```
import tensorflow as tf
print(tf.__version__)
```

```
2.15.0
```

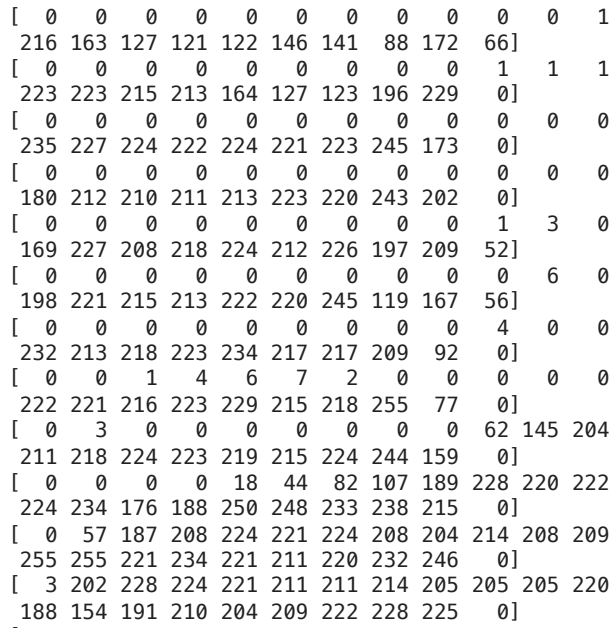
```
# data collection
mnist = tf.keras.datasets.fashion_mnist
```

```
# loading of data
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
```

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

```
# data visulaization
import matplotlib.pyplot as plt
plt.imshow(training_images[0])
print(training_labels[0])
print(training_images[0])
```

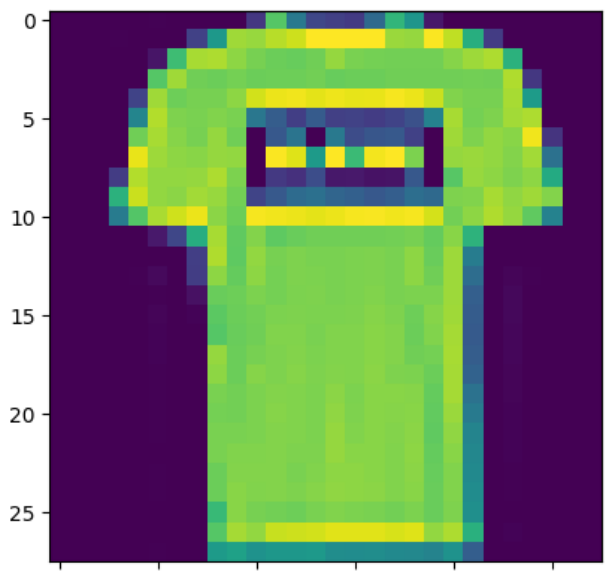
```
9
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   1   0   0  13  73   0
    0   1   4   0   0   0   0   1   1   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   3   0  36 136 127  62
   54   0   0   0   1   3   4   0   0   3]
[  0   0   0   0   0   0   0   0   0   0   0   0   6   0 102 204 176 134
  144 123  23   0   0   0   0  12  10   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0 155 236 207 178
  107 156 161 109  64  23  77 130  72  15]
[  0   0   0   0   0   0   0   0   0   0   0   1   0  69 207 223 218 216
  216 163 127 121 122 146 141  88 172  66]
[  0   0   0   0   0   0   0   0   0   1   1   1   0 200 232 232 233 229
  223 223 215 213 164 127 123 196 229   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0 183 225 216 223 228
  235 227 224 222 224 221 223 245 173   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0 193 228 218 213 198
  180 212 210 211 213 223 220 243 202   0]
[  0   0   0   0   0   0   0   0   0   1   3   0  12 219 220 212 218 192
  169 227 208 218 224 212 226 197 209  52]
[  0   0   0   0   0   0   0   0   0   0   6   0  99 244 222 220 218 203
  198 221 215 213 222 220 245 119 167  56]
[  0   0   0   0   0   0   0   0   0   4   0   0  55 236 228 230 228 240
  232 213 218 223 234 217 217 209  92   0]
[  0   0   1   4   6   7   2   0   0   0   0   0 237 226 217 223 222 219
  222 221 216 223 229 215 218 255  77   0]
[  0   3   0   0   0   0   0   0   0  62 145 204 228 207 213 221 218 208
  211 218 224 223 219 215 224 244 159   0]
[  0   0   0   0  18  44  82 107 189 228 220 222 217 226 200 205 211 230
  224 234 176 188 250 248 233 238 215   0]
[  0  57 187 208 224 221 224 208 204 214 208 209 200 159 245 193 206 223
  255 255 221 234 221 211 220 232 246   0]
[  3 202 228 224 221 211 211 214 205 205 205 220 240  80 150 255 229 221
  188 154 191 210 204 209 222 228 225   0]
[ 98 233 198 210 222 229 229 234 249 220 194 215 217 241  65  73 106 117
  168 219 221 215 217 223 223 224 229  29]
[ 75 204 212 204 193 205 211 225 216 185 197 206 198 213 240 195 227 245
  239 223 218 212 209 222 220 221 230  67]
[ 48 203 183 194 213 197 185 190 194 192 202 214 219 221 220 236 225 216
  199 206 186 181 177 172 181 205 206 115]
[  0 122 219 193 179 171 183 196 204 210 213 207 211 210 200 196 194 191
  195 191 198 192 176 156 167 177 210  92]
[  0   0  74 189 212 191 175 172 175 181 185 188 189 188 193 198 204 209
  210 210 211 188 188 194 192 216 170   0]
[  2   0   0   0  66 200 222 237 239 242 246 243 244 221 220 193 191 179
  182 182 181 176 166 168  99  58   0   0]
[  0   0   0   0   0   0   0  40  61  44  72  41  35   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]
```




```
# data visulaization-2
import matplotlib.pyplot as plt
plt.imshow(training_images[1])
print(training_labels[1])
print(training_images[1])
```

```
0
[[0.         0.         0.         0.         0.         0.00392157
  0.         0.         0.         0.         0.16078431 0.7372549
  0.40392157 0.21176471 0.18823529 0.16862745 0.34117647 0.65882353
  0.52156863 0.0627451  0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.00392157 0.         0.
  0.         0.19215686 0.53333333 0.85882353 0.84705882 0.89411765
  0.9254902  1.         1.         1.         1.         0.85098039
  0.84313725 0.99607843 0.90588235 0.62745098 0.17647059 0.
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.05490196
  0.69019608 0.87058824 0.87843137 0.83137255 0.79607843 0.77647059
  0.76862745 0.78431373 0.84313725 0.8        0.79215686 0.78823529
  0.78823529 0.78823529 0.81960784 0.85490196 0.87843137 0.64313725
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.7372549
  0.85882353 0.78431373 0.77647059 0.79215686 0.77647059 0.78039216
  0.78039216 0.78823529 0.76862745 0.77647059 0.77647059 0.78431373
  0.78431373 0.78431373 0.78431373 0.78823529 0.78431373 0.88235294
  0.16078431 0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.2        0.85882353
  0.78039216 0.79607843 0.79607843 0.83137255 0.93333333 0.97254902
  0.98039216 0.96078431 0.97647059 0.96470588 0.96862745 0.98823529
  0.97254902 0.92156863 0.81176471 0.79607843 0.79607843 0.87058824
  0.54901961 0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.45490196 0.88627451
  0.80784314 0.8        0.81176471 0.8        0.39607843 0.29411765
  0.18431373 0.28627451 0.18823529 0.19607843 0.17647059 0.2
  0.24705882 0.44313725 0.87058824 0.79215686 0.80784314 0.8627451
  0.87843137 0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.78431373 0.87058824
  0.81960784 0.79607843 0.84313725 0.78431373 0.         0.2745098
  0.38431373 0.         0.40392157 0.23137255 0.26666667 0.27843137
  0.19215686 0.         0.85882353 0.80784314 0.83921569 0.82352941
  0.98039216 0.14901961 0.         0.         0.        ]
 [0.         0.         0.         0.         0.96862745 0.85490196
  0.83137255 0.82352941 0.84313725 0.83921569 0.         0.99607843
  0.95294118 0.54509804 1.         0.68235294 0.98431373 1.
  0.80392157 0.         0.84313725 0.85098039 0.83921569 0.81568627
  0.8627451  0.37254902 0.         0.         0.        ]
 [0.         0.         0.         0.17647059 0.88627451 0.83921569
  0.83921569 0.84313725 0.87843137 0.80392157 0.         0.16470588
  0.1372549  0.23529412 0.0627451  0.06666667 0.04705882 0.05098039
  0.2745098  0.         0.74117647 0.84705882 0.83137255 0.80784314
  0.83137255 0.61176471 0.         0.         0.        ]
 [0.         0.         0.         0.64313725 0.92156863 0.83921569
  0.82745098 0.8627451  0.84705882 0.78823529 0.20392157 0.27843137
  0.34901961 0.36862745 0.3254902  0.30588235 0.2745098  0.29803922
  0.36078431 0.34117647 0.80784314 0.81176471 0.87058824 0.83529412
  0.85882353 0.81568627 0.         0.         0.        ]
 [0.         0.         0.         0.41568627 0.73333333 0.8745098
  0.92941176 0.97254902 0.82745098 0.77647059 0.98823529 0.98039216
  0.97254902 0.96078431 0.97254902 0.98823529 0.99215686 0.98039216
  0.98823529 0.9372549  0.78823529 0.83137255 0.88235294 0.84313725
  0.75686275 0.44313725 0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.06666667
  0.21176471 0.62352941 0.87058824 0.75686275 0.81568627 0.75294118
  0.77254902 0.78431373 0.78431373 0.78431373 0.78431373 0.78823529
  0.79607843 0.76470588 0.82352941 0.64705882 0.         0.
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.18431373 0.88235294 0.75294118 0.83921569 0.79607843
  0.80784314 0.8        0.8        0.80392157 0.80784314 0.8
  0.83137255 0.77254902 0.85490196 0.41960784 0.         0.
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.00392157 0.02352941
  0.         0.18039216 0.83137255 0.76470588 0.83137255 0.79215686
  0.80784314 0.80392157 0.8        0.80392157 0.80784314 0.8
  0.83137255 0.78431373 0.85490196 0.35686275 0.         0.01176471
  0.00392157 0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.00392157
  0.         0.04313725 0.77254902 0.78039216 0.80392157 0.79215686
  0.80392157 0.80784314 0.8        0.80392157 0.81176471 0.8
  0.80392157 0.80392157 0.85490196 0.30196078 0.         0.01960784
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.01176471
  0.         0.00784314 0.74901961 0.77647059 0.78823529 0.80392157
  0.80784314 0.80392157 0.80392157 0.80784314 0.81960784 0.80784314
  0.78039216 0.81960784 0.85882353 0.29019608 0.         0.01960784
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.00784314
  0.         0.         0.7372549  0.77254902 0.78431373 0.81176471
  0.81176471 0.8        0.81176471 0.81176471 0.82352941 0.81568627
  0.77647059 0.81176471 0.86666667 0.28235294 0.         0.01568627
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.00784314
  0.         0.         0.84313725 0.77647059 0.79607843 0.80784314
  0.81568627 0.80392157 0.81176471 0.81176471 0.82352941 0.81568627
  0.78431373 0.79215686 0.87058824 0.29411765 0.         0.01568627
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.00392157
  0.         0.         0.83137255 0.77647059 0.81960784 0.80784314
  0.81960784 0.80784314 0.81568627 0.81176471 0.82745098 0.80784314
  0.80392157 0.77647059 0.86666667 0.31372549 0.         0.01176471
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.00392157
  0.         0.         0.8        0.78823529 0.80392157 0.81568627
  0.81176471 0.80392157 0.82745098 0.80392157 0.82352941 0.82352941
  0.81960784 0.76470588 0.86666667 0.37647059 0.         0.01176471
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.00392157
  0.         0.         0.79215686 0.78823529 0.80392157 0.81960784
  0.81176471 0.80392157 0.83529412 0.80784314 0.82352941 0.81960784
  0.82352941 0.76078431 0.85098039 0.41176471 0.         0.00784314
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.00392157
  0.         0.         0.8        0.8        0.80392157 0.81568627
  0.81176471 0.80392157 0.84313725 0.81176471 0.82352941 0.81568627
  0.82745098 0.75686275 0.83529412 0.45098039 0.         0.00784314
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.8        0.81176471 0.81176471 0.81568627
  0.80784314 0.80784314 0.84313725 0.82352941 0.82352941 0.81176471
  0.83137255 0.76470588 0.82352941 0.4627451  0.         0.00784314
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.00392157
  0.         0.         0.77647059 0.81568627 0.81568627 0.81568627
  0.8        0.81176471 0.83137255 0.83137255 0.82352941 0.81176471
  0.82745098 0.76862745 0.81176471 0.4745098  0.         0.00392157
```

```
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.00392157
 0.         0.         0.77647059 0.82352941 0.81176471 0.81568627
 0.80784314 0.81960784 0.83529412 0.83137255 0.82745098 0.81176471
 0.82352941 0.77254902 0.81176471 0.48627451 0.         0.00392157
 0.         0.         0.         0.         ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.6745098  0.82352941 0.79607843 0.78823529
 0.78039216 0.8        0.81176471 0.80392157 0.8        0.78823529
 0.80392157 0.77254902 0.80784314 0.49803922 0.         0.
 0.         0.         0.         0.         ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.7372549  0.86666667 0.83921569 0.91764706
 0.9254902  0.93333333 0.95686275 0.95686275 0.95686275 0.94117647
 0.95294118 0.83921569 0.87843137 0.63529412 0.         0.00784314
 0.         0.         0.         0.         ]
[0.         0.         0.         0.         0.         0.00392157
 0.         0.         0.54509804 0.57254902 0.50980392 0.52941176
 0.52941176 0.5372549  0.49019608 0.48627451 0.49019608 0.4745098
 0.46666667 0.44705882 0.50980392 0.29803922 0.         0.
 0.         0.         0.         0.         ]]
```




```
# data normalisation
training_images  = training_images / 255.0
test_images = test_images / 255.0
```

```
# design the model with 3 layers model
# Sequential defines a sequence of layers in the neural network.
# Flatten takes a square and turns it into a one-dimensional vector.
# Dense adds a layer of neurons.
# Activation functions tell each layer of neurons what to do:
# Relu effectively means that if X is greater than 0 return X, else return 0. It only passes values of 0 or greater to the next layer in the network.
# Softmax takes a set of values, and effectively picks the biggest one.
model = tf.keras.models.Sequential([tf.keras.layers.Flatten(), tf.keras.layers.Dense(128, activation=tf.nn.relu), tf.keras.layers.Dense(10, activation=t
```

```
# Model building and training
model.compile(optimizer = tf.keras.optimizers.Adam(), loss = 'sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(training_images, training_labels, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 9s 4ms/step - loss: 0.4989 - accuracy: 0.8249
Epoch 2/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3728 - accuracy: 0.8651
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3352 - accuracy: 0.8783
Epoch 4/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3091 - accuracy: 0.8849
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2919 - accuracy: 0.8911
<keras.src.callbacks.History at 0x786cd715ea70>
```

```
# Test the model
model.evaluate(test_images, test_labels)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.3523 - accuracy: 0.8755
[0.3523348569869995, 0.8755000233650208]
```

```
classifications = model.predict(test_images)
print(classifications[0])
```

```
313/313 [==============================] - 1s 2ms/step
[3.8789972e-06 2.3702633e-08 1.3109383e-07 3.4619757e-10 8.2389079e-08
 1.9439885e-03 7.2308494e-06 1.7319907e-02 3.9995222e-05 9.8068482e-01]
```

```
print(test_labels[0])
```

```
9
```

```
classifications = model.predict(test_images)

print(classifications[1])
```

```
313/313 [==============================] - 1s 3ms/step
[9.1460715e-05 9.1194536e-09 9.9451387e-01 4.0946375e-09 4.9880268e-03
 1.9634667e-06 4.0468070e-04 1.1837766e-12 2.0608709e-08 1.5658051e-11]
```

```
print(test_labels[1])
```

```
2
```

```
# Hidden Layers increased
import tensorflow as tf
print(tf.__version__)

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(1024, activation=tf.nn.relu),
                                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy')

model.fit(training_images, training_labels, epochs=5)

model.evaluate(test_images, test_labels)

classifications = model.predict(test_images)

print(classifications[0])
print(test_labels[0])
```

```
2.15.0
Epoch 1/5
1875/1875 [==============================] - 24s 13ms/step - loss: 0.4716
Epoch 2/5
1875/1875 [==============================] - 24s 13ms/step - loss: 0.3603
Epoch 3/5
1875/1875 [==============================] - 24s 13ms/step - loss: 0.3215
Epoch 4/5
1875/1875 [==============================] - 24s 13ms/step - loss: 0.2957
Epoch 5/5
1875/1875 [==============================] - 23s 12ms/step - loss: 0.2790
313/313 [==============================] - 1s 4ms/step - loss: 0.3348
313/313 [==============================] - 1s 3ms/step
[2.78907319e-09 1.65446745e-09 8.78008846e-11 6.76895389e-12
 3.28106409e-09 1.16543255e-04 2.97101499e-09 7.88777322e-03
 2.04979909e-08 9.91995633e-01]
9
```

```
import tensorflow as tf
print(tf.__version__)

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.Dense(64, activation=tf.nn.relu),tf.keras.layers.Dense(10, activation=tf.nn.softmax)])

# This version has the 'flatten' removed. Replace the above with this one to see the error.
#model = tf.keras.models.Sequential([tf.keras.layers.Dense(64, activation=tf.nn.relu),
#                                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])


model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy')

model.fit(training_images, training_labels, epochs=5)

model.evaluate(test_images, test_labels)

classifications = model.predict(test_images)

print(classifications[1])
print(test_labels[1])
```

```
2.15.0
Epoch 1/5
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-dc9366bc37a3> in <cell line: 22>()
     20               loss = 'sparse_categorical_crossentropy')
     21
---> 22 model.fit(training_images, training_labels, epochs=5)
     23
     24 model.evaluate(test_images, test_labels)

 1 frames
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py in tf__train_function(iterator)
     13                 try:
     14                     do_return = True
---> 15                     retval_ = ag__.converted_call(ag__.ld(step_function), (ag__.ld(self),
ag__.ld(iterator)), None, fscope)
     16                 except:
     17                     do_return = False

ValueError: in user code:

    File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py", line 1401, in
train_function  *
        return step_function(self, iterator)
    File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py", line 1384, in step_function
**
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py", line 1373, in run_step  **
        outputs = model.train_step(data)
    File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py", line 1151, in train_step
        loss = self.compute_loss(x, y, y_pred, sample_weight)
    File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py", line 1209, in compute_loss
        return self.compiled_loss(
    File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/compile_utils.py", line 277, in __call__
        loss_value = loss_obj(y_t, y_p, sample_weight=sw)
    File "/usr/local/lib/python3.10/dist-packages/keras/src/losses.py", line 143, in __call__
        losses = call_fn(y_true, y_pred)
    File "/usr/local/lib/python3.10/dist-packages/keras/src/losses.py", line 270, in call  **
        return ag_fn(y_true, y_pred, **self._fn_kwargs)
    File "/usr/local/lib/python3.10/dist-packages/keras/src/losses.py", line 2454, in
sparse_categorical_crossentropy
        return backend.sparse_categorical_crossentropy(
    File "/usr/local/lib/python3.10/dist-packages/keras/src/backend.py", line 5775, in
sparse_categorical_crossentropy
        res = tf.nn.sparse_softmax_cross_entropy_with_logits(
```

```
## Error detection details:
## The Flatten layer is needed in this case because the input data for the neural network is a two-dimensional array (a 28x28 image), but the Dense laye
## Without the Flatten layer, the input data would be passed to the Dense layers as a two-dimensional array, which would result in an error.

### Here's a more detailed explanation:

##The Dense layer expects input data in the following format: [batch_size, features]. In this case, the batch size is the number of images being process
##The input data for this neural network is a two-dimensional array with the shape (28, 28). This means that each image is represented as a 28x28 grid o
##To convert the two-dimensional input data into one-dimensional data, we need to flatten it. This means that we need to take all of the pixels in each
##The Flatten layer does this for us. It takes the two-dimensional input data and reshapes it into a one-dimensional array with the shape [batch_size, 78
##The one-dimensional output from the Flatten layer can then be passed to the Dense layers.
##In general, the Flatten layer is used whenever you have input data that is not already in the correct format for the Dense layers. For example, if you
```

```
# Another checking
import tensorflow as tf
print(tf.__version__)

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(64, activation=tf.nn.relu),
                                    tf.keras.layers.Dense(5, activation=tf.nn.softmax)])

# Replace the above model definiton with this one to see the network with 5 output layers
# And you'll see errors as a result!
# model = tf.keras.models.Sequential([tf.keras.layers.Flatten(),
#                                     tf.keras.layers.Dense(64, activation=tf.nn.relu),
#                                     tf.keras.layers.Dense(5, activation=tf.nn.softmax)])

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy')

model.fit(training_images, training_labels, epochs=5)

model.evaluate(test_images, test_labels)

classifications = model.predict(test_images)

print(classifications[0])
print(test_labels[0])
```

```
2.15.0
Epoch 1/5
---------------------------------------------------------------------------
InvalidArgumentError                      Traceback (most recent call last)
<ipython-input-17-33c9e637b54b> in <cell line: 25>()
     23               loss = 'sparse_categorical_crossentropy')
     24
---> 25 model.fit(training_images, training_labels, epochs=5)
     26
     27 model.evaluate(test_images, test_labels)

 1 frames
/usr/local/lib/python3.10/dist-packages/tensorflow/python/eager/execute.py in quick_execute(op_name,
num_outputs, inputs, attrs, ctx, name)
     51   try:
     52     ctx.ensure_initialized()
---> 53     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     54                                         inputs, attrs, num_outputs)
     55   except core._NotOkStatusException as e:

InvalidArgumentError: Graph execution error:

Detected at node
sparse_categorical_crossentropy/SparseSoftmaxCrossEntropyWithLogits/SparseSoftmaxCrossEntropyWithLogits
defined at (most recent call last):
  File "/usr/lib/python3.10/runpy.py", line 196, in _run_module_as_main

  File "/usr/lib/python3.10/runpy.py", line 86, in _run_code

  File "/usr/local/lib/python3.10/dist-packages/colab_kernel_launcher.py", line 37, in <module>

  File "/usr/local/lib/python3.10/dist-packages/traitlets/config/application.py", line 992, in
launch_instance

  File "/usr/local/lib/python3.10/dist-packages/ipykernel/kernelapp.py", line 619, in start

  File "/usr/local/lib/python3.10/dist-packages/tornado/platform/asyncio.py", line 195, in start

  File "/usr/lib/python3.10/asyncio/base_events.py", line 603, in run_forever

  File "/usr/lib/python3.10/asyncio/base_events.py", line 1909, in _run_once

  File "/usr/lib/python3.10/asyncio/events.py", line 80, in _run

  File "/usr/local/lib/python3.10/dist-packages/tornado/ioloop.py", line 685, in <lambda>

  File "/usr/local/lib/python3.10/dist-packages/tornado/ioloop.py", line 738, in _run_callback

  File "/usr/local/lib/python3.10/dist-packages/tornado/gen.py", line 825, in inner

  File "/usr/local/lib/python3.10/dist-packages/tornado/gen.py", line 786, in run

  File "/usr/local/lib/python3.10/dist-packages/ipykernel/kernelbase.py", line 361, in process_one

  File "/usr/local/lib/python3.10/dist-packages/tornado/gen.py", line 234, in wrapper

  File "/usr/local/lib/python3.10/dist-packages/ipykernel/kernelbase.py", line 261, in dispatch_shell

  File "/usr/local/lib/python3.10/dist-packages/tornado/gen.py", line 234, in wrapper

  File "/usr/local/lib/python3.10/dist-packages/ipykernel/kernelbase.py", line 539, in execute_request

  File "/usr/local/lib/python3.10/dist-packages/tornado/gen.py", line 234, in wrapper

  File "/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py", line 302, in do_execute

  File "/usr/local/lib/python3.10/dist-packages/ipykernel/zmqshell.py", line 539, in run_cell

  File "/usr/local/lib/python3.10/dist-packages/IPython/core/interactiveshell.py", line 2975, in run_cell

  File "/usr/local/lib/python3.10/dist-packages/IPython/core/interactiveshell.py", line 3030, in _run_cell

  File "/usr/local/lib/python3.10/dist-packages/IPython/core/async_helpers.py", line 78, in
_pseudo_sync_runner

  File "/usr/local/lib/python3.10/dist-packages/IPython/core/interactiveshell.py", line 3257, in
run_cell_async

  File "/usr/local/lib/python3.10/dist-packages/IPython/core/interactiveshell.py", line 3473, in
run_ast_nodes

  File "/usr/local/lib/python3.10/dist-packages/IPython/core/interactiveshell.py", line 3553, in run_code

  File "<ipython-input-17-33c9e637b54b>", line 25, in <cell line: 25>

  File "/usr/local/lib/python3.10/dist-packages/keras/src/utils/traceback_utils.py", line 65, in
error_handler

  File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py", line 1807, in fit

  File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py", line 1401, in train_function

  File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py", line 1384, in step_function

  File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py", line 1373, in run_step

  File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py", line 1151, in train_step

  File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py", line 1209, in compute_loss

  File "/usr/local/lib/python3.10/dist-packages/keras/src/engine/compile_utils.py", line 277, in __call__

  File "/usr/local/lib/python3.10/dist-packages/keras/src/losses.py", line 143, in __call__

  File "/usr/local/lib/python3.10/dist-packages/keras/src/losses.py", line 270, in call

  File "/usr/local/lib/python3.10/dist-packages/keras/src/losses.py", line 2454, in
sparse_categorical_crossentropy
```

```
# The error message points to an issue with the sparse_categorical_crossentropy loss function.
# The error message states that a label value of 9 is outside the valid range of [0, 5).
# This means that the model is expecting labels between 0 and 4, but it encountered a label of 9.
print(training_labels)
print(test_labels)
```

```
[9 0 0 ... 3 0 5]
[9 2 1 ... 8 1 5]
```

```
# From above we can cnclude that there is need to change the output layer of the model to have 10 neurons instead of 5.
```

```
# Consider the effects of additional layers in the network.
# What will happen if you add another layer between the one with 512 and the final layer with 10?
# There isn't a significant impact -- because this is relatively simple data.
# For far more complex data (including color images to be classified as flowers that you'll see in the next lesson), extra layers are often necessary.
import tensorflow as tf
print(tf.__version__)

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(256, activation=tf.nn.relu),
                                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy')

model.fit(training_images, training_labels, epochs=5)

model.evaluate(test_images, test_labels)

classifications = model.predict(test_images)

print(classifications[0])
print(test_labels[0])
```

```
2.15.0
Epoch 1/5
1875/1875 [==============================] - 25s 12ms/step - loss: 0.4652
Epoch 2/5
1875/1875 [==============================] - 16s 9ms/step - loss: 0.3563
Epoch 3/5
1875/1875 [==============================] - 20s 10ms/step - loss: 0.3199
Epoch 4/5
1875/1875 [==============================] - 17s 9ms/step - loss: 0.2951
Epoch 5/5
1875/1875 [==============================] - 16s 9ms/step - loss: 0.2765
313/313 [==============================] - 1s 3ms/step - loss: 0.3292
313/313 [==============================] - 1s 3ms/step
[2.0697474e-08 9.0923482e-09 2.0312902e-09 5.5544780e-10 4.0997103e-10
 2.8566211e-03 2.4098862e-08 3.9385990e-03 1.4077395e-09 9.9320471e-01]
9
```

```
# Consider the impact of training for more or less epochs. Why do you think that would be the case?

# I have tried 15 epochs -- Got a model with a much better loss than the one with 5 Try 30 epochs
# Saw the loss value stops decreasing, and sometimes increases.
# This is a side effect of something called 'overfitting' which you can learn about [somewhere]
# It is something you need to keep an eye out for when training neural networks.
# There's no point in wasting your time training if you aren't improving your loss, right! :)
import tensorflow as tf
print(tf.__version__)

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(128, activation=tf.nn.relu),
                                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy')

model.fit(training_images, training_labels, epochs=30)

model.evaluate(test_images, test_labels)

classifications = model.predict(test_images)

print(classifications[34])
print(test_labels[34])
```

```
2.15.0
Epoch 1/30
1875/1875 [==============================] - 8s 4ms/step - loss: 0.5025
Epoch 2/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3805
Epoch 3/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3437
Epoch 4/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3172
Epoch 5/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3009
Epoch 6/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2829
Epoch 7/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2707
Epoch 8/30
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2597
Epoch 9/30
1875/1875 [==============================] - 7s 3ms/step - loss: 0.2506
Epoch 10/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2407
Epoch 11/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2372
Epoch 12/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2257
Epoch 13/30
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2191
Epoch 14/30
1875/1875 [==============================] - 7s 3ms/step - loss: 0.2130
```

```
Epoch 15/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2064
Epoch 16/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1996
Epoch 17/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1958
Epoch 18/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1909
Epoch 19/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1858
Epoch 20/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1807
Epoch 21/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1768
Epoch 22/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1732
Epoch 23/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1690
Epoch 24/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1663
Epoch 25/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1605
Epoch 26/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1582
Epoch 27/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1536
Epoch 28/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1506
Epoch 29/30
```

```
# Before I trained, normalized the data, going from values that were 0 through 255 to values that were 0 through 1.
# What would be the impact of removing that?
import tensorflow as tf
print(tf.__version__)
mnist = tf.keras.datasets.fashion_mnist
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()

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='adam', loss='sparse_categorical_crossentropy')
model.fit(training_images, training_labels, epochs=5)
model.evaluate(test_images, test_labels)
classifications = model.predict(test_images)
print(classifications[0])
print(test_labels[0])
```

```
2.15.0
Epoch 1/5
1875/1875 [==============================] - 15s 8ms/step - loss: 4.2990
Epoch 2/5
1875/1875 [==============================] - 13s 7ms/step - loss: 0.5265
Epoch 3/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.5068
Epoch 4/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.5003
Epoch 5/5
1875/1875 [==============================] - 14s 8ms/step - loss: 0.4715
313/313 [==============================] - 1s 3ms/step - loss: 0.5078
313/313 [==============================] - 1s 2ms/step
[1.9835710e-15 3.5707580e-15 8.0527958e-20 4.4003730e-16 1.5246095e-22
 6.7817359e-03 6.1527788e-20 2.4431892e-02 2.7594527e-11 9.6878636e-01]
9
```

```
# In a nutshell, normalization reduces the complexity of the problem your network is trying to solve.
```