DUKE
FUQUA
SCHOOL OF BUSINESS

Sudipta Dasmohapatra
sd345@duke.edu

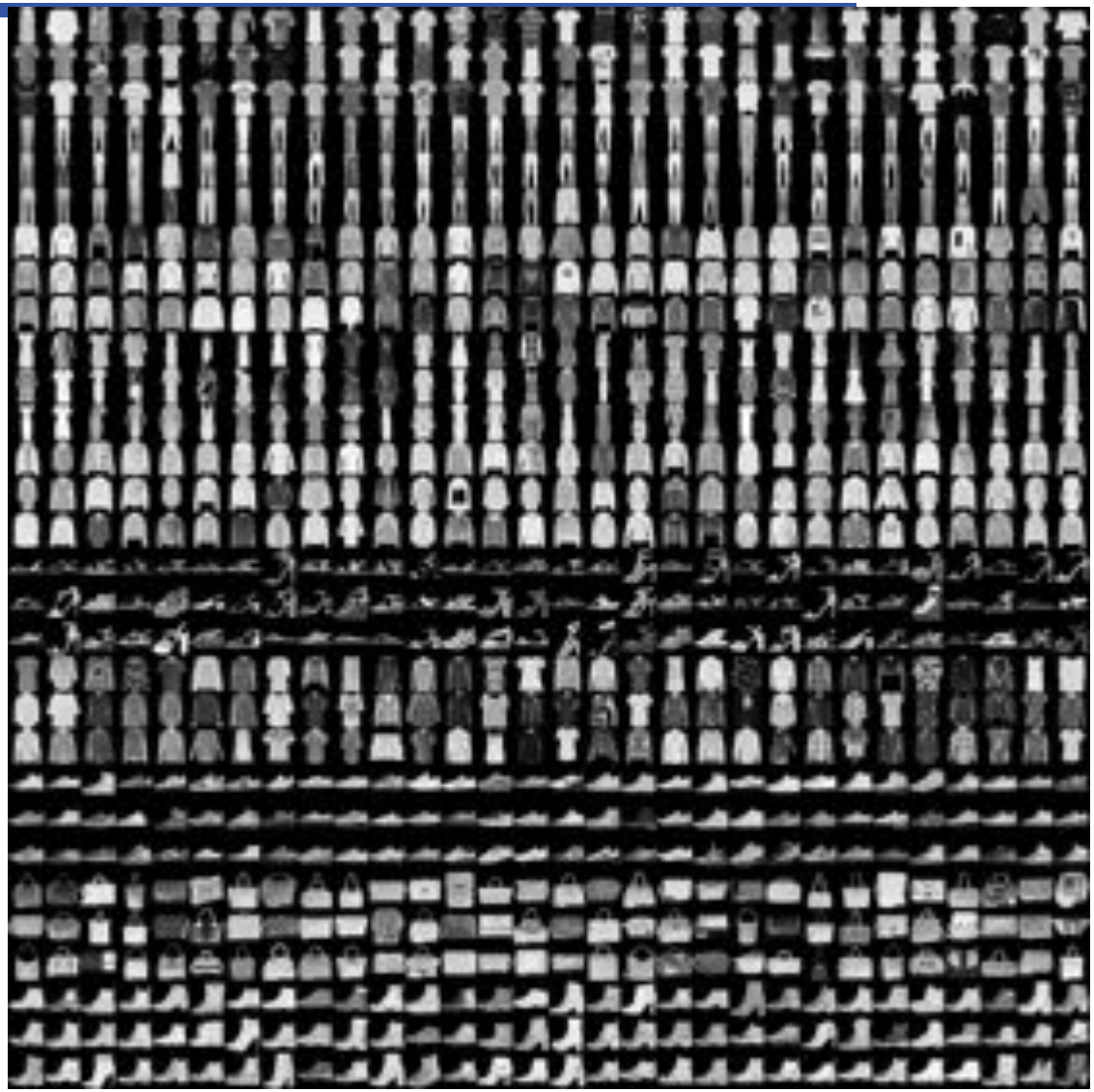
Neural Network and Deep Learning Workshop

Spring 2020

Example: The MNIST Fashion Database

Source: https://github.com/zalandoresearch/fashion-mnist

- The MNIST fashion dataset contains 70000 grayscale image in 10 categories
- The images show individual articles of clothing at low resolution (28x 28 pixels)
- Goal: Your goal in this project is to train a deep learning network (CNN) to classify these images of clothing



TensorFlow

- Deep Learning Library: Open source, Python, Google
- 117k + users
- TensorFlow.org: Blogs, YouTube discussions, DevSummit
- Entire ecosystem of tooling to work with
  Keras (API), TensorFlow.js (browser); Colaboratory (cloud), TensorBoard (visualization); TensorFlow Hub (graph-modules); TensorFlow Lite (phone)
- Competitors: PyTorch, MXNet

Example: Classify Images of Clothing

- The TensorFlow tutorials are written as Jupyter notebooks and run directly in Google Colab—a hosted notebook environment that requires no setup. Click the *Run in Google Colab* button.
- We will run the code for Deep Learning using Tensorflow using Google Collaborate (this provides a platform for you to run the code snippets and observe your output on google)

https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/r2/tutorials/quickstart/beginner.ipynb#scrollTo=-Z-zA3EIfKpy

CNN: Tensorflow Code on Google Collaborate

```
[3]  from __future__ import absolute_import, division, print_function

     # TensorFlow and tf.keras
     import tensorflow as tf
     from tensorflow import keras

     # Helper libraries
     import numpy as np
     import matplotlib.pyplot as plt

     print(tf.__version__)
```

```
1.13.1
```

This example uses tf.keras, a high-level API to build and train models in TensorFlow. The first function imports the Tensorflow and tf.keras and all the helper libraries for python

DUKE FUQUA SCHOOL OF BUSINESS

CNN: Tensorflow Code on Google Collaborate

DUKE FUQUA SCHOOL OF BUSINESS

CNN: Import the MNIST Fashion Dataset

Training: 60,000 images

Test: 10,000 images

You can access the Fashion MNIST directly from TensorFlow, just import and load the data from the github library: https://github.com/zalandoresearch/fashion-mnist

```
[4] fashion_mnist = keras.datasets.fashion_mnist

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
```

DUKE FUQUA SCHOOL OF BUSINESS

Labels for Images

Label	Class
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

The *labels* are an array of integers, ranging from 0 to 9. These correspond to the *class* of clothing the image represents.

```
[5]  class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
                    'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

Explore the data

```
[6]  train_images.shape
```

```
(60000, 28, 28)
```

```
[7]  len(train_labels)
```

```
60000
```

```
[8]  train_labels
```

```
array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)
```

```
[9]  test_images.shape
```

```
(10000, 28, 28)
```

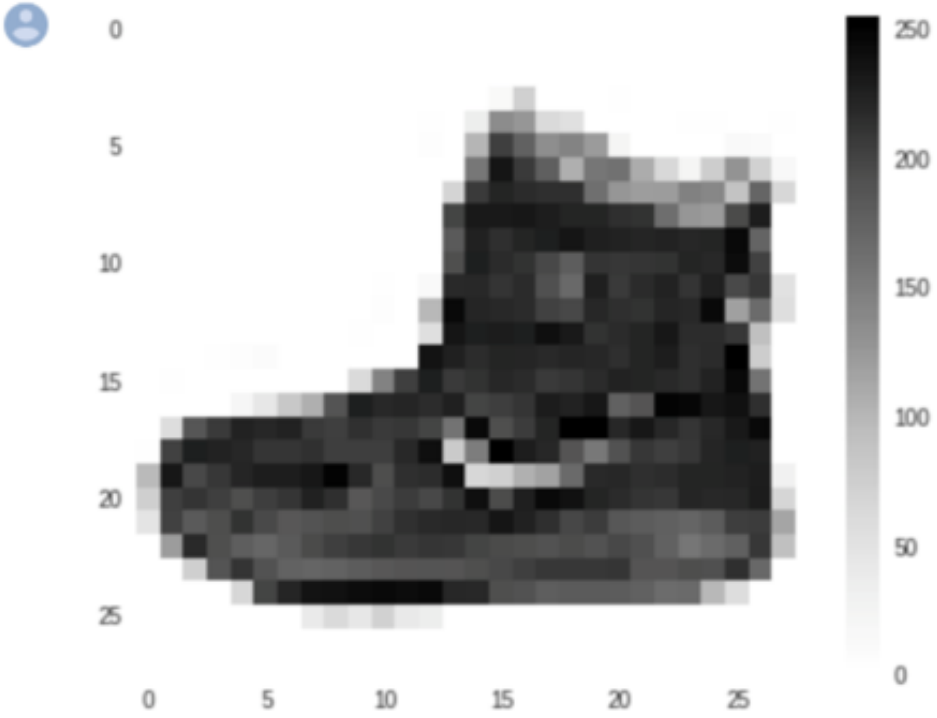
```
[10] len(test_labels)
```

```
10000
```

DUKE FUQUA SCHOOL OF BUSINESS

Pre-Process the Data

```
[11] plt.figure()
     plt.imshow(train_images[0])
     plt.colorbar()
     plt.grid(False)
     plt.show()
```




```
[12] train_images = train_images / 255.0

     test_images = test_images / 255.0
```

DUKE FUQUA SCHOOL OF BUSINESS

```
[13] plt.figure(figsize=(10,10))
     for i in range(25):
         plt.subplot(5,5,i+1)
         plt.xticks([])
         plt.yticks([])
         plt.grid(False)
         plt.imshow(train_images[i], cmap=plt.cm.binary)
         plt.xlabel(class_names[train_labels[i]])
     plt.show()
```

DUKE FUQUA SCHOOL OF BUSINESS

Build the Model: Set up Layers

```
[15] 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)
    ])
```

- First Layer: flatten layer (transforms images)
- Second ReLu Layer: 128 nodes
- Third Pooled Layer: Softmax method: 10 probability scores (10 classes)

DUKE FUQUA SCHOOL OF BUSINESS

Build the Model: Setting

Add:

- LOSS FUNCTION: Minimize loss to move model to the right direction
- OPTIMIZER: Measures how model is updated based on data and loss function
- METRICS: Monitor the training and testing steps (accuracy= fraction of images that are correctly classified)

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

DUKE FUQUA SCHOOL OF BUSINESS

Training the Model

Training the neural network model requires the following steps:

- Feed the training data to the model
- The model learns to associate images and labels
- Ask the model to make predictions about a test set
- Verify that the predictions match the labels from the test_labels array

Training the Model

Train the neural network; Indicate number of **epochs**

An **epoch** is a full iteration over samples. The number of **epochs** is how many times the algorithm is going to run. We indicated 5 iterations for the model.

You can change that – it should be at least 2 for best results.

```
[15] model.fit(train_images, train_labels, epochs=5)

Epoch 1/5
60000/60000 [==============================] - 7s 114us/sample - loss: 0.4954 - acc: 0.8263
Epoch 2/5
60000/60000 [==============================] - 6s 106us/sample - loss: 0.3744 - acc: 0.8648
Epoch 3/5
60000/60000 [==============================] - 6s 106us/sample - loss: 0.3355 - acc: 0.8778
Epoch 4/5
60000/60000 [==============================] - 6s 103us/sample - loss: 0.3115 - acc: 0.8853
Epoch 5/5
60000/60000 [==============================] - 6s 103us/sample - loss: 0.2927 - acc: 0.8913
<tensorflow.python.keras.callbacks.History at 0x7f3f3a185dd8>
```

89% accuracy on training data

Testing the Model

```
[16] test_loss, test_acc = model.evaluate(test_images, test_labels)

     print('Test accuracy:', test_acc)
```

87% accuracy on test data

```
10000/10000 [==============================] - 0s 46us/sample - loss: 0.3439 - acc: 0.8751
Test accuracy: 0.8751
```

DUKE FUQUA SCHOOL OF BUSINESS

Making Predictions

87% accuracy on test data

```
[17] predictions = model.predict(test_images)
```

```
[18] predictions[0]
```

```
array([6.81665333e-05, 9.80569297e-08, 4.42758761e-07, 2.44479043e-06,
       3.70949374e-06, 1.09513134e-01, 2.64959599e-05, 5.02966978e-02,
       1.98009261e-03, 8.38108659e-01], dtype=float32)
```

```
[19] np.argmax(predictions[0])
```

```
9
```

```
[20] test_labels[0]
```

```
9
```

Making Predictions

We can graph this to look at the full set of 10 channels

```
[21] def plot_image(i, predictions_array, true_label, img):
       predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
       plt.grid(False)
       plt.xticks([])
       plt.yticks([])

       plt.imshow(img, cmap=plt.cm.binary)

       predicted_label = np.argmax(predictions_array)
       if predicted_label == true_label:
         color = 'blue'
       else:
         color = 'red'

       plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                    100*np.max(predictions_array),
                                    class_names[true_label]),
                                    color=color)

     def plot_value_array(i, predictions_array, true_label):
       predictions_array, true_label = predictions_array[i], true_label[i]
       plt.grid(False)
       plt.xticks([])
       plt.yticks([])
       thisplot = plt.bar(range(10), predictions_array, color="#777777")
       plt.ylim([0, 1])
       predicted_label = np.argmax(predictions_array)

       thisplot[predicted_label].set_color('red')
       thisplot[true_label].set_color('blue')
```

DUKE FUQUA SCHOOL OF BUSINESS

Making Predictions

```
[22] i = 0
     plt.figure(figsize=(6,3))
     plt.subplot(1,2,1)
     plot_image(i, predictions, test_labels, test_images)
     plt.subplot(1,2,2)
     plot_value_array(i, predictions,  test_labels)
     plt.show()
```

Image labeled 0 in test set, the ankle boot class has been predicted as the highest probability at 84%. This image was actually an ankle boot (shown in parenthesis underneath the image) so the model predicted this image correctly

DUKE FUQUA SCHOOL OF BUSINESS

Making Predictions

```
[23] i = 12
     plt.figure(figsize=(6,3))
     plt.subplot(1,2,1)
     plot_image(i, predictions, test_labels, test_images)
     plt.subplot(1,2,2)
     plot_value_array(i, predictions,  test_labels)
     plt.show()
```

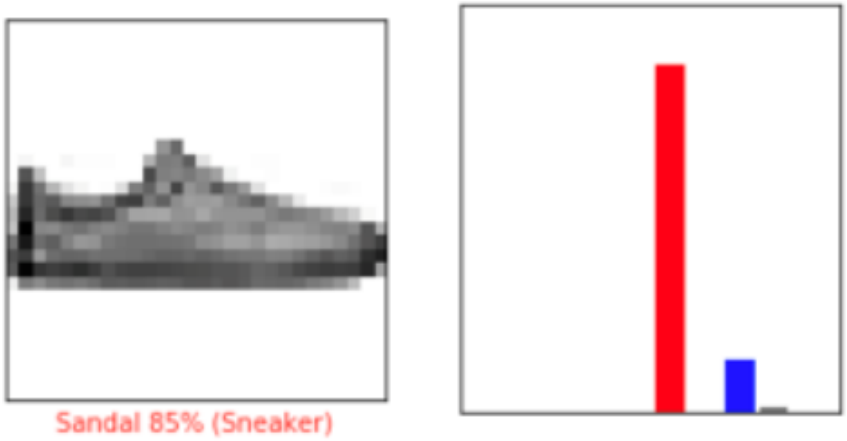


Image 12 in our test set is incorrectly predicted as a sandal with 85% accuracy or probability whereas in reality it is a sneaker.

Remember that your model is only 87% accurate so there will be some errors in prediction.

DUKE FUQUA SCHOOL OF BUSINESS

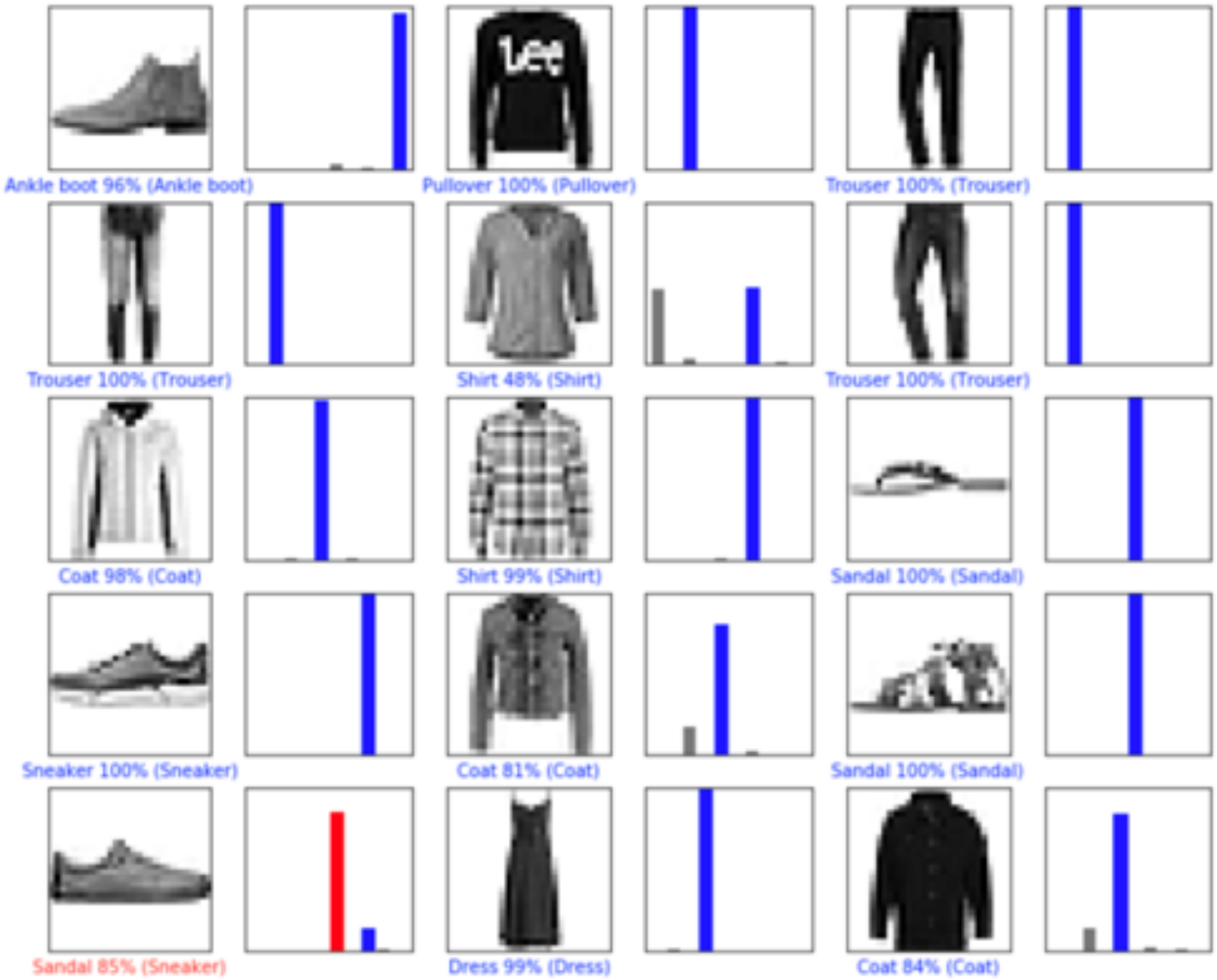
Making Predictions

```
[24] # Plot the first X test images, their predicted label, and the true label
     # Color correct predictions in blue, incorrect predictions in red
     num_rows = 5
     num_cols = 3
     num_images = num_rows*num_cols
     plt.figure(figsize=(2*2*num_cols, 2*num_rows))
     for i in range(num_images):
       plt.subplot(num_rows, 2*num_cols, 2*i+1)
       plot_image(i, predictions, test_labels, test_images)
       plt.subplot(num_rows, 2*num_cols, 2*i+2)
       plot_value_array(i, predictions, test_labels)
     plt.show()
```

We ask for plotting images in 5 rows with 3 columns so total of 15 images

DUKE FUQUA SCHOOL OF BUSINESS

Making Predictions



DUKE FUQUA SCHOOL OF BUSINESS

Summary

- Deep learning concepts and principles
- Neural networks and the modeling procedure (loss functions, back propagation)
- Activation functions
- Convolutional neural networks and recurrent neural networks
- Tensor flow and example

DUKE
FUQUA
SCHOOL OF BUSINESS

Thanks

sd345@duke.edu

DUKE FUQUA SCHOOL OF BUSINESS