deep-learning-keras

November 19, 2024

Installing the Necessary Libraries

[25]:
```
# !pip install tensorflow
# !pip install matplotlib
```

Importing the necessary libraries

[26]:
```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import warnings
warnings.filterwarnings('ignore')
```

Importing the dataset Importing the fashion mnist dataset which is an alternative to the numbers mnist dataset

[27]:
```
fashion_mnist = tf.keras.datasets.fashion_mnist.load_data()
```

Extracting the training and testing datasets from the fashion mnist dataset

[28]:
```
(X_train_full, y_train_full), (X_test, y_test) = fashion_mnist
```

Seperating the training set into training and validation datasets

[29]:
```
X_train, y_train = X_train_full[:-5000], y_train_full[:-5000]
X_valid, y_valid = X_train_full[-5000:], y_train_full[-5000:]
```

[30]:
```
X_train.shape, y_train.shape
```

[30]:
```
((55000, 28, 28), (55000,))
```

[31]:
```
X_valid.shape, y_valid.shape
```

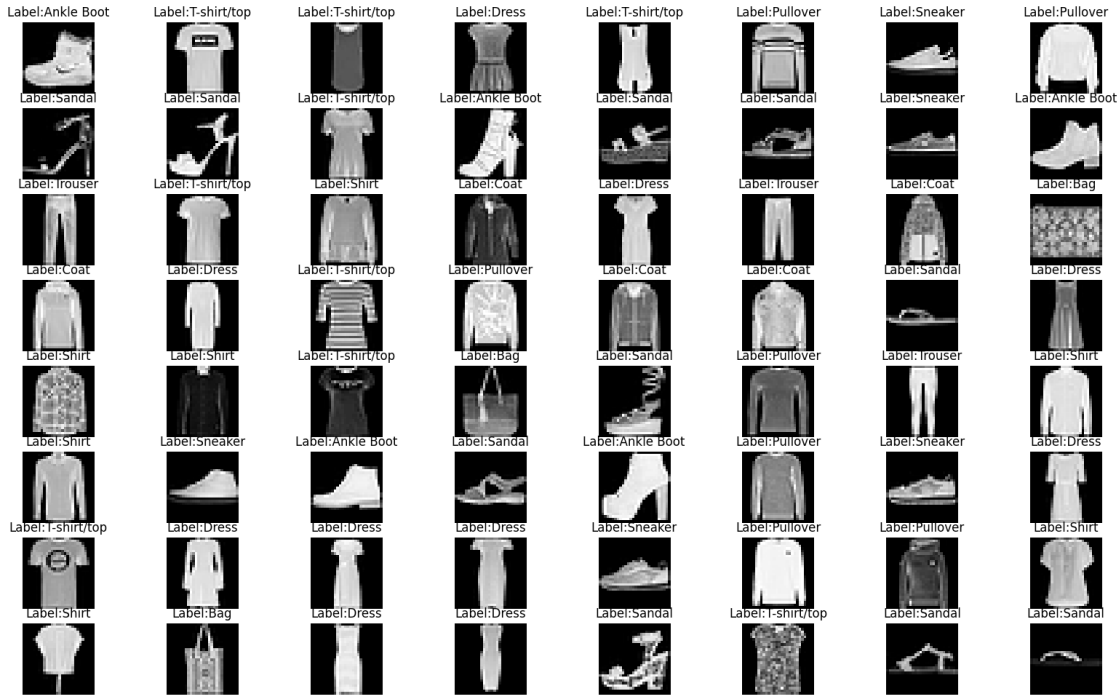
[31]:
```
((5000, 28, 28), (5000,))
```

Actual Classnames represented by name and index in array

[32]:
```
class_names = ["T-shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal",␣
↪"Shirt", "Sneaker", "Bag", "Ankle Boot"]
```

Visualizing the pixel data in images

[33]:
```
num_images = 64
factor = int(np.sqrt(num_images))
fig,ax = plt.subplots(nrows=factor, ncols=factor, figsize=(20,12))
idx_offset = 0 # take "num_images" starting from the index "idx_offset"
for i in range(factor):
    index = idx_offset+i*(factor)
    for j in range(factor):
        ax[i,j].imshow(X_train_full[index+j],cmap='gray')
        ax[i,j].set_title('Label:{0}'.
 ↪format(str(class_names[y_train_full[index+j]])))
        ax[i,j].set_axis_off()
```



Normalizing the pixel data Pixel values are ranging from 0 to 255. Dividing by 255 will result the values to lie from 0.0 to 1.0.

[34]:
```
X_train, X_valid, X_test = X_train/255, X_valid/255, X_test/255
```

Visualizing the pixel values of the first image

[35]:
```
print(X_train[0])
```

```
[[0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
```

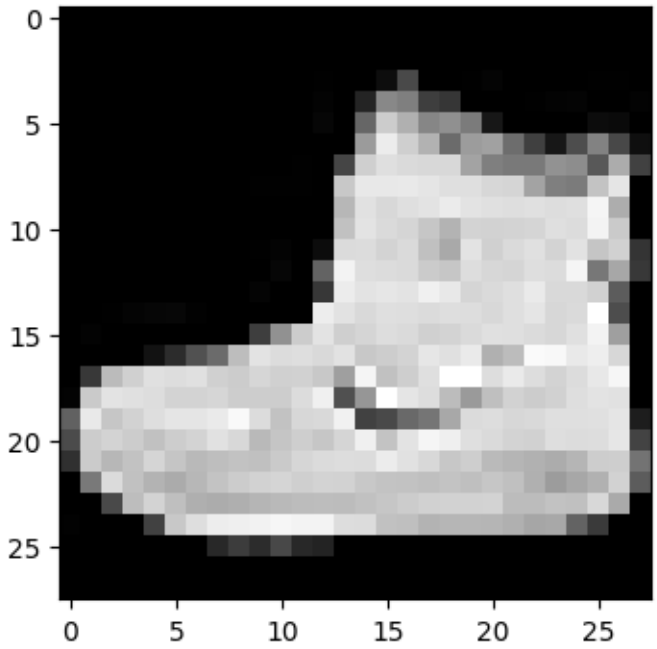
```
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         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.         0.
 0.00392157 0.         0.         0.05098039 0.28627451 0.
 0.         0.00392157 0.01568627 0.         0.         0.
 0.         0.00392157 0.00392157 0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.01176471 0.         0.14117647 0.53333333 0.49803922 0.24313725
 0.21176471 0.         0.         0.         0.00392157 0.01176471
 0.01568627 0.         0.         0.01176471]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.02352941 0.         0.4        0.8        0.69019608 0.5254902
 0.56470588 0.48235294 0.09019608 0.         0.         0.
 0.         0.04705882 0.03921569 0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.60784314 0.9254902  0.81176471 0.69803922
 0.41960784 0.61176471 0.63137255 0.42745098 0.25098039 0.09019608
 0.30196078 0.50980392 0.28235294 0.05882353]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.00392157
 0.         0.27058824 0.81176471 0.8745098  0.85490196 0.84705882
 0.84705882 0.63921569 0.49803922 0.4745098  0.47843137 0.57254902
 0.55294118 0.34509804 0.6745098  0.25882353]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.00392157 0.00392157 0.00392157
 0.         0.78431373 0.90980392 0.90980392 0.91372549 0.89803922
 0.8745098  0.8745098  0.84313725 0.83529412 0.64313725 0.49803922
 0.48235294 0.76862745 0.89803922 0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.71764706 0.88235294 0.84705882 0.8745098  0.89411765
 0.92156863 0.89019608 0.87843137 0.87058824 0.87843137 0.86666667
 0.8745098  0.96078431 0.67843137 0.        ]
[0.         0.         0.         0.         0.         0.
```

```
 0.         0.         0.         0.         0.         0.
 0.         0.75686275 0.89411765 0.85490196 0.83529412 0.77647059
 0.70588235 0.83137255 0.82352941 0.82745098 0.83529412 0.8745098
 0.8627451  0.95294118 0.79215686 0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.00392157 0.01176471 0.
 0.04705882 0.85882353 0.8627451  0.83137255 0.85490196 0.75294118
 0.6627451  0.89019608 0.81568627 0.85490196 0.87843137 0.83137255
 0.88627451 0.77254902 0.81960784 0.20392157]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.02352941 0.
 0.38823529 0.95686275 0.87058824 0.8627451  0.85490196 0.79607843
 0.77647059 0.86666667 0.84313725 0.83529412 0.87058824 0.8627451
 0.96078431 0.46666667 0.65490196 0.21960784]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.01568627 0.         0.
 0.21568627 0.9254902  0.89411765 0.90196078 0.89411765 0.94117647
 0.90980392 0.83529412 0.85490196 0.8745098  0.91764706 0.85098039
 0.85098039 0.81960784 0.36078431 0.        ]
[0.         0.         0.00392157 0.01568627 0.02352941 0.02745098
 0.00784314 0.         0.         0.         0.         0.
 0.92941176 0.88627451 0.85098039 0.8745098  0.87058824 0.85882353
 0.87058824 0.86666667 0.84705882 0.8745098  0.89803922 0.84313725
 0.85490196 1.         0.30196078 0.        ]
[0.         0.01176471 0.         0.         0.         0.
 0.         0.         0.         0.24313725 0.56862745 0.8
 0.89411765 0.81176471 0.83529412 0.86666667 0.85490196 0.81568627
 0.82745098 0.85490196 0.87843137 0.8745098  0.85882353 0.84313725
 0.87843137 0.95686275 0.62352941 0.        ]
[0.         0.         0.         0.         0.07058824 0.17254902
 0.32156863 0.41960784 0.74117647 0.89411765 0.8627451  0.87058824
 0.85098039 0.88627451 0.78431373 0.80392157 0.82745098 0.90196078
 0.87843137 0.91764706 0.69019608 0.7372549  0.98039216 0.97254902
 0.91372549 0.93333333 0.84313725 0.        ]
[0.         0.22352941 0.73333333 0.81568627 0.87843137 0.86666667
 0.87843137 0.81568627 0.8        0.83921569 0.81568627 0.81960784
 0.78431373 0.62352941 0.96078431 0.75686275 0.80784314 0.8745098
 1.         1.         0.86666667 0.91764706 0.86666667 0.82745098
 0.8627451  0.90980392 0.96470588 0.        ]
[0.01176471 0.79215686 0.89411765 0.87843137 0.86666667 0.82745098
 0.82745098 0.83921569 0.80392157 0.80392157 0.80392157 0.8627451
 0.94117647 0.31372549 0.58823529 1.         0.89803922 0.86666667
 0.7372549  0.60392157 0.74901961 0.82352941 0.8        0.81960784
 0.87058824 0.89411765 0.88235294 0.        ]
[0.38431373 0.91372549 0.77647059 0.82352941 0.87058824 0.89803922
 0.89803922 0.91764706 0.97647059 0.8627451  0.76078431 0.84313725
 0.85098039 0.94509804 0.25490196 0.28627451 0.41568627 0.45882353
 0.65882353 0.85882353 0.86666667 0.84313725 0.85098039 0.8745098
```

```
 0.8745098  0.87843137 0.89803922 0.11372549]
[0.29411765 0.8        0.83137255 0.8        0.75686275 0.80392157
 0.82745098 0.88235294 0.84705882 0.7254902  0.77254902 0.80784314
 0.77647059 0.83529412 0.94117647 0.76470588 0.89019608 0.96078431
 0.9372549  0.8745098  0.85490196 0.83137255 0.81960784 0.87058824
 0.8627451  0.86666667 0.90196078 0.2627451 ]
[0.18823529 0.79607843 0.71764706 0.76078431 0.83529412 0.77254902
 0.7254902  0.74509804 0.76078431 0.75294118 0.79215686 0.83921569
 0.85882353 0.86666667 0.8627451  0.9254902  0.88235294 0.84705882
 0.78039216 0.80784314 0.72941176 0.70980392 0.69411765 0.6745098
 0.70980392 0.80392157 0.80784314 0.45098039]
[0.         0.47843137 0.85882353 0.75686275 0.70196078 0.67058824
 0.71764706 0.76862745 0.8        0.82352941 0.83529412 0.81176471
 0.82745098 0.82352941 0.78431373 0.76862745 0.76078431 0.74901961
 0.76470588 0.74901961 0.77647059 0.75294118 0.69019608 0.61176471
 0.65490196 0.69411765 0.82352941 0.36078431]
[0.         0.         0.29019608 0.74117647 0.83137255 0.74901961
 0.68627451 0.6745098  0.68627451 0.70980392 0.7254902  0.7372549
 0.74117647 0.7372549  0.75686275 0.77647059 0.8        0.81960784
 0.82352941 0.82352941 0.82745098 0.7372549  0.7372549  0.76078431
 0.75294118 0.84705882 0.66666667 0.        ]
[0.00784314 0.         0.         0.         0.25882353 0.78431373
 0.87058824 0.92941176 0.9372549  0.94901961 0.96470588 0.95294118
 0.95686275 0.86666667 0.8627451  0.75686275 0.74901961 0.70196078
 0.71372549 0.71372549 0.70980392 0.69019608 0.65098039 0.65882353
 0.38823529 0.22745098 0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.15686275 0.23921569 0.17254902 0.28235294 0.16078431
 0.1372549  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.        ]]
```

Image plot of the first image

[36]:
```
plt.figure(figsize=(4,4))
plt.imshow(X_train[0].reshape(28,28),cmap='gray')
plt.show()
```



[37]:
```
class_names[y_train[0]]
```

[37]:
```
'Ankle Boot'
```

Model Building

- Sequential Model : Simplest kind of keras model for neural networks that are just composed of a single stack of layers connected sequentially.
- First Layer (Input Layer) : Specify the input shape, which doesn't include the batch size only the shape of the instances. Keras needs to know the shape of the inputs so it can determine the shape of the connection weight matrix of the first hidden layer.
- Flattern Layer : Role is convert each input image to 1D array. Recieves a batch of shape [32, 28, 28], will reshape it to [32, 784].
- Dense Hidden Layer : 300 neurons and will use the ReLU activation function. Each dense layer manages it's own weight matrix containing all the connection weights between the neurons and their inputs. It also manages a vector of bias terms (one per neuron).
- Dense Hidden Layer : 100 neurons with ReLU activation function.
- Dense Output Layer : 10 neurons (one per class) using the softmax activation function because the classes are exclusive.

[38]:
```
tf.random.set_seed(42)
model = tf.keras.Sequential()
model.add(tf.keras.layers.Input(shape=[28, 28]))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(300, activation='relu'))
model.add(tf.keras.layers.Dense(100, activation='relu'))
```

```
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

```
2024-11-19 22:51:49.145414: E
external/local_xla/xla/stream_executor/cuda/cuda_driver.cc:152] failed call to
cuInit: INTERNAL: CUDA error: Failed call to cuInit: UNKNOWN ERROR (303)
```

[39]:
```
"""
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=[28, 28]),
    tf.keras.layers.Dense(300, activation='relu'),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')])
"""
```

[39]:
```
"\nmodel = tf.keras.Sequential([\n    tf.keras.layers.Flatten(input_shape=[28,
28]),\n    tf.keras.layers.Dense(300, activation='relu'),\n
tf.keras.layers.Dense(100, activation='relu'),\n    tf.keras.layers.Dense(10,
activation='softmax')])\n"
```

[40]:
```
model.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 300)	235,500
dense_1 (Dense)	(None, 100)	30,100
dense_2 (Dense)	(None, 10)	1,010

```
Total params: 266,610 (1.02 MB)

Trainable params: 266,610 (1.02 MB)

Non-trainable params: 0 (0.00 B)
```

Inspecting the model layers

[42]:
```
model.layers
```

[42]:
```
[<Flatten name=flatten, built=True>,
 <Dense name=dense, built=True>,
 <Dense name=dense_1, built=True>,
 <Dense name=dense_2, built=True>]
```

[43]:
```python
hidden1 = model.layers[1]
```

[44]:
```python
hidden1.name
```

[44]:
```
'dense'
```

[45]:
```python
model.get_layer('dense') is hidden1
```

[45]:
```
True
```

Weights and biases of the first hidden layer

[46]:
```python
weights, biases = hidden1.get_weights()
weights, weights.shape
```

[46]:
```
(array([[-0.06174467,  0.00136631, -0.00458926, …, -0.02218592,
         -0.05540018,  0.01706963],
        [-0.01772485,  0.06739265, -0.05860752, …,  0.04124127,
          0.00802088, -0.02619709],
        [-0.0175525 ,  0.01896753, -0.02102911, …,  0.05805399,
         -0.00629631, -0.01752633],
        …,
        [ 0.05329123, -0.03537329,  0.04880107, …, -0.00063659,
          0.02894461,  0.02290244],
        [ 0.04168116,  0.06314105, -0.02075434, …, -0.04146532,
         -0.04206529,  0.03249374],
        [-0.06970996, -0.03089734, -0.05079816, …,  0.04138362,
         -0.04707992, -0.0712704 ]], dtype=float32),
 (784, 300))
```

[47]:
```python
biases, biases.shape
```

[47]:
```
(array([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., 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., 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., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32),
 (300,))
```

Model Compilation using appropriate metrics

[32]:
```
model.compile(loss="sparse_categorical_crossentropy", optimizer="sgd",␣
 ↪metrics=["accuracy"])
```

Model fitting step

[33]:
```
history = model.fit(X_train, y_train, epochs=30, validation_data=(X_valid,␣
 ↪y_valid))
```

```
Epoch 1/30

2024-11-03 21:40:55.964101: W
external/local_xla/xla/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
172480000 exceeds 10% of free system memory.

1719/1719              5s 3ms/step -
accuracy: 0.6777 - loss: 1.0218 - val_accuracy: 0.8212 - val_loss: 0.5101
Epoch 2/30
1719/1719              4s 2ms/step -
accuracy: 0.8278 - loss: 0.5044 - val_accuracy: 0.8338 - val_loss: 0.4601
Epoch 3/30
1719/1719              4s 2ms/step -
accuracy: 0.8441 - loss: 0.4488 - val_accuracy: 0.8400 - val_loss: 0.4371
Epoch 4/30
1719/1719              4s 2ms/step -
accuracy: 0.8529 - loss: 0.4183 - val_accuracy: 0.8440 - val_loss: 0.4215
Epoch 5/30
1719/1719              4s 2ms/step -
accuracy: 0.8591 - loss: 0.3966 - val_accuracy: 0.8484 - val_loss: 0.4108
Epoch 6/30
1719/1719              4s 2ms/step -
accuracy: 0.8657 - loss: 0.3797 - val_accuracy: 0.8504 - val_loss: 0.4017
Epoch 7/30
1719/1719              4s 2ms/step -
accuracy: 0.8702 - loss: 0.3653 - val_accuracy: 0.8528 - val_loss: 0.3931
Epoch 8/30
1719/1719              4s 2ms/step -
accuracy: 0.8754 - loss: 0.3531 - val_accuracy: 0.8554 - val_loss: 0.3876
```

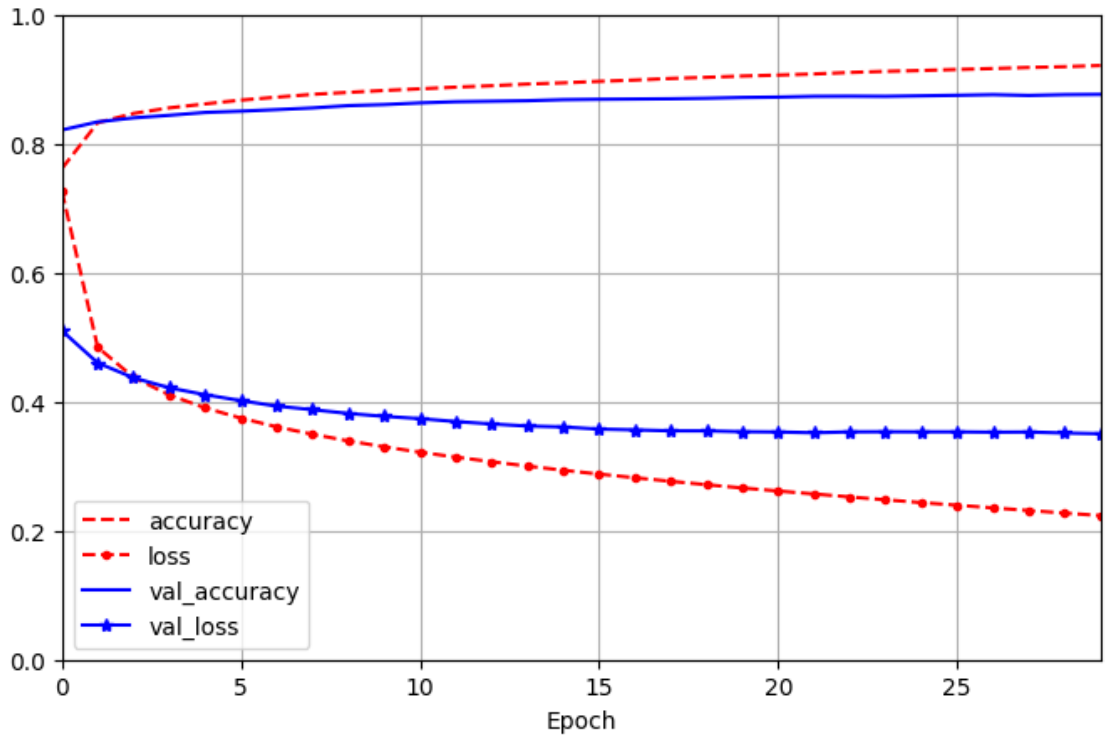
```
Epoch 9/30
1719/1719            4s 2ms/step -
accuracy: 0.8783 - loss: 0.3424 - val_accuracy: 0.8590 - val_loss: 0.3817
Epoch 10/30
1719/1719            4s 2ms/step -
accuracy: 0.8813 - loss: 0.3327 - val_accuracy: 0.8606 - val_loss: 0.3773
Epoch 11/30
1719/1719            4s 2ms/step -
accuracy: 0.8840 - loss: 0.3241 - val_accuracy: 0.8632 - val_loss: 0.3737
Epoch 12/30
1719/1719            4s 2ms/step -
accuracy: 0.8870 - loss: 0.3162 - val_accuracy: 0.8650 - val_loss: 0.3692
Epoch 13/30
1719/1719            4s 2ms/step -
accuracy: 0.8889 - loss: 0.3088 - val_accuracy: 0.8658 - val_loss: 0.3654
Epoch 14/30
1719/1719            4s 2ms/step -
accuracy: 0.8918 - loss: 0.3017 - val_accuracy: 0.8666 - val_loss: 0.3623
Epoch 15/30
1719/1719            4s 2ms/step -
accuracy: 0.8939 - loss: 0.2951 - val_accuracy: 0.8682 - val_loss: 0.3608
Epoch 16/30
1719/1719            4s 2ms/step -
accuracy: 0.8963 - loss: 0.2890 - val_accuracy: 0.8688 - val_loss: 0.3575
Epoch 17/30
1719/1719            4s 2ms/step -
accuracy: 0.8985 - loss: 0.2830 - val_accuracy: 0.8692 - val_loss: 0.3563
Epoch 18/30
1719/1719            4s 2ms/step -
accuracy: 0.9013 - loss: 0.2774 - val_accuracy: 0.8696 - val_loss: 0.3548
Epoch 19/30
1719/1719            4s 2ms/step -
accuracy: 0.9032 - loss: 0.2720 - val_accuracy: 0.8704 - val_loss: 0.3548
Epoch 20/30
1719/1719            4s 2ms/step -
accuracy: 0.9055 - loss: 0.2668 - val_accuracy: 0.8716 - val_loss: 0.3535
Epoch 21/30
1719/1719            4s 2ms/step -
accuracy: 0.9073 - loss: 0.2621 - val_accuracy: 0.8722 - val_loss: 0.3530
Epoch 22/30
1719/1719            4s 2ms/step -
accuracy: 0.9090 - loss: 0.2573 - val_accuracy: 0.8734 - val_loss: 0.3518
Epoch 23/30
1719/1719            4s 2ms/step -
accuracy: 0.9115 - loss: 0.2527 - val_accuracy: 0.8736 - val_loss: 0.3532
Epoch 24/30
1719/1719            4s 2ms/step -
accuracy: 0.9132 - loss: 0.2481 - val_accuracy: 0.8734 - val_loss: 0.3534
```

```
Epoch 25/30
1719/1719             4s 2ms/step -
accuracy: 0.9145 - loss: 0.2437 - val_accuracy: 0.8742 - val_loss: 0.3532
Epoch 26/30
1719/1719             4s 2ms/step -
accuracy: 0.9165 - loss: 0.2393 - val_accuracy: 0.8750 - val_loss: 0.3532
Epoch 27/30
1719/1719             4s 2ms/step -
accuracy: 0.9179 - loss: 0.2351 - val_accuracy: 0.8760 - val_loss: 0.3527
Epoch 28/30
1719/1719             4s 2ms/step -
accuracy: 0.9195 - loss: 0.2310 - val_accuracy: 0.8748 - val_loss: 0.3530
Epoch 29/30
1719/1719             4s 2ms/step -
accuracy: 0.9203 - loss: 0.2270 - val_accuracy: 0.8760 - val_loss: 0.3514
Epoch 30/30
1719/1719             4s 2ms/step -
accuracy: 0.9227 - loss: 0.2232 - val_accuracy: 0.8764 - val_loss: 0.3504
```

Plot of Testing set Accuracy, Validation accuracy, Test loss and Validation set Loss

[36]:
```
import matplotlib.pyplot as plt
import pandas as pd

pd.DataFrame(history.history).plot(figsize=(8, 5), xlim=[0, 29], ylim=[0, 1],␣
 ↪grid=True, xlabel="Epoch", style=["r--", "r--.","b-", "b-*"])
plt.show()
```



Model Prediction and Evaluation

[37]:
```
model.evaluate(X_test, y_test)
```

```
313/313             0s 892us/step -
accuracy: 0.8734 - loss: 0.3707
```

[37]:
```
[0.36567622423171997, 0.8747000098228455]
```

Seperate a group of 3 datapoints and predict them seperately using the model probabilities using numpy

[38]:
```
X_new = X_test[:3]
y_proba = model.predict(X_new)
y_proba.round(2)
```

```
1/1             0s 50ms/step
```

[38]:
```
array([[0.  , 0.  , 0.  , 0.  , 0.  , 0.22, 0.  , 0.01, 0.  , 0.77],
       [0.  , 0.  , 1.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ],
       [0.  , 1.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ]],
      dtype=float32)
```

[39]:
```
import numpy as np
y_pred = y_proba.argmax(axis=-1)
y_pred
```

[39]:
```
array([9, 2, 1])
```

[40]:
```
np.array(class_names)[y_pred]
```

[40]:
```
array(['Ankle Boot', 'Pullover', 'Trouser'], dtype='<U11')
```

[41]:
```
y_new = y_test[:3]
y_new
```

[41]:
```
array([9, 2, 1], dtype=uint8)
```