On Device Machine Learning using Raspberry Pi

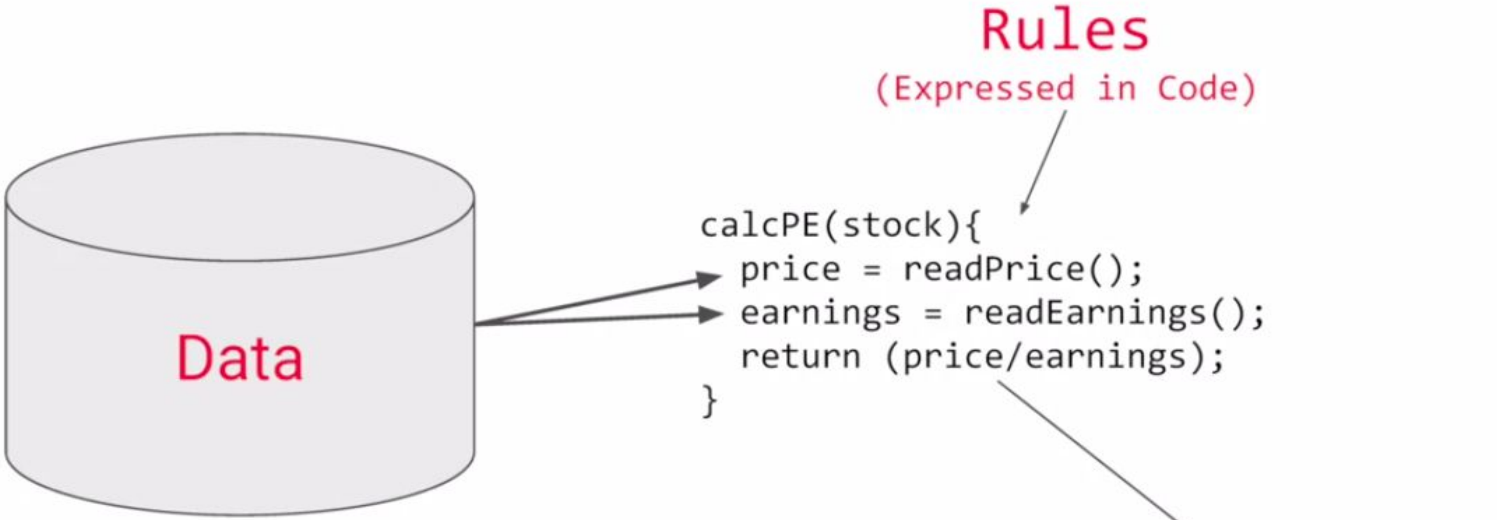
Tun Tun Win

Disclaimer

All images and Contents are from Coursera, Introduction to Tensorflow for Artificial Intelligence, Machine Learning, Instructurer Laurence Maroney, Google AI and TensorFlow

Part I

Hello World!



Data

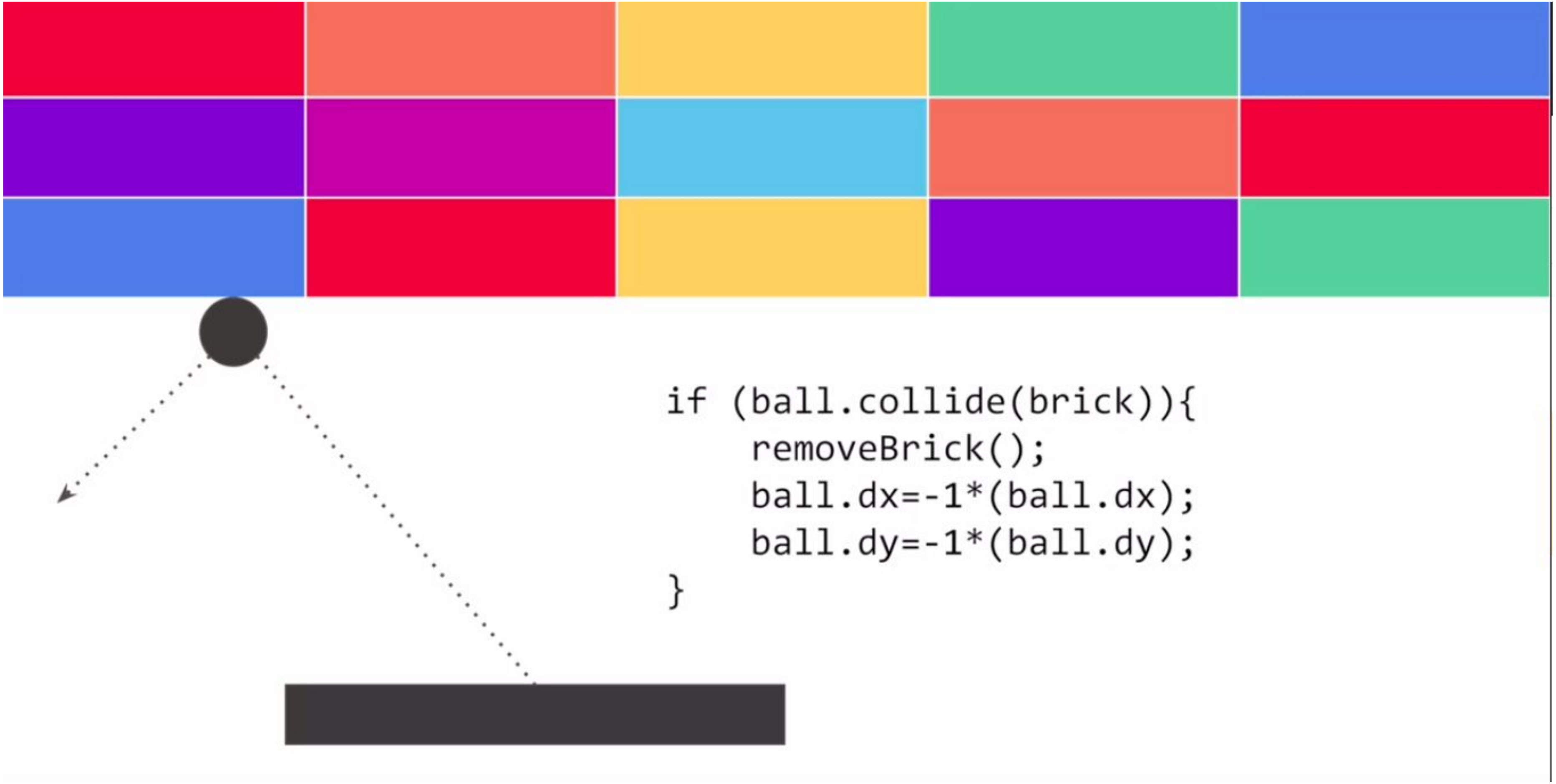
Rules

(Expressed in Code)

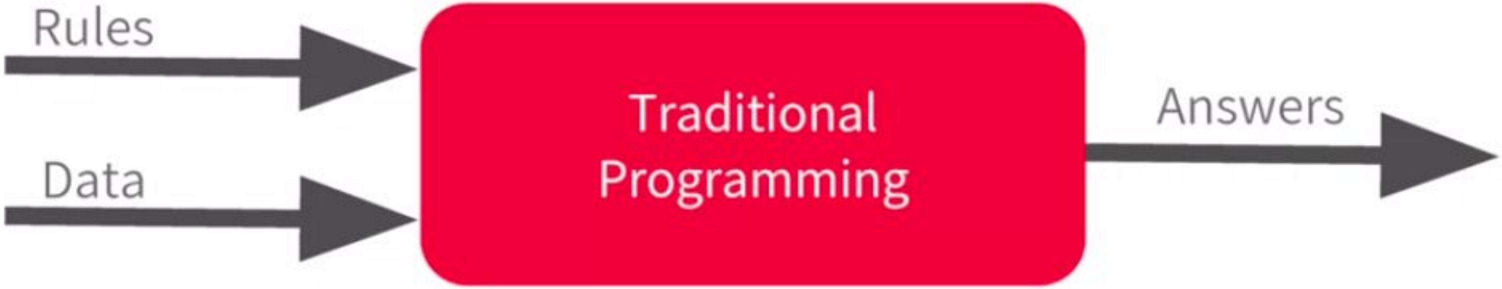
```
calcPE(stock){
  price = readPrice();
  earnings = readEarnings();
  return (price/earnings);
}
```

Answers

(Returned From Code)



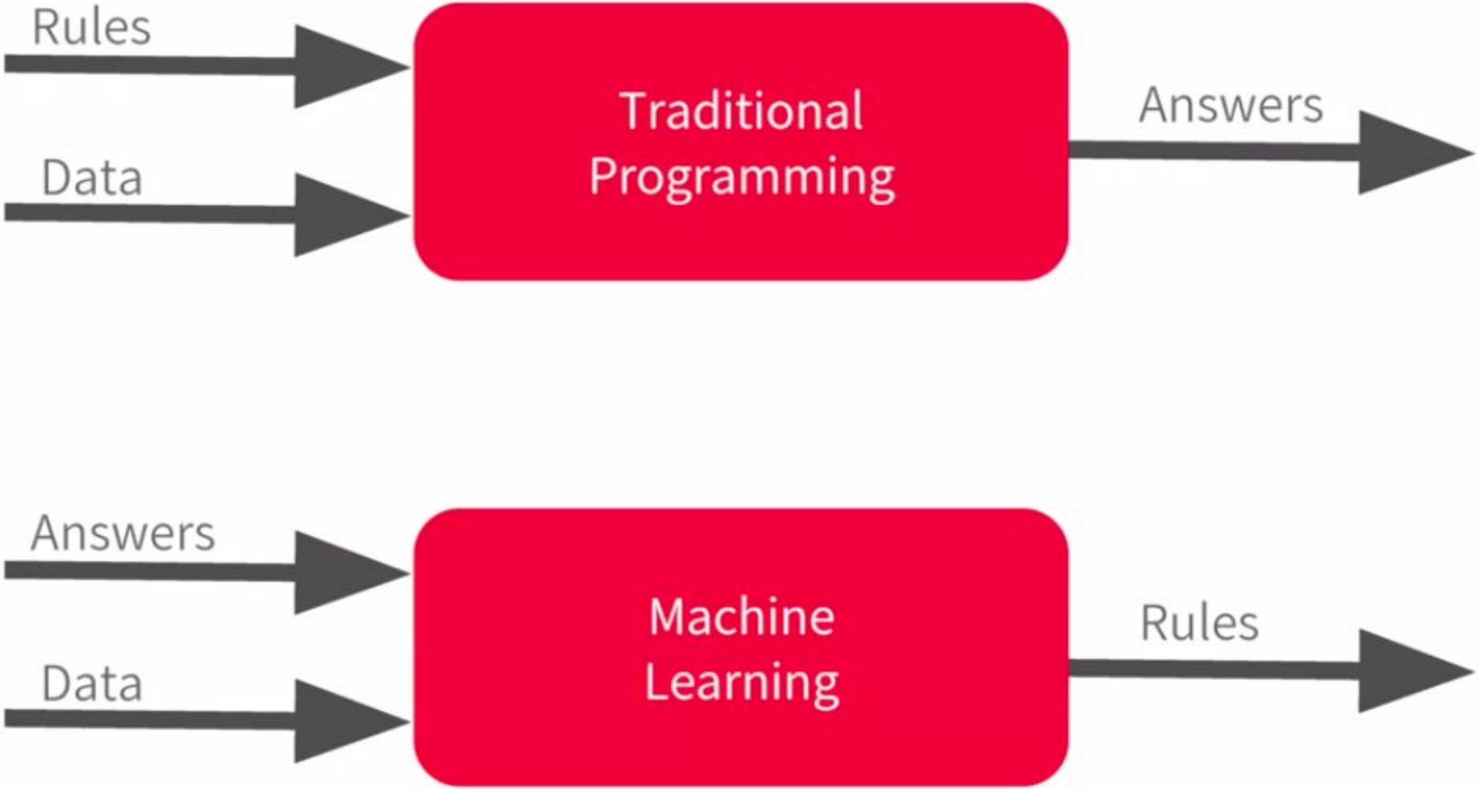
```
if (ball.collide(brick)){
    removeBrick();
    ball.dx=-1*(ball.dx);
    ball.dy=-1*(ball.dy);
}
```


Rules

Data

Traditional Programming

Answers



Rules

Data

Traditional Programming

Answers

Answers

Data

Machine Learning

Rules

Activity Recognition

```
if(speed<4){
    status=WALKING;
}
```

```
if(speed<4){
    status=WALKING;
} else {
    status=RUNNING;
}
```

```
if(speed<4){
    status=WALKING;
} else if(speed<12){
    status=RUNNING;
} else {
    status=BIKING;
}
```

```
// Oh crap
```

Activity Recognition

```
0101001010100101010
1001010101001011101
0100101010010101001
0101001010100101010
```

Label = WALKING

```
1010100101001010101
0101010010010010001
0010011111010101111
1010100100111101011
```

Label = RUNNING

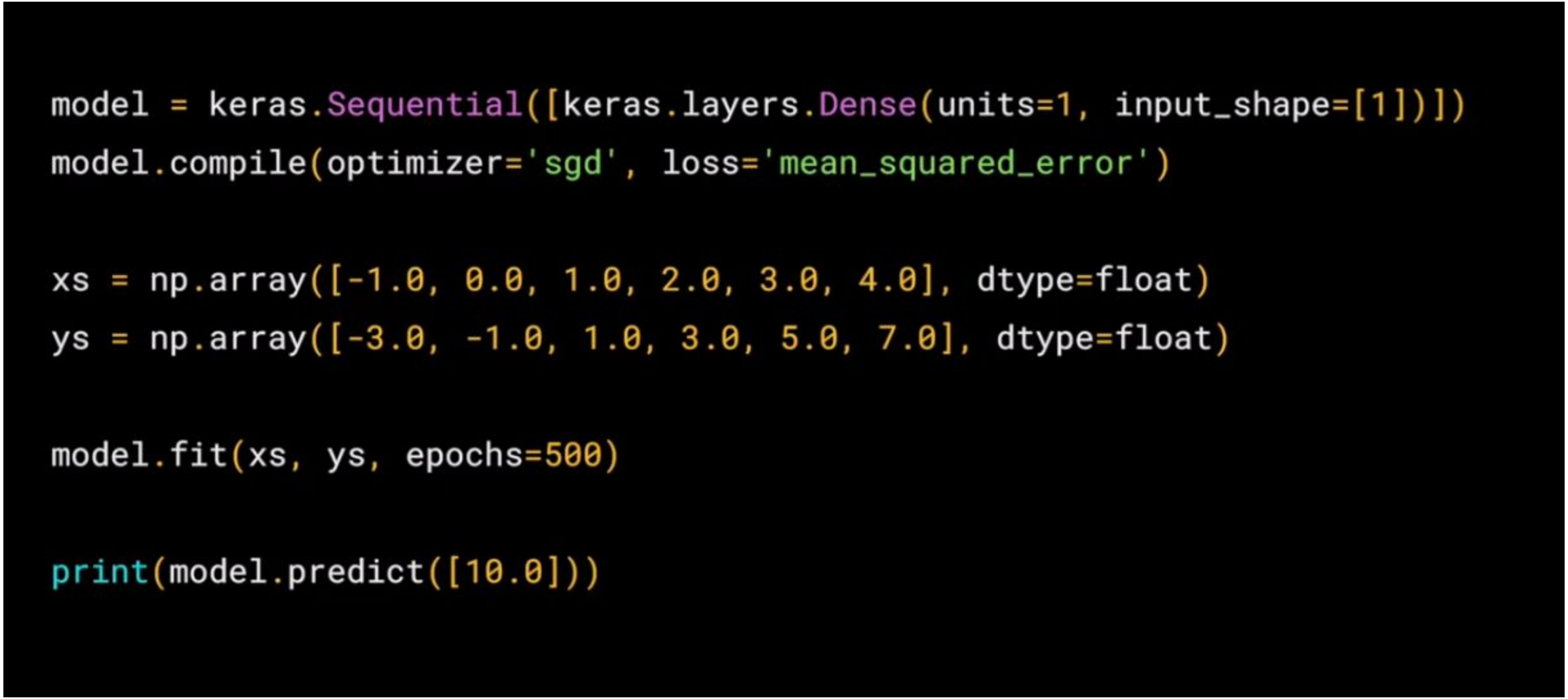
```
1001010011111010101
1101010111010101110
1010101111010101011
1111110001111010101
```

Label = BIKING

```
1111111111010011101
0011111010111110101
0101110101010101110
1010101010100111110
```

Label = GOLFING
(Sort of)

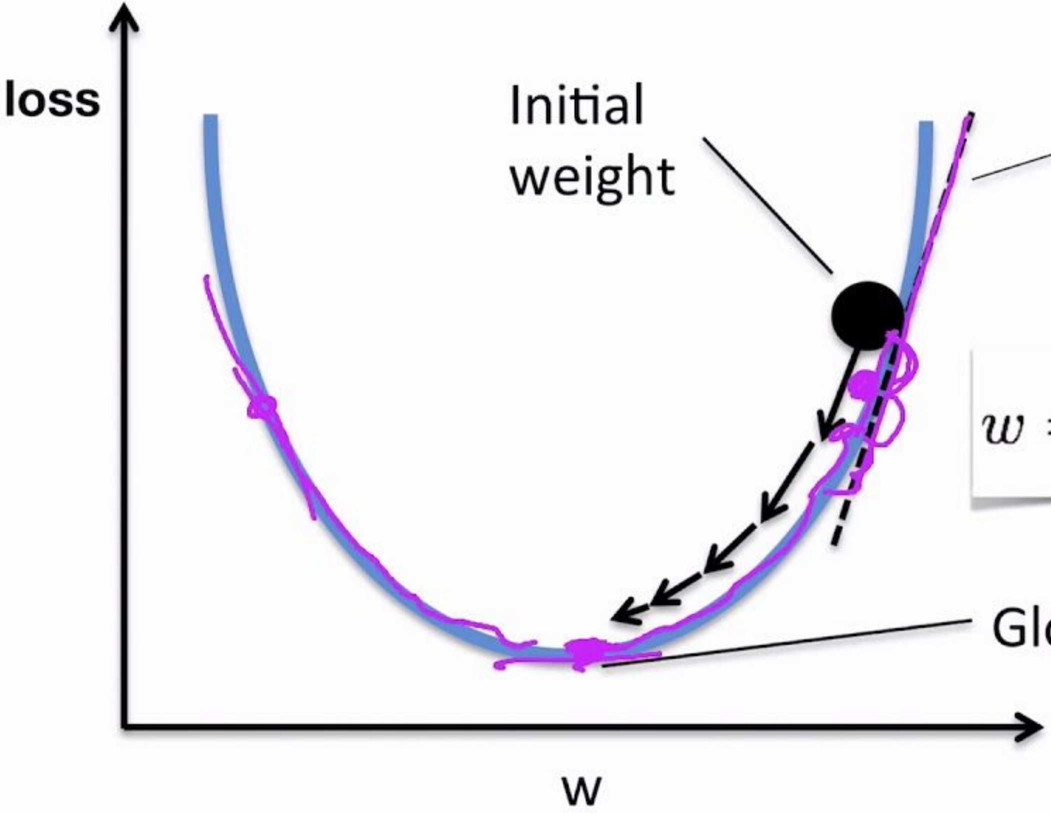
```
X = -1,  0, 1, 2, 3, 4
Y = -3, -1, 1, 3, 5, 7
```



```
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))
```

1 dense= 1 layer, 1 unit = 1 neurons, 1 input_shape = 1

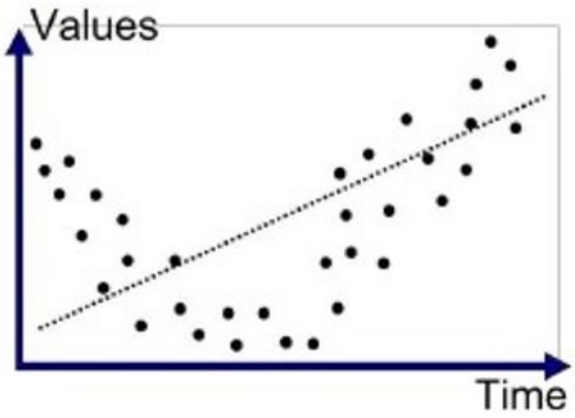
Gradient descent algorithm



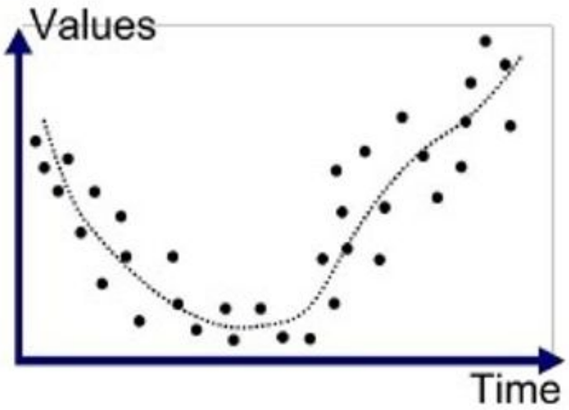
Gradient $\frac{\partial loss}{\partial w}$

$$w = \underline{w} - \overset{=0.01}{\alpha} \boxed{\frac{\partial loss}{\partial w}}$$

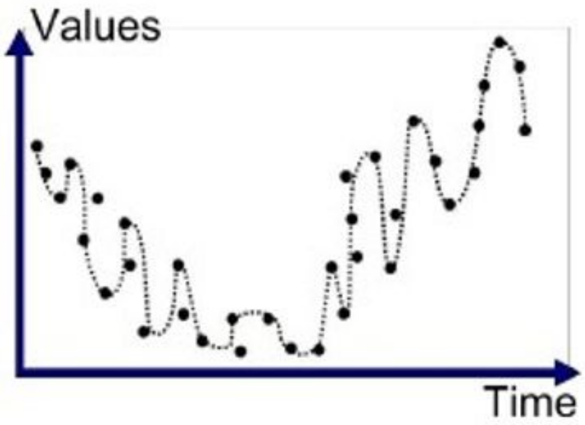
$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \widehat{y} \right)}_{\substack{\text{The square of the difference} \\ \text{between actual and} \\ \text{predicted}}}^{2}$$



Underfitted



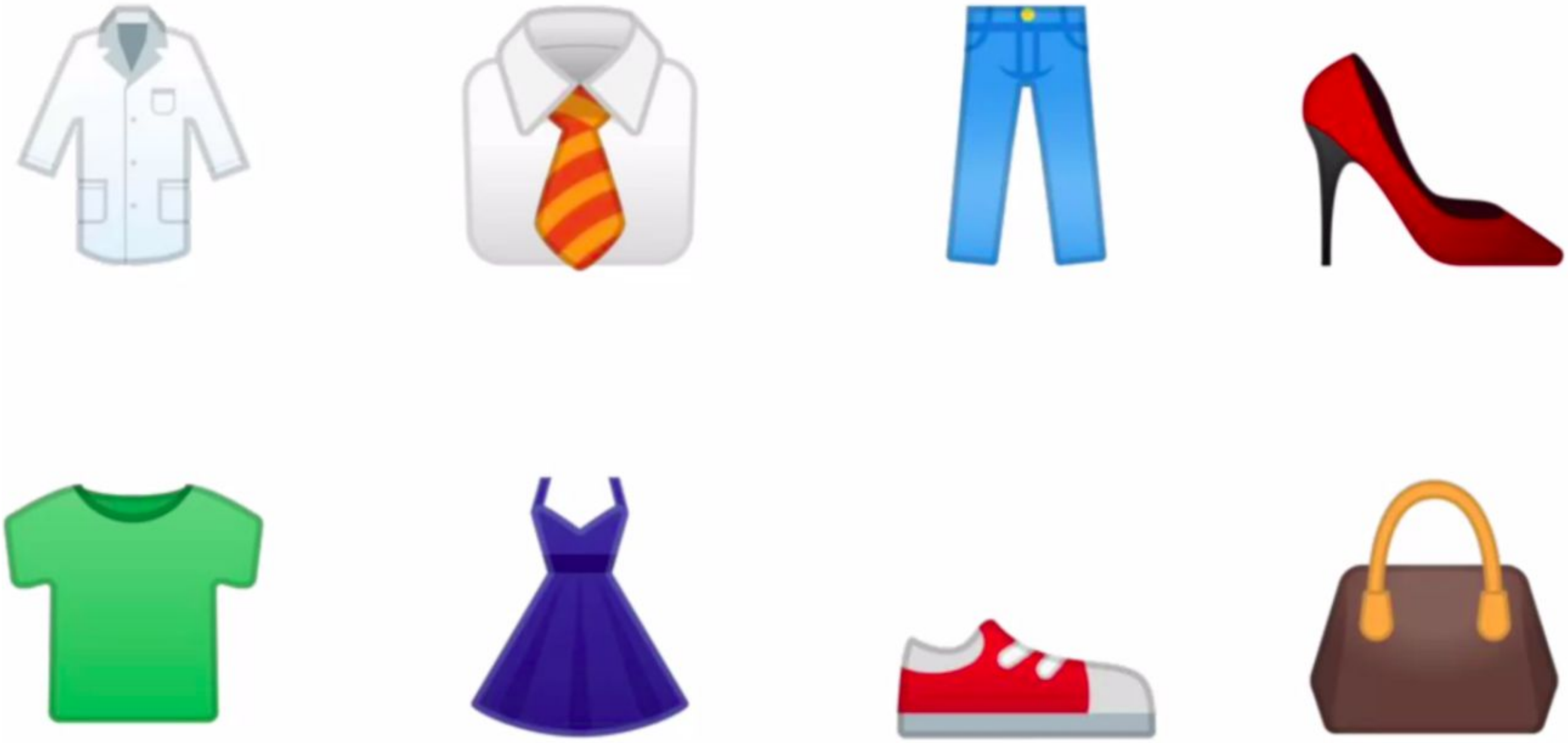
Good Fit/Robust



Overfitted

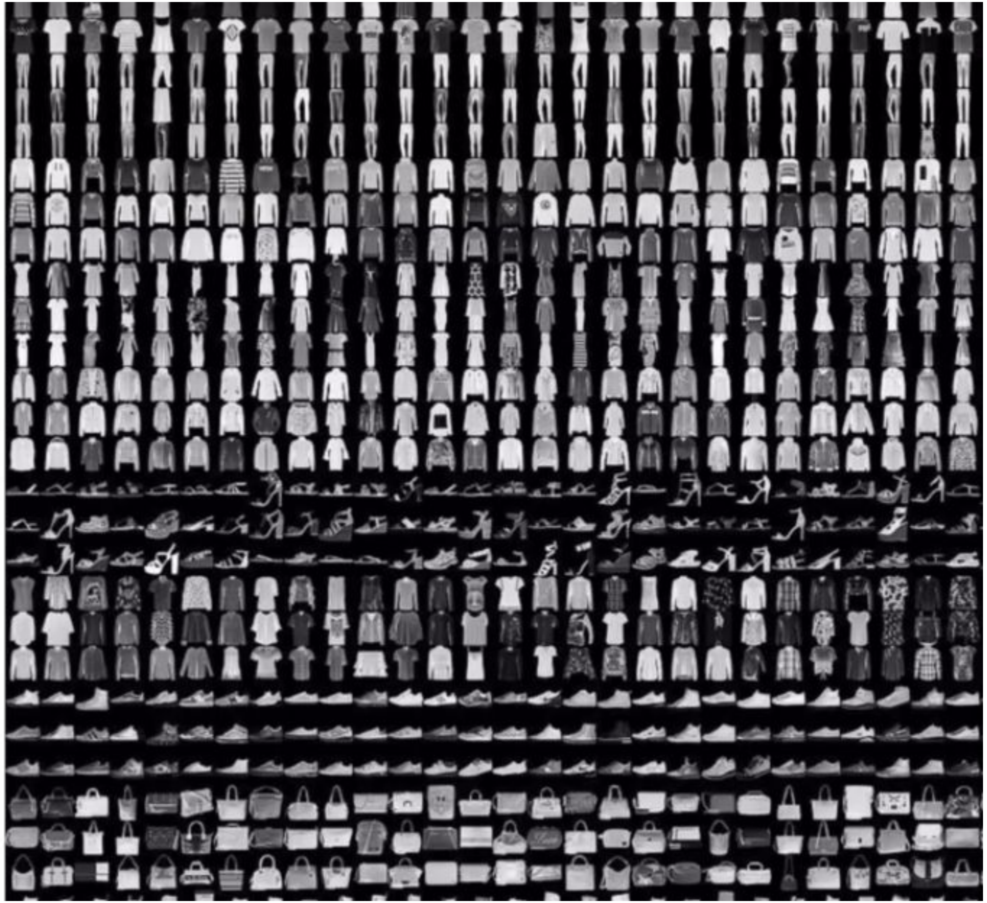
Part II

Computer Vision



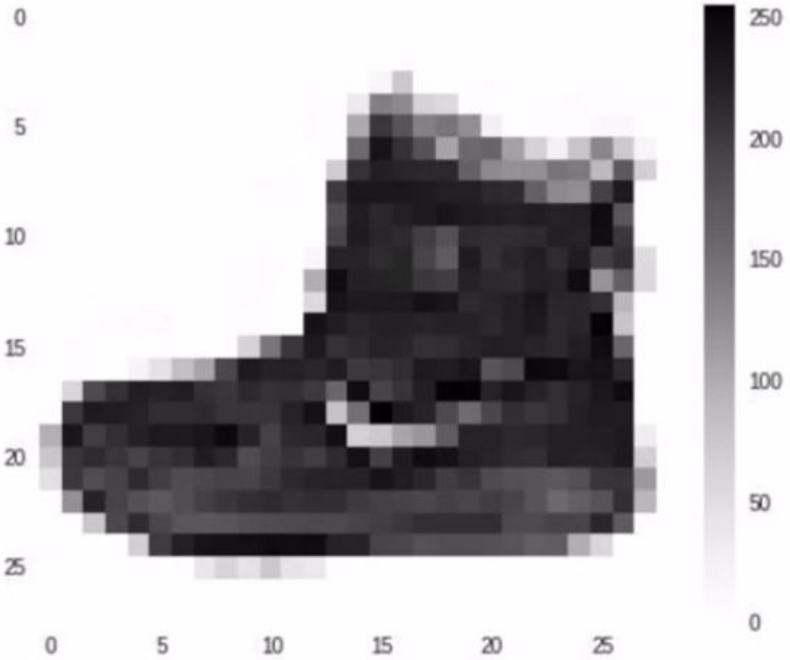
Fashion MNIST

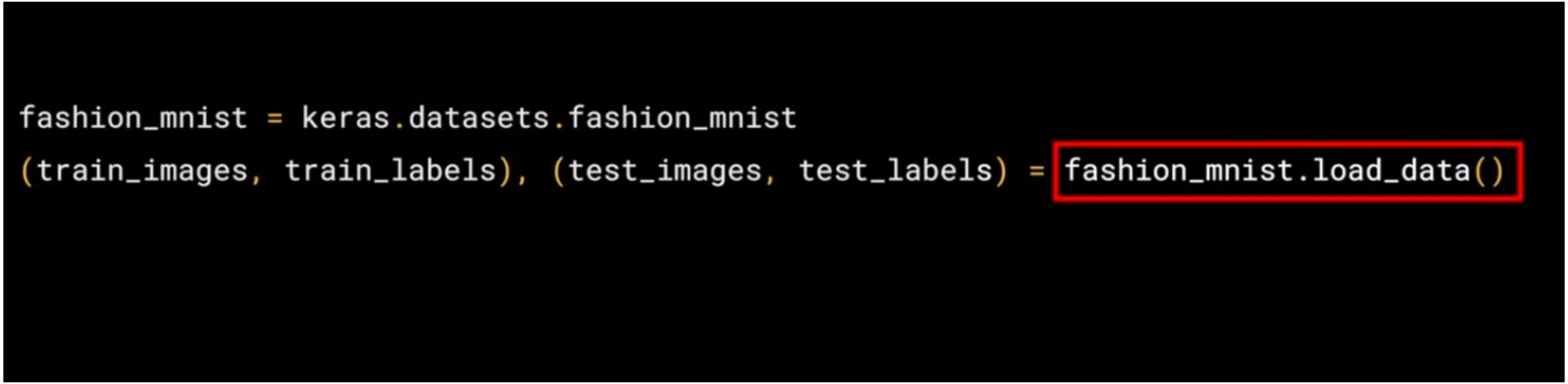
- 70k Images
- 10 Categories
- Images are 28x28
- Can train a neural net!



Fashion MNIST

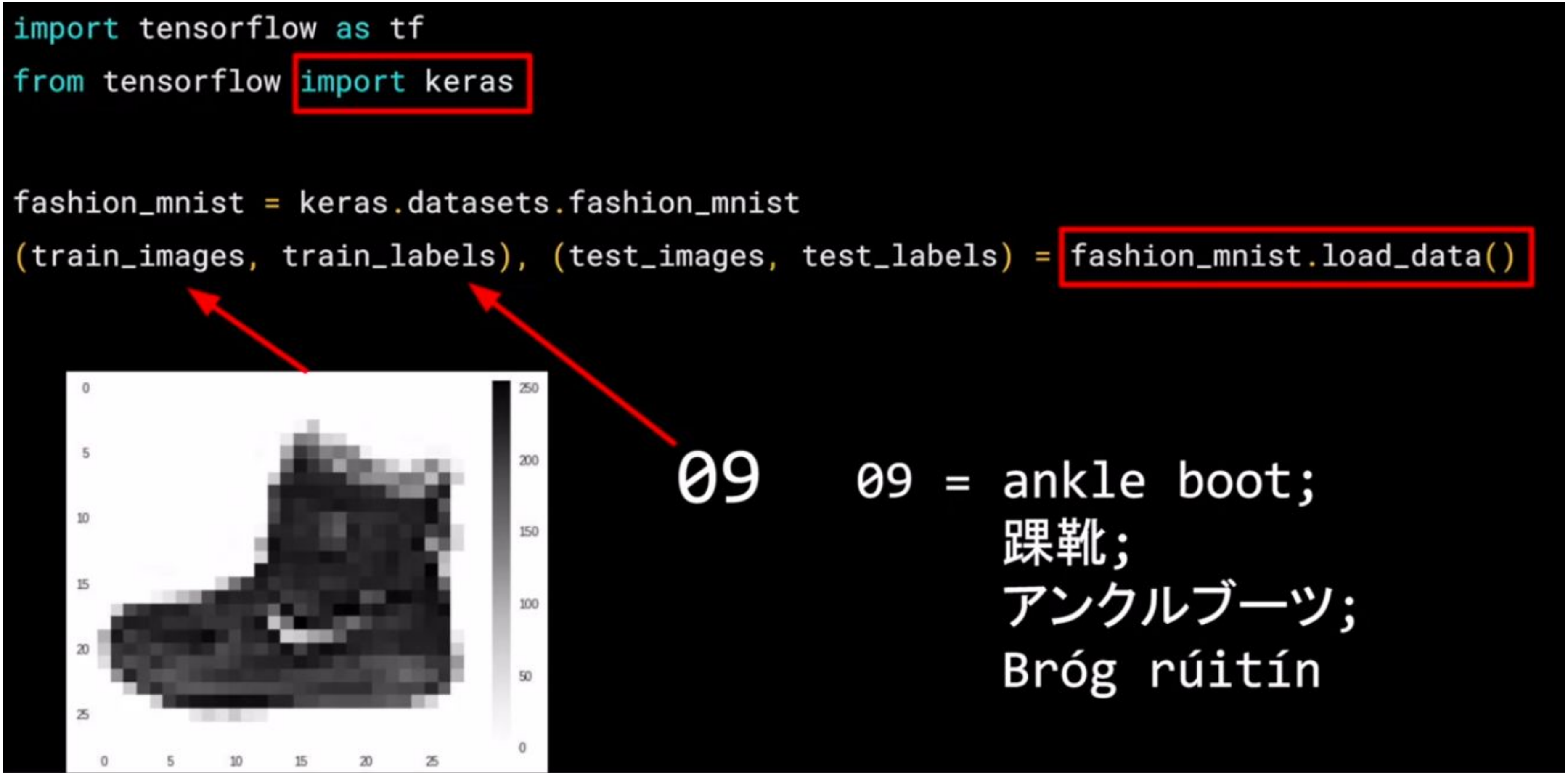
- 70k Images
- 10 Categories
- Images are 28x28
- Can train a neural net!





```
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

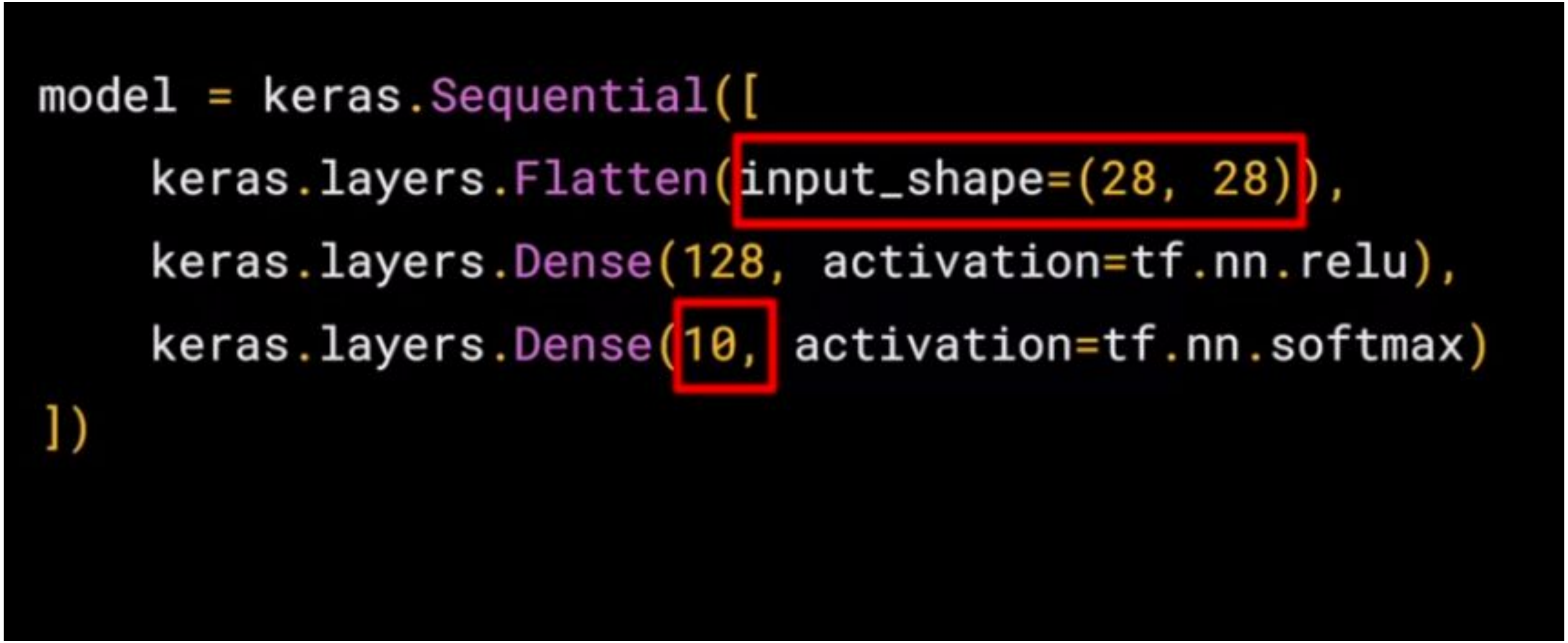
Train =
Test =



```
import tensorflow as tf
from tensorflow import keras


fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

09

09 = ankle boot;
踝靴;
アンクルブーツ;
Bróg rúitín



```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

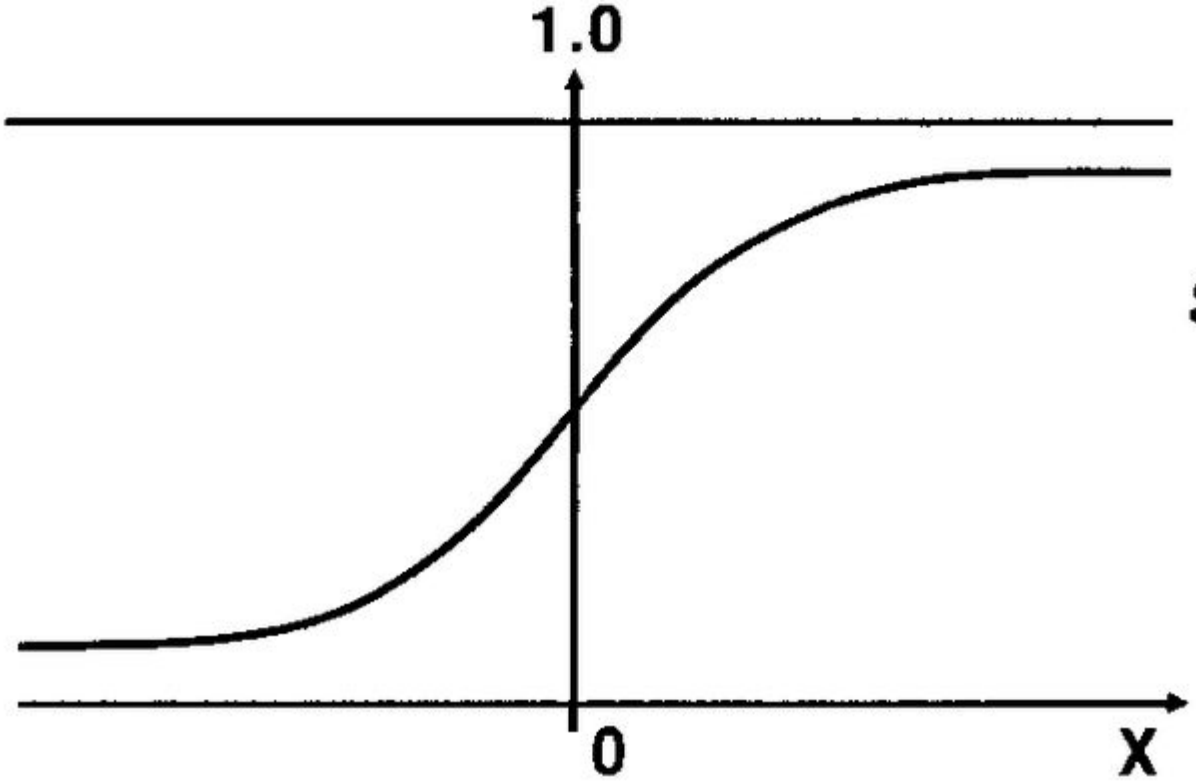
W0*X0+W1*X1+W2*X2+.......+WN*XN = 9-----> Label

Confused???

Optimizer
Loss Function
Activation Function

Anythings Else??

Activation Function



Sigmoid

$$f(x) = \frac{1}{1+e^{-x}}$$

Machine Learning Week 3

Andrew Ng

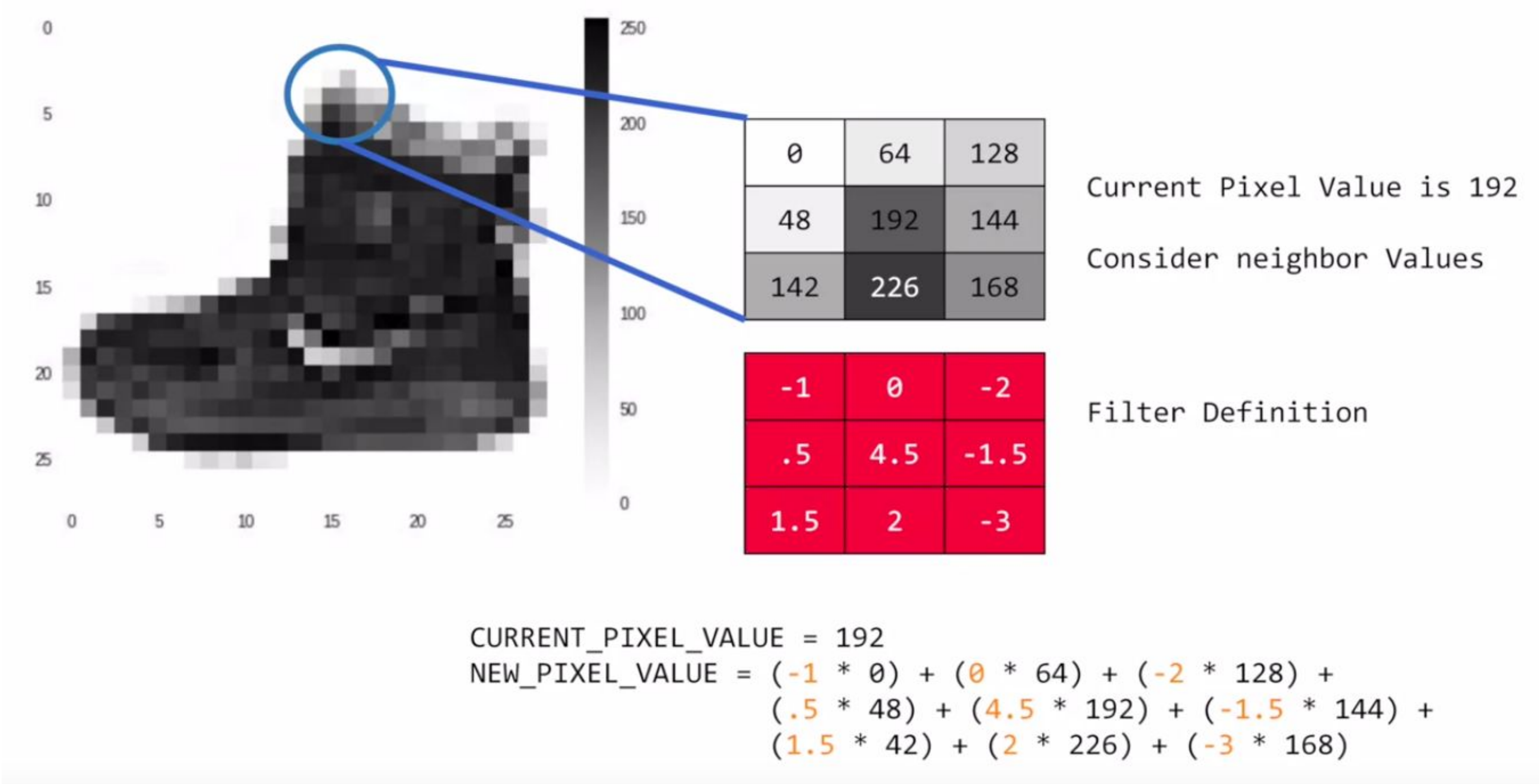
$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Binary Cross-Entropy / Log Loss

Part III

Convolutional Neural Network

Convolution



0	64	128
48	192	144
142	226	168

Current Pixel Value is 192

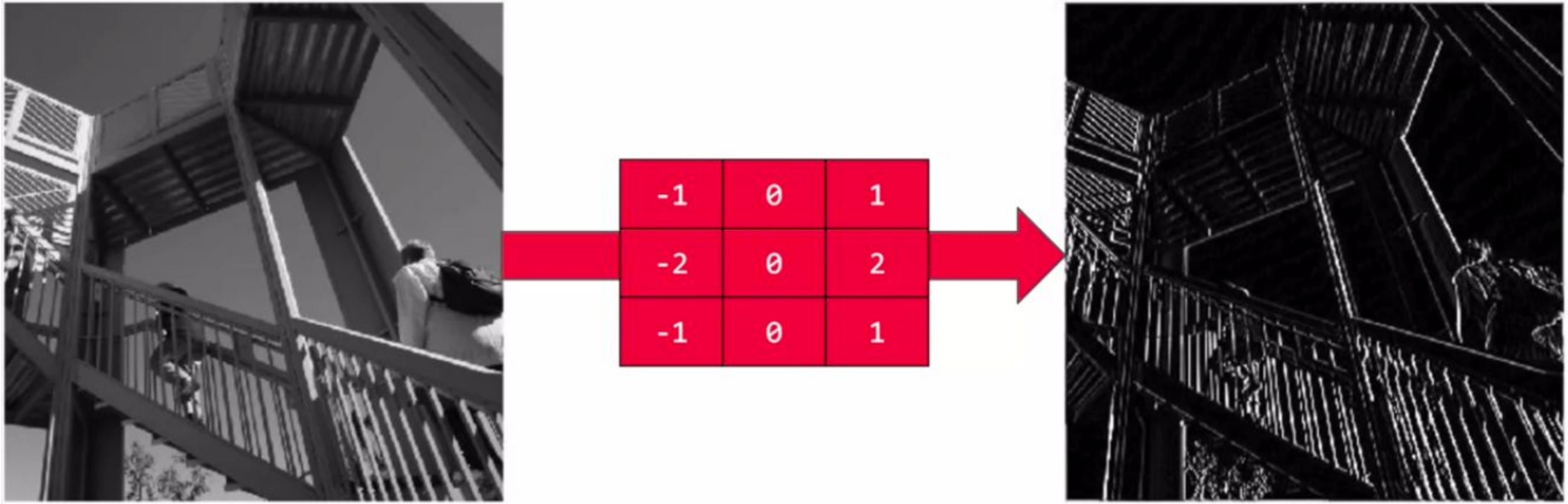
Consider neighbor Values

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

Filter Definition

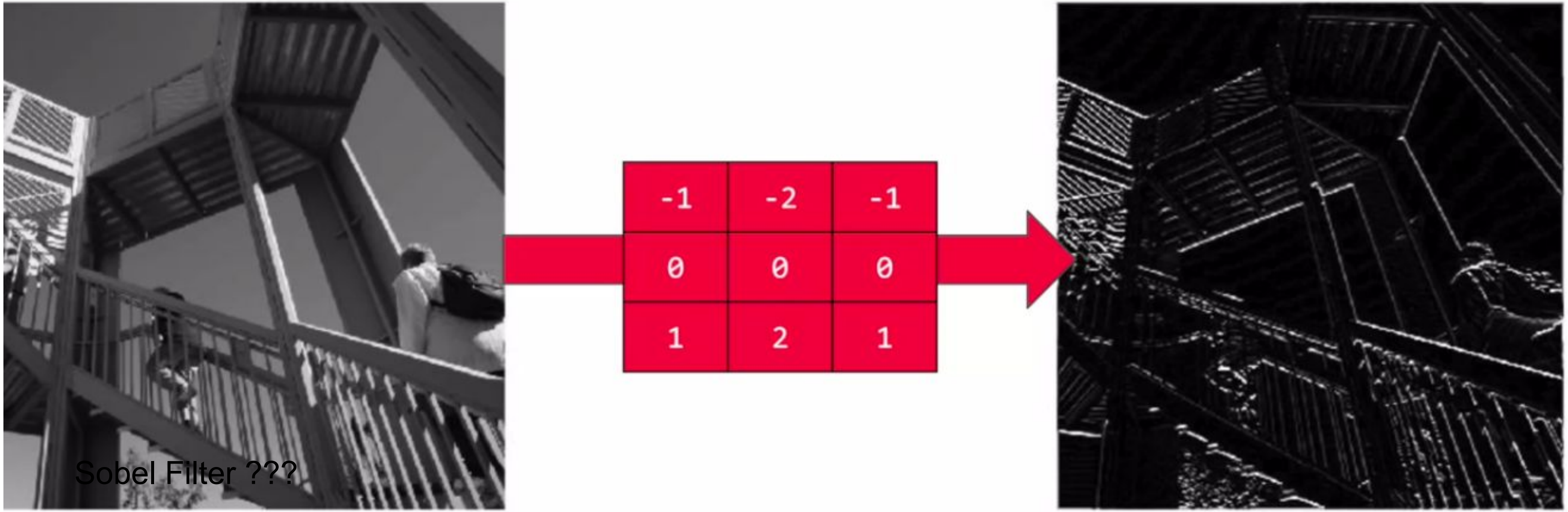
```
CURRENT_PIXEL_VALUE = 192
NEW_PIXEL_VALUE = (-1 * 0) + (0 * 64) + (-2 * 128) +
                  (.5 * 48) + (4.5 * 192) + (-1.5 * 144) +
                  (1.5 * 42) + (2 * 226) + (-3 * 168)
```

Convolution



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

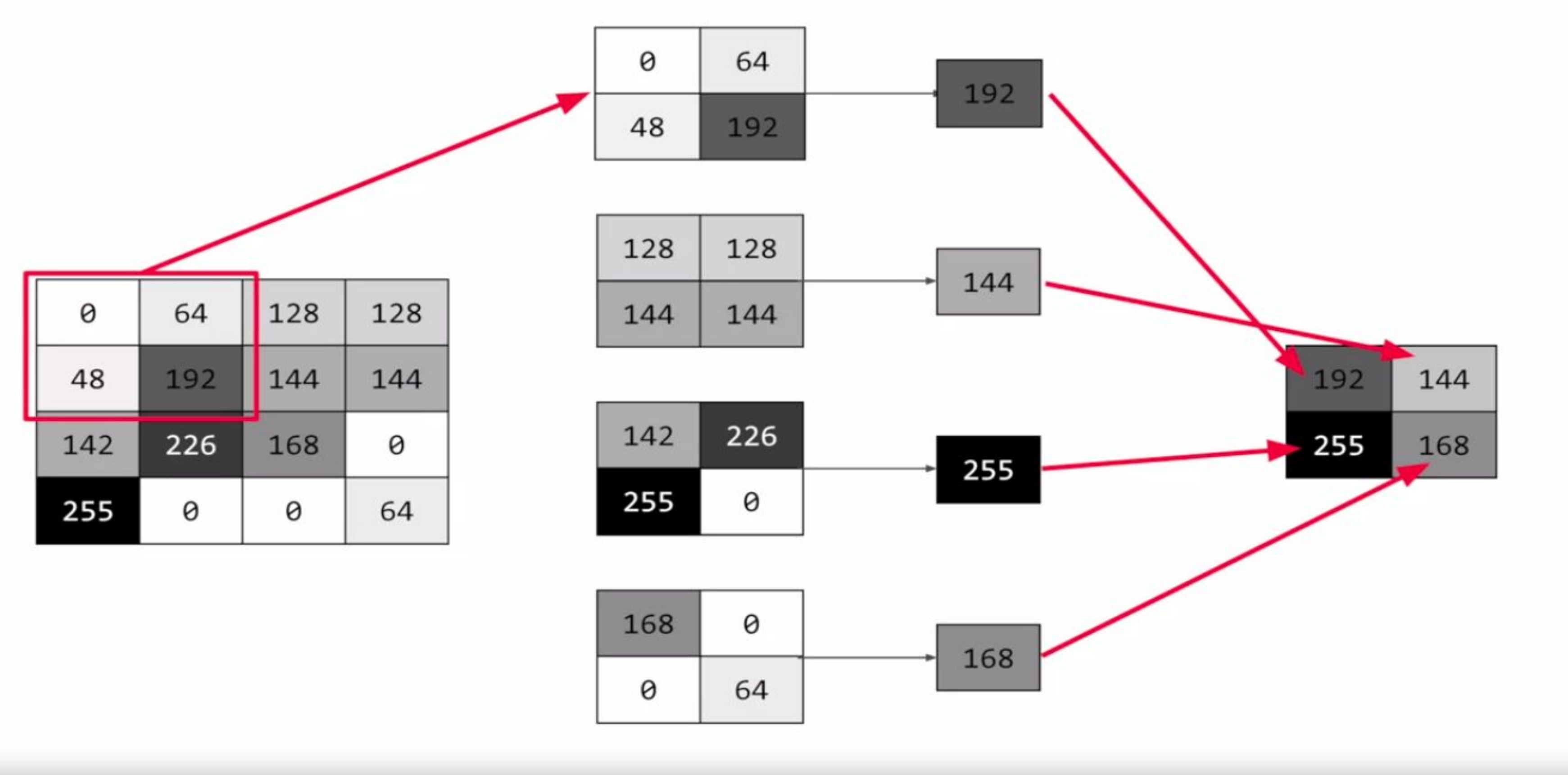
Convolution



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

Sobel Filter ???

Pooling



0	64	128	128
48	192	144	144
142	226	168	0
255	0	0	64

0	64
48	192

192

128	128
144	144

144

142	226
255	0

255

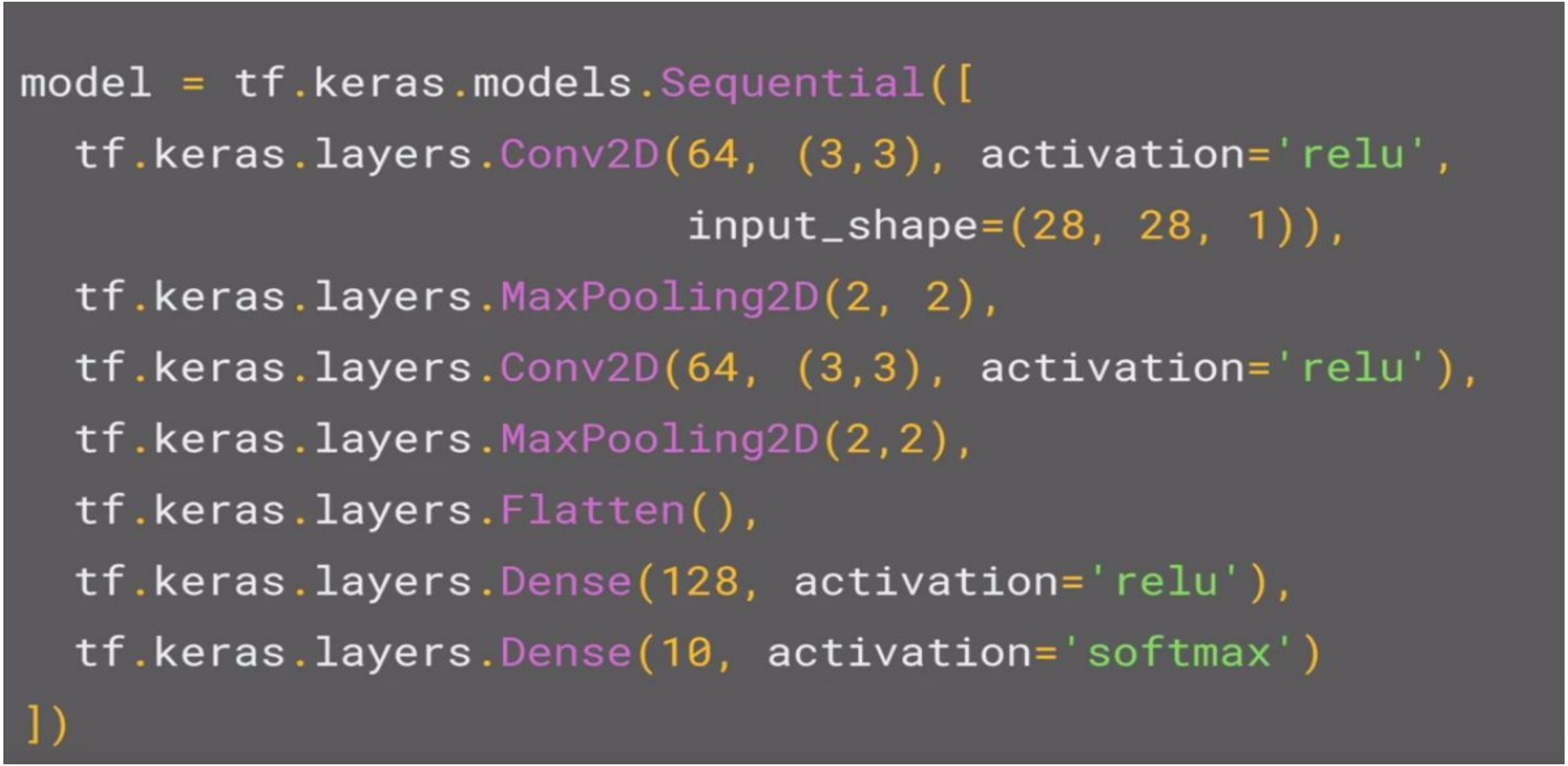
168	0
0	64

168

192	144
255	168

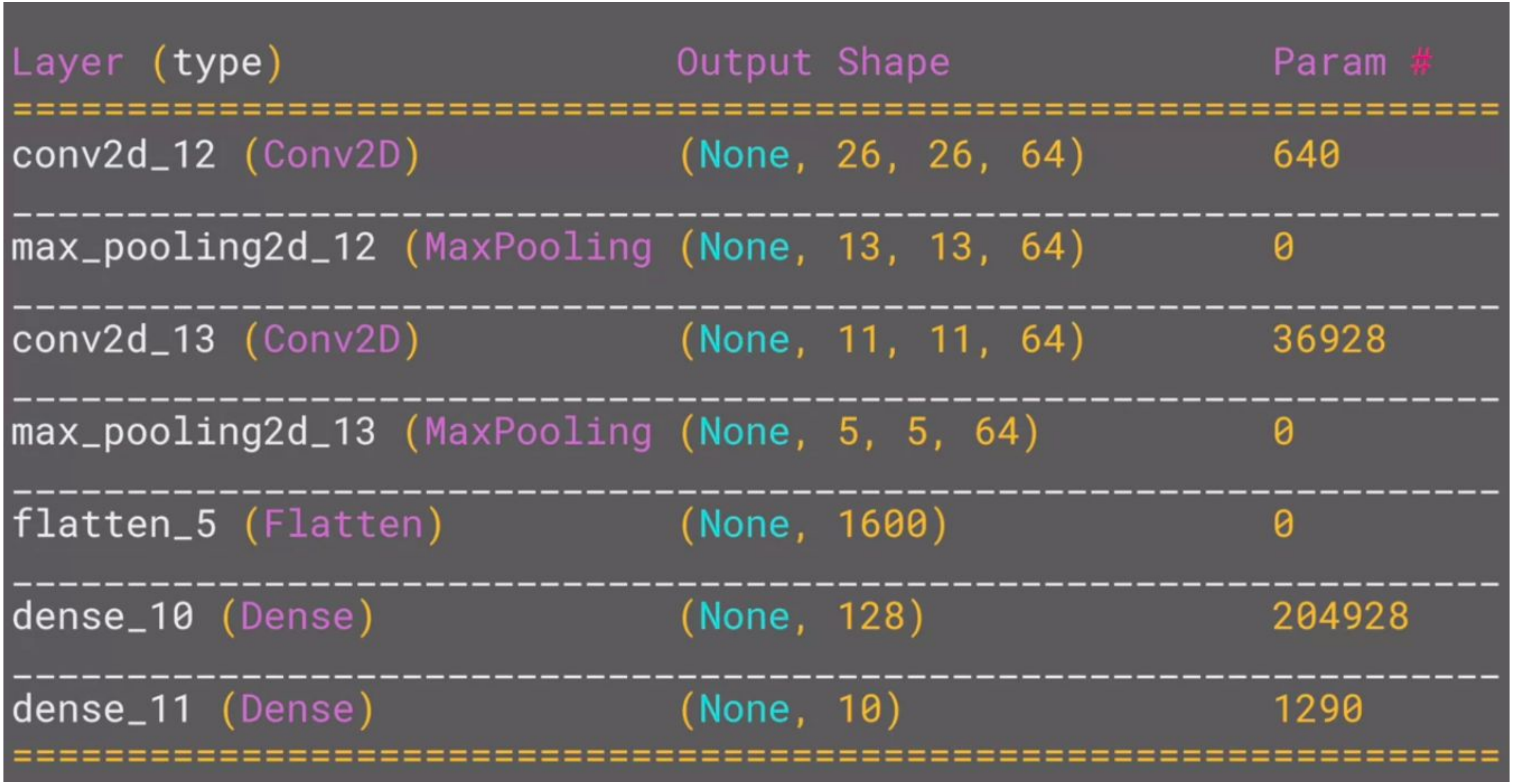
Reducing the size

Code

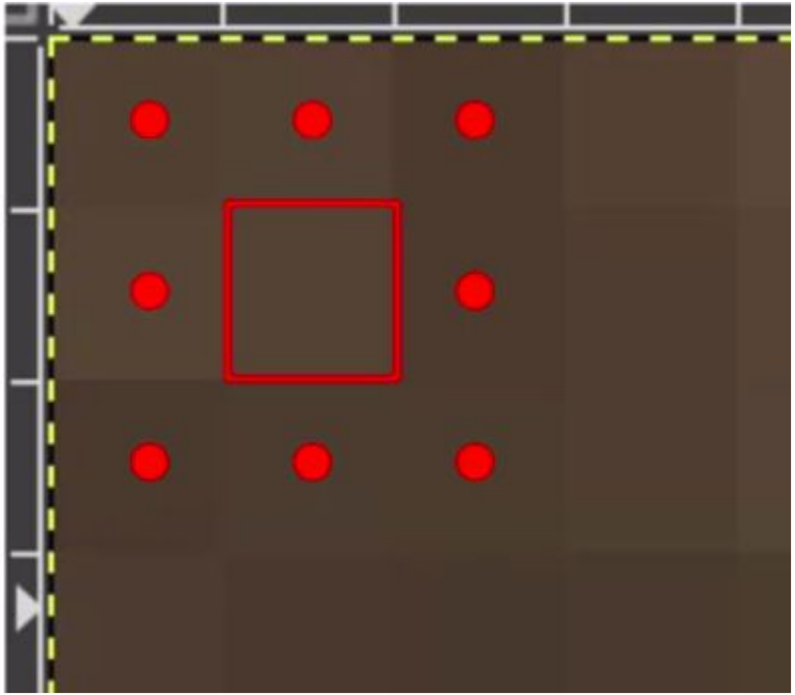


```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(64, (3,3), activation='relu',
                         input_shape=(28, 28, 1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

model.summary()



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 26, 26, 64)        640
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 11, 11, 64)        36928
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 5, 5, 64)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_10 (Dense)             (None, 128)               204928
_________________________________________________________________
dense_11 (Dense)             (None, 10)                1290
=================================================================
```



X-2
Y-2 in output

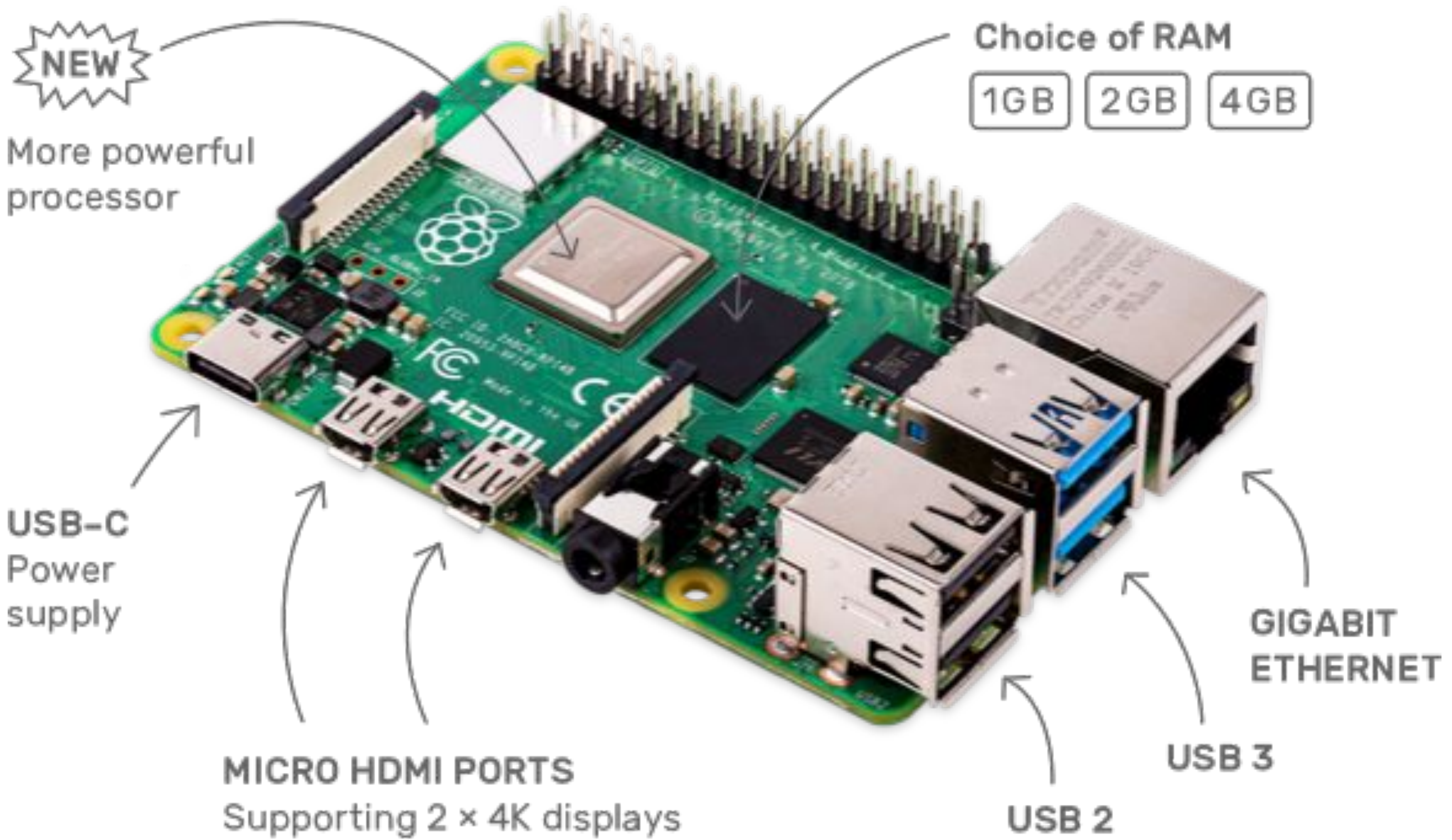
28x28, 64 filters,---> 26x26,64 filters

Original images are convolved with 64 filters, so we got 64 different images

Part IV

Raspberry Pi

https://medium.com/analytics-vidhya/quick-setup-instructions-for-installing-pytorch-and-fastai-on-raspberry-pi-4-5ffbe45e0ae3


NEW
More powerful processor
Choice of RAM
1GB
2GB
4GB
USB-C
Power supply
MICRO HDMI PORTS
Supporting 2 × 4K displays
USB 2
USB 3
GIGABIT ETHERNET

```
sudo apt install libopenblas-dev libblas-dev m4 cmake cython python3-dev python3-yaml python3-setuptools python3-wheel python3-pillow python3-numpy
```

```
sudo apt install libatlas3-base
```

```
sudo pip3 install numpy
```

```
python3 -m pip install Pillow==6.1
```

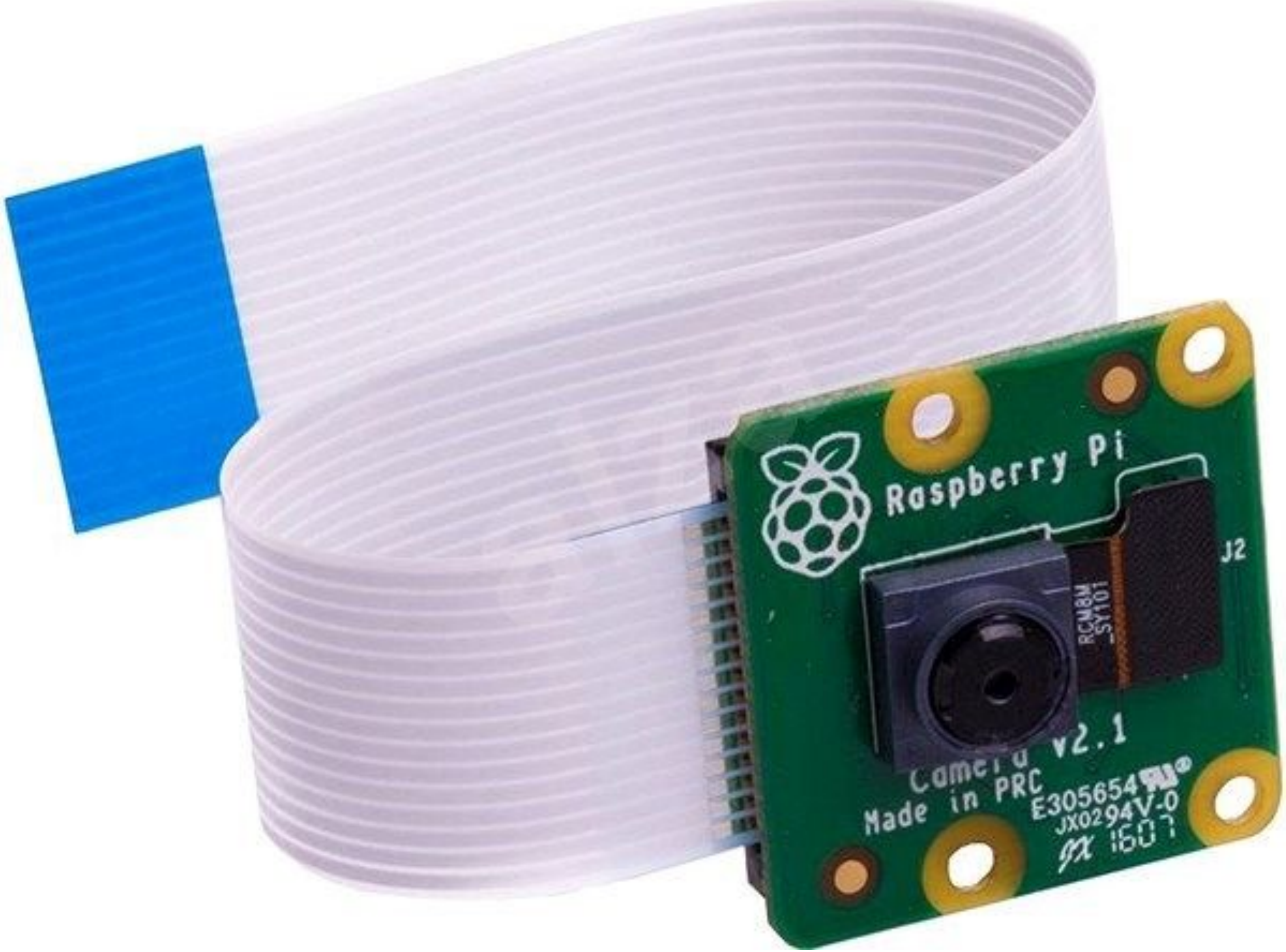
```
pip3 install torch-1.3.0a0+deadc27-cp37-cp37m-linux_armv7l.whl
```

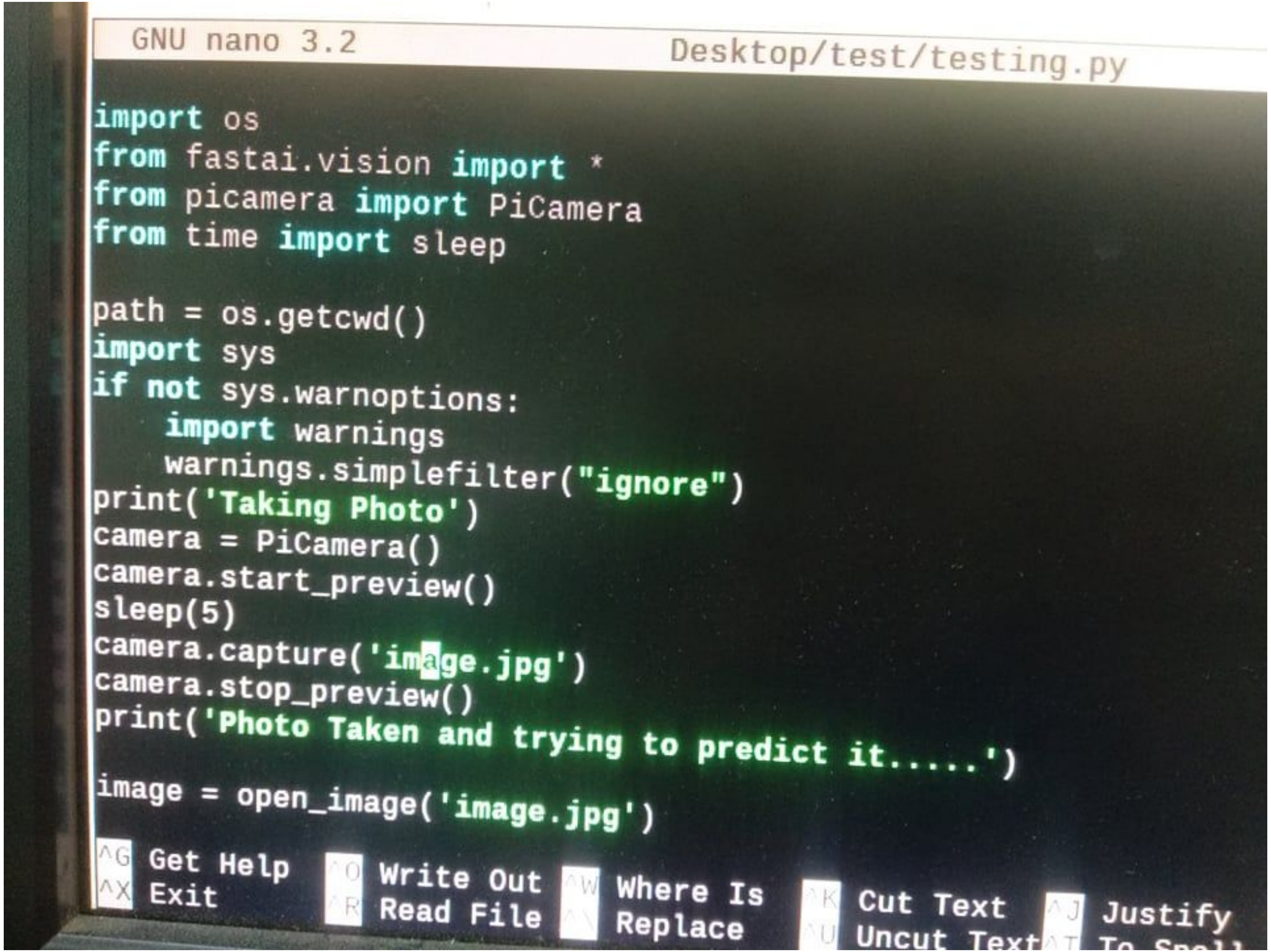
```
pip3 install torchvision-0.4.0a0+d31eafa-cp37-cp37m-linux_armv7l.whl
```

```
pip3 install fastai --no-deps
```

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

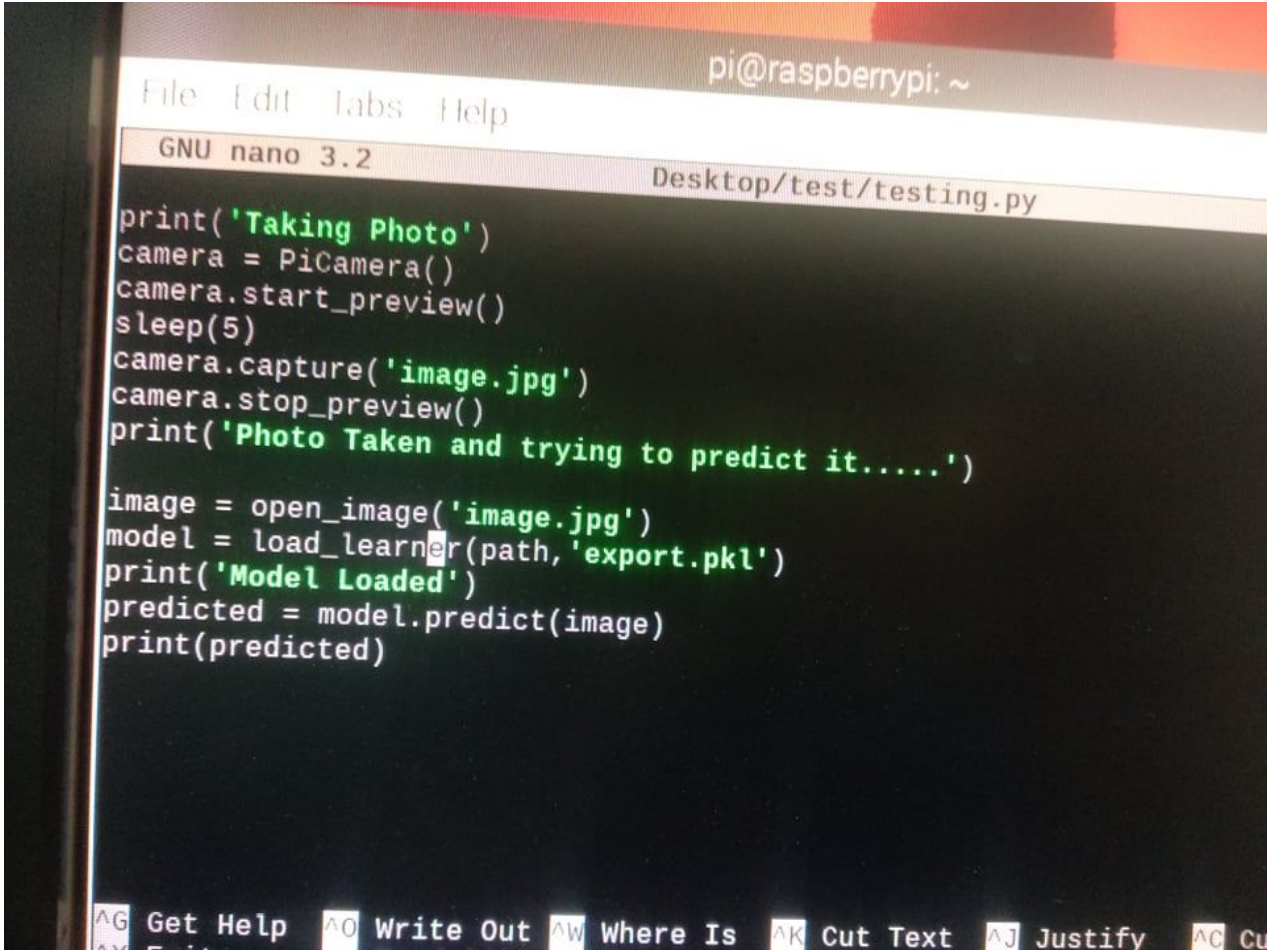
https://www.raspberrypi.org/products/raspberry-pi-3-model-b/


Raspberry Pi
J2
RCM8M
SY101
Camera V2.1
Made in PRC
E305654
JX0294V-0
1607



```
import os
from fastai.vision import *
from picamera import PiCamera
from time import sleep

path = os.getcwd()
import sys
if not sys.warnoptions:
    import warnings
    warnings.simplefilter("ignore")
print('Taking Photo')
camera = PiCamera()
camera.start_preview()
sleep(5)
camera.capture('image.jpg')
camera.stop_preview()
print('Photo Taken and trying to predict it.....')

image = open_image('image.jpg')
```

Thank you, my students



```
  GNU nano 3.2                    Desktop/test/testing.py

print('Taking Photo')
camera = PiCamera()
camera.start_preview()
sleep(5)
camera.capture('image.jpg')
camera.stop_preview()
print('Photo Taken and trying to predict it.....')

image = open_image('image.jpg')
model = load_learner(path,'export.pkl')
print('Model Loaded')
predicted = model.predict(image)
print(predicted)
```

Thank you so much,

Faculty Members, Student Volunteers

Thanks Online Resources