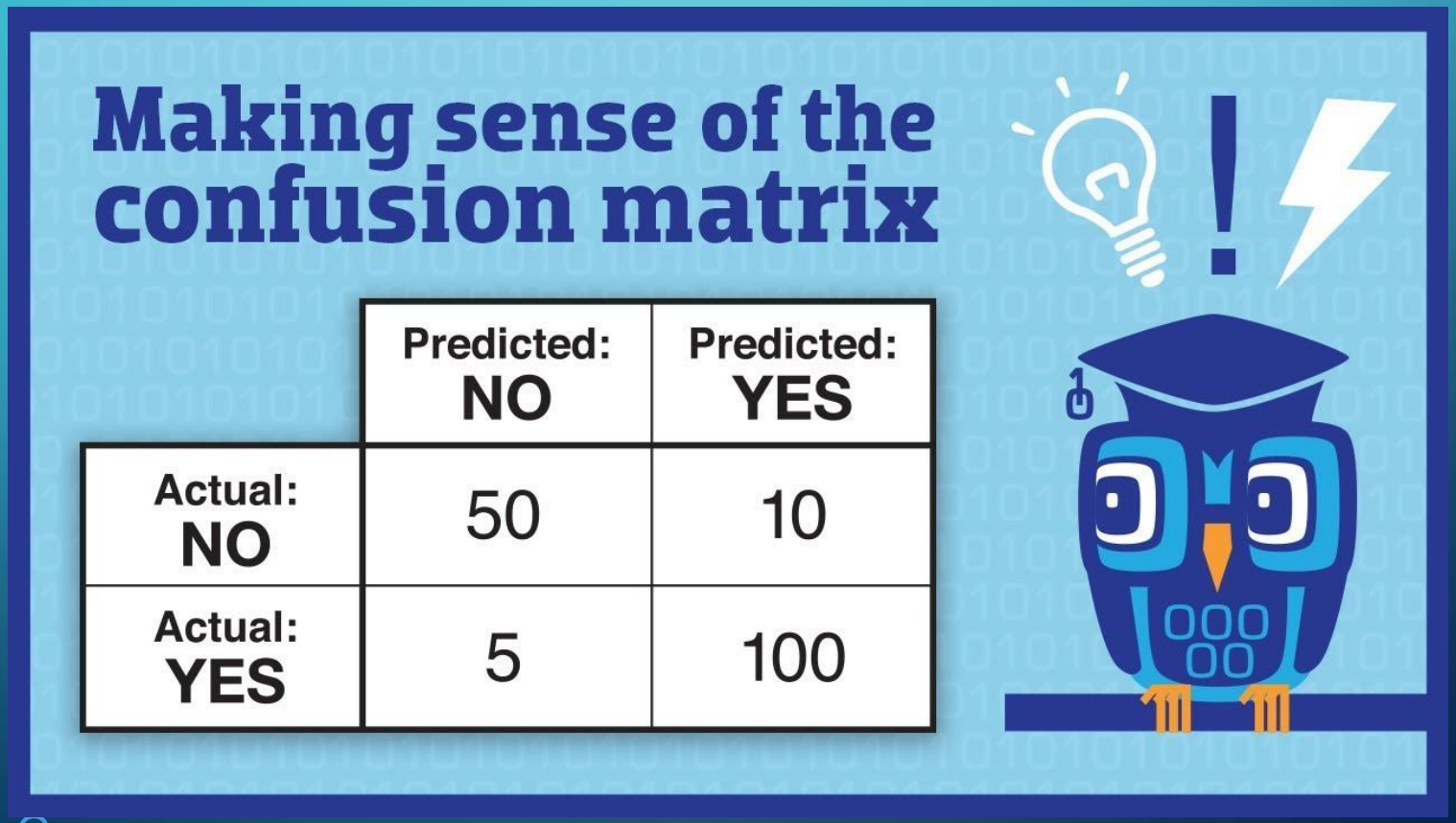
```
# display confusion matrix
# we take argmax on one-hot encoding results
from sklearn.metrics import confusion_matrix
answer = np.argmax(Y_test, axis = 1)

pred = model.predict(X_test)
pred = np.argmax(pred, axis = 1)

print('\n== CONFUSION MATRIX ==')
print(confusion_matrix(answer, pred))
```

NUMPY.ARGMAX()

```
import numpy as np

my_array = [[10, 14, 12], [13, 11, 15]]

# flatten the array
print(np.argmax(my_array))

# axis = 0 means along the column
print(np.argmax(my_array, axis = 0))

# axis = 1 means along the row
print(np.argmax(my_array, axis = 1))
```

CONFUSION MATRIX

- Useful in multi-class classification



3. FASHION-ITEMS.PY (12/12)

```
# calculate F1 score using confusion matrix
from sklearn.metrics import f1_score
print("\nF1 score:", f1_score(answer, pred, average = 'micr
o'))

# testing the model after learning with a sample
sample = X_train[MY_SAMPLE]
sample = sample.reshape(1, 28, 28, 1)
pred = model.predict(sample)
print("\nPrediction after learning:",
        labels[np.argmax(pred)], "\n")
```

F1 SCORE

- Popular metric for classification accuracy calculation

$$precision = \frac{true\ positive}{true\ positive + false\ positive}$$

$$recall = \frac{true\ positive}{true\ positive + false\ negative}$$

$$F1\ score = \frac{2 \times precision \times recall}{precision + recall}$$