Classifying-Fashion-MNIST

November 11, 2022

1 Classifying Fashion-MNIST

1.1 Import Resources

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'

import numpy as np
import matplotlib.pyplot as plt

import tensorflow as tf
import tensorflow_datasets as tfds
tfds.disable_progress_bar()
```

[3]:
```
import logging
logger = tf.get_logger()
logger.setLevel(logging.ERROR)
```

[4]:
```
print('Using:')
print('\t\u2022 TensorFlow version:', tf.__version__)
print('\t\u2022 tf.keras version:', tf.keras.__version__)
print('\t\u2022 Running on GPU' if tf.test.is_gpu_available() else '\t\u2022 GPU␣
↪device not found. Running on CPU')
```

```
Using:
        • TensorFlow version: 2.9.1
        • tf.keras version: 2.9.0
        • GPU device not found. Running on CPU
```

1.2 Load the Dataset

[6]:
```
train_split = 60
test_val_split = 20
```

```
dataset, dataset_info = tfds.load('fashion_mnist', split=['train[:
↪60%]','train[60%:80%]','train[80%:]'], as_supervised=True, with_info=True)

training_set, validation_set, test_set = dataset
```

```
Downloading and preparing dataset Unknown size (download: Unknown size,
generated: Unknown size, total: Unknown size) to
C:\Users\LENOVO\tensorflow_datasets\fashion_mnist\3.0.1...
Dataset fashion_mnist downloaded and prepared to
C:\Users\LENOVO\tensorflow_datasets\fashion_mnist\3.0.1. Subsequent calls will
reuse this data.
```

1.3 Explore the Dataset

[7]:
```
# Display dataset
dataset
```

[7]:
```
[<PrefetchDataset element_spec=(TensorSpec(shape=(28, 28, 1), dtype=tf.uint8,
name=None), TensorSpec(shape=(), dtype=tf.int64, name=None))>,
 <PrefetchDataset element_spec=(TensorSpec(shape=(28, 28, 1), dtype=tf.uint8,
name=None), TensorSpec(shape=(), dtype=tf.int64, name=None))>,
 <PrefetchDataset element_spec=(TensorSpec(shape=(28, 28, 1), dtype=tf.uint8,
name=None), TensorSpec(shape=(), dtype=tf.int64, name=None))>]
```

[8]:
```
# Display dataset_info
dataset_info
```

[8]:
```
tfds.core.DatasetInfo(
    name='fashion_mnist',
    full_name='fashion_mnist/3.0.1',
    description="""
    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.
    """,
    homepage='https://github.com/zalandoresearch/fashion-mnist',
    data_path='C:\\Users\\LENOVO\\tensorflow_datasets\\fashion_mnist\\3.0.1',
    file_format=tfrecord,
    download_size=29.45 MiB,
    dataset_size=36.42 MiB,
    features=FeaturesDict({
        'image': Image(shape=(28, 28, 1), dtype=tf.uint8),
        'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=10),
    }),
    supervised_keys=('image', 'label'),
```

```
    disable_shuffling=False,
    splits={
        'test': <SplitInfo num_examples=10000, num_shards=1>,
        'train': <SplitInfo num_examples=60000, num_shards=1>,
    },
    citation="""@article{DBLP:journals/corr/abs-1708-07747,
      author    = {Han Xiao and
                   Kashif Rasul and
                   Roland Vollgraf},
      title     = {Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine
Learning
                   Algorithms},
      journal   = {CoRR},
      volume    = {abs/1708.07747},
      year      = {2017},
      url       = {http://arxiv.org/abs/1708.07747},
      archivePrefix = {arXiv},
      eprint    = {1708.07747},
      timestamp = {Mon, 13 Aug 2018 16:47:27 +0200},
      biburl    = {https://dblp.org/rec/bib/journals/corr/abs-1708-07747},
      bibsource = {dblp computer science bibliography, https://dblp.org}
    }""",
)
```

[9]:
```
total_examples = dataset_info.splits['train'].num_examples + dataset_info.
 ↪splits['test'].num_examples

num_training_examples = (total_examples * train_split) // 100
num_validation_examples = (total_examples * test_val_split) // 100
num_test_examples = num_validation_examples

print('There are {:,} images in the training set'.format(num_training_examples))
print('There are {:,} images in the validation set'.
 ↪format(num_validation_examples))
print('There are {:,} images in the test set'.format(num_test_examples))
```

```
There are 42,000 images in the training set
There are 14,000 images in the validation set
There are 14,000 images in the test set
```

The images in this dataset are 28×28 arrays, with pixel values in the range `[0, 255]`. The *labels* are an array of integers, in the range `[0, 9]`. These correspond to the *class* of clothing the image represents:

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

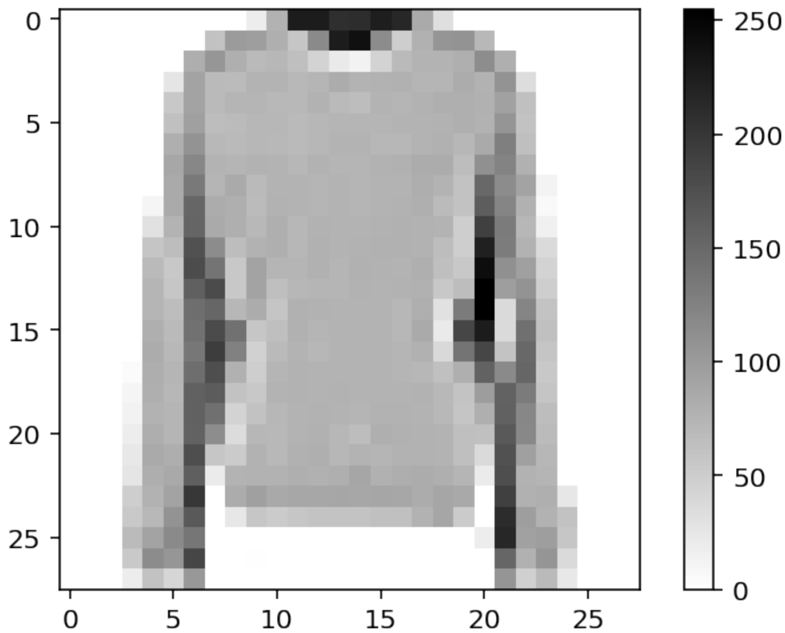
Each image is mapped to a single label. Since the *class names* are not included with the dataset, we create them here to use later when plotting the images:

[10]:
```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal',      'Shirt',   'Sneaker',  'Bag',   'Ankle boot']
```

[11]:
```
for image, label in training_set.take(1):
    image = image.numpy().squeeze()
    label = label.numpy()

plt.imshow(image, cmap= plt.cm.binary)
plt.colorbar()
plt.show()

print('The label of this image is:', label)
print('The class name of this image is:', class_names[label])
```




```
The label of this image is: 2
The class name of this image is: Pullover
```

1.4 Create Pipeline

```
[12]: def normalize(image, label):
          image = tf.cast(image, tf.float32)
          image /= 255
          return image, label

      batch_size = 64

      training_batches = training_set.cache().shuffle(num_training_examples//4).
       ↪batch(batch_size).map(normalize).prefetch(1)
      validation_batches = validation_set.cache().batch(batch_size).map(normalize).
       ↪prefetch(1)
      testing_batches = test_set.cache().batch(batch_size).map(normalize).prefetch(1)
```

1.5 Build the Model

```
[13]: model = tf.keras.Sequential([
              tf.keras.layers.Flatten(input_shape=(28,28,1)),
              tf.keras.layers.Dense(256, activation = 'relu'),
              tf.keras.layers.Dense(128, activation = 'relu'),
              tf.keras.layers.Dense(64, activation = 'relu'),
```

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

1.6 Train the Model

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

EPOCHS = 30

history = model.fit(training_batches,
                    epochs = EPOCHS,
                    validation_data=validation_batches)
```

```
Epoch 1/30
563/563 [==============================] - 4s 6ms/step - loss: 0.5491 -
accuracy: 0.8075 - val_loss: 0.4463 - val_accuracy: 0.8406
Epoch 2/30
563/563 [==============================] - 2s 4ms/step - loss: 0.3957 -
accuracy: 0.8559 - val_loss: 0.3826 - val_accuracy: 0.8614
Epoch 3/30
563/563 [==============================] - 2s 4ms/step - loss: 0.3508 -
accuracy: 0.8718 - val_loss: 0.3425 - val_accuracy: 0.8752
Epoch 4/30
563/563 [==============================] - 2s 4ms/step - loss: 0.3267 -
accuracy: 0.8792 - val_loss: 0.3599 - val_accuracy: 0.8749
Epoch 5/30
563/563 [==============================] - 2s 4ms/step - loss: 0.3052 -
accuracy: 0.8871 - val_loss: 0.3421 - val_accuracy: 0.8763
Epoch 6/30
563/563 [==============================] - 2s 4ms/step - loss: 0.2877 -
accuracy: 0.8935 - val_loss: 0.3527 - val_accuracy: 0.8728
Epoch 7/30
563/563 [==============================] - 2s 4ms/step - loss: 0.2794 -
accuracy: 0.8964 - val_loss: 0.3206 - val_accuracy: 0.8875
Epoch 8/30
563/563 [==============================] - 2s 4ms/step - loss: 0.2650 -
accuracy: 0.9005 - val_loss: 0.3600 - val_accuracy: 0.8748
Epoch 9/30
563/563 [==============================] - 2s 4ms/step - loss: 0.2507 -
accuracy: 0.9046 - val_loss: 0.3056 - val_accuracy: 0.8884
Epoch 10/30
563/563 [==============================] - 2s 4ms/step - loss: 0.2458 -
accuracy: 0.9062 - val_loss: 0.3284 - val_accuracy: 0.8847
Epoch 11/30
563/563 [==============================] - 2s 4ms/step - loss: 0.2339 -
accuracy: 0.9112 - val_loss: 0.3377 - val_accuracy: 0.8844
```

```
Epoch 12/30
563/563 [==============================] - 2s 4ms/step - loss: 0.2246 -
accuracy: 0.9138 - val_loss: 0.3506 - val_accuracy: 0.8792
Epoch 13/30
563/563 [==============================] - 2s 4ms/step - loss: 0.2183 -
accuracy: 0.9162 - val_loss: 0.3180 - val_accuracy: 0.8888
Epoch 14/30
563/563 [==============================] - 2s 4ms/step - loss: 0.2067 -
accuracy: 0.9202 - val_loss: 0.3386 - val_accuracy: 0.8890
Epoch 15/30
563/563 [==============================] - 2s 4ms/step - loss: 0.2027 -
accuracy: 0.9233 - val_loss: 0.3259 - val_accuracy: 0.8912
Epoch 16/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1941 -
accuracy: 0.9267 - val_loss: 0.3504 - val_accuracy: 0.8842
Epoch 17/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1940 -
accuracy: 0.9267 - val_loss: 0.3253 - val_accuracy: 0.8938
Epoch 18/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1836 -
accuracy: 0.9293 - val_loss: 0.3766 - val_accuracy: 0.8796
Epoch 19/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1723 -
accuracy: 0.9338 - val_loss: 0.3634 - val_accuracy: 0.8912
Epoch 20/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1748 -
accuracy: 0.9326 - val_loss: 0.3404 - val_accuracy: 0.8946
Epoch 21/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1669 -
accuracy: 0.9351 - val_loss: 0.3740 - val_accuracy: 0.8885
Epoch 22/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1639 -
accuracy: 0.9362 - val_loss: 0.3663 - val_accuracy: 0.8919
Epoch 23/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1544 -
accuracy: 0.9404 - val_loss: 0.3423 - val_accuracy: 0.8968
Epoch 24/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1462 -
accuracy: 0.9428 - val_loss: 0.3737 - val_accuracy: 0.8895
Epoch 25/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1436 -
accuracy: 0.9451 - val_loss: 0.3572 - val_accuracy: 0.8932
Epoch 26/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1437 -
accuracy: 0.9445 - val_loss: 0.3871 - val_accuracy: 0.8947
Epoch 27/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1362 -
accuracy: 0.9465 - val_loss: 0.3790 - val_accuracy: 0.9003
```

```
Epoch 28/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1347 -
accuracy: 0.9472 - val_loss: 0.4164 - val_accuracy: 0.8886
Epoch 29/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1233 -
accuracy: 0.9527 - val_loss: 0.3944 - val_accuracy: 0.8943
Epoch 30/30
563/563 [==============================] - 2s 4ms/step - loss: 0.1248 -
accuracy: 0.9536 - val_loss: 0.4238 - val_accuracy: 0.8981
```

1.7 Evaluate Loss and Accuracy on the Test Set

[15]:
```
loss, accuracy = model.evaluate(testing_batches)

print('\nLoss on the TEST Set: {:,.3f}'.format(loss))
print('Accuracy on the TEST Set: {:.3%}'.format(accuracy))
```

```
188/188 [==============================] - 1s 4ms/step - loss: 0.4134 -
accuracy: 0.8955

Loss on the TEST Set: 0.413
Accuracy on the TEST Set: 89.550%
```

1.8 Loss and Validation Plots

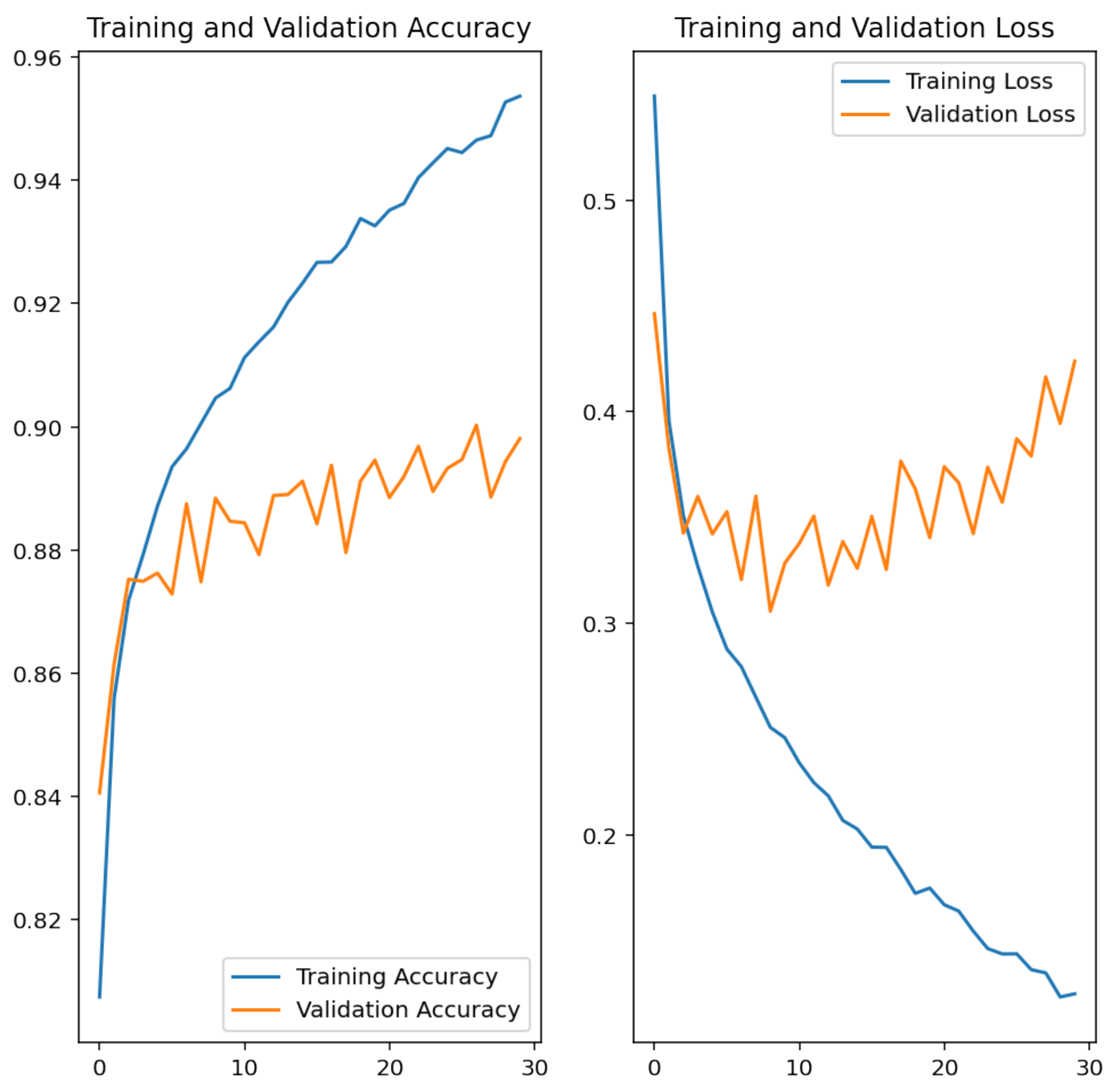
[16]:
```
training_accuracy = history.history['accuracy']
validation_accuracy = history.history['val_accuracy']

training_loss = history.history['loss']
validation_loss = history.history['val_loss']

epochs_range=range(EPOCHS)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, training_accuracy, label='Training Accuracy')
plt.plot(epochs_range, validation_accuracy, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, training_loss, label='Training Loss')
plt.plot(epochs_range, validation_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



1.9 Early Stopping

[17]:
```
model = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=(28,28,1)),
        tf.keras.layers.Dense(256, activation = 'relu'),
        tf.keras.layers.Dense(128, activation = 'relu'),
        tf.keras.layers.Dense(64, activation = 'relu'),
        tf.keras.layers.Dense(10, activation = 'softmax')
])

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

```python
# Stop training when there is no improvement in the validation loss for 5␣
 ↪consecutive epochs
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)

history = model.fit(training_batches,
                    epochs = 100,
                    validation_data=validation_batches,
                    callbacks=[early_stopping])
```

```
Epoch 1/100
563/563 [==============================] - 3s 4ms/step - loss: 0.5489 -
accuracy: 0.8048 - val_loss: 0.4292 - val_accuracy: 0.8393
Epoch 2/100
563/563 [==============================] - 2s 4ms/step - loss: 0.3916 -
accuracy: 0.8575 - val_loss: 0.3855 - val_accuracy: 0.8584
Epoch 3/100
563/563 [==============================] - 2s 4ms/step - loss: 0.3534 -
accuracy: 0.8706 - val_loss: 0.3741 - val_accuracy: 0.8683
Epoch 4/100
563/563 [==============================] - 3s 5ms/step - loss: 0.3265 -
accuracy: 0.8806 - val_loss: 0.3523 - val_accuracy: 0.8729
Epoch 5/100
563/563 [==============================] - 2s 4ms/step - loss: 0.3066 -
accuracy: 0.8859 - val_loss: 0.3425 - val_accuracy: 0.8780
Epoch 6/100
563/563 [==============================] - 2s 4ms/step - loss: 0.2892 -
accuracy: 0.8907 - val_loss: 0.3599 - val_accuracy: 0.8725
Epoch 7/100
563/563 [==============================] - 2s 4ms/step - loss: 0.2758 -
accuracy: 0.8975 - val_loss: 0.3409 - val_accuracy: 0.8748
Epoch 8/100
563/563 [==============================] - 2s 4ms/step - loss: 0.2653 -
accuracy: 0.9003 - val_loss: 0.3200 - val_accuracy: 0.8865
Epoch 9/100
563/563 [==============================] - 2s 4ms/step - loss: 0.2540 -
accuracy: 0.9040 - val_loss: 0.3253 - val_accuracy: 0.8867
Epoch 10/100
563/563 [==============================] - 2s 4ms/step - loss: 0.2475 -
accuracy: 0.9053 - val_loss: 0.3540 - val_accuracy: 0.8785
Epoch 11/100
563/563 [==============================] - 2s 4ms/step - loss: 0.2350 -
accuracy: 0.9115 - val_loss: 0.3224 - val_accuracy: 0.8870
Epoch 12/100
563/563 [==============================] - 3s 4ms/step - loss: 0.2275 -
accuracy: 0.9150 - val_loss: 0.3345 - val_accuracy: 0.8867
Epoch 13/100
563/563 [==============================] - 2s 4ms/step - loss: 0.2202 -
```

```
accuracy: 0.9184 - val_loss: 0.3302 - val_accuracy: 0.8876
```

[18]:

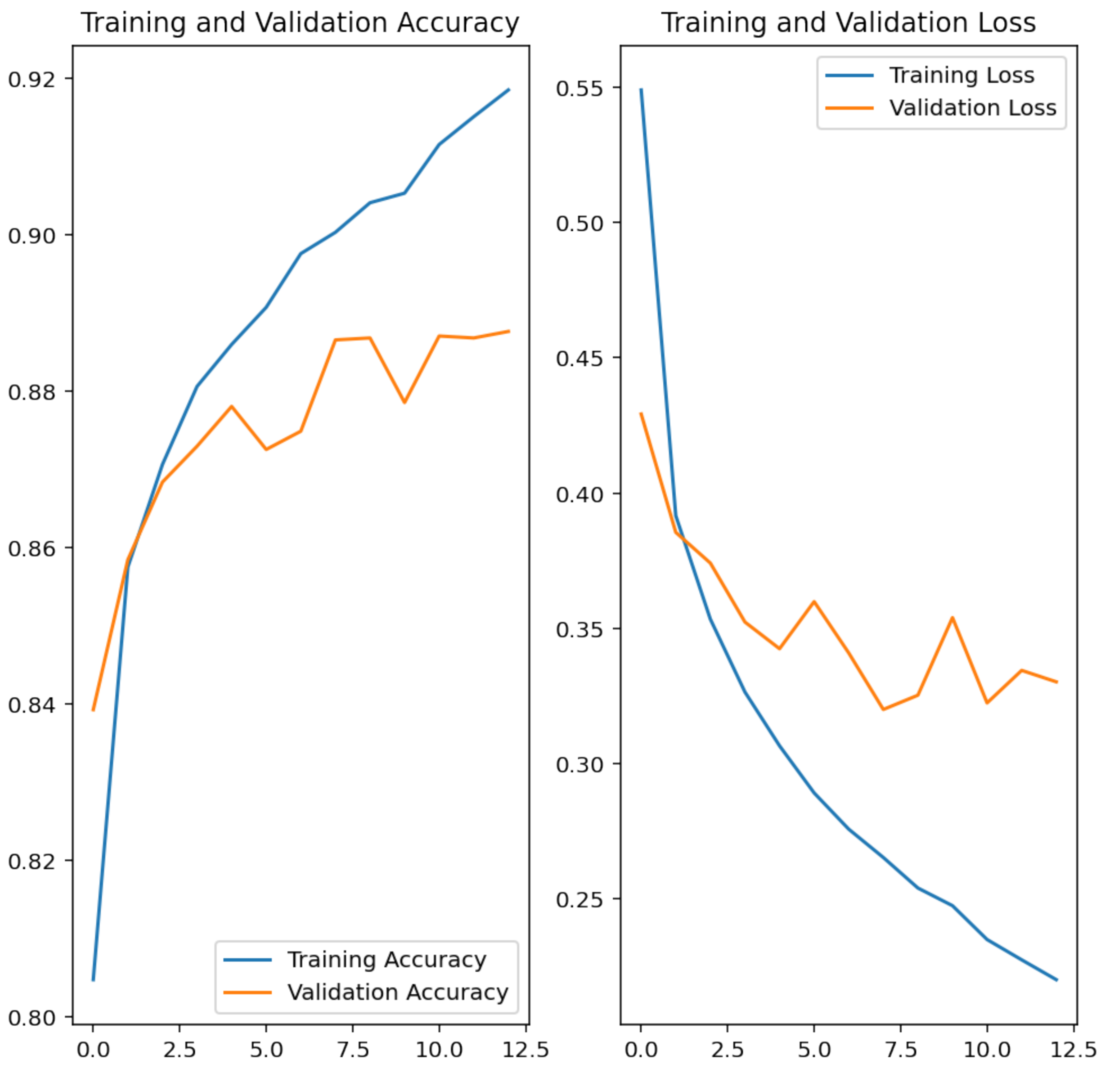
```python
training_accuracy = history.history['accuracy']
validation_accuracy = history.history['val_accuracy']

training_loss = history.history['loss']
validation_loss = history.history['val_loss']

epochs_range=range(len(training_accuracy))

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, training_accuracy, label='Training Accuracy')
plt.plot(epochs_range, validation_accuracy, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, training_loss, label='Training Loss')
plt.plot(epochs_range, validation_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



1.10 Dropout

[19]:
```
model = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=(28,28,1)),
        tf.keras.layers.Dense(256, activation = 'relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(128, activation = 'relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(64, activation = 'relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(10, activation = 'softmax')
])
```

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


history = model.fit(training_batches,
                    epochs = 30,
                    validation_data=validation_batches)
```

```
Epoch 1/30
563/563 [==============================] - 3s 5ms/step - loss: 0.6822 -
accuracy: 0.7566 - val_loss: 0.4395 - val_accuracy: 0.8422
Epoch 2/30
563/563 [==============================] - 3s 5ms/step - loss: 0.4745 -
accuracy: 0.8324 - val_loss: 0.4032 - val_accuracy: 0.8525
Epoch 3/30
563/563 [==============================] - 3s 5ms/step - loss: 0.4240 -
accuracy: 0.8461 - val_loss: 0.3955 - val_accuracy: 0.8533
Epoch 4/30
563/563 [==============================] - 3s 5ms/step - loss: 0.4002 -
accuracy: 0.8557 - val_loss: 0.3587 - val_accuracy: 0.8688
Epoch 5/30
563/563 [==============================] - 3s 5ms/step - loss: 0.3741 -
accuracy: 0.8650 - val_loss: 0.3556 - val_accuracy: 0.8707
Epoch 6/30
563/563 [==============================] - 3s 5ms/step - loss: 0.3618 -
accuracy: 0.8681 - val_loss: 0.3711 - val_accuracy: 0.8612
Epoch 7/30
563/563 [==============================] - 3s 5ms/step - loss: 0.3498 -
accuracy: 0.8711 - val_loss: 0.3414 - val_accuracy: 0.8777
Epoch 8/30
563/563 [==============================] - 3s 5ms/step - loss: 0.3386 -
accuracy: 0.8760 - val_loss: 0.3531 - val_accuracy: 0.8701
Epoch 9/30
563/563 [==============================] - 3s 5ms/step - loss: 0.3272 -
accuracy: 0.8809 - val_loss: 0.3199 - val_accuracy: 0.8861
Epoch 10/30
563/563 [==============================] - 3s 5ms/step - loss: 0.3223 -
accuracy: 0.8819 - val_loss: 0.3417 - val_accuracy: 0.8816
Epoch 11/30
563/563 [==============================] - 3s 4ms/step - loss: 0.3092 -
accuracy: 0.8852 - val_loss: 0.3225 - val_accuracy: 0.8848
Epoch 12/30
563/563 [==============================] - 3s 5ms/step - loss: 0.3016 -
accuracy: 0.8890 - val_loss: 0.3295 - val_accuracy: 0.8830
Epoch 13/30
563/563 [==============================] - 3s 5ms/step - loss: 0.3023 -
accuracy: 0.8898 - val_loss: 0.3539 - val_accuracy: 0.8767
```

```
Epoch 14/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2896 -
accuracy: 0.8936 - val_loss: 0.3262 - val_accuracy: 0.8863
Epoch 15/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2865 -
accuracy: 0.8935 - val_loss: 0.3336 - val_accuracy: 0.8867
Epoch 16/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2818 -
accuracy: 0.8956 - val_loss: 0.3223 - val_accuracy: 0.8871
Epoch 17/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2748 -
accuracy: 0.8971 - val_loss: 0.3252 - val_accuracy: 0.8891
Epoch 18/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2733 -
accuracy: 0.8979 - val_loss: 0.3103 - val_accuracy: 0.8906
Epoch 19/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2693 -
accuracy: 0.9004 - val_loss: 0.3266 - val_accuracy: 0.8879
Epoch 20/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2650 -
accuracy: 0.9017 - val_loss: 0.3255 - val_accuracy: 0.8920
Epoch 21/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2589 -
accuracy: 0.9038 - val_loss: 0.3233 - val_accuracy: 0.8829
Epoch 22/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2539 -
accuracy: 0.9051 - val_loss: 0.3131 - val_accuracy: 0.8922
Epoch 23/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2530 -
accuracy: 0.9075 - val_loss: 0.3279 - val_accuracy: 0.8831
Epoch 24/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2437 -
accuracy: 0.9073 - val_loss: 0.3240 - val_accuracy: 0.8917
Epoch 25/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2441 -
accuracy: 0.9089 - val_loss: 0.3177 - val_accuracy: 0.8928
Epoch 26/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2455 -
accuracy: 0.9081 - val_loss: 0.3108 - val_accuracy: 0.8942
Epoch 27/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2381 -
accuracy: 0.9111 - val_loss: 0.3187 - val_accuracy: 0.8917
Epoch 28/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2347 -
accuracy: 0.9120 - val_loss: 0.3259 - val_accuracy: 0.8918
Epoch 29/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2353 -
accuracy: 0.9109 - val_loss: 0.3122 - val_accuracy: 0.8937
```

```
Epoch 30/30
563/563 [==============================] - 3s 5ms/step - loss: 0.2272 -
accuracy: 0.9147 - val_loss: 0.3146 - val_accuracy: 0.8900
```

[20]:

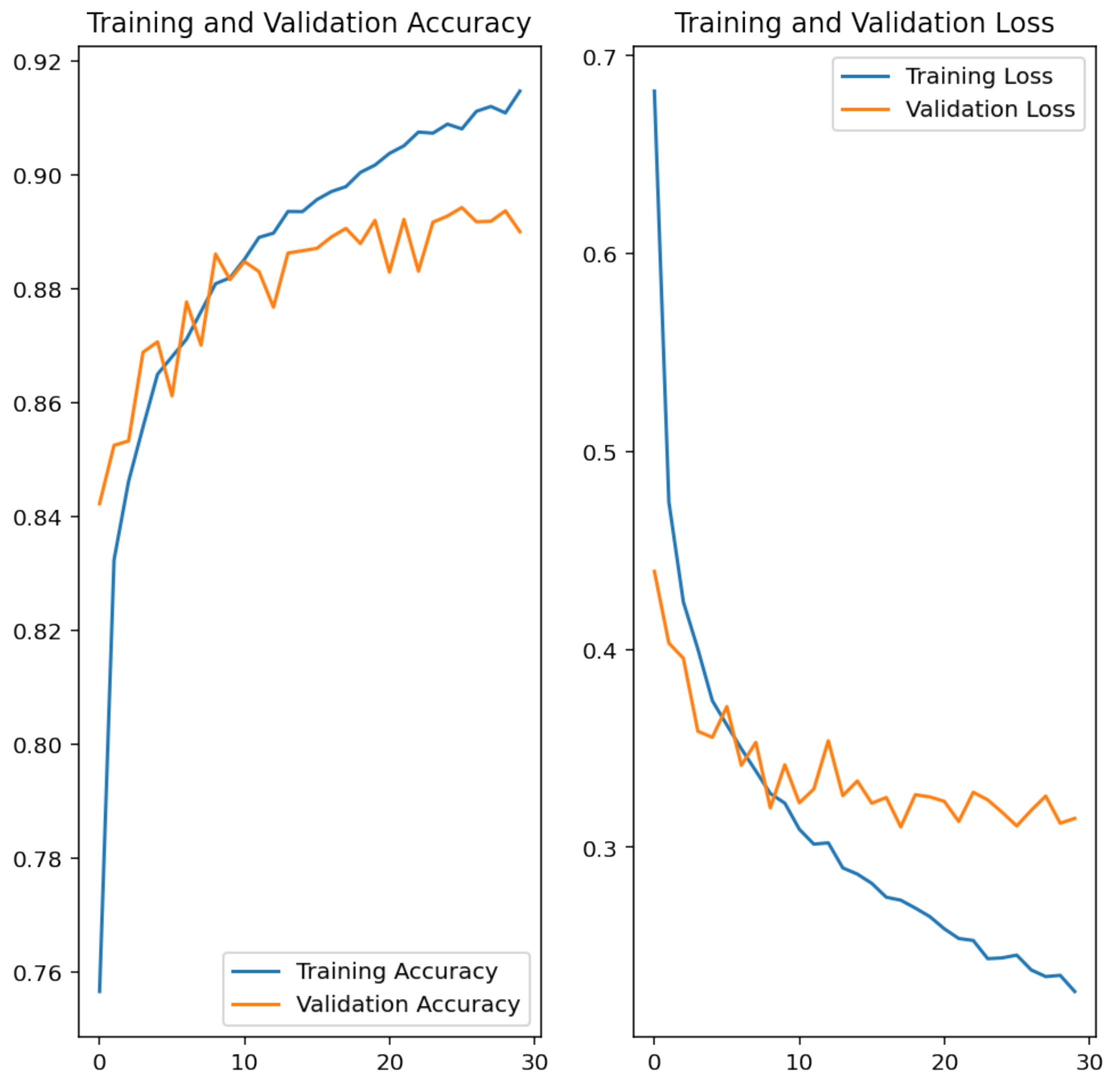
```
training_accuracy = history.history['accuracy']
validation_accuracy = history.history['val_accuracy']

training_loss = history.history['loss']
validation_loss = history.history['val_loss']

epochs_range=range(len(training_accuracy))

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, training_accuracy, label='Training Accuracy')
plt.plot(epochs_range, validation_accuracy, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, training_loss, label='Training Loss')
plt.plot(epochs_range, validation_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



1.11 Inference

[21]:
```
for image_batch, label_batch in testing_batches.take(1):
    ps = model.predict(image_batch)
    images = image_batch.numpy().squeeze()
    labels = label_batch.numpy()


plt.figure(figsize=(10,15))

for n in range(30):
    plt.subplot(6,5,n+1)
    plt.imshow(images[n], cmap = plt.cm.binary)
```

```
    color = 'green' if np.argmax(ps[n]) == labels[n] else 'red'
    plt.title(class_names[np.argmax(ps[n])], color=color)
    plt.axis('off')
```

```
2/2 [==============================] - 0s 3ms/step
```


Sneaker
Shirt
Sandal
Sneaker
Pullover
Pullover
T-shirt/top
Dress
Sandal
Coat
Bag
Bag
Dress
Sandal
T-shirt/top
Shirt
Bag
Dress
Sneaker
Ankle boot
Coat
Sneaker
Sneaker
Ankle boot
T-shirt/top
Coat
T-shirt/top
Sneaker
T-shirt/top
Ankle boot

```
[ ]:
```