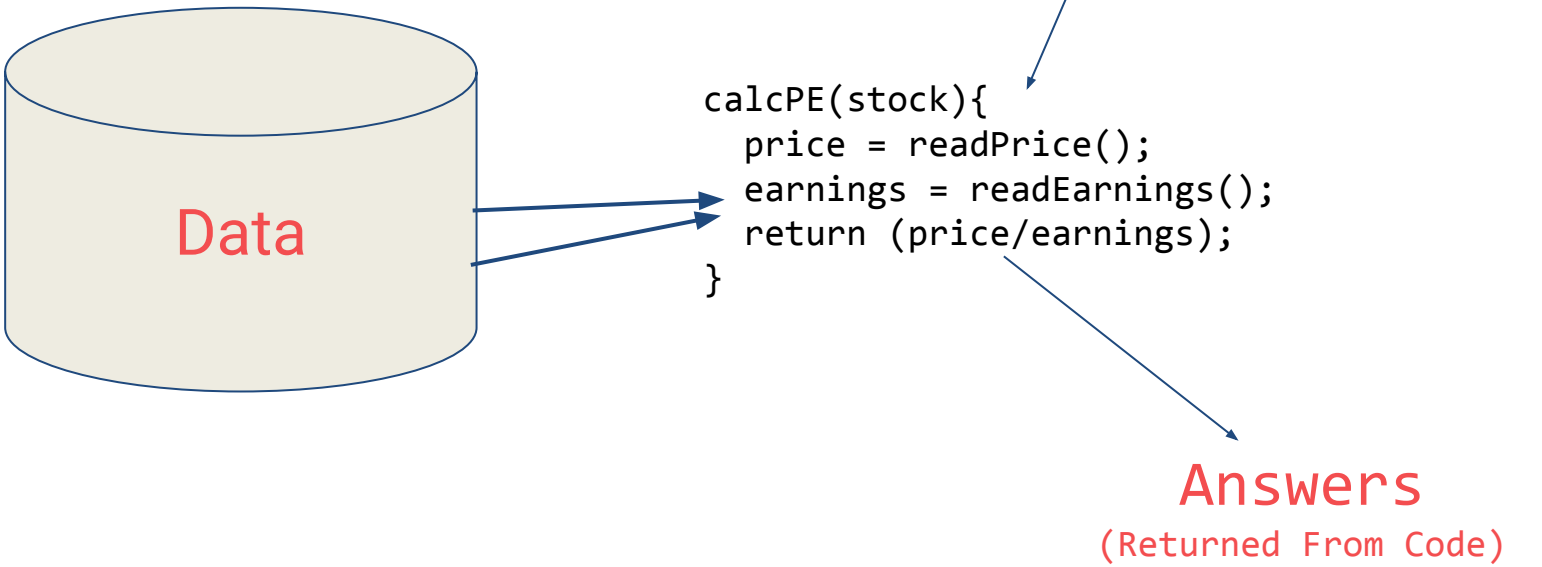
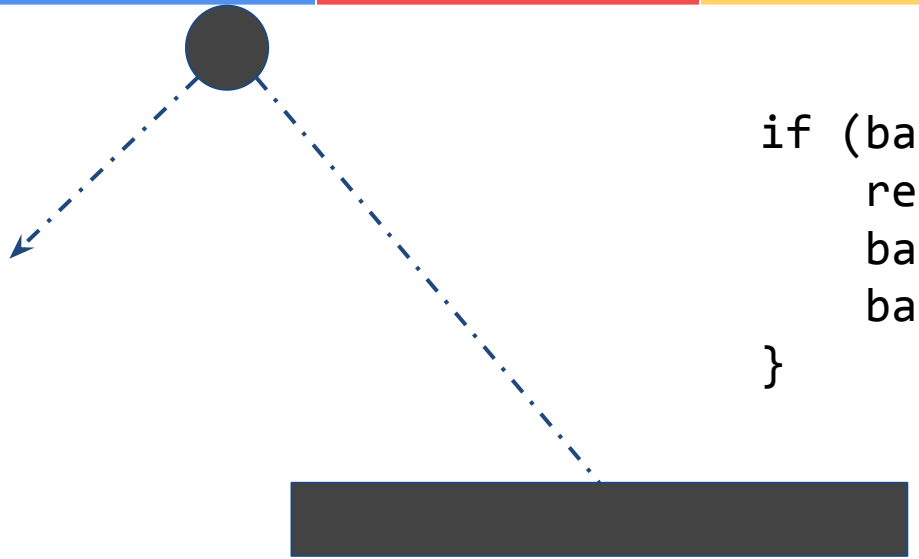
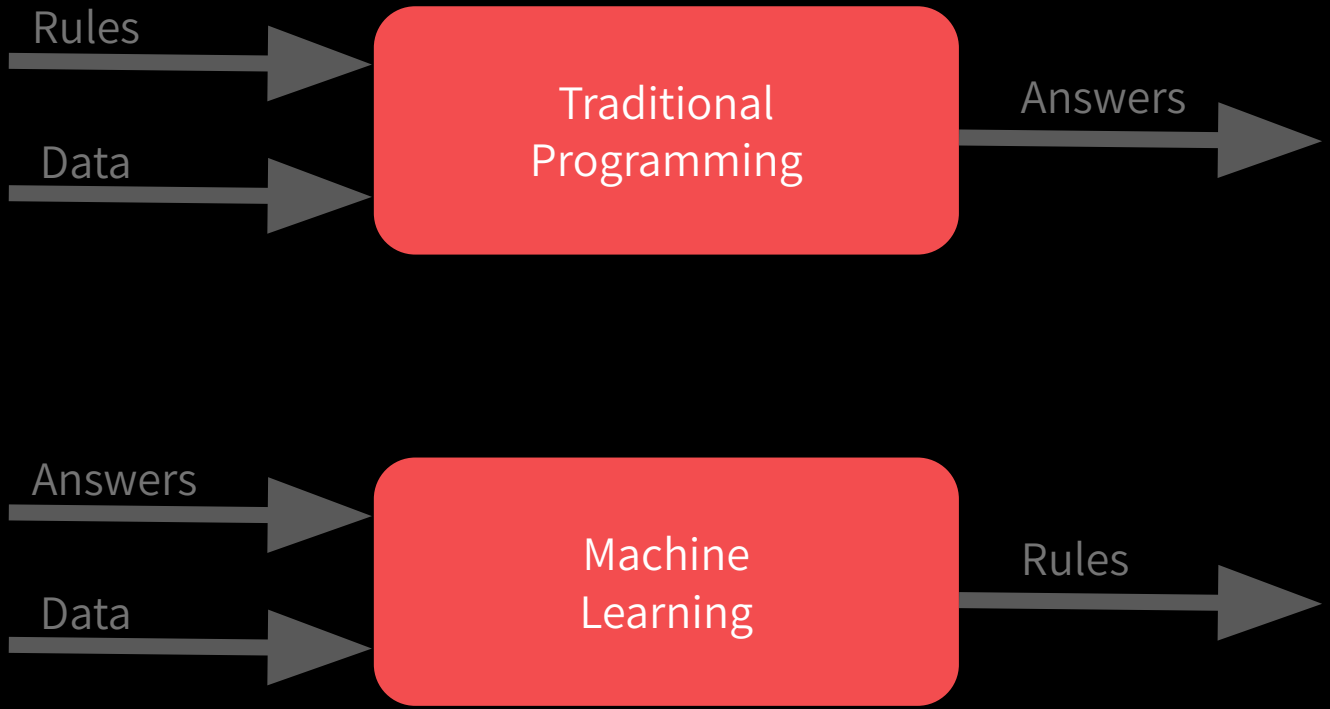
Copyright Notice


Rules
(Expressed in Code)
Data
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;
}
```

Activity Recognition

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

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

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;
}
```

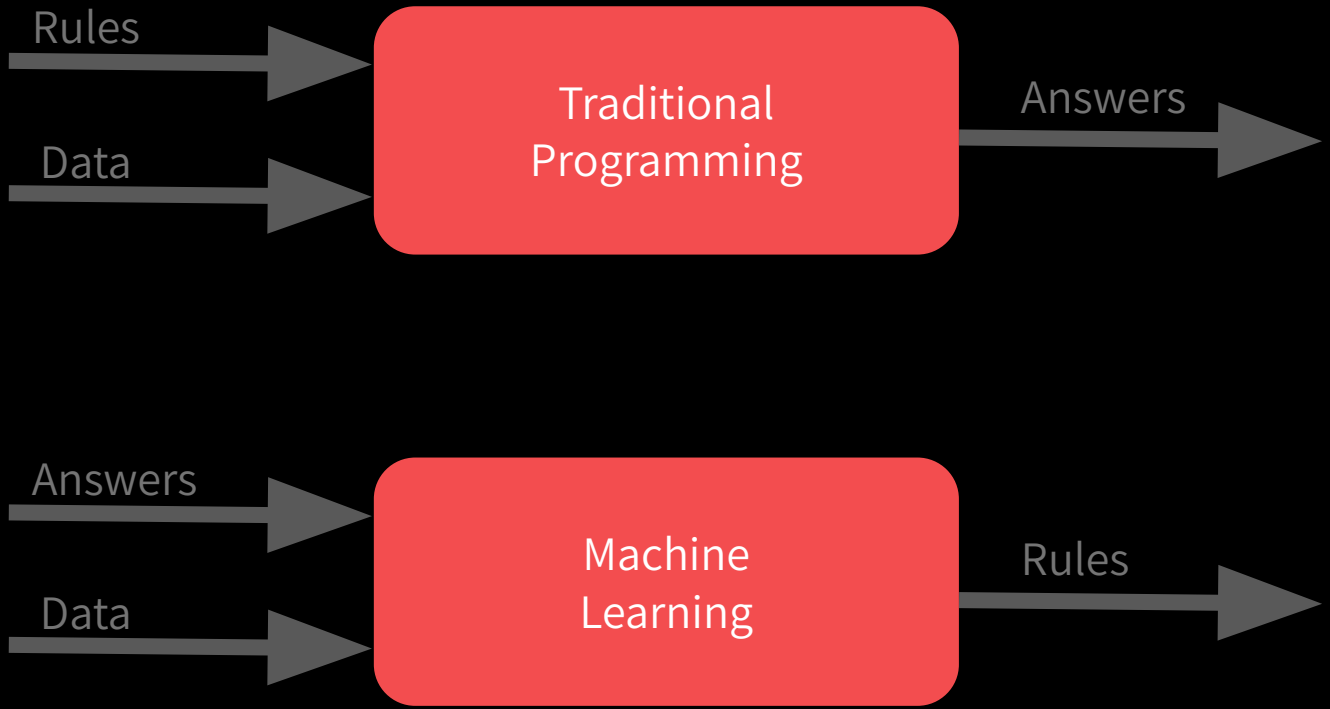
Activity Recognition

```
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
```


Rules
Data
Traditional Programming
Answers
Answers
Data
Machine Learning
Rules

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 = tf.keras.Sequential([
    tf.keras.Input(shape=(1,)),
    tf.keras.layers.Dense(units=1)
])
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(1,)),
    tf.keras.layers.Dense(units=1)
])
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(1,)),
    tf.keras.layers.Dense(units=1)
])
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(1,)),
    tf.keras.layers.Dense(units=1)
])
```

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

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(1,)),
    tf.keras.layers.Dense(units=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 = tf.keras.Sequential([
    tf.keras.Input(shape=(1,)),
    tf.keras.layers.Dense(units=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)
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(1,)),
    tf.keras.layers.Dense(units=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)

model.predict(np.array([10.0]))
```

Copyright Notice

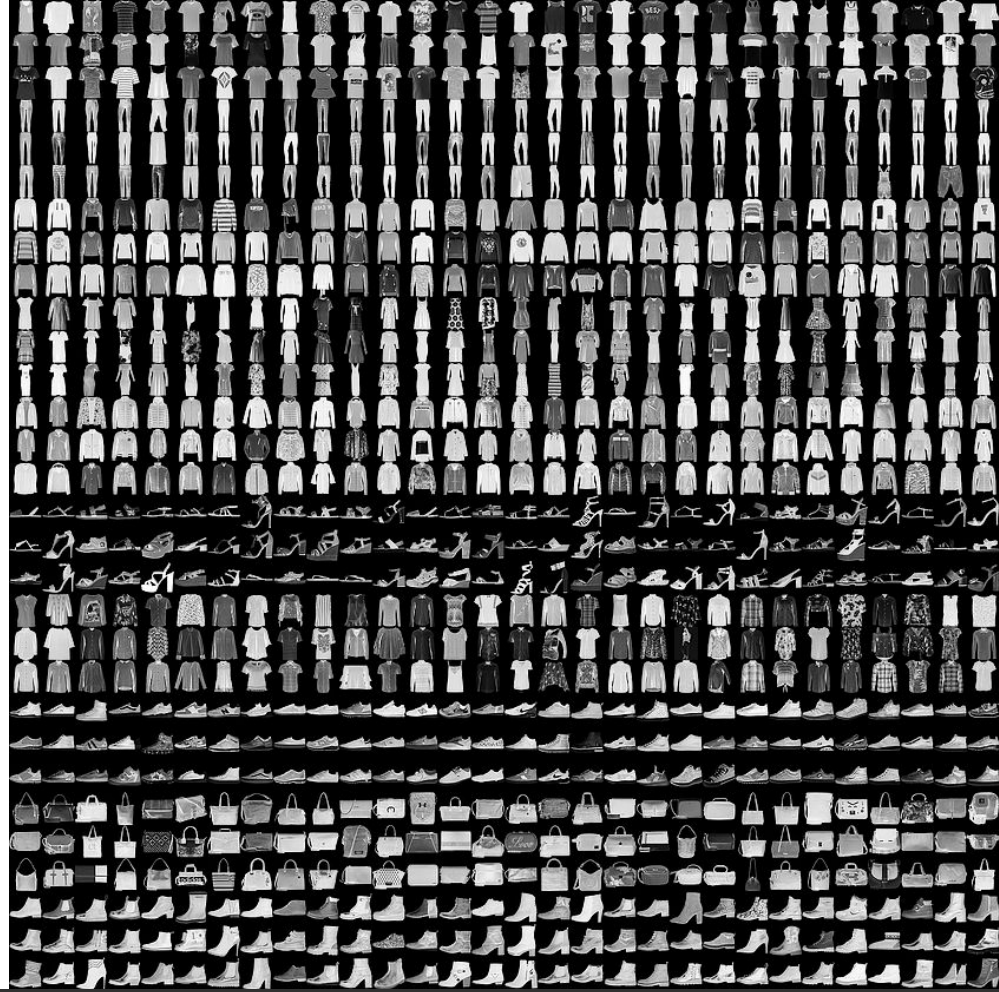






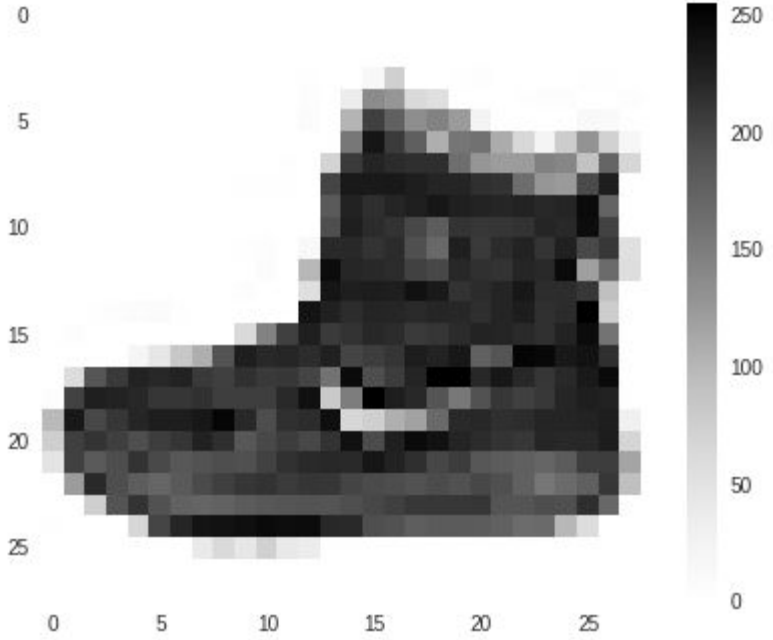

Fashion MNIST

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



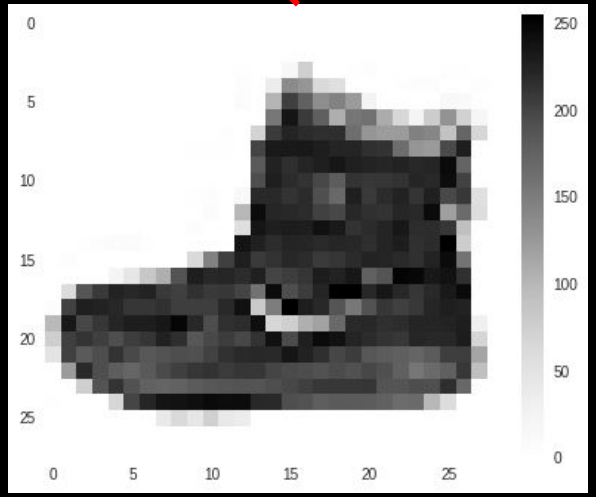
Fashion MNIST

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



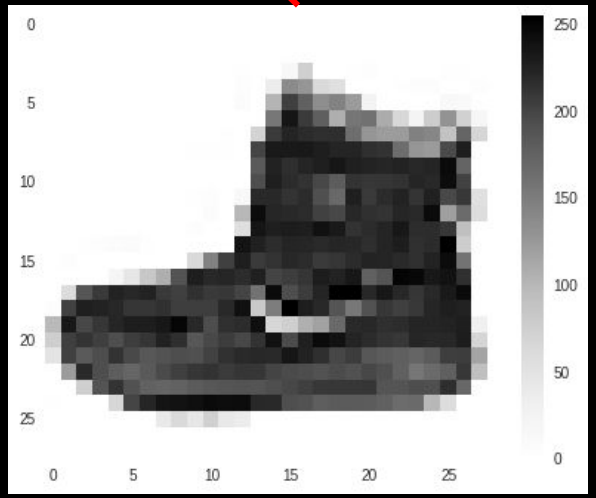
```
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

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



09

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



09

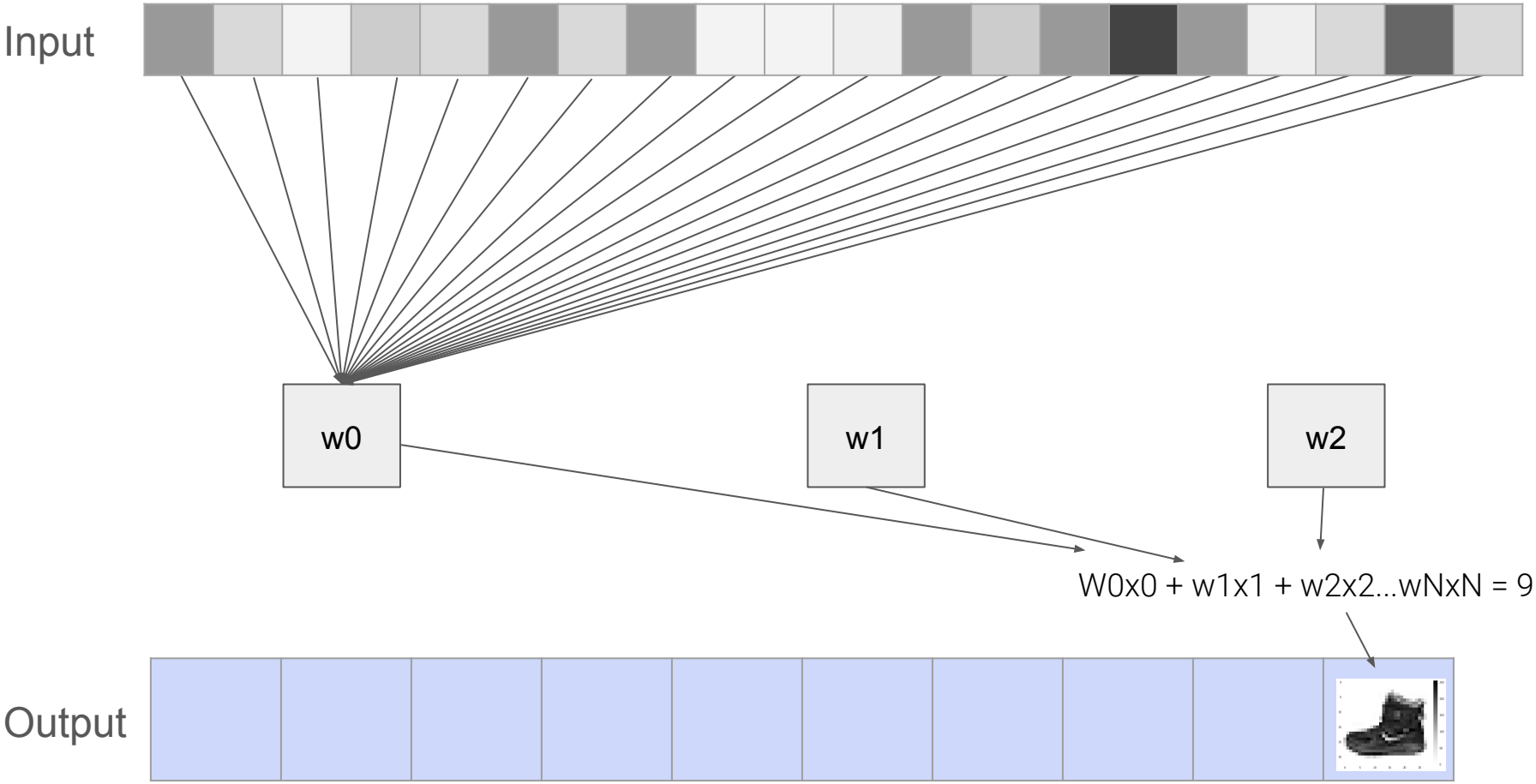
09 = ankle boot;
踝靴;
アンクルブーツ;
Bróg rúitín

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

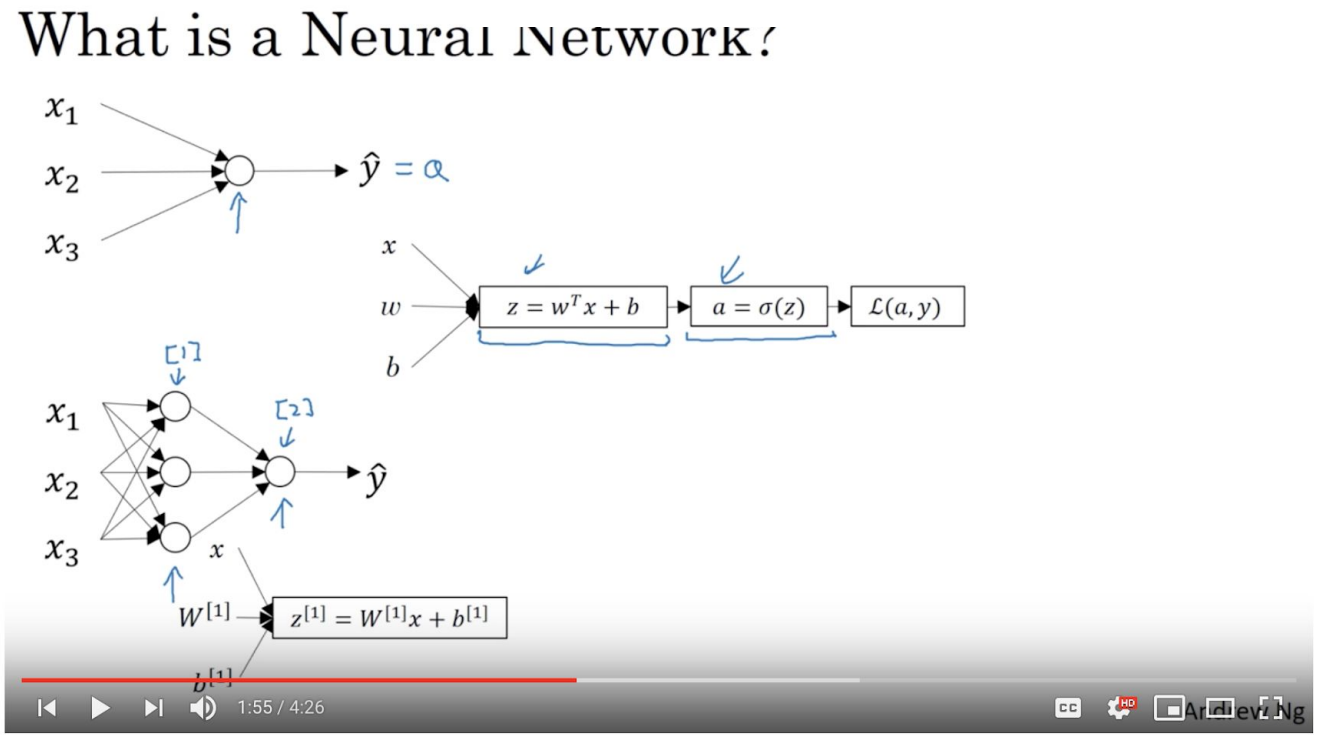
```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(28, 28)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

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


Input
w0
w1
w2
W0x0 + w1x1 + w2x2...wNxN = 9
Output

https://youtu.be/fXOsFF95ifk



Neural Network Overview (C1W3L01)

11,067 views

43 0 SHARE SAVE

Neural Networks and Deep Learning (Course 1 of

Deeplearning.ai - 25 / 43

Complete User Registration system using PHP and MySQL...

Awa Melvine

5.7M views

```
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
training_images = training_images / 255.0
test_images = test_images / 255.0
model = tf.keras.Sequential([
    tf.keras.Input(shape=(28, 28)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, 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)
```

```
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
training_images = training_images / 255.0
test_images = test_images / 255.0
model = tf.keras.Sequential([
    tf.keras.Input(shape=(28, 28)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, 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=None):
    if logs['loss'] < 0.4:
      print('Loss is low so cancelling training!')
      self.model.stop_training = True
```

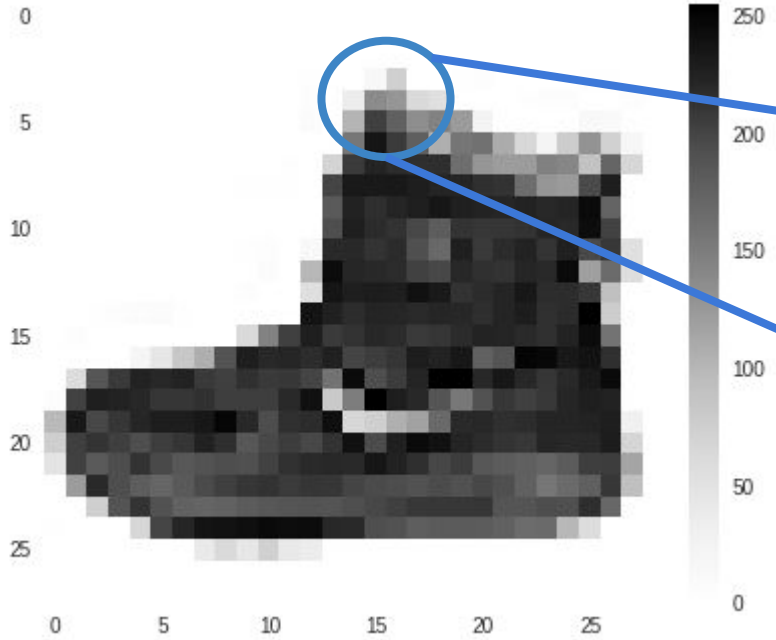
```
class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs=None):
    if logs['loss'] < 0.4:
      print('Loss is low so cancelling training!')
      self.model.stop_training = True
```

```
class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs=None):
    if logs['loss'] < 0.4:
      print('Loss is low so cancelling training!')
      self.model.stop_training = True
```

```
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
training_images = training_images / 255.0
test_images = test_images / 255.0
model = tf.keras.Sequential([
    tf.keras.input(shape=(28, 28)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, 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)
```

```
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
training_images = training_images / 255.0
test_images = test_images / 255.0
model = tf.keras.Sequential([
    tf.keras.input(shape=(28, 28)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, 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()])
```

Copyright Notice



0	64	128
48	192	144
142	226	168

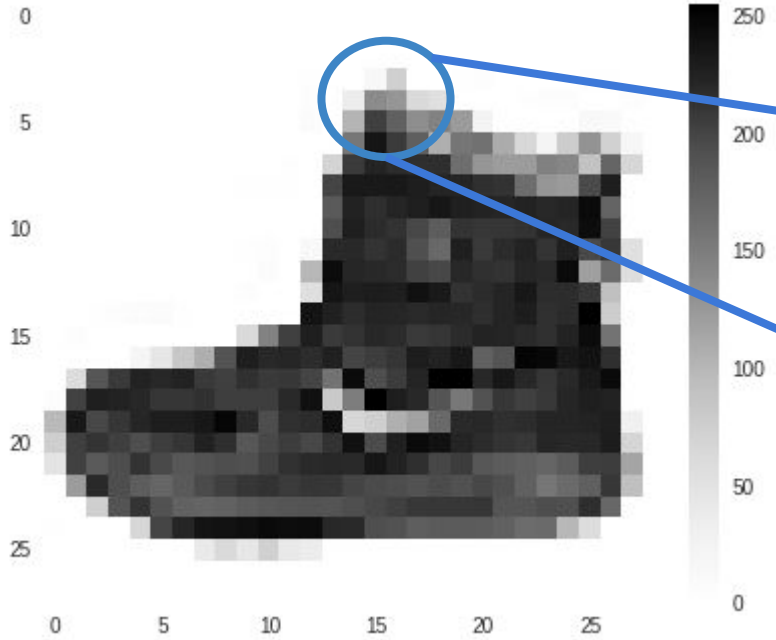
Current Pixel Value is 192

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 * 142) + (2 * 226) + (-3 * 168)
```



0	64	128
48	192	144
142	226	168

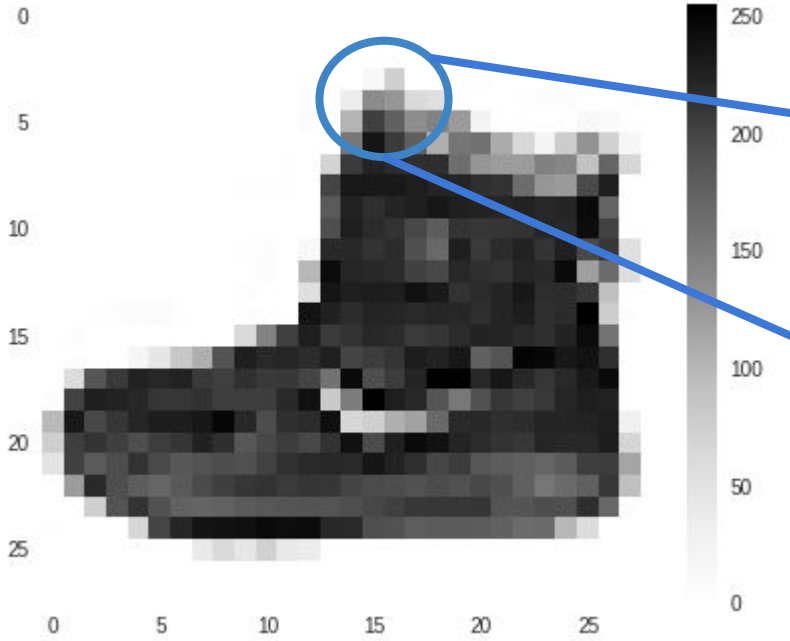
Current Pixel Value is 192

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 * 142) + (2 * 226) + (-3 * 168)
```



0	64	128
48	192	144
142	226	168

Current Pixel Value is 192

Consider neighbor Values

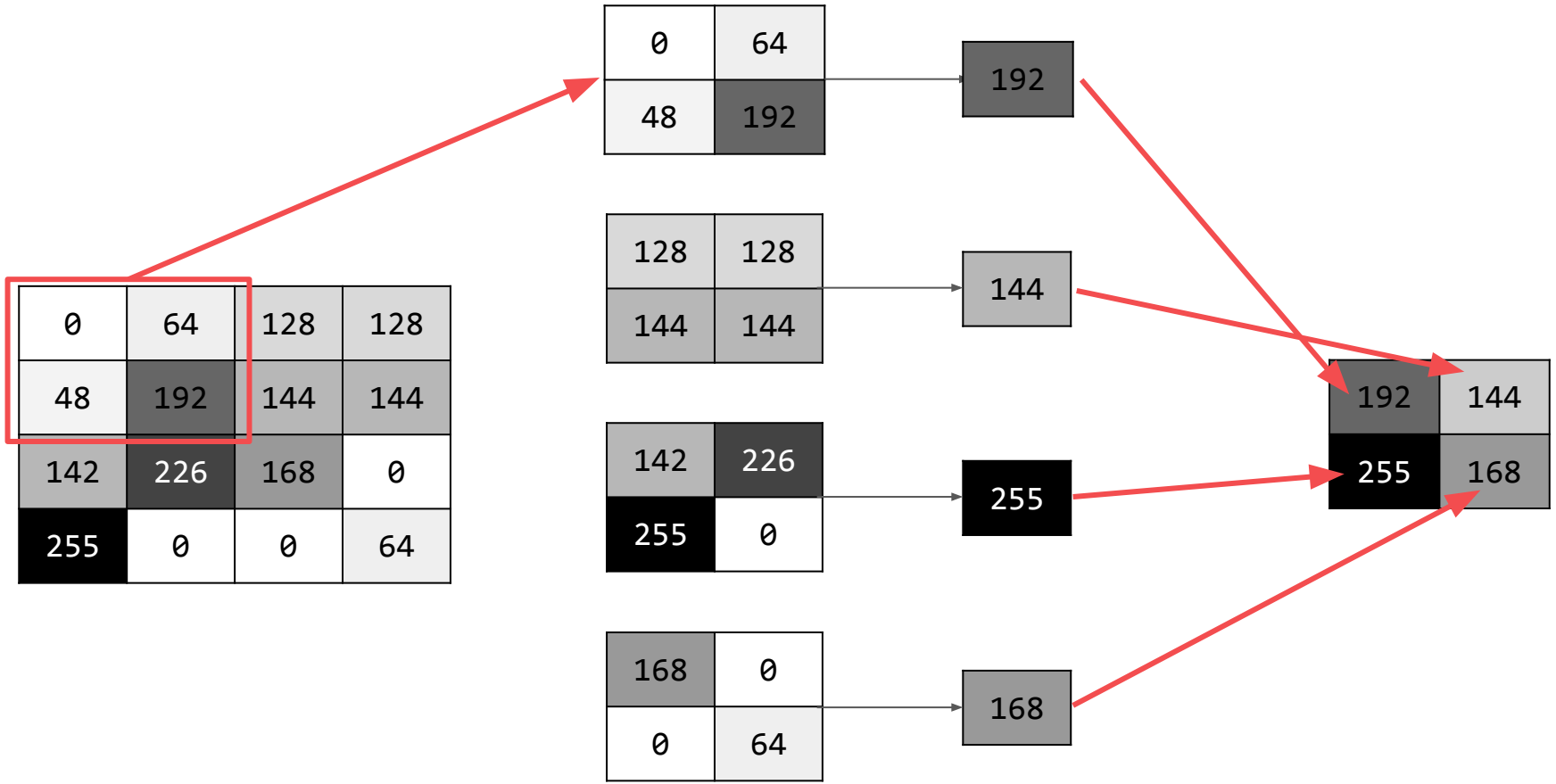
-1	0	-2
.5	4.5	-1.5
1.5	2	-3

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 * 142) + (2 * 226) + (-3 * 168)
```

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

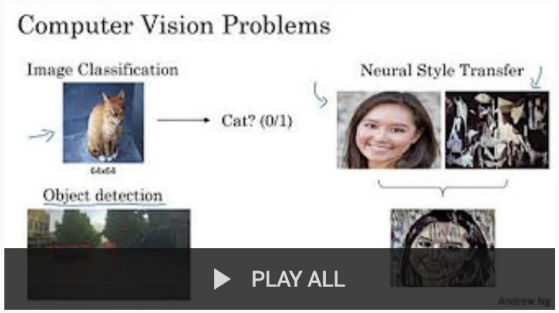
-1	-2	-1
0	0	0
1	2	1


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

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(28, 28)),
    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.Sequential([
  tf.keras.Input(shape=(28, 28, 1)),
  tf.keras.layers.Conv2D(64, (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(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

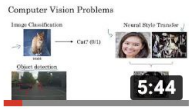
```
model = tf.keras.Sequential([
  tf.keras.Input(shape=(28, 28, 1)),
  tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128,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



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

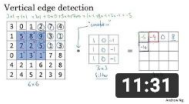
42 videos • 415,722 views • Last updated on Nov 7, 2017

Deeplearning.ai SUBSCRIBE 28K

1 

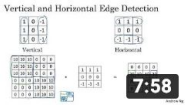
C4W1L01 Computer Vision

Deeplearning.ai

2 

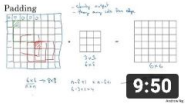
C4W1L02 Edge Detection Examples

Deeplearning.ai

3 

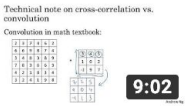
C4W1L03 More Edge Detection

Deeplearning.ai

4 

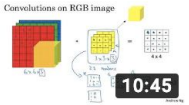
C4W1L04 Padding

Deeplearning.ai

5 

C4W1L05 Strided Convolutions

Deeplearning.ai

6 

C4W1L06 Convolutions Over Volumes

Deeplearning.ai

https://bit.ly/2UGa7uH

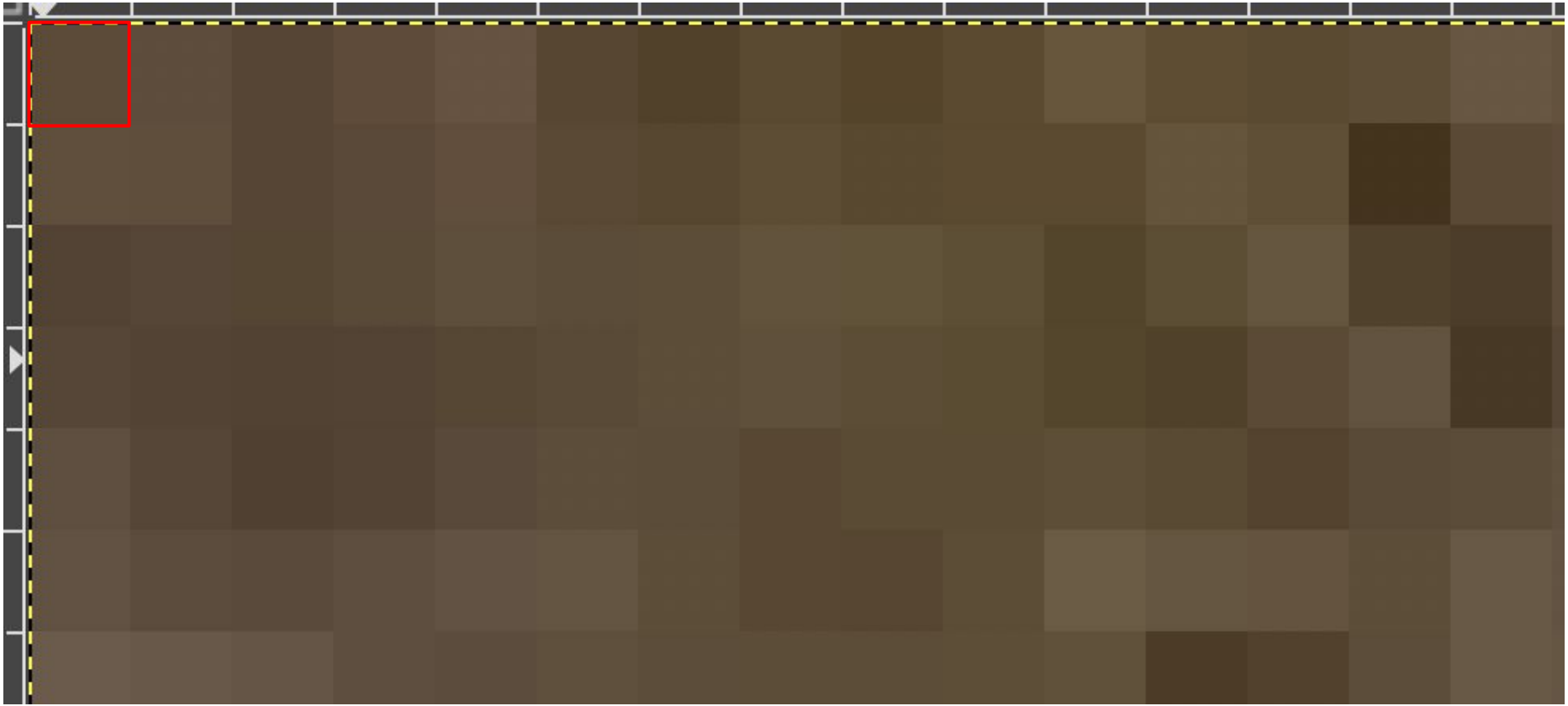
```
model = tf.keras.Sequential([
  tf.keras.Input(shape=(28, 28, 1)),
  tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128,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

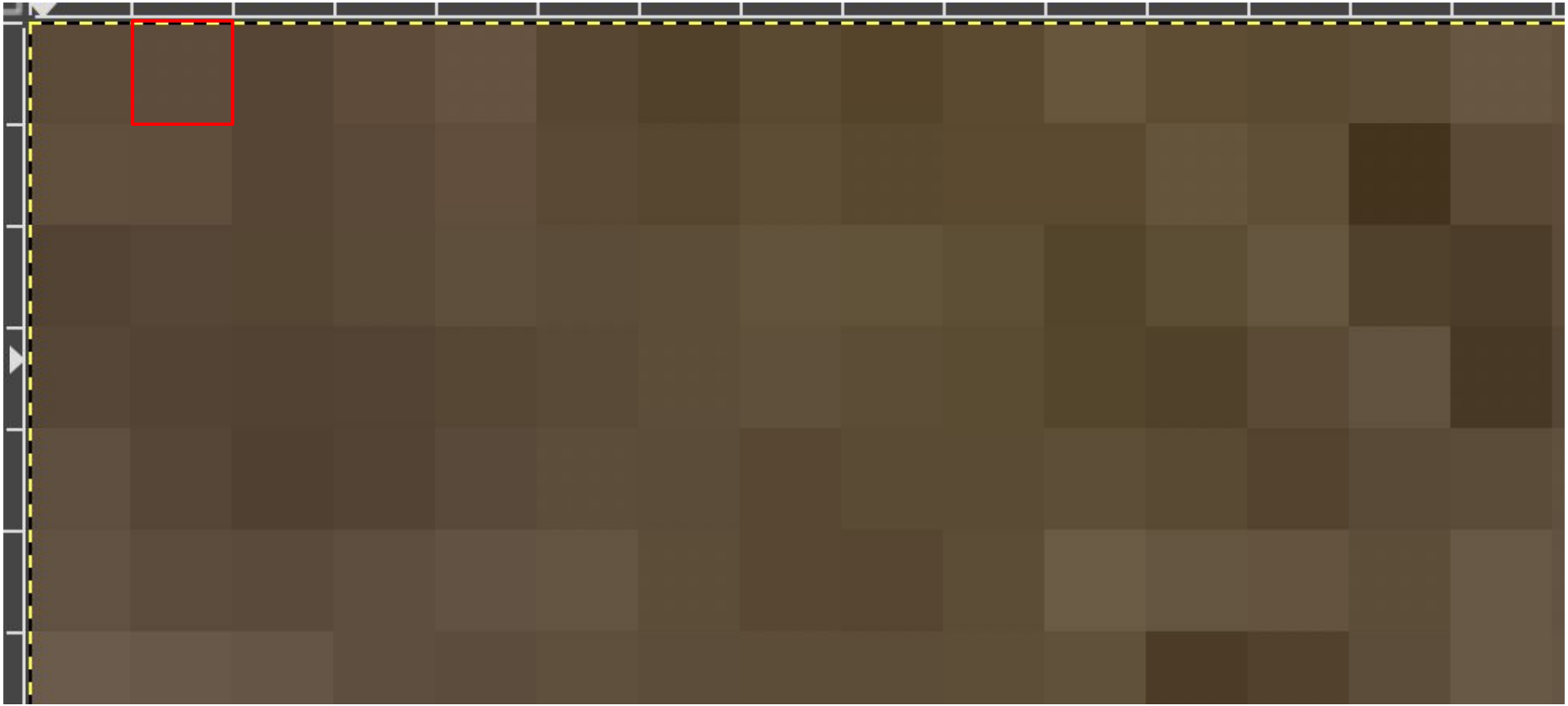
```
model = tf.keras.Sequential([
  tf.keras.Input(shape=(28, 28, 1)),
  tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128,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

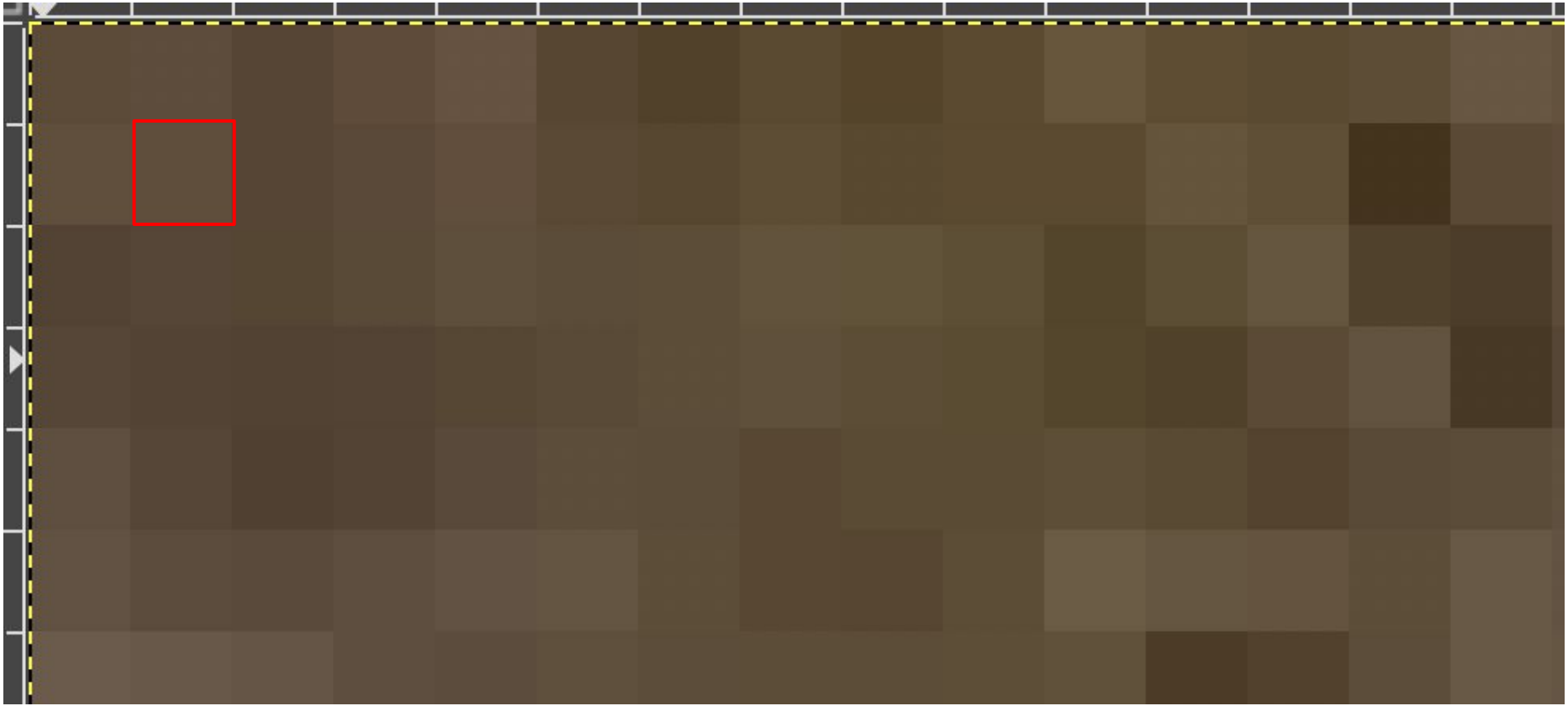
```
model.summary()
```

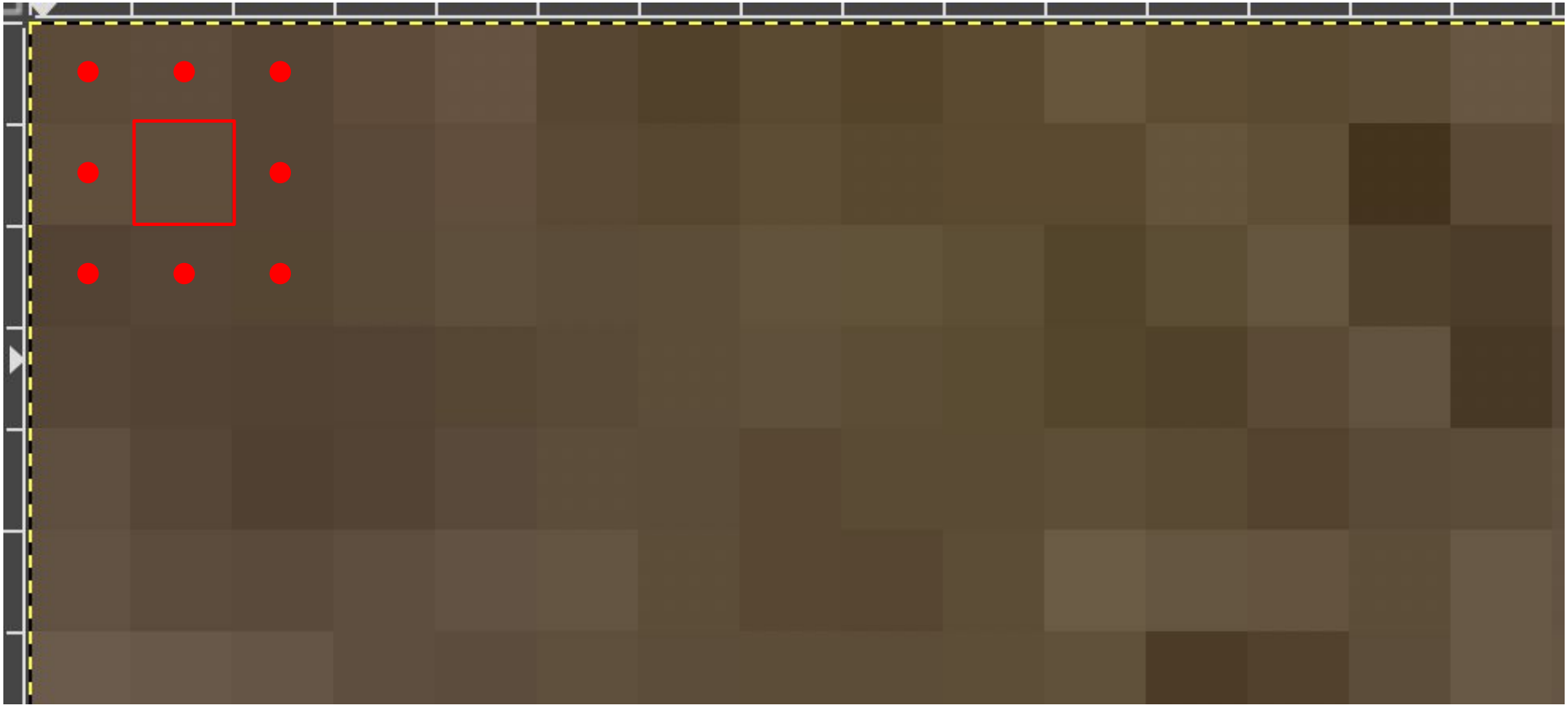
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 26, 26, 64)        640
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 13, 13, 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

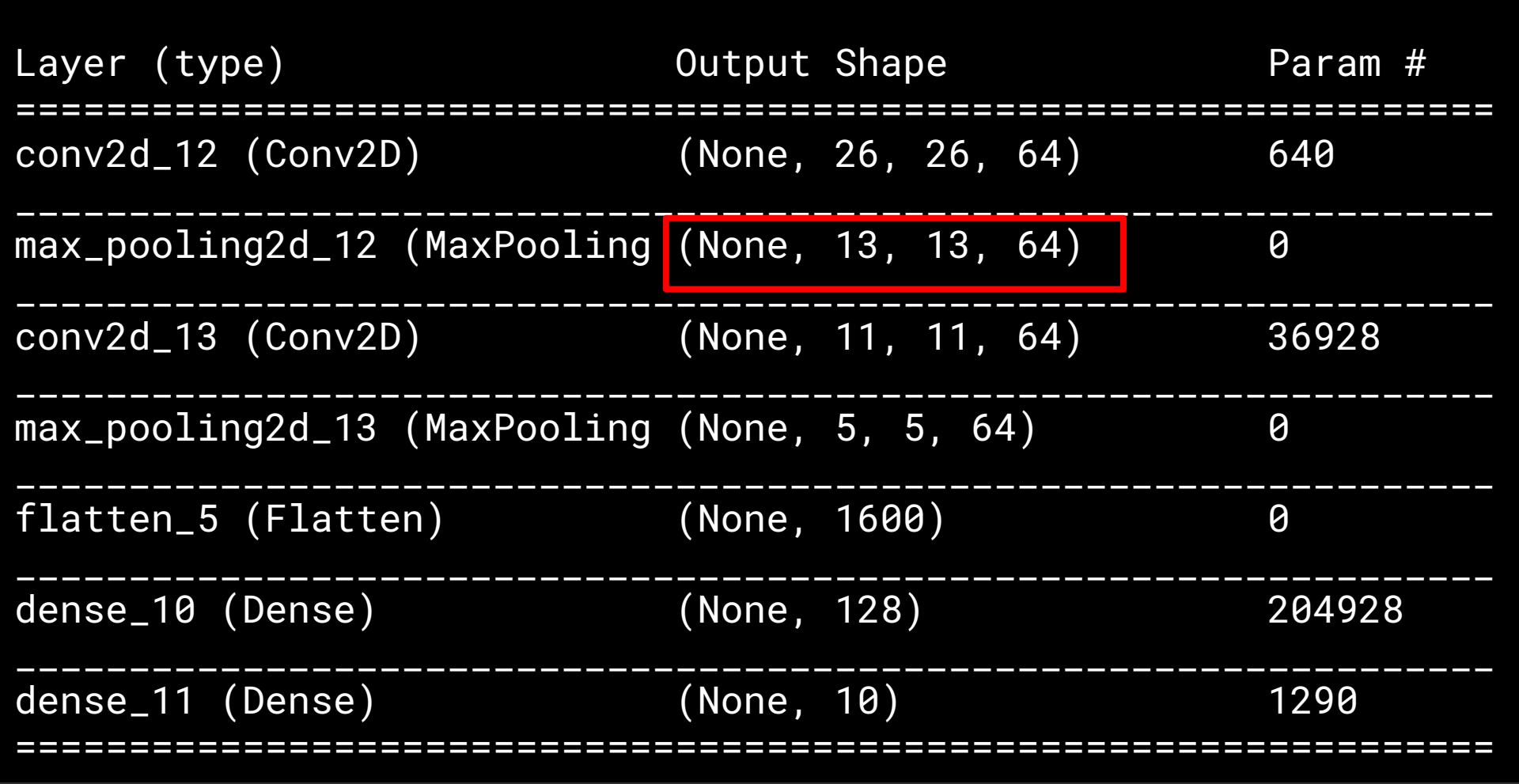








Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 26, 26, 64)	640
max_pooling2d_12 (MaxPooling	(None, 13, 13, 64)	0
conv2d_13 (Conv2D)	(None, 11, 11, 64)	36928
max_pooling2d_13 (MaxPooling	(None, 5, 5, 64)	0
flatten_5 (Flatten)	(None, 1600)	0
dense_10 (Dense)	(None, 128)	204928
dense_11 (Dense)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

```
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
=================================================================
```

```
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
=================================================================
```

Copyright Notice

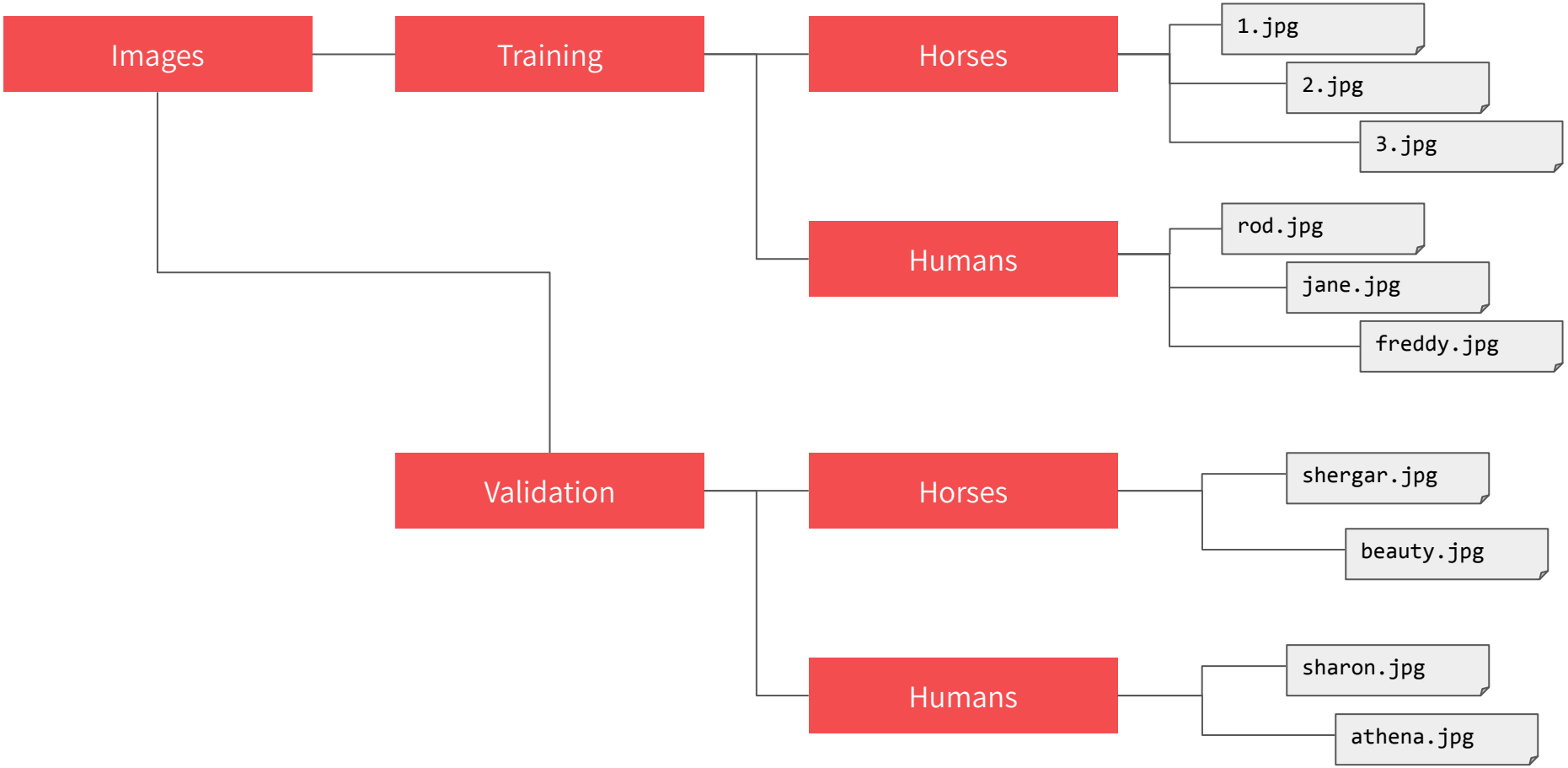


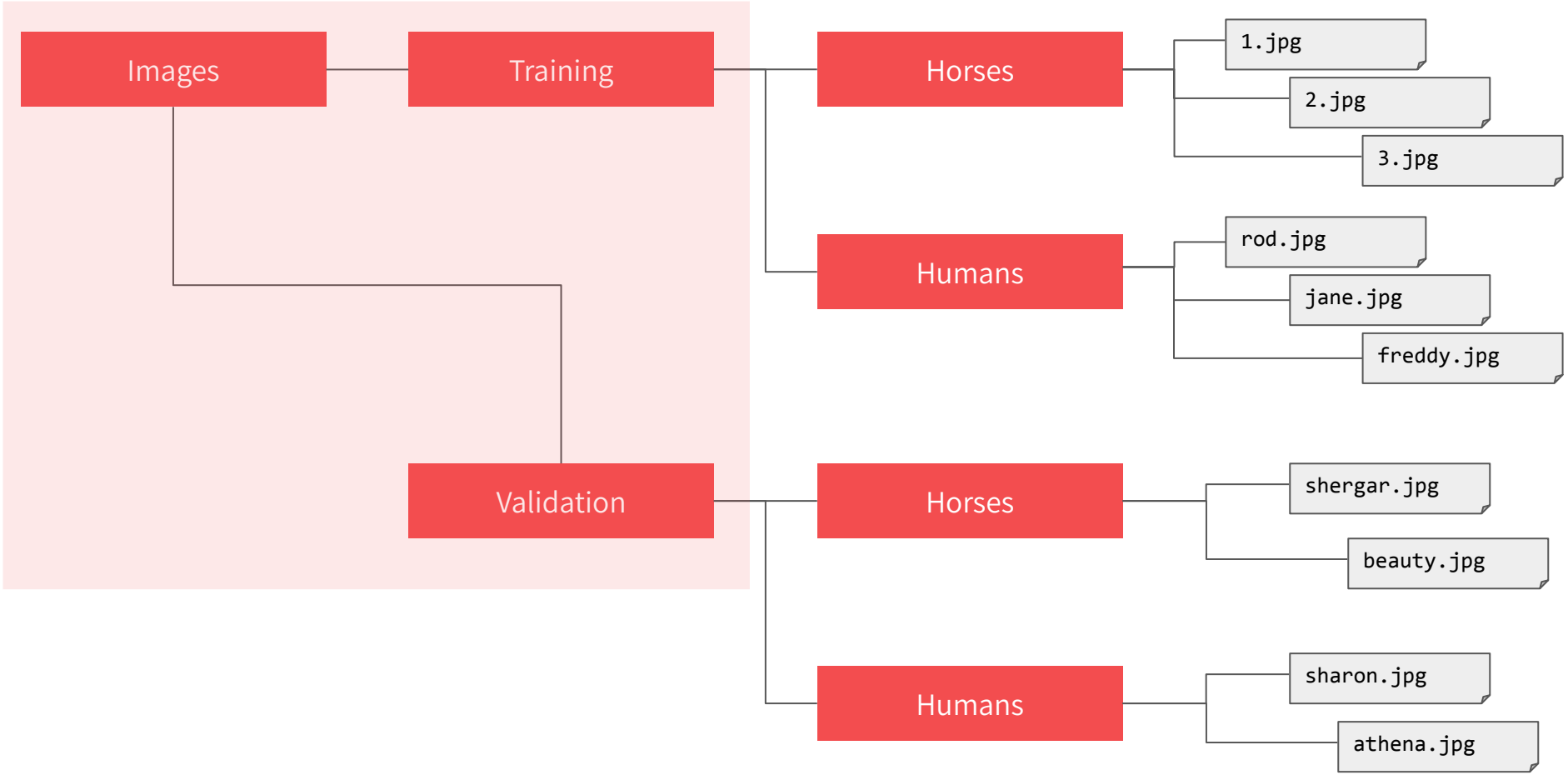


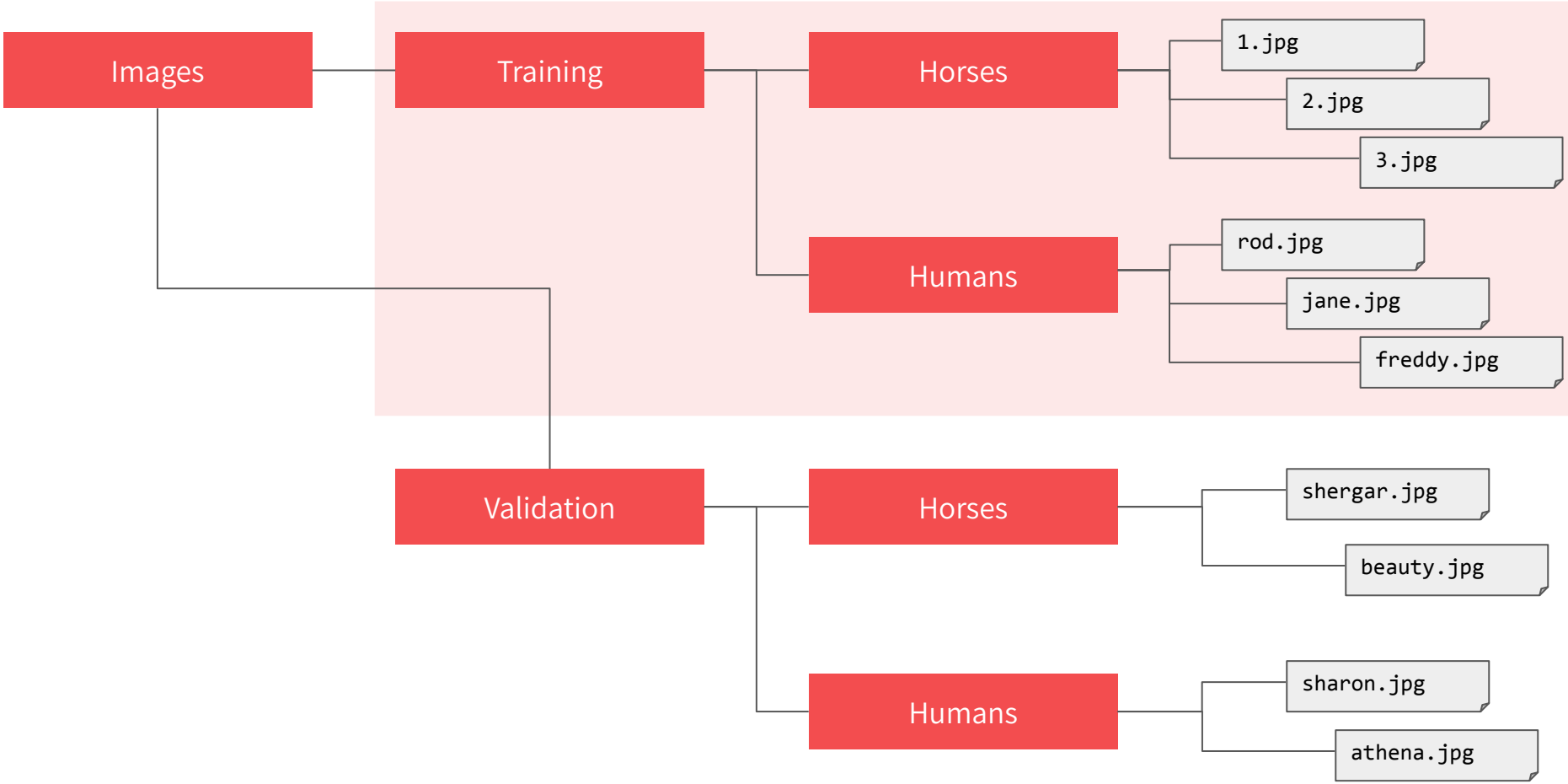


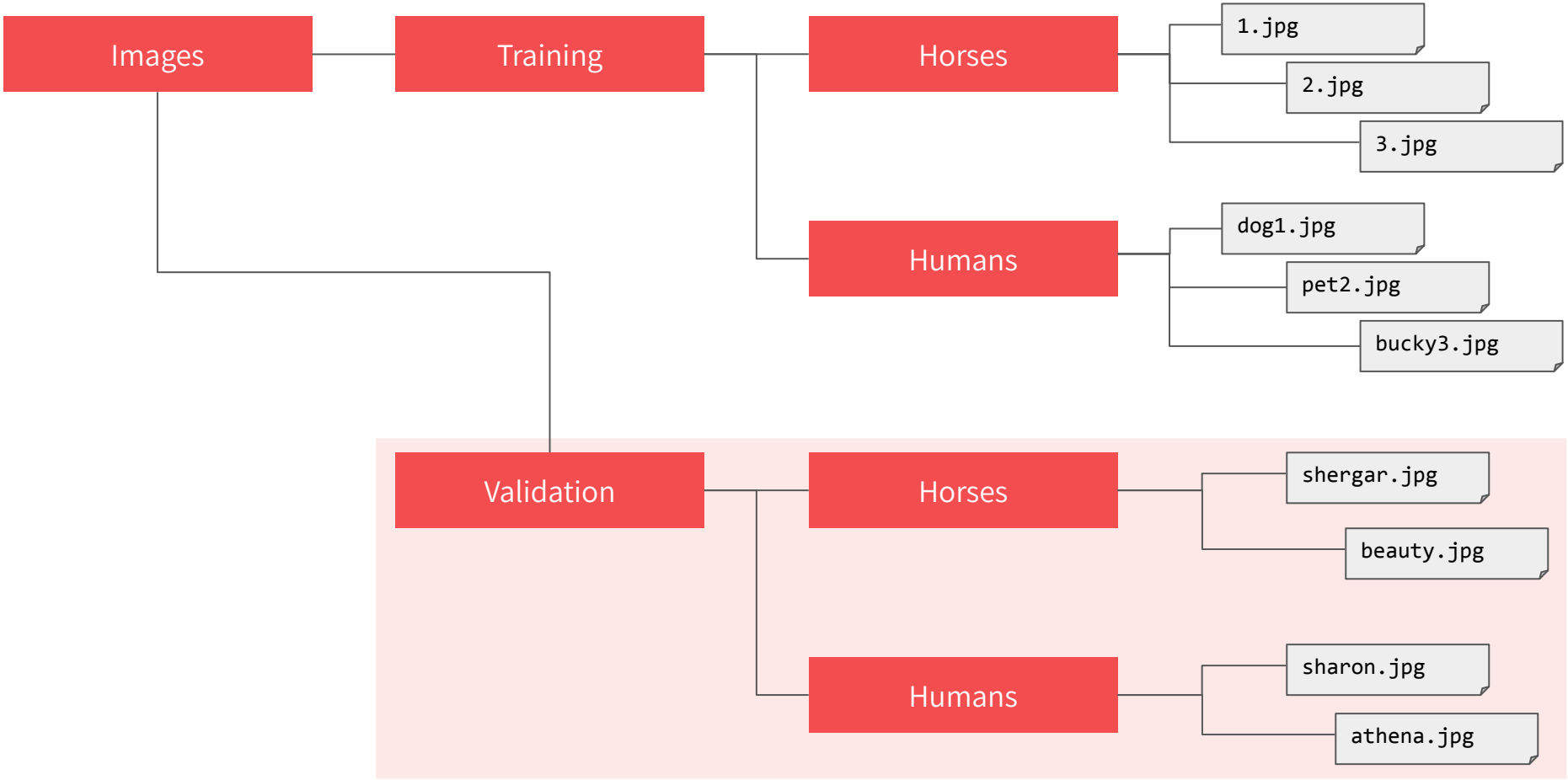
tf.data API

tf.keras.utils.image_dataset_from_directory


Images
Training
Horses
1.jpg
2.jpg
3.jpg
Humans
rod.jpg
jane.jpg
freddy.jpg
Validation
Horses
shergar.jpg
beauty.jpg
Humans
sharon.jpg
athena.jpg


Images
Training
Horses
1.jpg
2.jpg
3.jpg
Humans
rod.jpg
jane.jpg
freddy.jpg
Validation
Horses
shergar.jpg
beauty.jpg
Humans
sharon.jpg
athena.jpg


Images
Training
Horses
1.jpg
2.jpg
3.jpg
Humans
rod.jpg
jane.jpg
freddy.jpg
Validation
Horses
shergar.jpg
beauty.jpg
Humans
sharon.jpg
athena.jpg


Images
Training
Horses
1.jpg
2.jpg
3.jpg
Humans
dog1.jpg
pet2.jpg
bucky3.jpg
Validation
Horses
shergar.jpg
beauty.jpg
Humans
sharon.jpg
athena.jpg

```
train_dataset = tf.keras.utils.image_dataset_from_directory(
    TRAIN_DIR,
    image_size=(300, 300),
    batch_size=128,
    label_mode='binary'
    )
```

```
train_dataset = tf.keras.utils.image_dataset_from_directory(
    TRAIN_DIR,
    image_size=(300, 300),
    batch_size=128,
    label_mode='binary'
    )
```

```
train_dataset = tf.keras.utils.image_dataset_from_directory(
    TRAIN_DIR,
    image_size=(300, 300),
    batch_size=128,
    label_mode='binary'
    )
```

```
train_dataset = tf.keras.utils.image_dataset_from_directory(
    TRAIN_DIR,
    image_size=(300, 300),
    batch_size=128,
    label_mode='binary'
    )
```

tf.data.Dataset

https://www.tensorflow.org/api_docs/python/tf/data/Dataset

```
rescale_layer = tf.keras.layers.Rescaling(scale=1./255)

train_dataset_scaled = train_dataset.map(
    lambda image, label: (rescale_layer(image), label))
```

```
rescale_layer = tf.keras.layers.Rescaling(scale=1./255)

train_dataset_scaled = train_dataset.map(
    lambda image, label: (rescale_layer(image), label))
```

```
rescale_layer = tf.keras.layers.Rescaling(scale=1./255)

train_dataset_scaled = train_dataset.map(
    lambda image, label: (rescale_layer(image), label))
```

```
train_dataset_final = (train_dataset_scaled
                       .cache()
                       .shuffle(buffer_size=1000)
                       .prefetch(buffer_size=tf.data.AUTOTUNE))
```

```
train_dataset_final = (train_dataset_scaled
                       .cache()
                       .shuffle(buffer_size=1000)
                       .prefetch(buffer_size=tf.data.AUTOTUNE))
```

```
train_dataset_final = (train_dataset_scaled
                       .cache()
                       .shuffle(buffer_size=1000)
                       .prefetch(buffer_size=tf.data.AUTOTUNE))
```

```
train_dataset_final = (train_dataset_scaled
                       .cache()
                       .shuffle(buffer_size=1000)
                       .prefetch(buffer_size=tf.data.AUTOTUNE))
```

```
validation_dataset = tf.keras.utils.image_dataset_from_directory(
    VAL_DIR,
    image_size=(300, 300),
    batch_size=32,
    label_mode='binary'
    )

validation_dataset_scaled = validation_dataset.map(lambda image, label:
(rescale_layer(image), label))

# Configure the validation dataset
validation_dataset_final = (validation_dataset_scaled
                            .cache()
                            .prefetch(buffer_size=tf.data.AUTOTUNE))
```

```
validation_dataset = tf.keras.utils.image_dataset_from_directory(
    VAL_DIR,
    image_size=(300, 300),
    batch_size=128,
    label_mode='binary'
    )

validation_dataset_scaled = validation_dataset.map(lambda image, label:
(rescale_layer(image), label))

# Configure the validation dataset
validation_dataset_final = (validation_dataset_scaled
 .cache()
 .prefetch(buffer_size=tf.data.AUTOTUNE))
```

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(300, 300, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu'),
    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')
])
```

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(300, 300, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu'),
    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')
])
```

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(300, 300, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu'),
    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')
])
```

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(300, 300, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu'),
    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
```

```
model.compile(loss='binary_crossentropy',
              optimizer=tf.keras.optimizers.RMSprop(learning_rate=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_dataset_final,
      epochs=15,
      validation_data=validation_dataset_final,
      verbose=2)
```

```
history = model.fit(
      train_dataset_final,
      epochs=15,
      validation_data=validation_dataset_final,
      verbose=2)
```

```
history = model.fit(
      train_dataset_final,
      epochs=15,
      validation_data=validation_dataset_final,
      verbose=2)
```

```
history = model.fit(
      train_dataset_final,
      epochs=15,
      validation_data=validation_dataset_final,
      verbose=2)
```

```
history = model.fit(
      train_dataset_final,
      epochs=15,
      validation_data=validation_dataset_final,
      verbose=2)
```

```
from google.colab import files

uploaded = files.upload()

for filename in uploaded.keys():

    # predicting images
    path = '/content/' + filename
    image = tf.keras.utils.load_img(path, target_size=(300, 300))
    image = tf.keras.utils.img_to_array(image)
    image = rescale_layer(image)
    image = np.expand_dims(image, axis=0)

    prediction = model.predict(image, verbose=0)[0][0]
    print(f'\nmodel output: {prediction}')

    if prediction > 0.5:
        print(filename + " is a human")
    else:
        print(filename + " is a horse")
```

```
from google.colab import files

uploaded = files.upload()

for filename in uploaded.keys():

    # predicting images
    path = '/content/' + filename
    image = tf.keras.utils.load_img(path, target_size=(300, 300))
    image = tf.keras.utils.img_to_array(image)
    image = rescale_layer(image)
    image = np.expand_dims(image, axis=0)

    prediction = model.predict(image, verbose=0)[0][0]
    print(f'\nmodel output: {prediction}')

    if prediction > 0.5:
        print(filename + " is a human")
    else:
        print(filename + " is a horse")
```

```
from google.colab import files

uploaded = files.upload()

for filename in uploaded.keys():

    # predicting images
    path = '/content/' + filename
    image = tf.keras.utils.load_img(path, target_size=(300, 300))
    image = tf.keras.utils.img_to_array(image)
    image = rescale_layer(image)
    image = np.expand_dims(image, axis=0)

    prediction = model.predict(image, verbose=0)[0][0]
    print(f'\nmodel output: {prediction}')

    if prediction > 0.5:
        print(filename + " is a human")
    else:
        print(filename + " is a horse")
```

```
from google.colab import files

uploaded = files.upload()

for filename in uploaded.keys():

    # predicting images
    path = '/content/' + filename
    image = tf.keras.utils.load_img(path, target_size=(300, 300))
    image = tf.keras.utils.img_to_array(image)
    image = rescale_layer(image)
    image = np.expand_dims(image, axis=0)

    prediction = model.predict(image, verbose=0)[0][0]
    print(f'\nmodel output: {prediction}')

    if prediction > 0.5:
        print(filename + " is a human")
    else:
        print(filename + " is a horse")
```