



CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

Introduction to CNNs

Ariel Rokem

Senior Data Scientist, University of Washington



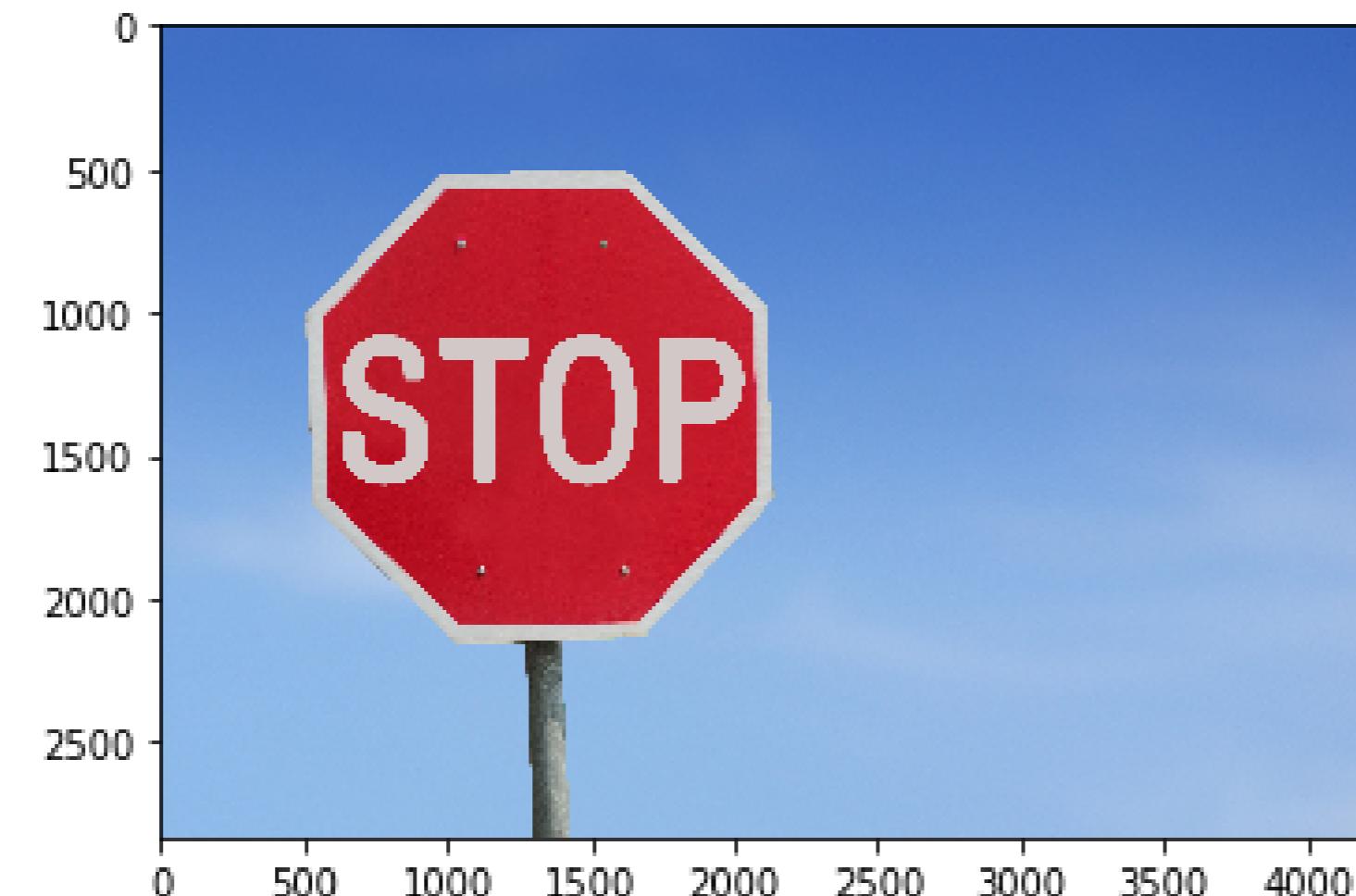


Software and pre-requisites

- [DataCamp's Deep Learning course](#)
- Machine learning:
 - Overfitting
 - Model evaluation
 - Cross-validation

Images as data

```
import matplotlib.pyplot as plt  
data = plt.imread('stop_sign.jpg')  
plt.imshow(data)  
plt.show()
```



Images as data

```
data.shape  
(2832, 4256, 3)
```

Images as data

```
data[1000, 1500]  
array([0.73333333, 0.07843137, 0.14509804])
```



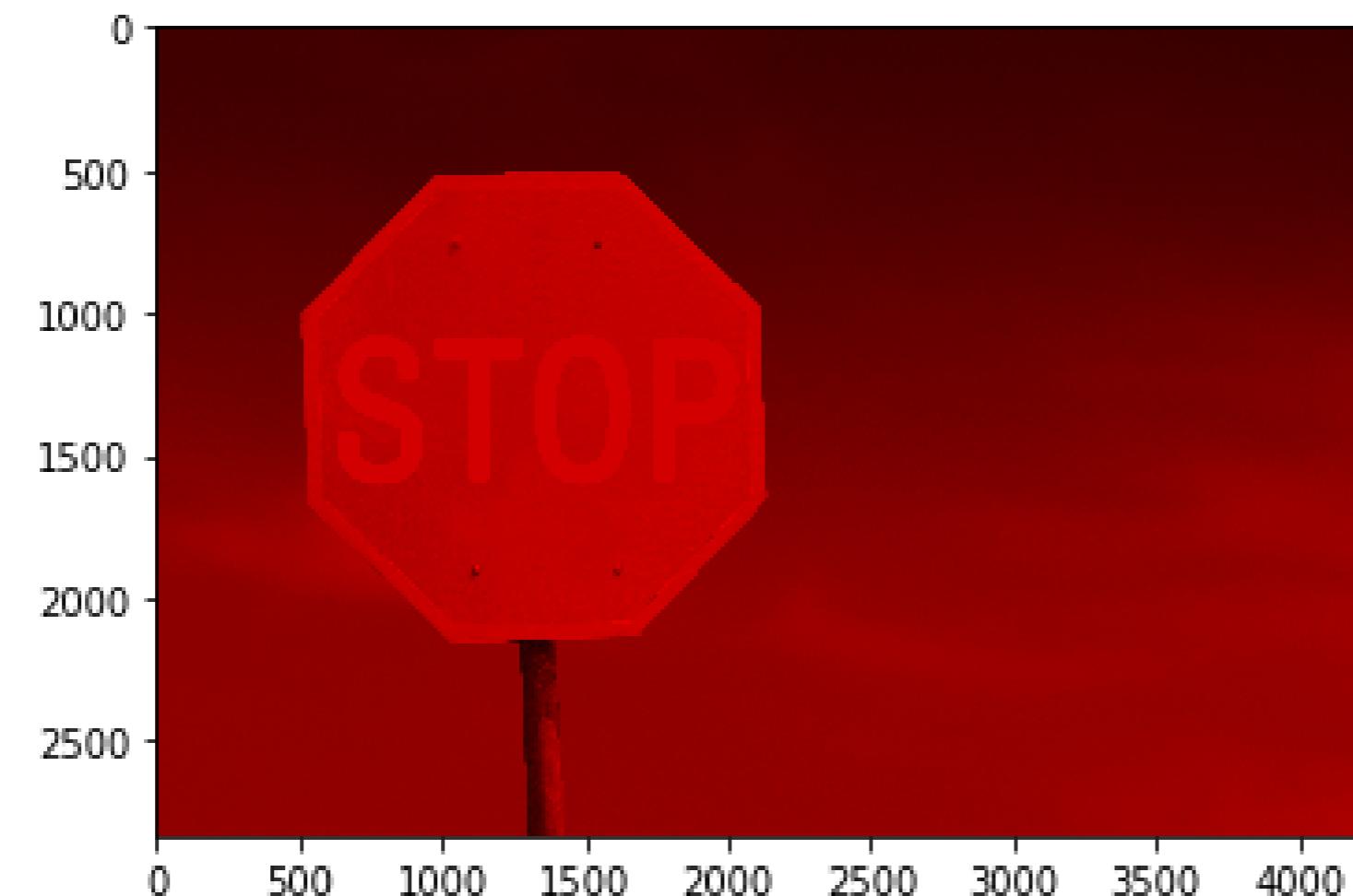
Images as data

```
data[250, 3500]  
array([0.25882353, 0.43921569, 0.77254902])
```



Modifying image data

```
data[:, :, 1] = 0  
data[:, :, 2] = 0  
plt.imshow(data)  
plt.show()
```

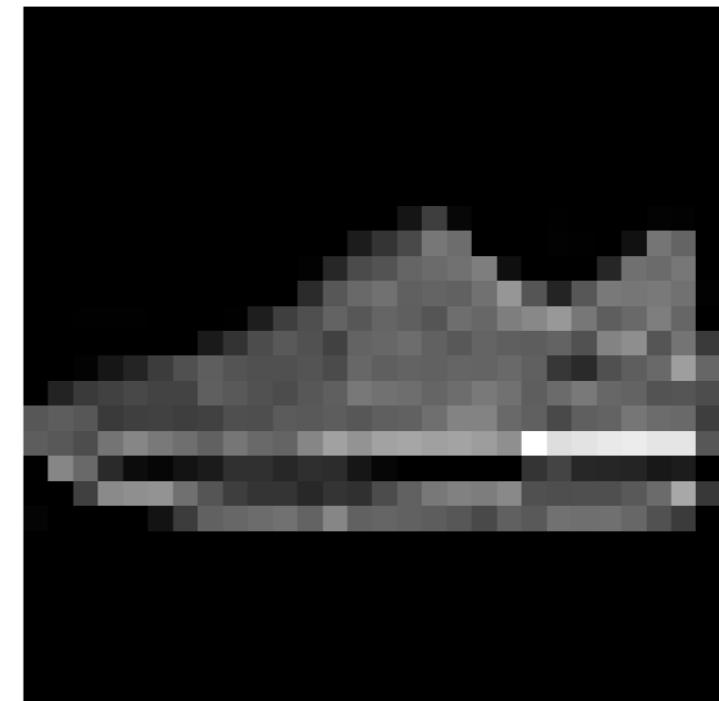
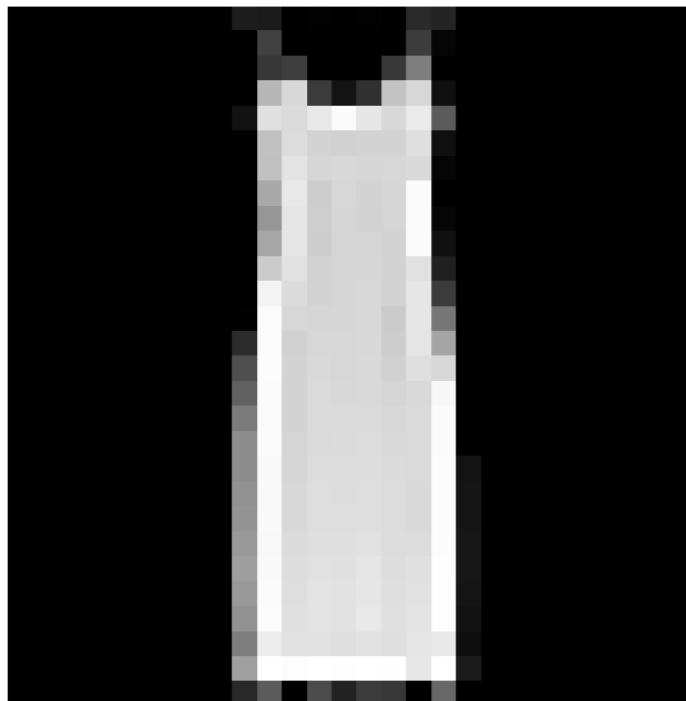


Changing an image

```
data[200:1200, 200:1200, :] = [0, 1, 0]  
  
plt.imshow(data)  
plt.show()
```



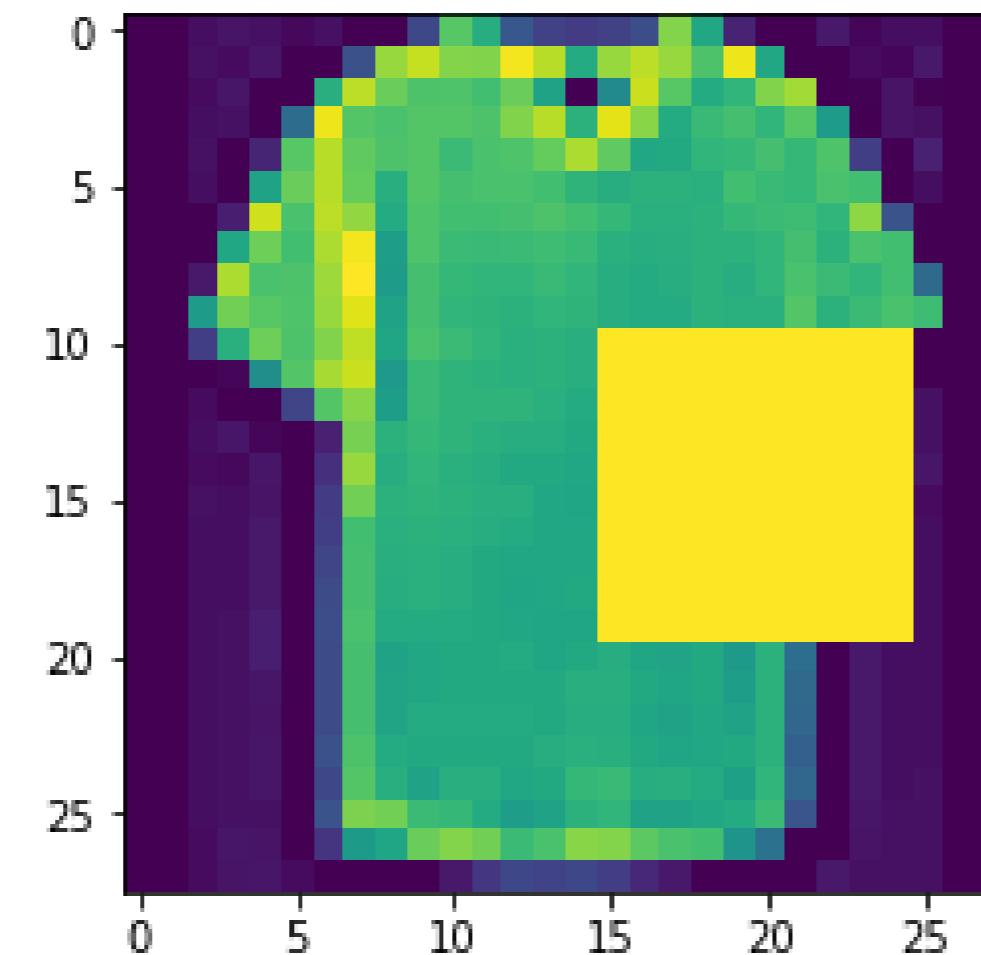
Black and white images



Black and white images

Black and white images

```
tshirt[10:20, 15:25] = 1  
  
plt.imshow(tshirt)  
plt.show()
```





CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

Let's practice!



CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

Classifying images

Ariel Rokem

Senior Data Scientist, University of Washington

Image classification

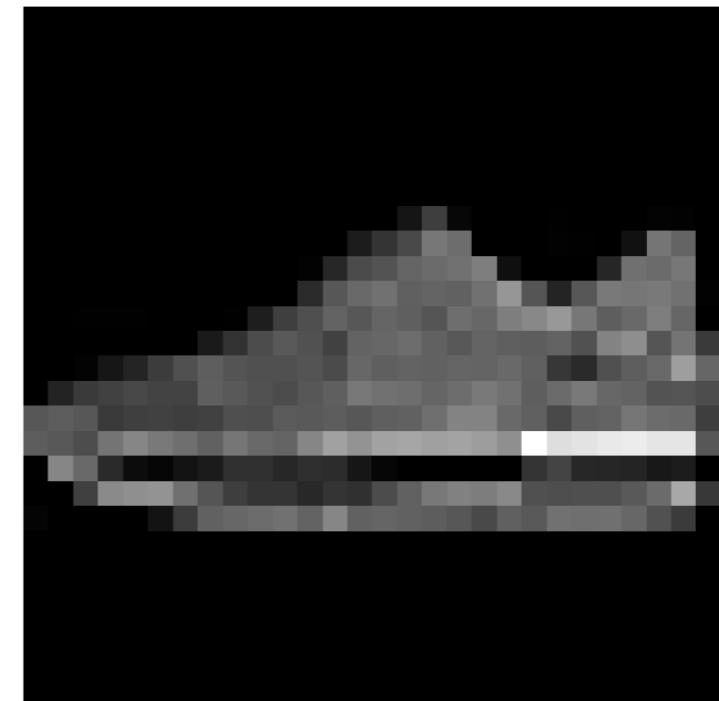
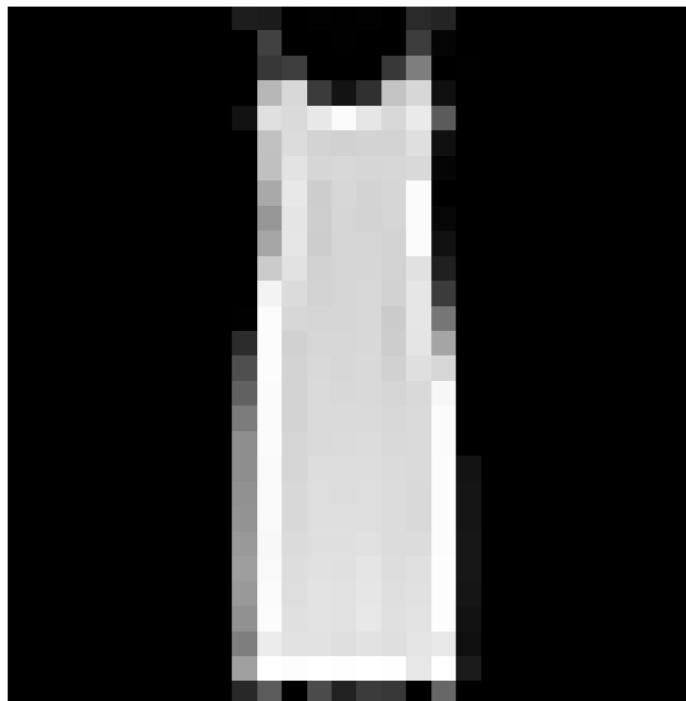


Image classification: training

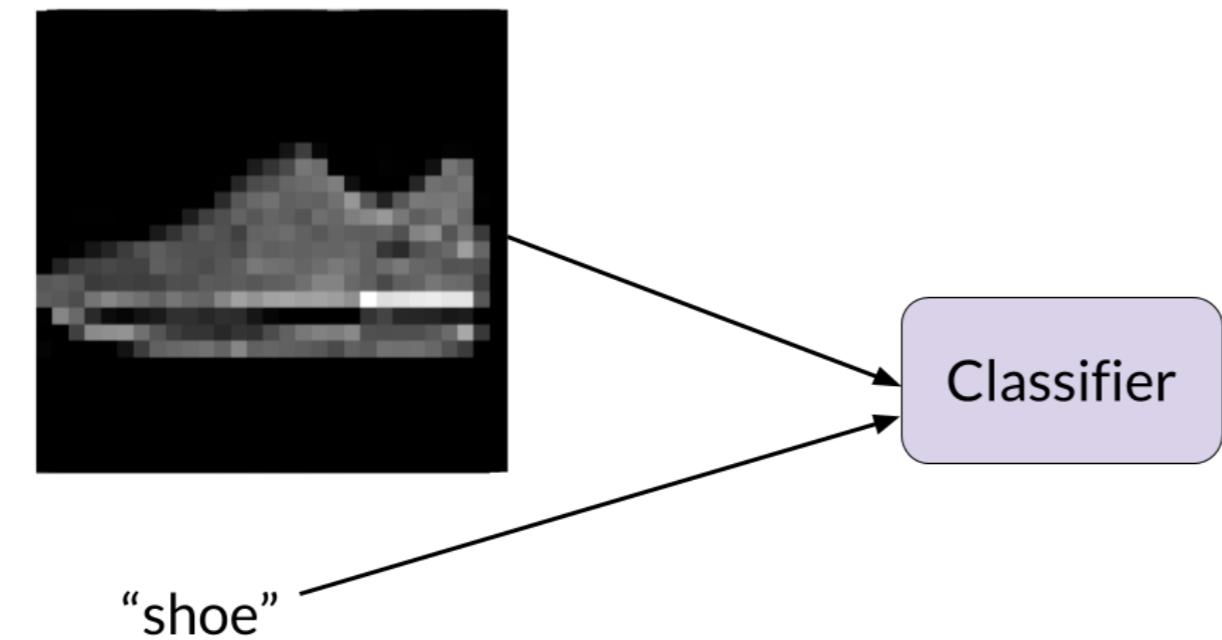


Image classification: training

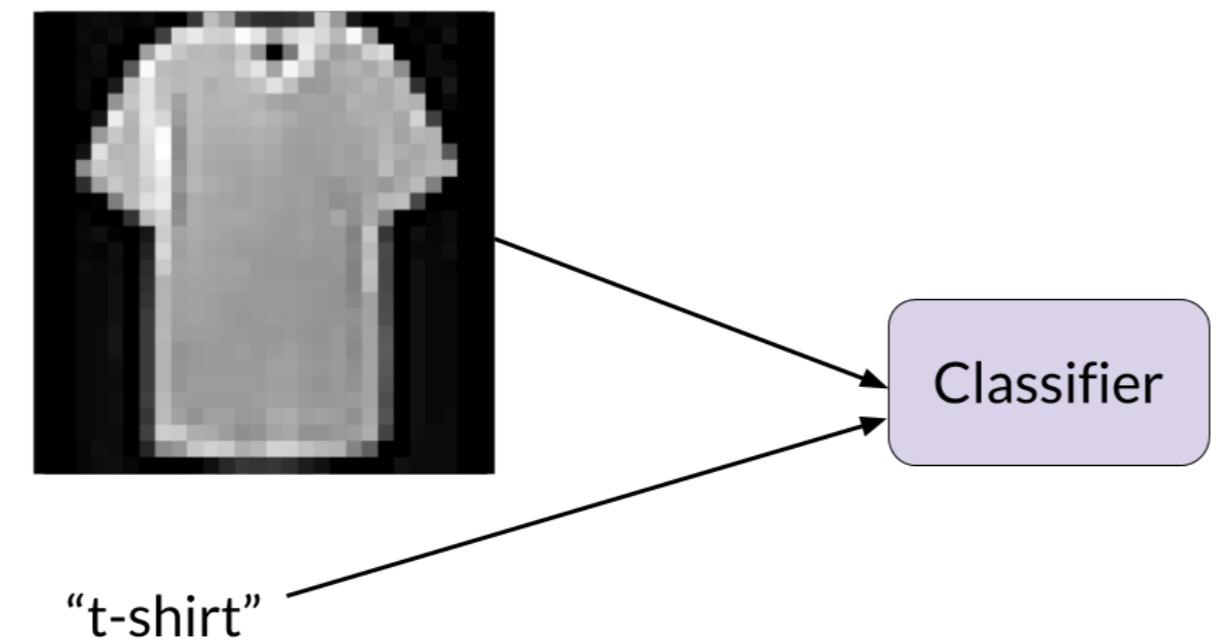


Image classification: training

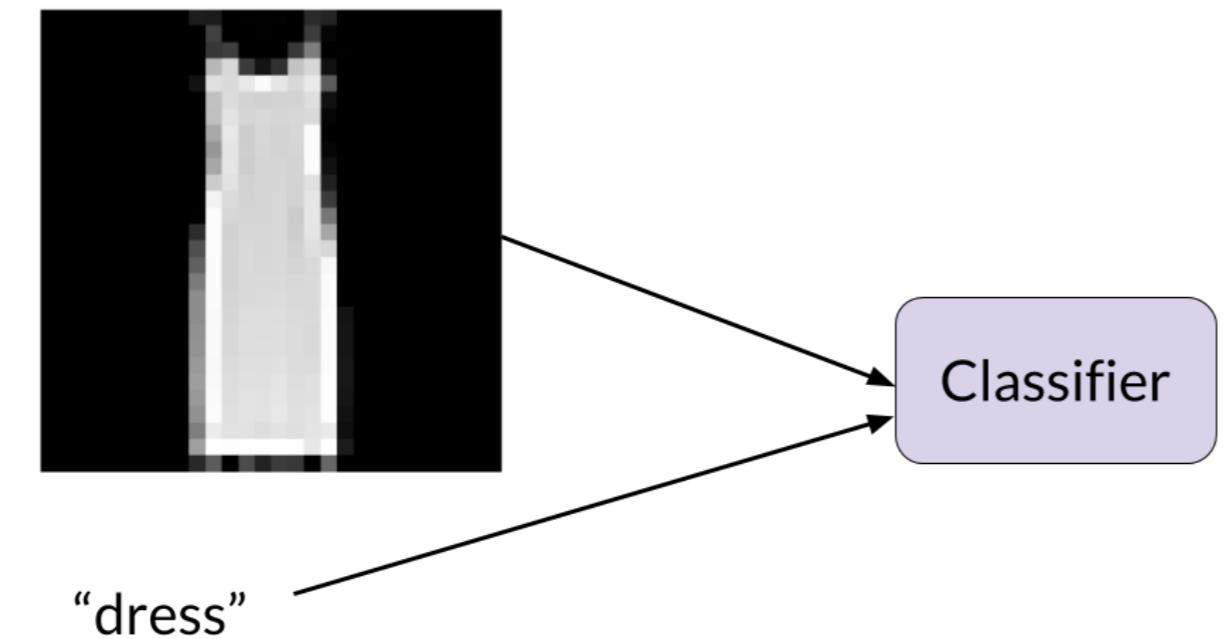


Image classification: evaluation

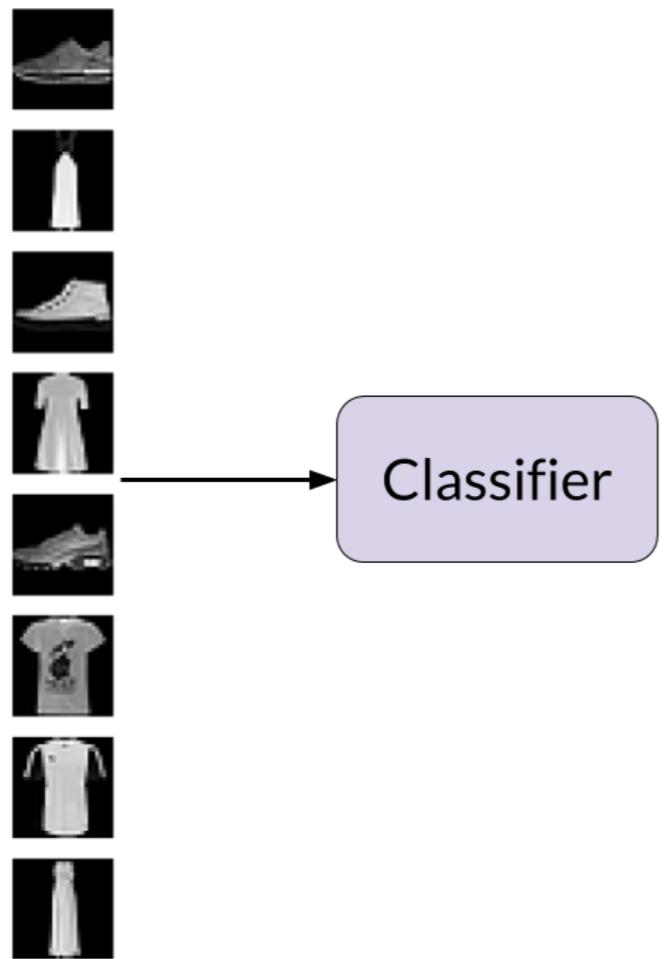
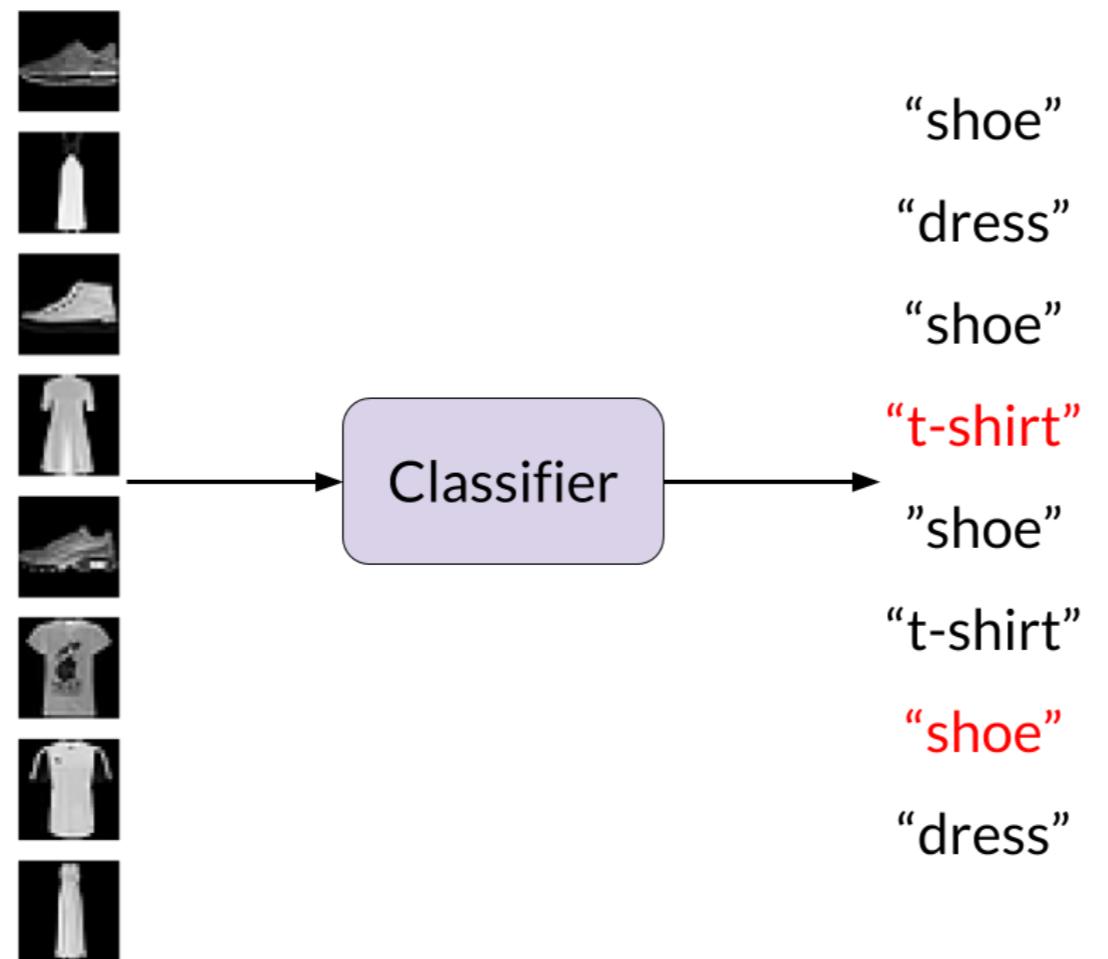


Image classification: evaluation



Representing class data: one-hot encoding

```
labels = ["shoe", "dress", "shoe", "t-shirt",
          "shoe", "t-shirt", "shoe", "dress"]
```

Representing class data: one-hot encoding

```
array([[0., 0., 1.],    <= shoe
       [0., 1., 0.],    <= dress
       [0., 0., 1.],    <= shoe
       [1., 0., 0.],    <= t-shirt
       [0., 0., 1.],    <= shoe
       [1., 0., 0.],    <= t-shirt
       [0., 0., 1.],    <= shoe
       [0., 1., 0.]])   <= dress
```

One-hot encoding

```
categories = np.array(["t-shirt", "dress", "shoe"])

n_categories = 3
ohe_labels = np.zeros((len(labels), n_categories))

for ii in range(len(labels)):

    jj = np.where(categories == labels[ii])

    ohe_labels[ii, jj] = 1
```

One-hot encoding: testing predictions

```
test  
  
array([[0., 0., 1.],  
       [0., 1., 0.],  
       [0., 0., 1.],  
       [0., 1., 0.],  
       [0., 0., 1.],  
       [0., 0., 1.],  
       [0., 0., 1.],  
       [0., 0., 1.],  
       [0., 1., 0.]])
```

```
(test * prediction).sum()  
6.0
```

```
prediction  
  
array([[0., 0., 1.],  
       [0., 1., 0.],  
       [0., 0., 1.],  
       [1., 0., 0.], <= incorrect  
       [0., 0., 1.],  
       [1., 0., 0.],  
       [0., 0., 1.], <= incorrect  
       [0., 1., 0.]])
```



CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

Let's practice!



CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

Image classification with Keras

Ariel Rokem

Senior Data Scientist, University of Washington

Keras for image classification

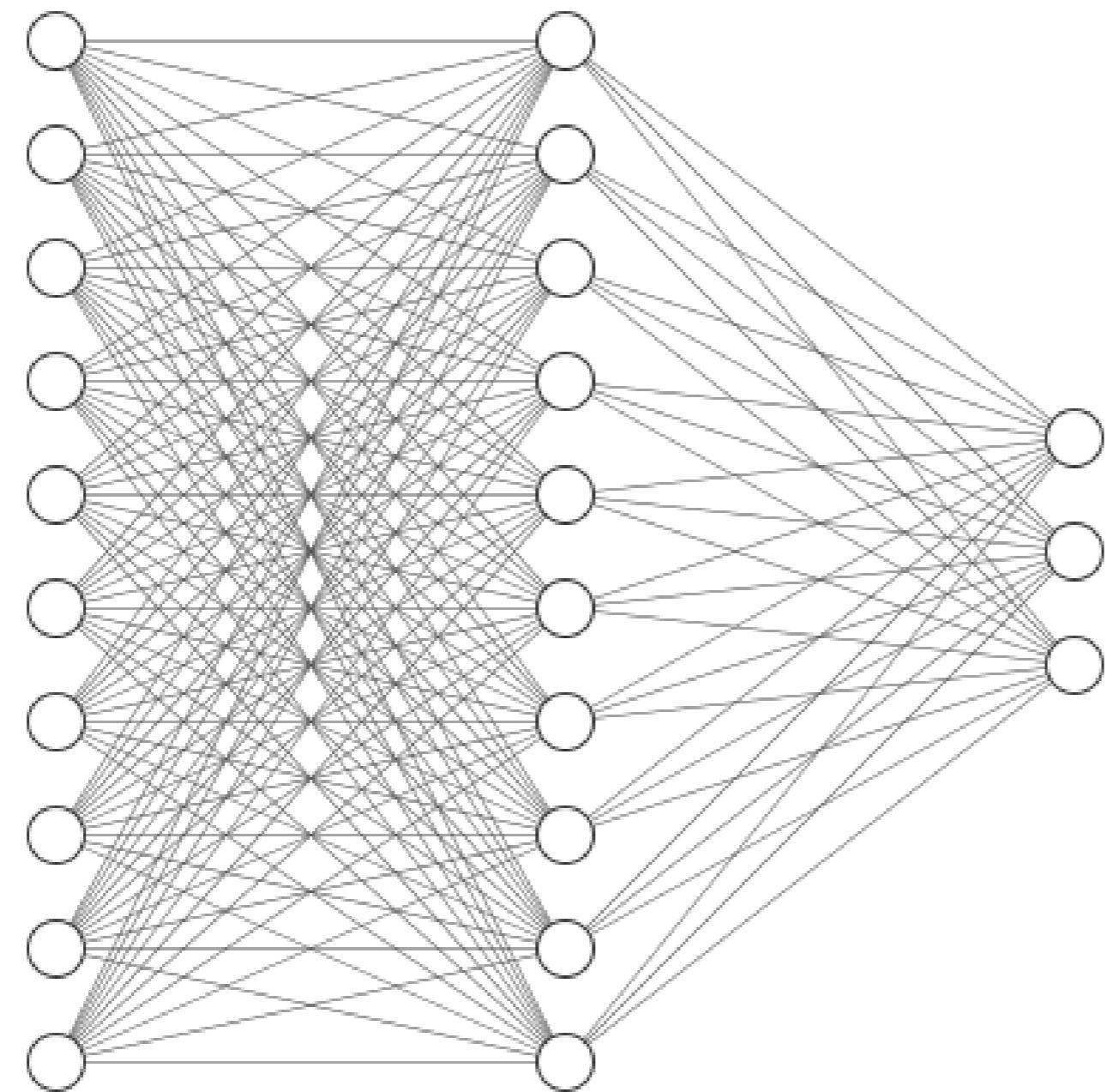
```
from keras.models import Sequential  
model = Sequential()
```

Keras for image classification

```
from keras.layers import Dense  
  
train_data.shape  
(50, 28, 28, 1)
```

Keras for image classification

```
model.add(Dense(10, activation='relu',  
    input_shape=(784,)))  
  
model.add(Dense(10, activation='relu'))  
  
model.add(Dense(3, activation='softmax'))
```

Input Layer $\in \mathbb{R}^{10}$ Hidden Layer $\in \mathbb{R}^{10}$ Output Layer $\in \mathbb{R}^3$

Keras for image classification

```
model.compile(optimizer='adam',  
              loss='categorical_crossentropy',  
              metrics=['accuracy'])
```

Keras for image classification

```
train_data = train_data.reshape(50, 784)
```

Keras for image classification

```
model.fit(train_data, train_labels,  
          validation_split=0.2,  
          epochs=3)
```

Keras for image classification

```
model.fit(train_data, train_labels,  
          validation_split=0.2,  
          epochs=3)  
  
Train on 40 samples, validate on 10 samples  
Epoch 1/3  
  
32/40 [=====>.....] - ETA: 0s - loss: 1.0117 - acc: 0.4688  
40/40 [=====] - 0s 4ms/step - loss: 1.0438 - acc: 0.4250 - val_loss: 0.9668 - val_acc: 0.4000  
Epoch 2/3  
  
32/40 [=====>.....] - ETA: 0s - loss: 0.9556 - acc: 0.5312  
40/40 [=====] - 0s 195us/step - loss: 0.9404 - acc: 0.5750 - val_loss: 0.9068 - val_acc: 0.4000  
Epoch 3/3  
  
32/40 [=====>.....] - ETA: 0s - loss: 0.9143 - acc: 0.5938  
40/40 [=====] - 0s 189us/step - loss: 0.8726 - acc: 0.6750 - val_loss: 0.8452 - val_acc: 0.4000
```

Keras for image classification

```
test_data = test_data.reshape((10, 784))  
model.evaluate(test_data, test_labels)  
  
10/10 [=====] - 0s 335us/step  
[1.0191701650619507, 0.400000059604645]
```



CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

Let's practice!