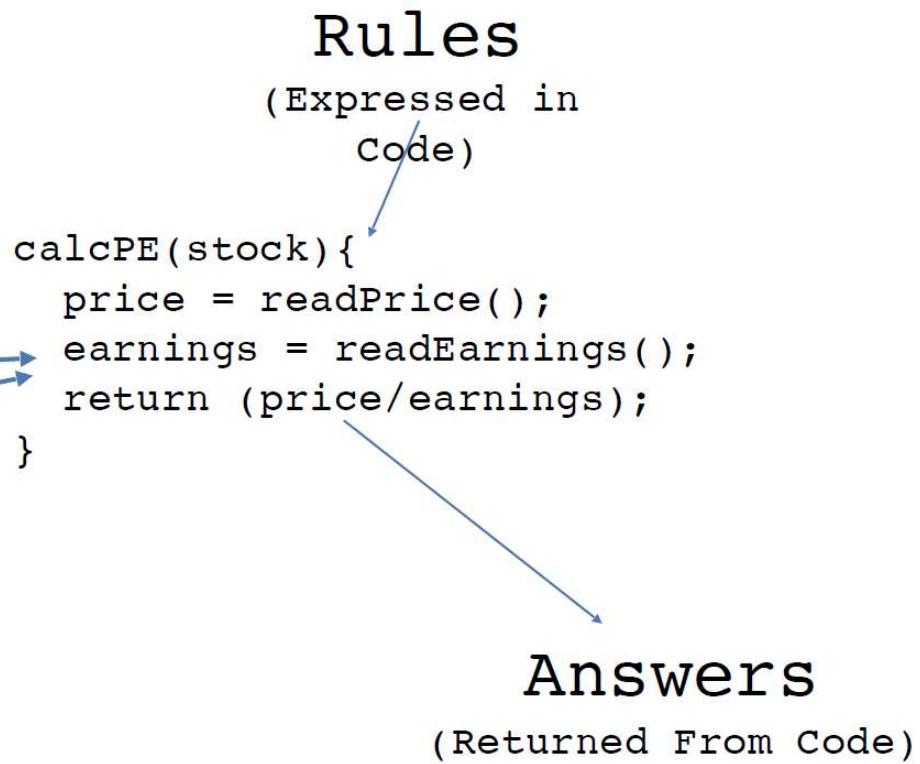
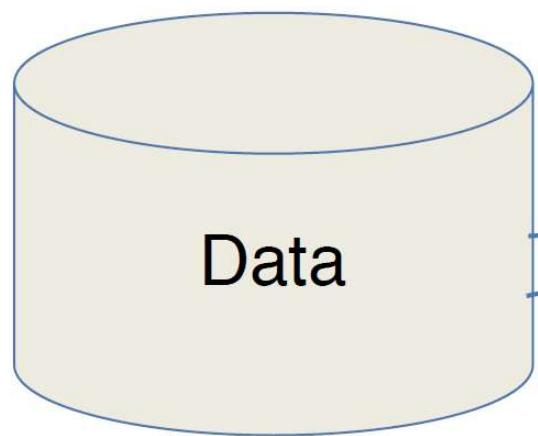


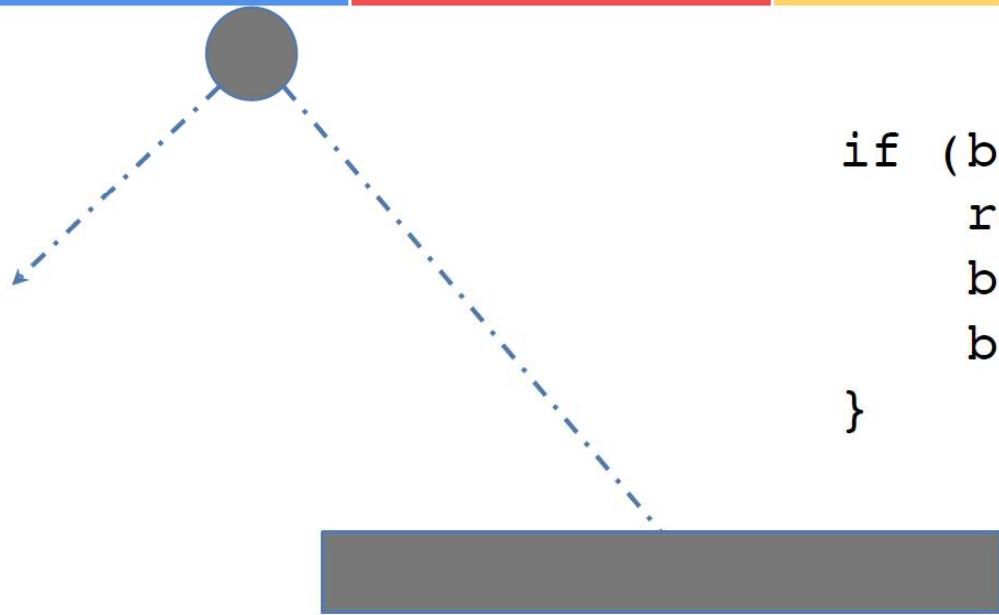
Copyright Notice

These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>





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





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



01010010101001010
10100101010100101
11010100101010010
10100101010010101
00101010
Label =
WALKING



10101001010010101
01010101001001001
0001001001111010
10111110101001001
1110101011110111
Label =
RUNNING



1001010011110101
01110101011101010
11101010101111010
10101111111100011
11010101
Label = BIKING

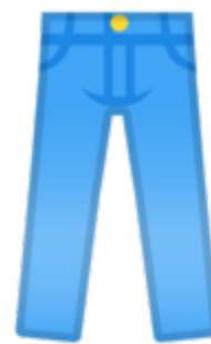


1111111110100111
0100111101011111
01010101110101010
10111010101010101
00111110
Label =
GOLFING (Sort
of)

X = -1, 0, 1, 2, 3, 4

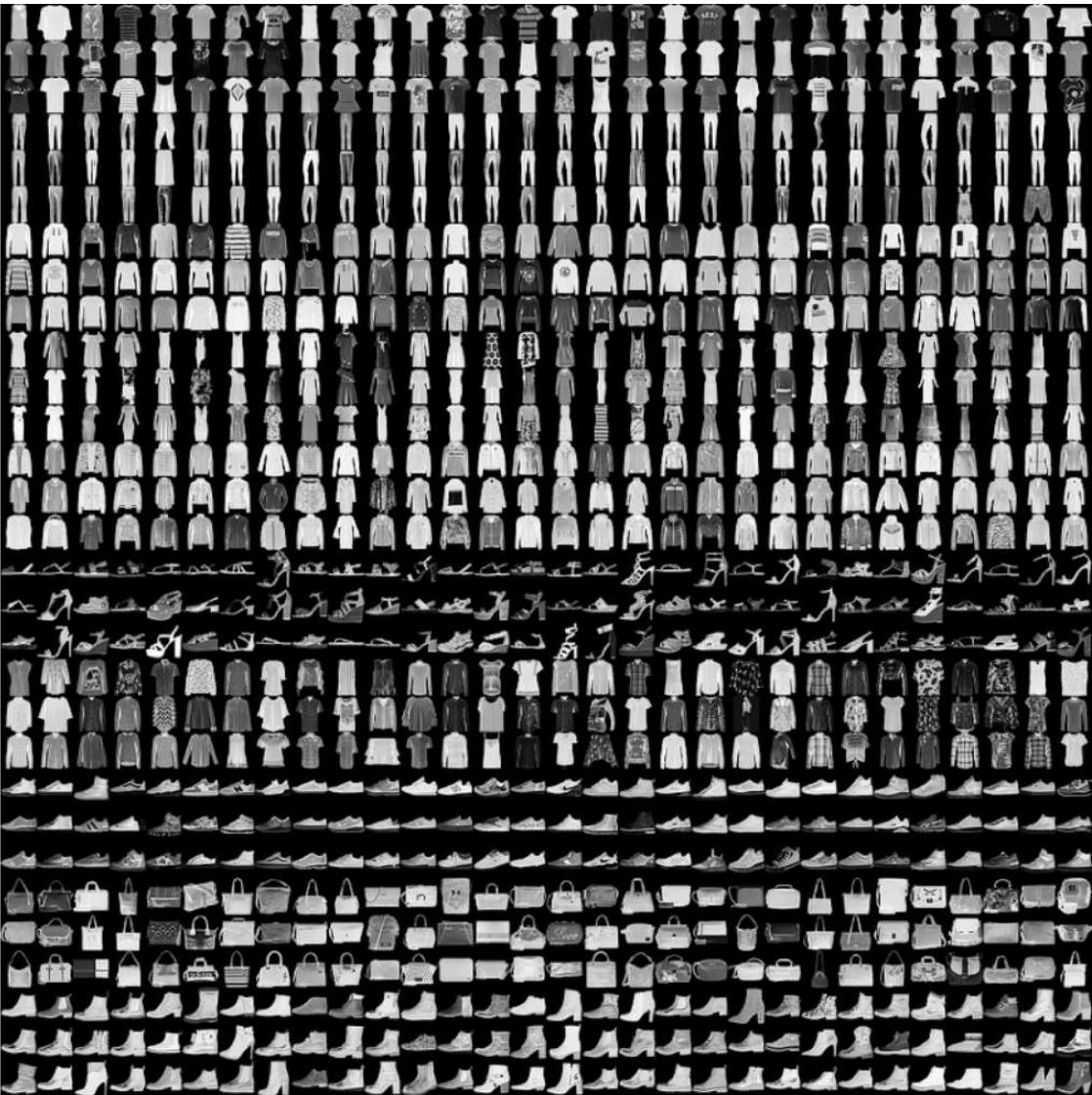
Y = -3, -1, 1, 3, 5, 7

```
model = tf.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]))
```



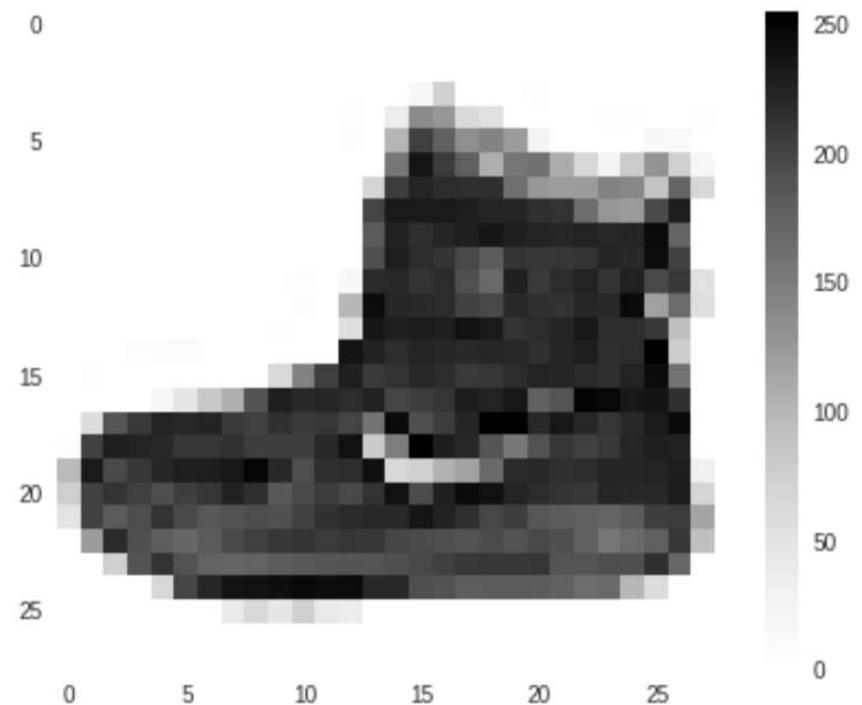
Fashion MNIST

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

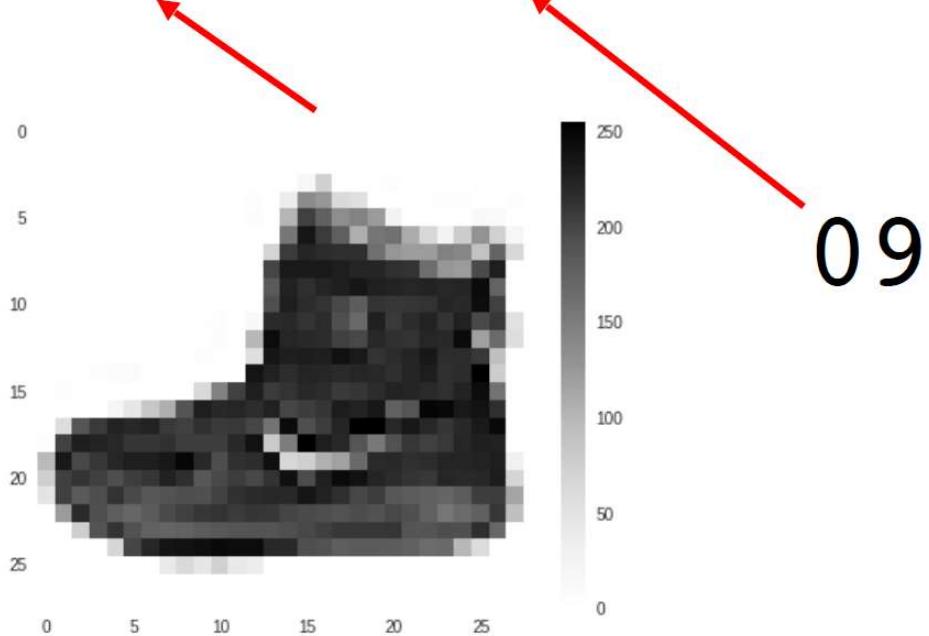


Fashion MNIST

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



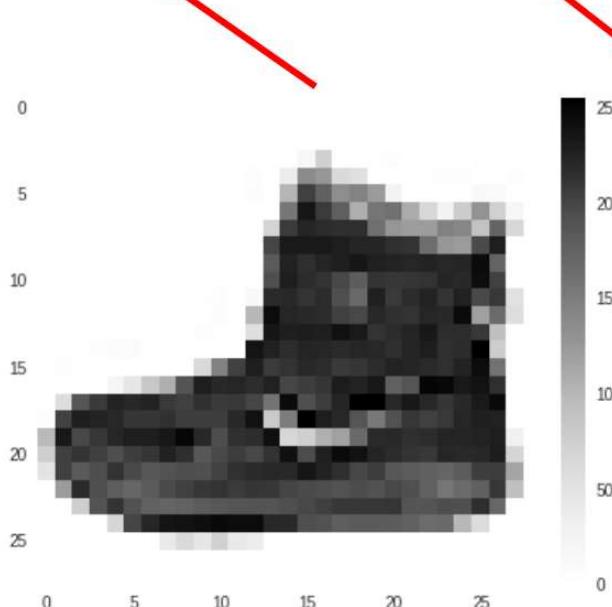
```
fashion_mnist = tf.keras.datasets.fashion_mnist  
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```



09

```
import tensorflow as tf  
from tensorflow import keras
```

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



09

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

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

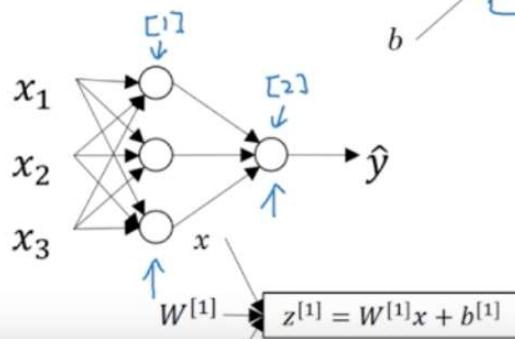
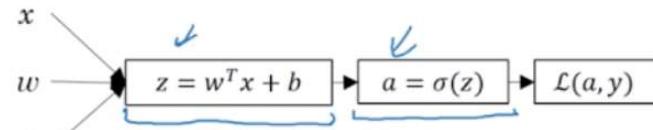
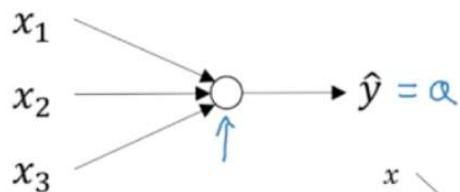


$$w_0 \times 0 + w_1 \times 1 + w_2 \times 2 + \dots + w_N \times N = 9$$



<https://youtu.be/fXOsFF95ifk>

What is a Neural Network?



1:55 / 4:26

CC RD Ar dev Ng

Neural Network Overview (C1W3L01)

11,067 views

43 0 SHARE SAVE ...

Neural Networks and Deep Learning (Course 1 of Deeplearning.ai - 25 / 43

- ▶ One hidden layer Neural Network 4:27
- 26 Neural Network Representations (C1W3L02) 5:15
- 27 Computing Neural Network Output (C1W3L03) 9:58
- 28 Vectorizing Across Multiple Examples (C1W3L04) 9:06
- 29 Explanation For Vectorized Implementation (C1W3L05) 7:38
- 30 Activation Functions (C1W3L06) 10:57

Why Non-linear Activation Functions



Complete User Registration system using PHP and MySQL...

Awa Melvine
5.7M views

32:43

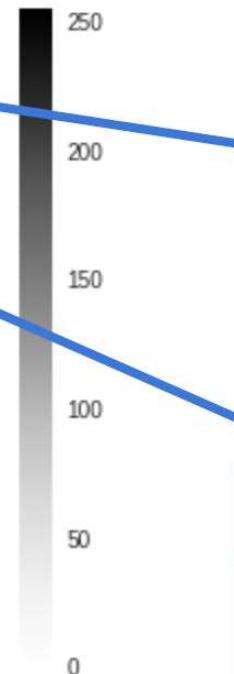
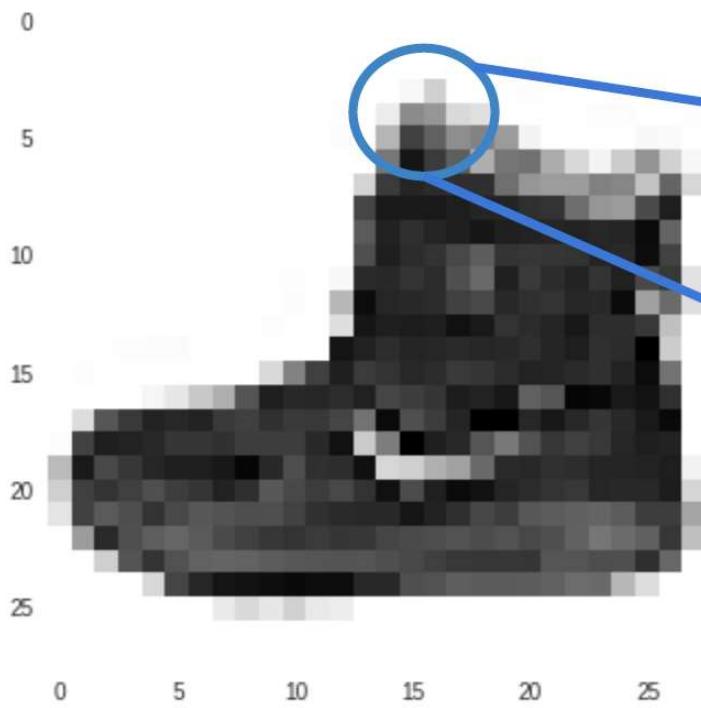
```
mnist = tf.keras.datasets.fashion_mnist
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
training_images=training_images/255.0
test_images=test_images/255.0
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer=tf.optimizers.Adam(), loss='sparse_categorical_crossentropy')
model.fit(training_images, training_labels, epochs=5)
```

```
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('loss')<0.4):
            print("\nLoss is low so cancelling training!")
            self.model.stop_training = True
```

```
mnist = tf.keras.datasets.fashion_mnist
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
training_images=training_images/255.0
test_images=test_images/255.0
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer=tf.optimizers.Adam(), loss='sparse_categorical_crossentropy')
model.fit(training_images, training_labels, epochs=5)
```

```
callbacks = myCallback()
```

```
mnist = tf.keras.datasets.fashion_mnist
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
training_images=training_images/255.0
test_images=test_images/255.0
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer=tf.optimizers.Adam(), loss='sparse_categorical_crossentropy')
model.fit(training_images, training_labels, epochs=5)
```



0	64	128
48	192	144
142	226	168

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

Current Pixel Value is 192

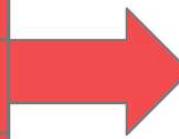
Consider neighbor values

Filter Definition

$$\begin{aligned}
 \text{CURRENT_PIXEL_VALUE} &= 192 \\
 \text{NEW_PIXEL_VALUE} &= (-1 * 0) + (0 * 64) + (-2 * 128) + \\
 &\quad (.5 * 48) + (4.5 * 192) + (-1.5 * 144) \\
 &+ \\
 &\quad (1.5 * 142) + (2 * 226) + (-3 * 168)
 \end{aligned}$$



$$\begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

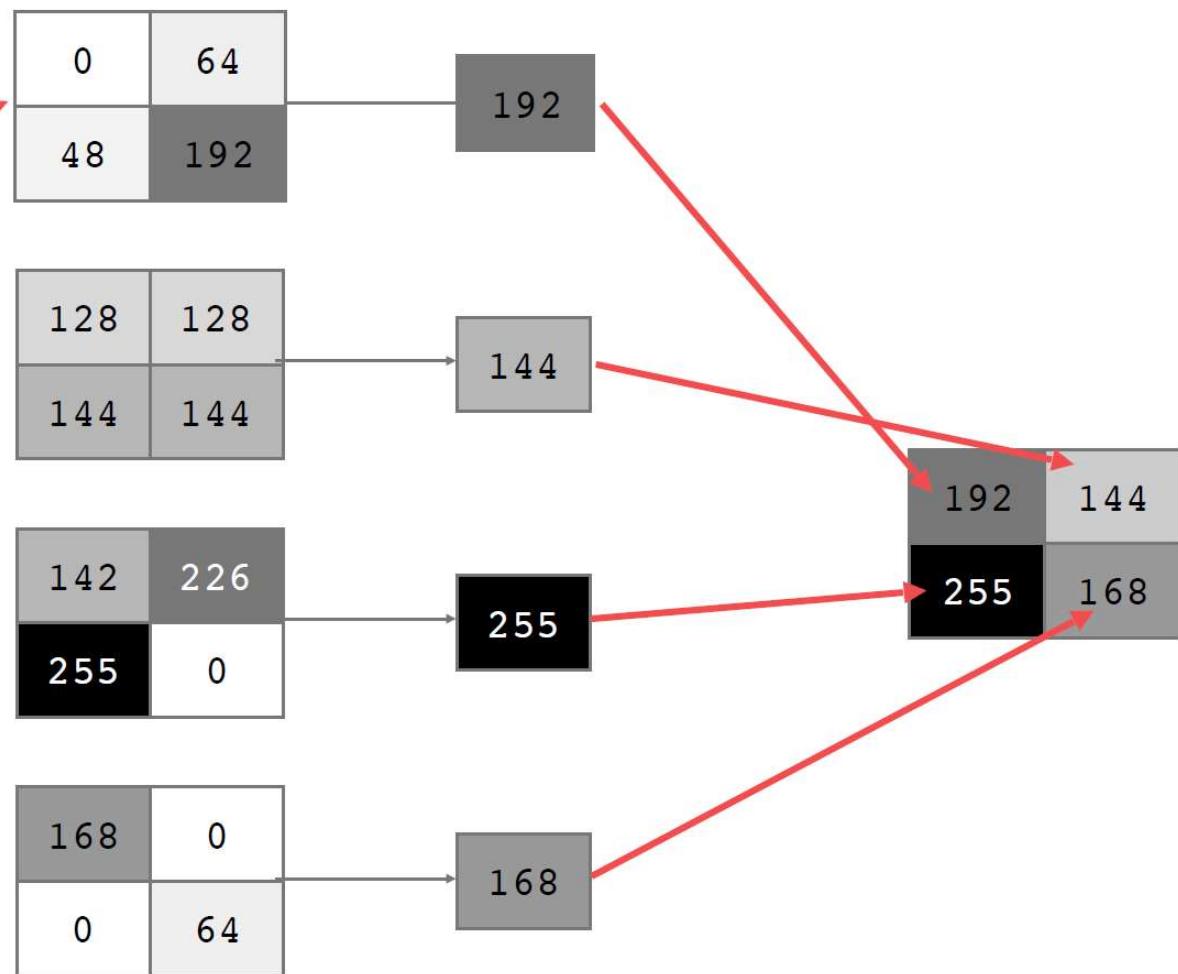




$$\begin{array}{ccc} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{array}$$



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



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

```
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')
])
```

Computer Vision Problems

Image Classification



Cat? (0/1)

Neural Style Transfer



Object detection



▶ PLAY ALL

Convolutional Neural Networks (Course 4 of the Deep Learning Specialization)

42 videos • 415,722 views • Last updated on Nov 7, 2017



Deeplearning.ai

SUBSCRIBE 28K

<https://bit.ly/2UGa7uH>

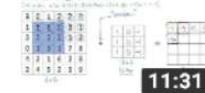
Computer Vision Problems

1

C4W1L01 Computer Vision

Deeplearning.ai

Vertical edge detection



11:31

C4W1L02 Edge Detection Examples

Deeplearning.ai

Virtual and Horizontal Edge Detection



7:58

C4W1L03 More Edge Detection

Deeplearning.ai

Padding

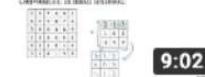


9:50

C4W1L04 Padding

Deeplearning.ai

Technical note on cross-correlation vs. convolution



9:02

C4W1L05 Strided Convolutions

Deeplearning.ai

Convolutions on RGB image



10:45

C4W1L06 Convolutions Over Volumes

Deeplearning.ai

```
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')
])
```

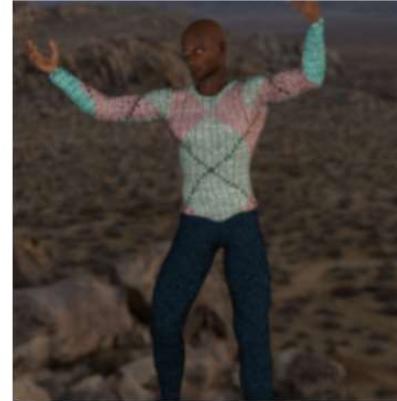
`model.summary()`

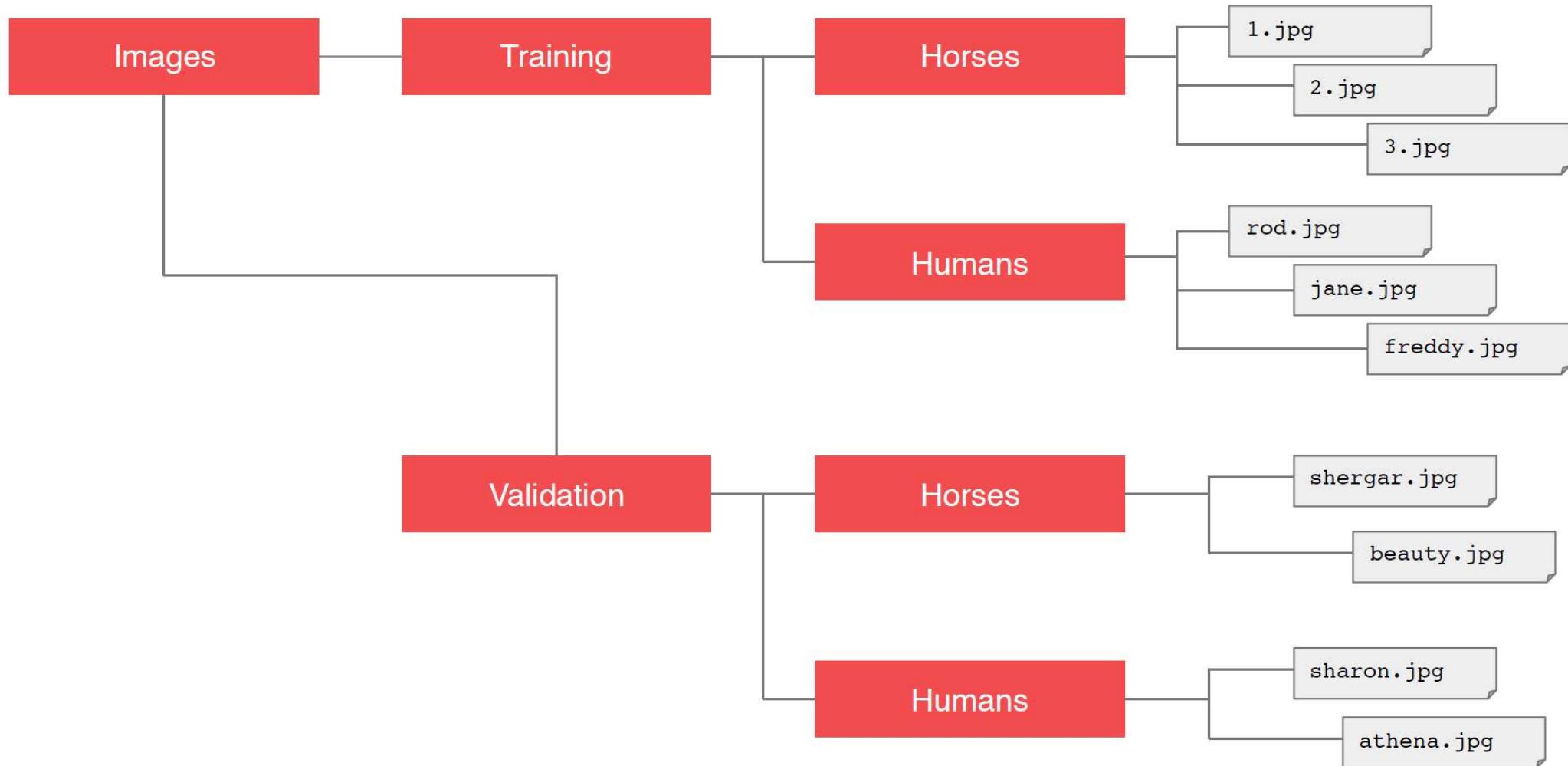
Layer (type)	Output Shape	Param #
=====		
conv2d_12 (Conv2D)	(None, 26, 26, 64)	640
max_pooling2d_12 (MaxPooling)	(None, 13, 13, 64)	0
conv2d_13 (Conv2D)	(None, 11, 11, 64)	36928
max_pooling2d_13 (MaxPooling)	(None, 5, 5, 64)	0
flatten_5 (Flatten)	(None, 1600)	0
dense_10 (Dense)	(None, 128)	204928
dense_11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





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





```
from tensorflow.keras.preprocessing.image  
import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(rescale=1./255)
```

```
train_generator = train_datagen.flow_from_directory(  
    train_dir,  
    target_size=(300, 300),  
    batch_size=128,  
    class_mode='binary')
```

```
test_datagen = ImageDataGenerator(rescale=1./255)

validation_generator = test_datagen.flow_from_directory(
    validation_dir,
    target_size=(300, 300),
    batch_size=32,
    class_mode='binary')
```

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu',
                         input_shape=(300, 300, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    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(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Layer (type)	Output Shape	Param #
=====		
conv2d_5 (Conv2D)	(None, 298, 298, 16)	448
max_pooling2d_5 (MaxPooling2)	(None, 149, 149, 16)	0
conv2d_6 (Conv2D)	(None, 147, 147, 32)	4640
max_pooling2d_6 (MaxPooling2)	(None, 73, 73, 32)	0
conv2d_7 (Conv2D)	(None, 71, 71, 64)	18496
max_pooling2d_7 (MaxPooling2)	(None, 35, 35, 64)	0
flatten_1 (Flatten)	(None, 78400)	0
dense_2 (Dense)	(None, 512)	40141312
dense_3 (Dense)	(None, 1)	513
=====		

Total params: 40,165,409

Trainable params: 40,165,409

Non-trainable params: 0

```
from tensorflow.keras.optimizers import RMSprop  
  
model.compile(loss='binary_crossentropy',  
               optimizer=RMSprop(lr=0.001),  
               metrics=['accuracy'])
```

<https://youtu.be/zLRB4oupj6g>



2.1.4 Gradient Descent in Practice II Learning Rate by Andrew Ng

```
history = model.fit(  
    train_generator,  
    steps_per_epoch=8,  
    epochs=15,  
    validation_data=validation_generator,  
    validation_steps=8,  
    verbose=2)
```

```
import numpy as np
from google.colab import files
from keras.preprocessing import image

uploaded = files.upload()

for fn in uploaded.keys():

    # predicting images
    path = '/content/' + fn
    img = image.load_img(path, target_size=(300, 300))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    images = np.vstack([x])
    classes = model.predict(images, batch_size=10)
    print(classes[0])
    if classes[0]>0.5:
        print(fn + " is a human")
    else:
        print(fn + " is a horse")
```