Deep Learning for Image Classification Assessment

Building an image classifier with Keras and Convolutional Neural Networks for the Fashion MNIST dataset. This data set includes 10 labels of different clothing types with 28 by 28 *grayscale* images. There is a training set of 60,000 images and 10,000 test images.**

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

The Data

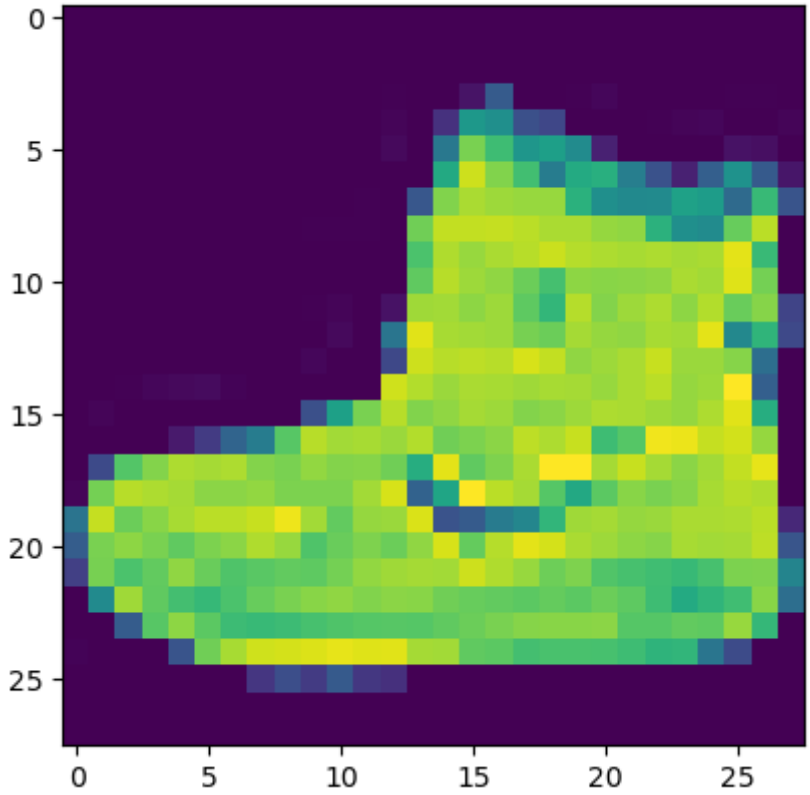
In [56]:
```
from tensorflow.keras.datasets import fashion_mnist

(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

Visualizing the Data

In [57]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [116…
```
plt.imshow(x_train[0])
```

Out[116]:
```
<matplotlib.image.AxesImage at 0x2a1da9b2910>
```



Preprocessing the Data

In [59]:
```
x_train.max()
x_test.max()
```

Out[59]:
```
255
```

In [60]:
```
x_train = x_train/255
```

In [61]:
```
x_test = x_test/255
```

In [62]:
```
x_train.max()
```

Out[62]:
```
1.0
```

In [63]:
```
import numpy as np
```

In [64]:
```
x_train = x_train.reshape(60000,28,28,1)
x_test = x_test.reshape(10000,28,28,1)
```

In [65]:
```
x_train.shape
```

Out[65]:
```
(60000, 28, 28, 1)
```

In [66]:
```
x_test.shape
```

Out[66]:
```
(10000, 28, 28, 1)
```

In [67]:
```
from tensorflow.keras.utils import to_categorical
```

In [69]:
```
y_cat_test = to_categorical(y_test)
```

In [70]:
```
y_cat_train
```

Out[70]:
```
array([[0., 0., 0., ..., 0., 0., 1.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

Building the Model

In [71]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, MaxPool2D, Dense
```

In [108…
```
model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(4,4),input_shape=(28, 28, 1), activation=
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dense(10, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accu
```

In [109…
```
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_14 (Conv2D)          (None, 25, 25, 32)        544

 max_pooling2d_14 (MaxPoolin  (None, 12, 12, 32)       0
 g2D)

 flatten_7 (Flatten)         (None, 4608)              0

 dense_14 (Dense)            (None, 128)               589952

 dense_15 (Dense)            (None, 10)                1290

=================================================================
Total params: 591,786
Trainable params: 591,786
Non-trainable params: 0
_________________________________________________________________
```

Training the Model

In [110…
```
model.fit(x_train,y_cat_train, epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.4193 - accurac
y: 0.8497
Epoch 2/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2818 - accurac
y: 0.8981
Epoch 3/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2402 - accurac
y: 0.9115
Epoch 4/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2120 - accurac
y: 0.9230
Epoch 5/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1897 - accurac
y: 0.9313
Epoch 6/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1725 - accurac
y: 0.9372
Epoch 7/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1560 - accurac
y: 0.9449
Epoch 8/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1445 - accurac
y: 0.9479
Epoch 9/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1319 - accurac
y: 0.9535
Epoch 10/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1203 - accurac
y: 0.9569
```

Out[110]: <keras.callbacks.History at 0x2a1f176adf0>

Evaluating the Model

In [111…
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [112…
```
predictions = model.predict(x_test)
```

```
313/313 [==============================] - 1s 2ms/step
```

In [113…
```
predictions_arg = np.argmax(predictions, axis=1)
```

In [114…
```
predictions_arg[0]
```

Out[114]: 9

In [115…
```
print(classification_report(y_test, predictions_arg))
```

	precision	recall	f1-score	support
0	0.83	0.91	0.87	1000
1	0.98	0.98	0.98	1000
2	0.83	0.89	0.86	1000
3	0.89	0.94	0.91	1000
4	0.89	0.81	0.85	1000
5	0.97	0.98	0.98	1000
6	0.80	0.70	0.75	1000
7	0.94	0.98	0.96	1000
8	0.98	0.98	0.98	1000
9	0.98	0.95	0.96	1000
accuracy			0.91	10000
macro avg	0.91	0.91	0.91	10000
weighted avg	0.91	0.91	0.91	10000