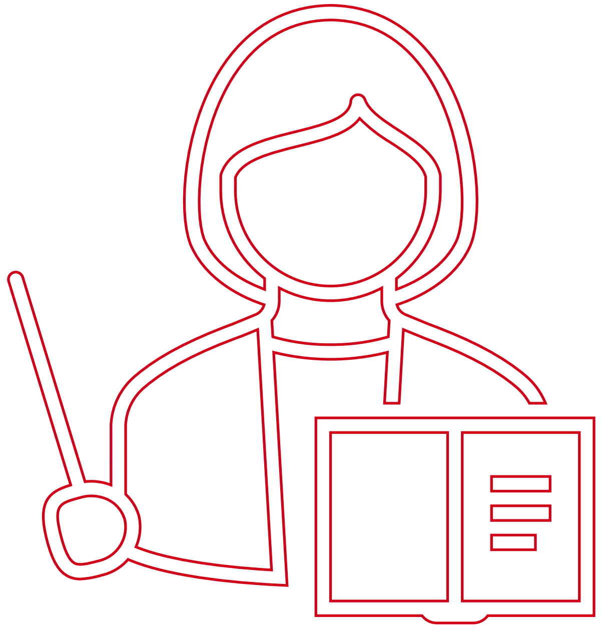
Session 7

1. Your first steps in Python for Machine Learning
2. The Iris ML problem
3. Exploring the Iris dataset
4. Simple statistical classifier generation in python
5. Neuronal Network SLP/ MLP in Python
6. Create your own dataset, data collection, and processing
7. **Implementing supervised / unsupervised learning on datasets.**
8. Machine learning example and briefing of final project



Topics for today

1. Task 6 review
2. Exploring an example with Fashion-Minst dataset
3. Defining Task 7
    1. How to classify your own data with help of Keras **TensorFlow**.
    2. Investigating options for improving accuracy.

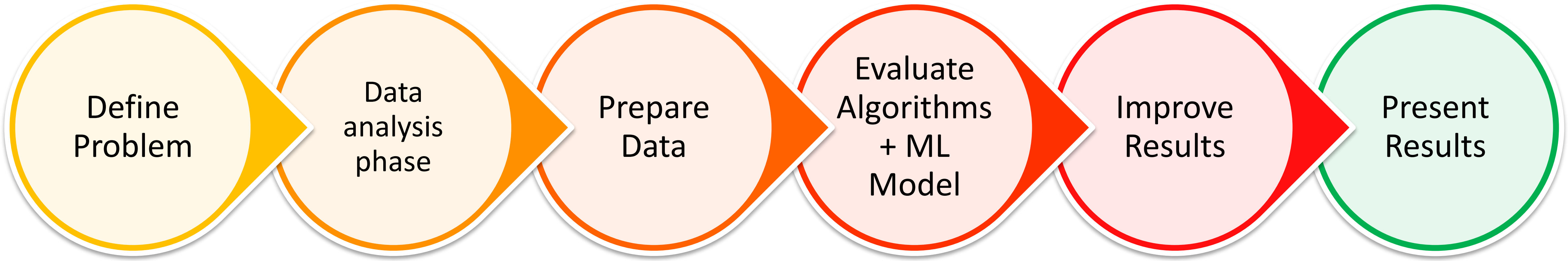
Task 6 Solution

Lets review your dataset!

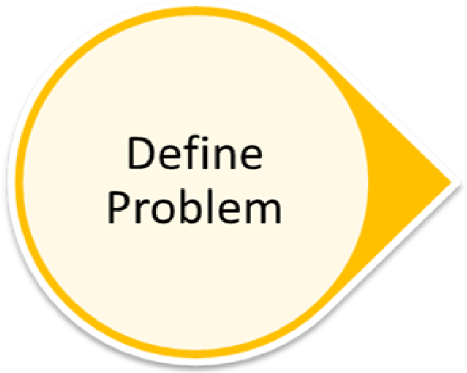
Neural Network

Task 7: TensorFlow

Remember our overarching Machine Learning process



Exploring the Fashion MNIST dataset example with TensorFlow



Context
Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Zalando intends Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

The original MNIST dataset contains a lot of handwritten digits. Members of the AI/ML/Data Science community love this dataset and use it as a benchmark to validate their algorithms. In fact, MNIST is often the first dataset researchers try. "If it doesn't work on MNIST, it won't work at all", they said. "Well, if it does work on MNIST, it may still fail on others."

Zalando seeks to replace the original MNIST dataset

Exploring the Fashion MNIST dataset example with TensorFlow

Data analysis phase

1. The datasets consists of 60,000 examples and a test set of 10,000 examples
2. Each image (sample) is 28 pixels in height and 28 pixels in width (28*28=784pixels)
3. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker.
4. This pixel-value is an integer between 0 and 255.
5. Each row is a separate image.
6. The training and test data sets have 785 columns.
7. Column 1 is the class label.
8. Remaining columns are pixel numbers (784 total).
9. Each value is the darkness of the pixel (1 to 255)

To locate a pixel on the image, suppose that we have decomposed x as x = i * 28 + j, where i and j are integers between 0 and 27. The pixel is located on row i and column j of a 28 x 28 matrix.
For example, pixel31 indicates the pixel that is in the fourth column from the left, and the second row from the top, as in the ascii-diagram below.

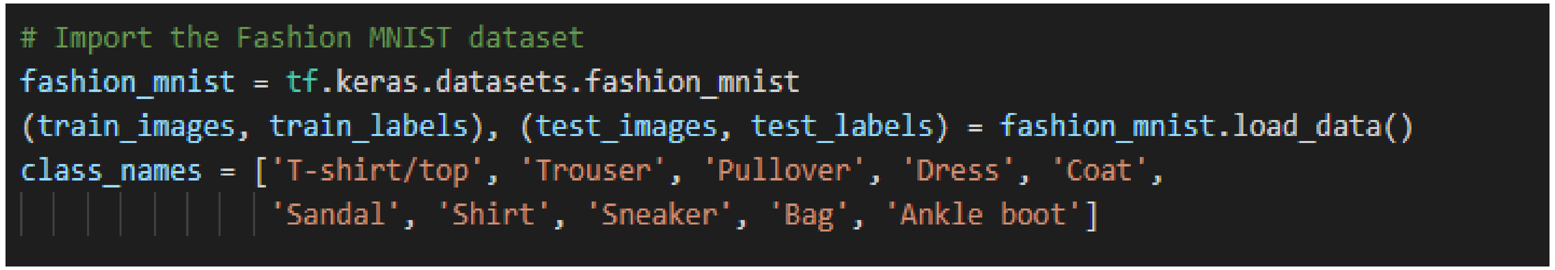
Exploring the Fashion MNIST dataset example with TensorFlow

Prepare Data

Resource on Moodle: Session7_Example_fashion_mnist.py

Step1 : Import the Fashion MNIST dataset

For original tutorial see - https://www.tensorflow.org/tutorials/keras/classification

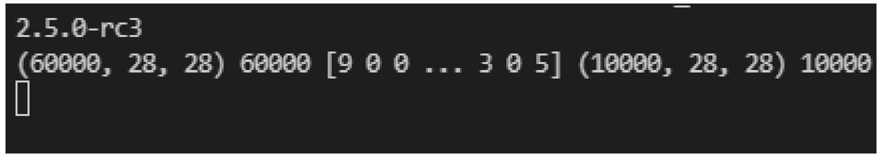


```
# Import the Fashion MNIST dataset
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

Resource for **tf.keras.datasets.fashion_mnist.load_data** = https://www.tensorflow.org/api_docs/python/tf/keras/datasets/fashion_mnist/load_data

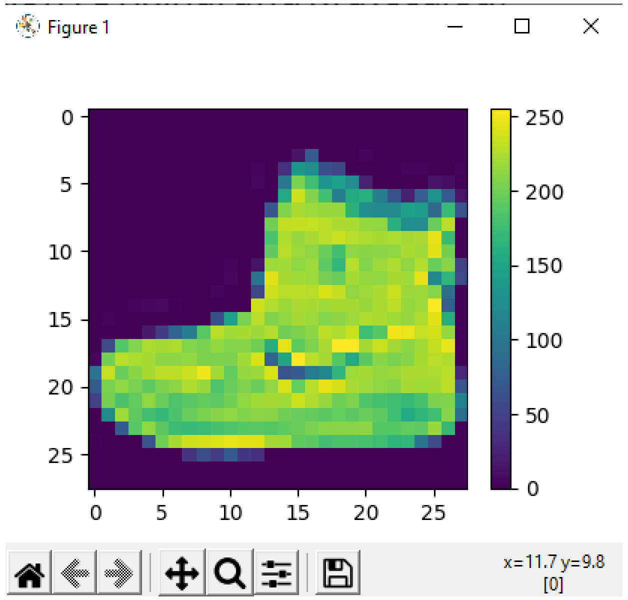
Exploring the Fashion MNIST dataset example with TensorFlow

Step 2: Explore and process the data

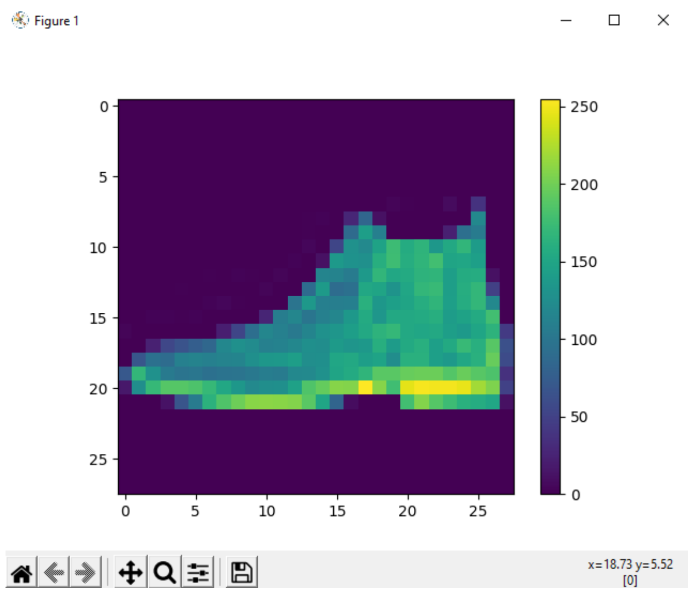


```
2.5.0-rc3
(60000, 28, 28) 60000 [9 0 0 ... 3 0 5] (10000, 28, 28) 10000
```

train_image[0]

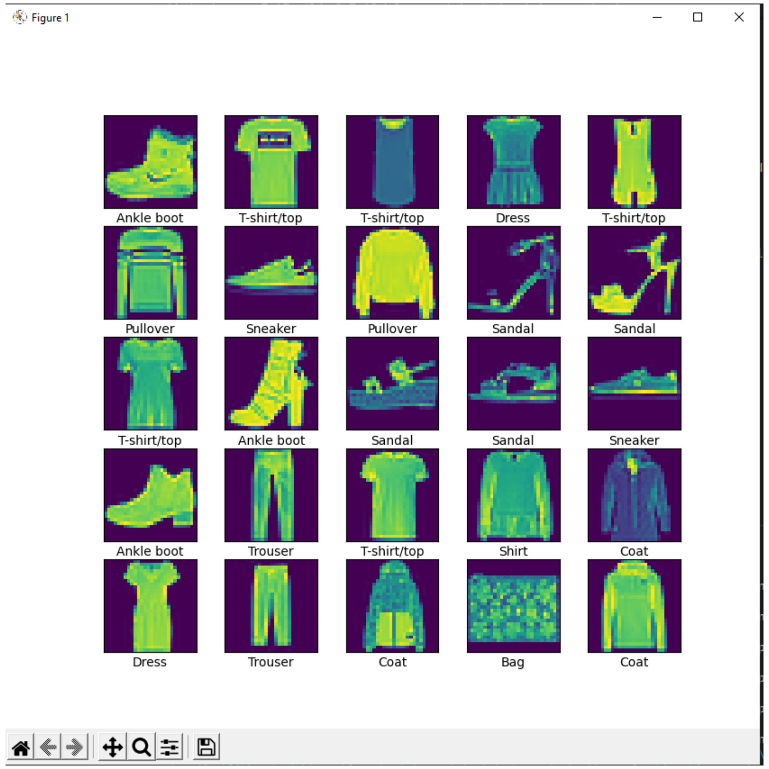


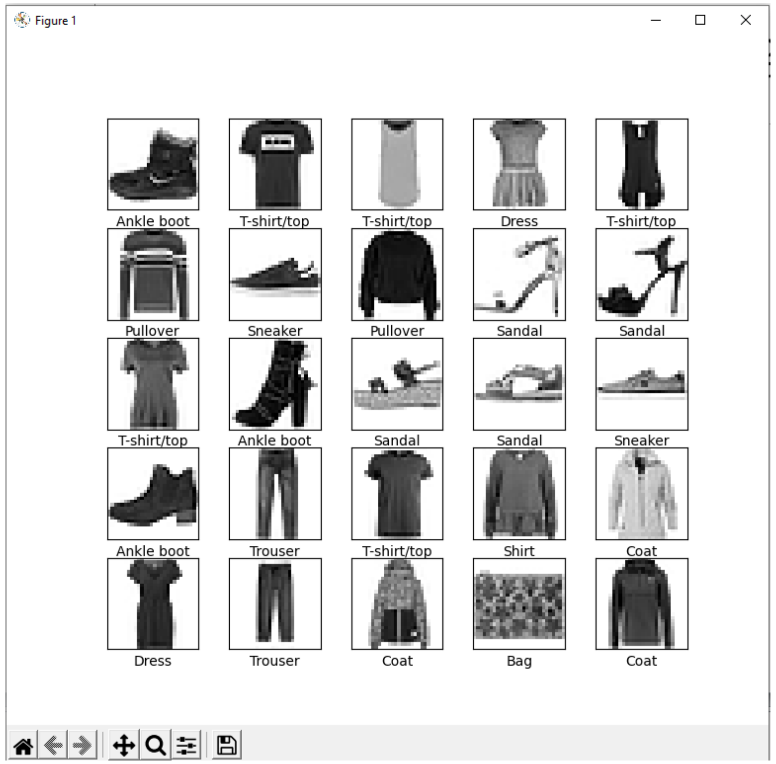
test_image[0]



Exploring the Fashion MNIST dataset example with TensorFlow

To verify that the data is in the correct format and that you're ready to build and train the network, let's display the first 25 images from the *training set* and display the class name below each image.

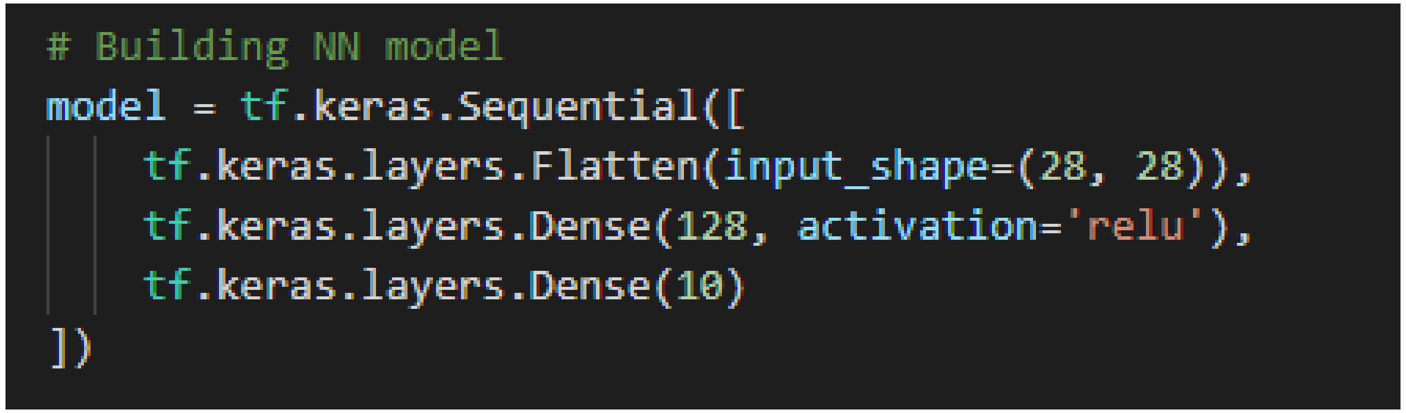




Exploring the Fashion MNIST dataset example with TensorFlow

Evaluate Algorithms + compile ML Model

Step 1: Spot check algorithm



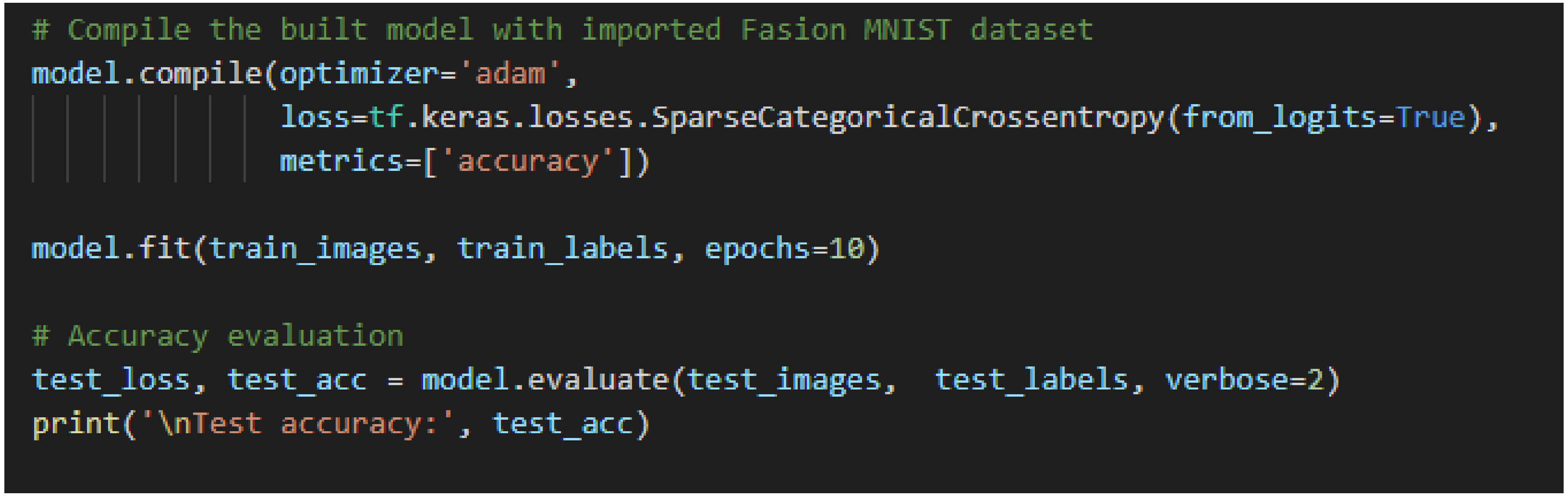
```
# Building NN model
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])
```

Important before we can train, we need to COMPILE THE MODEL!
Before the model is ready for training, it needs a few more settings. These are added during the model's compile step:

- **Loss function** —This measures how accurate the model is during training. You want to minimize this function to "steer" the model in the right direction.
- **Optimizer** —This is how the model is updated based on the data it sees and its loss function.
- **Metrics** —Used to monitor the training and testing steps. The following example uses accuracy, the fraction of the images that are correctly classified.

Exploring the Fashion MNIST dataset example with TensorFlow

Step 2: COMPILE THE ML MODEL



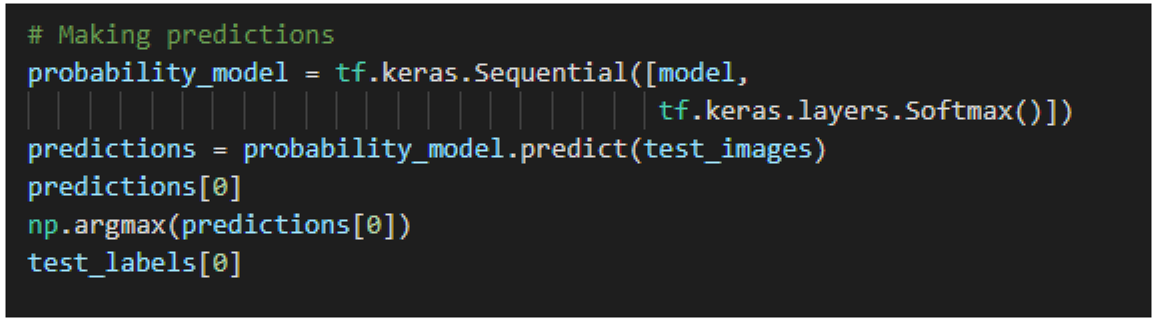
```
# Compile the built model with imported Fasion MNIST dataset
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=10)

# Accuracy evaluation
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print('\nTest accuracy:', test_acc)
```

Exploring the Fashion MNIST dataset example with TensorFlow

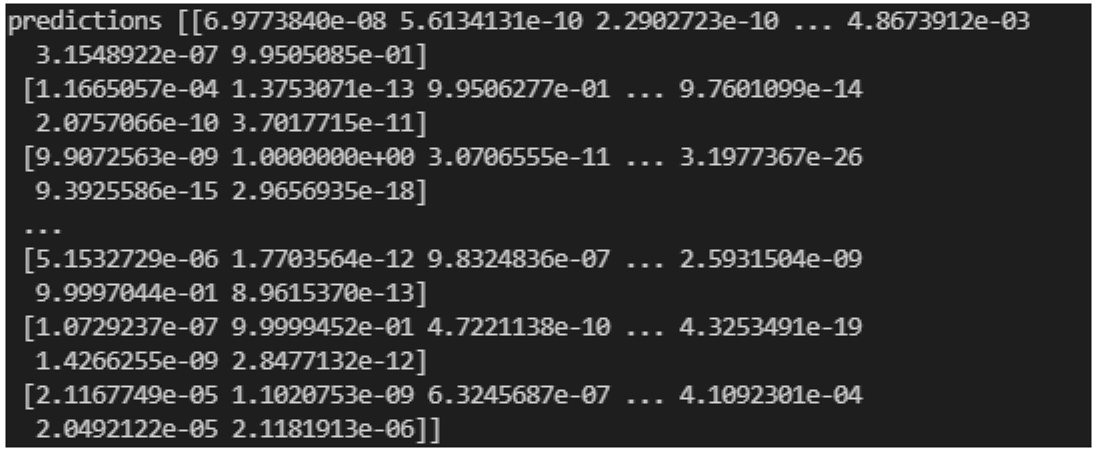
Step 3: Train the Model

Training the neural network model requires the following steps:

1. Feed the training data to the model.
2. The model needs to learn to associate images and labels.
3. Next you ask the model to make predictions about a test set
4. You then verify that the predictions match the labels from the test set.



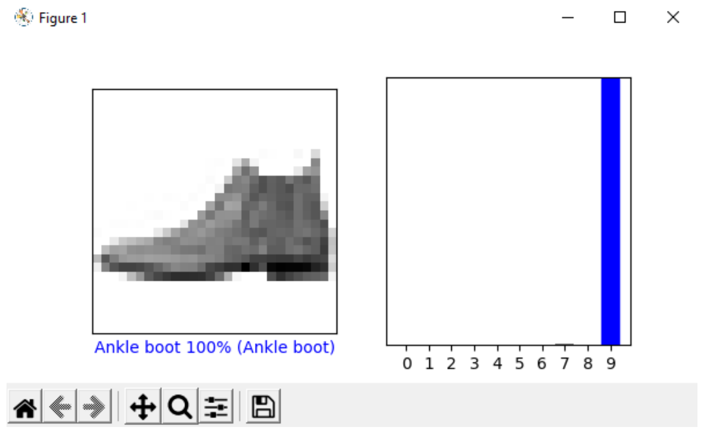
```
# Making predictions
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)
predictions[0]
np.argmax(predictions[0])
test_labels[0]
```

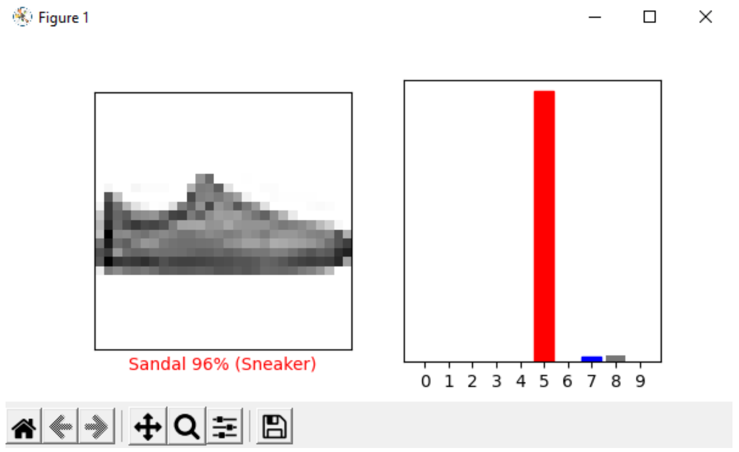


```
predictions [[6.9773840e-08 5.6134131e-10 2.2902723e-10 ... 4.8673912e-03
  3.1548922e-07 9.9505085e-01]
 [1.1665057e-04 1.3753071e-13 9.9506277e-01 ... 9.7601099e-14
  2.0757066e-10 3.7017715e-11]
 [9.9072563e-09 1.0000000e+00 3.0706555e-11 ... 3.1977367e-26
  9.3925586e-15 2.9656935e-18]
 ...
 [5.1532729e-06 1.7703564e-12 9.8324836e-07 ... 2.5931504e-09
  9.9997044e-01 8.9615370e-13]
 [1.0729237e-07 9.9999452e-01 4.7221138e-10 ... 4.3253491e-19
  1.4266255e-09 2.8477132e-12]
 [2.1167749e-05 1.1020753e-09 6.3245687e-07 ... 4.1092301e-04
  2.0492122e-05 2.1181913e-06]]
```

Exploring the Fashion MNIST dataset example with TensorFlow

Step 3: Train the Model Results

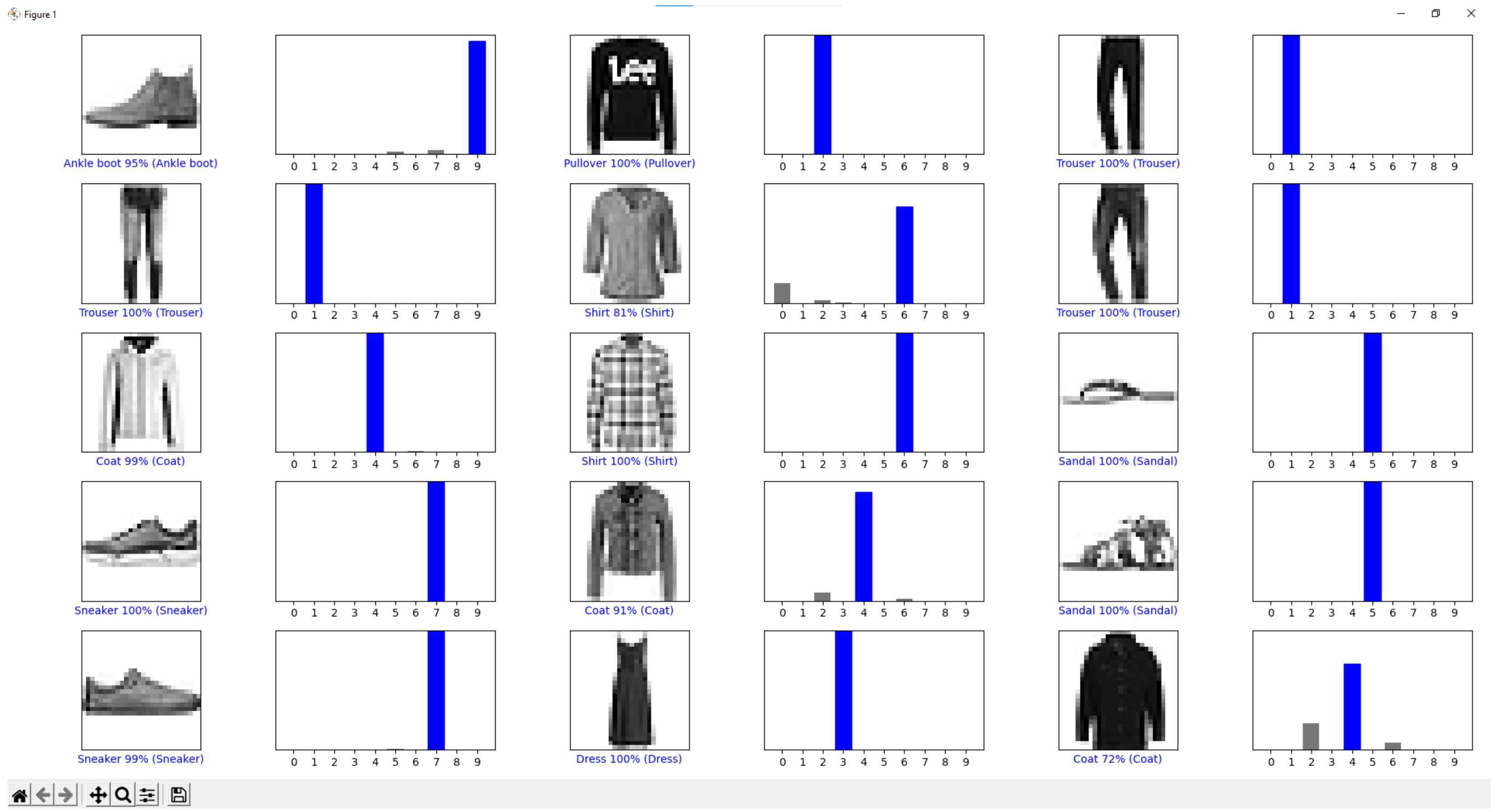




```
predictions 1st test image and winner [6.97738400e-08 5.61341307e-10 2.29027228e-10 7.75141284e-09
 1.50154822e-09 8.12685321e-05 1.09073824e-07 4.86739120e-03
 3.15489217e-07 9.95050848e-01] 9 9
predictions 12th test image and winner [1.6476913e-10 3.8446513e-09 3.9075574e-09 1.4675928e-07 4.7285031e-10
 9.6093887e-01 2.7022354e-08 1.6613457e-02 2.2447506e-02 2.5323011e-08] 5 7
```

Exploring the Fashion MNIST dataset example with TensorFlow

Part of Step 3: Evaluate the results

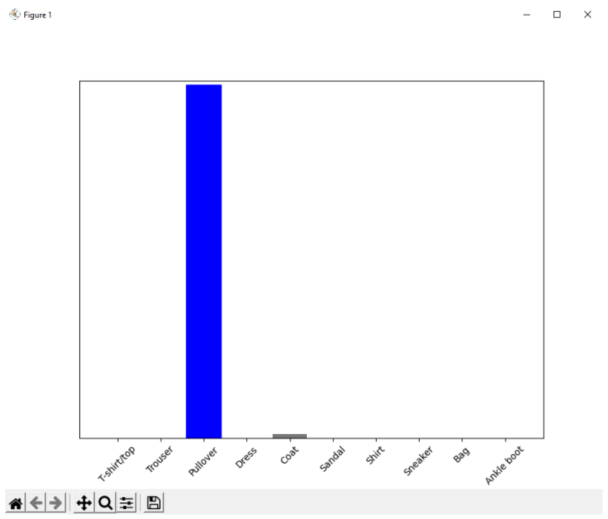


Exploring the Fashion MNIST dataset example with TensorFlow

Making Use of the ML model to make predictions

Present Results

```python
# Using the trained ANN model
# Grab an image from the test dataset.
img = test_images[1]
print(img.shape)
# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))
print(img.shape)

predictions_single = probability_model.predict(img)
print(predictions_single)

plot_value_array(1, predictions_single[0], test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)
plt.show()
print(np.argmax(predictions_single[0]))
```



```
(28, 28)
(1, 28, 28)
[[1.16650685e-04 1.37532021e-13 9.95062768e-01 2.27005484e-10
  4.49377531e-03 6.52373086e-11 3.26856156e-04 9.76010993e-14
  2.07570267e-10 3.70177812e-11]]
2
```

Wait, didn't we miss a step????

Improve Results

How do we improve accuracy???

1. More data?
2. Increasing the epochs/ training cycles? [Overfitting?]
3. Different Optimization methods?
4. More hidden layers?

We are looking forward to hearing your answers!!!

Task 7: Improve the Supervised Kitchenware ML model

We have created a Supervised classification ML model, now your team needs, understand, implement and improve it!

The aim of this task is to familiarize how to:

1. Initializing a Supervised ANN model to classify your own data with the help of TensorFlow + Kera
2. Train and test the model on your own dataset
3. Methods to improve the accuracy of the model with combined datasets.

We make use of our own pre-processed image database. The objects of interest for classifications are still cups, bowls, and plates.

Task 7.1

1. Implement the provided code

2. Train and test the Kitchenware ML Model on your Data.

Task 7.2

Improve accuracy by adding the data created by the other groups

Task 7.3

Explore different methods to improve the accuracy result in TensorFlow + Keras.

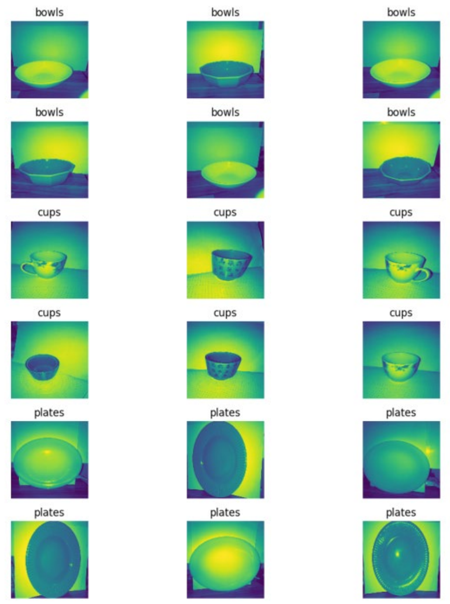
Task 7.1: Implement the provided code

This is the continuation of from Task 6. For Task 7.1:

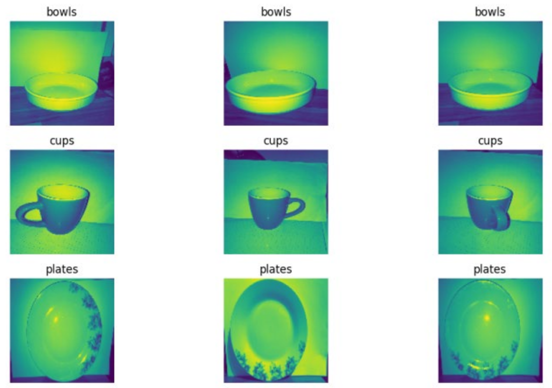
a. Implement the provided code – **Task7_Train_test_classify.py**
b. Train and test the Kitchenware ML Model on the **previously provided data** + **your data**

The example shown here only uses the provided data.

Provided training data

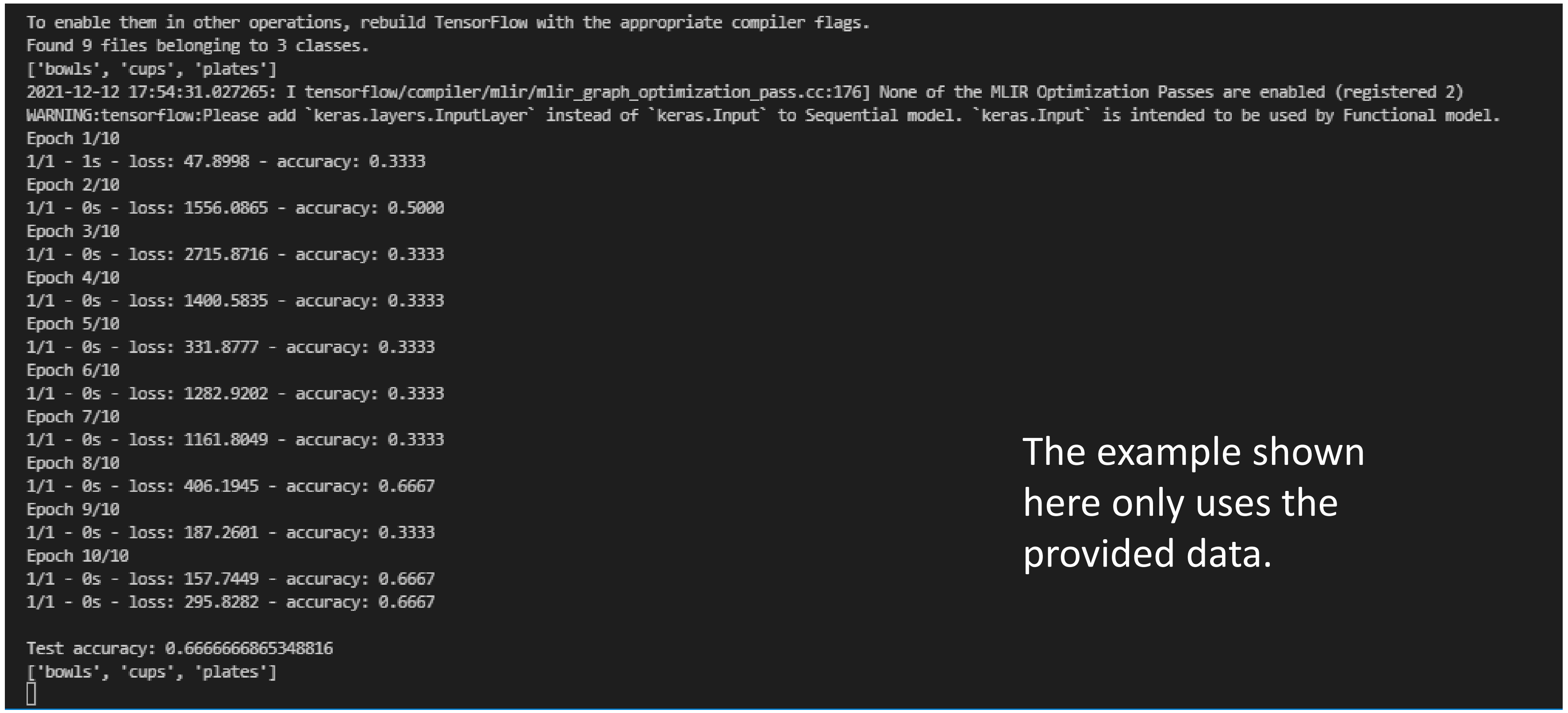


Provided testing data



Task 7.1: Implement the provided code

Making use of Keras TensorFlow to train and evaluate accuracy



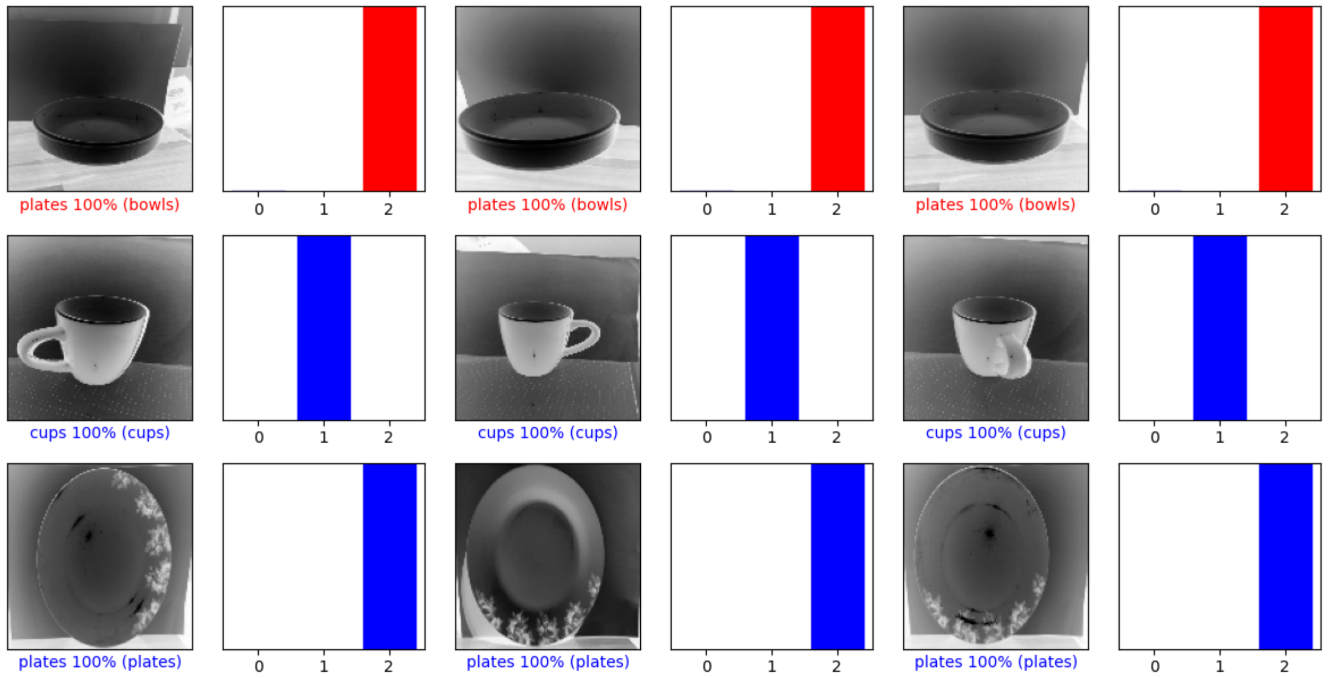
```
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Found 9 files belonging to 3 classes.
['bowls', 'cups', 'plates']
2021-12-12 17:54:31.027265: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:176] None of the MLIR Optimization Passes are enabled (registered 2)
WARNING:tensorflow:Please add `keras.layers.InputLayer` instead of `keras.Input` to Sequential model. `keras.Input` is intended to be used by Functional model.
Epoch 1/10
1/1 - 1s - loss: 47.8998 - accuracy: 0.3333
Epoch 2/10
1/1 - 0s - loss: 1556.0865 - accuracy: 0.5000
Epoch 3/10
1/1 - 0s - loss: 2715.8716 - accuracy: 0.3333
Epoch 4/10
1/1 - 0s - loss: 1400.5835 - accuracy: 0.3333
Epoch 5/10
1/1 - 0s - loss: 331.8777 - accuracy: 0.3333
Epoch 6/10
1/1 - 0s - loss: 1282.9202 - accuracy: 0.3333
Epoch 7/10
1/1 - 0s - loss: 1161.8049 - accuracy: 0.3333
Epoch 8/10
1/1 - 0s - loss: 406.1945 - accuracy: 0.6667
Epoch 9/10
1/1 - 0s - loss: 187.2601 - accuracy: 0.3333
Epoch 10/10
1/1 - 0s - loss: 157.7449 - accuracy: 0.6667
1/1 - 0s - loss: 295.8282 - accuracy: 0.6667

Test accuracy: 0.6666666865348816
['bowls', 'cups', 'plates']
```

The example shown here only uses the provided data.

Task 7.1: Implement the provided code

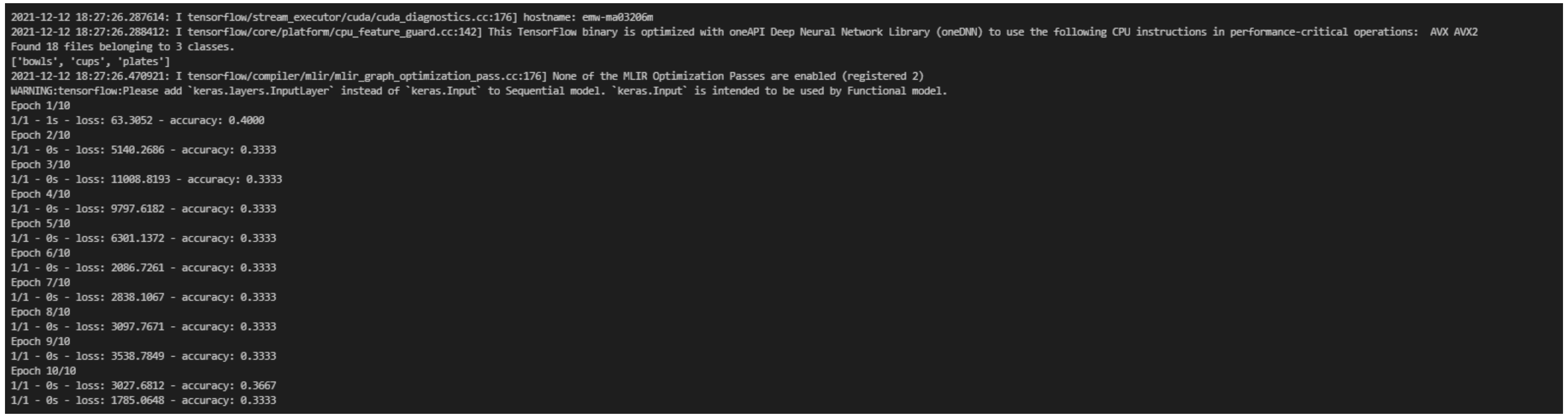
Making use of Keras TensorFlow to visualize the test images with classification

The example shown here only uses the provided data.

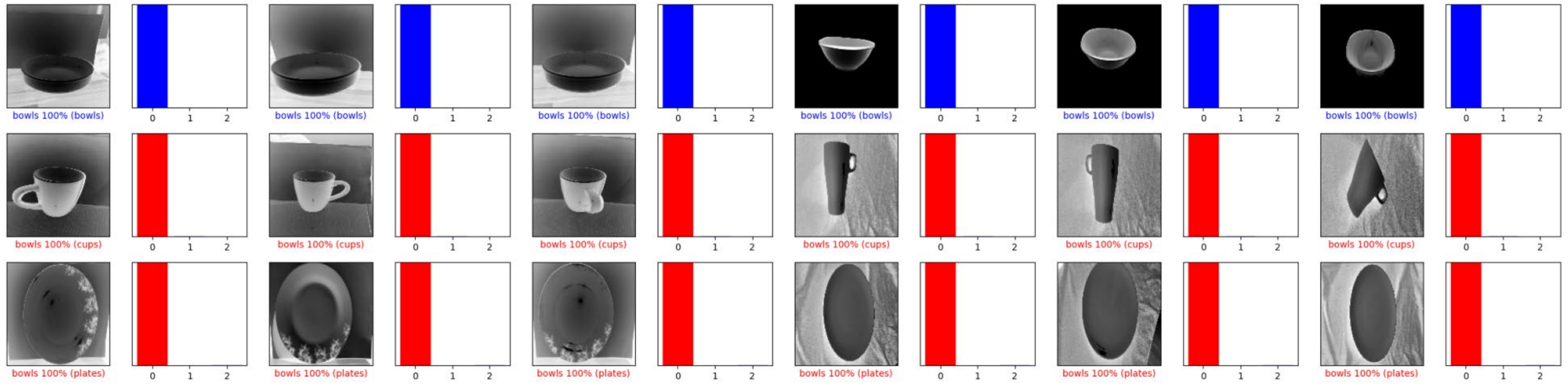


→ Our 66% accuracy is due to misclassification of bowls as plates

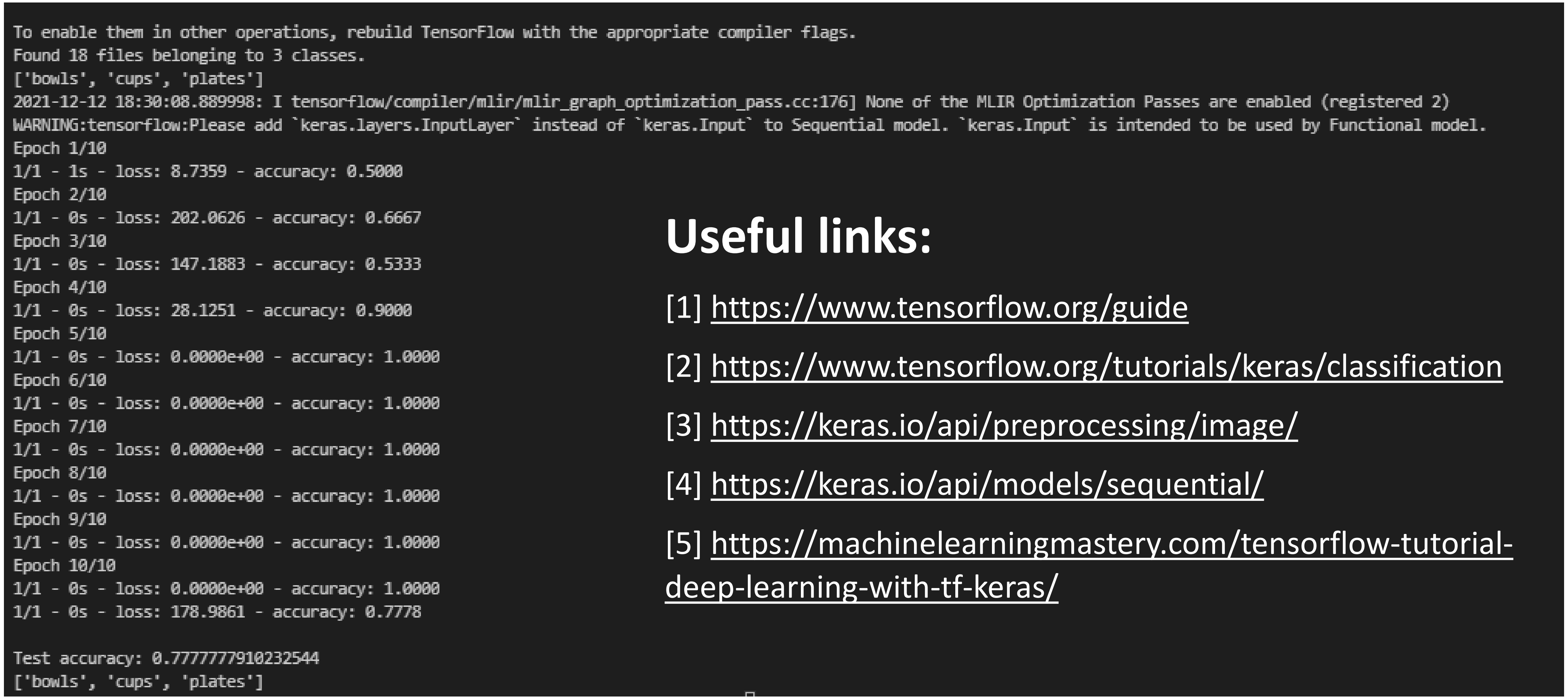
Task 7.2: Improve accuracy by adding the data created by the other groups



```
2021-12-12 18:27:26.287614: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: emw-ma03206m
2021-12-12 18:27:26.288412: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX AVX2
Found 18 files belonging to 3 classes.
['bowls', 'cups', 'plates']
2021-12-12 18:27:26.470921: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:176] None of the MLIR Optimization Passes are enabled (registered 2)
WARNING:tensorflow:Please add `keras.layers.InputLayer` instead of `keras.Input` to Sequential model. `keras.Input` is intended to be used by Functional model.
Epoch 1/10
1/1 - 1s - loss: 63.3052 - accuracy: 0.4000
Epoch 2/10
1/1 - 0s - loss: 5140.2686 - accuracy: 0.3333
Epoch 3/10
1/1 - 0s - loss: 11008.8193 - accuracy: 0.3333
Epoch 4/10
1/1 - 0s - loss: 9797.6182 - accuracy: 0.3333
Epoch 5/10
1/1 - 0s - loss: 6301.1372 - accuracy: 0.3333
Epoch 6/10
1/1 - 0s - loss: 2086.7261 - accuracy: 0.3333
Epoch 7/10
1/1 - 0s - loss: 2838.1067 - accuracy: 0.3333
Epoch 8/10
1/1 - 0s - loss: 3097.7671 - accuracy: 0.3333
Epoch 9/10
1/1 - 0s - loss: 3538.7849 - accuracy: 0.3333
Epoch 10/10
1/1 - 0s - loss: 3027.6812 - accuracy: 0.3667
1/1 - 0s - loss: 1785.0648 - accuracy: 0.3333
```



Task 7.3: Explore different methods to improve the accuracy



```
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Found 18 files belonging to 3 classes.
['bowls', 'cups', 'plates']
2021-12-12 18:30:08.889998: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:176] None of the MLIR Optimization Passes are enabled (registered 2)
WARNING:tensorflow:Please add `keras.layers.InputLayer` instead of `keras.Input` to Sequential model. `keras.Input` is intended to be used by Functional model.
Epoch 1/10
1/1 - 1s - loss: 8.7359 - accuracy: 0.5000
Epoch 2/10
1/1 - 0s - loss: 202.0626 - accuracy: 0.6667
Epoch 3/10
1/1 - 0s - loss: 147.1883 - accuracy: 0.5333
Epoch 4/10
1/1 - 0s - loss: 28.1251 - accuracy: 0.9000
Epoch 5/10
1/1 - 0s - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 6/10
1/1 - 0s - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 7/10
1/1 - 0s - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 8/10
1/1 - 0s - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 9/10
1/1 - 0s - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 10/10
1/1 - 0s - loss: 0.0000e+00 - accuracy: 1.0000
1/1 - 0s - loss: 178.9861 - accuracy: 0.7778

Test accuracy: 0.7777777910232544
['bowls', 'cups', 'plates']
```

Useful links:

[1] https://www.tensorflow.org/guide

[2] https://www.tensorflow.org/tutorials/keras/classification

[3] https://keras.io/api/preprocessing/image/

[4] https://keras.io/api/models/sequential/

[5] https://machinelearningmastery.com/tensorflow-tutorial-deep-learning-with-tf-keras/

References, extra reading links, and resources

Fashion MNIST

- Research Project by Kashif Rasul, Han Xiao (ex-member) & Roland Vollgraf
  https://research.zalando.com/project/fashion_mnist/fashion_mnist/
- https://github.com/zalandoresearch/fashion-mnist
- https://www.tensorflow.org/tutorials/keras/classification