Final Project - Classification of Clothing Items

Pragya Jalan

Importing Libraries

```
import keras
import numpy as np
from keras.datasets import fashion_mnist
from matplotlib import pyplot
from scipy.misc import toimage
from sklearn.metrics import confusion_matrix
from collections import OrderedDict
import itertools

import cv2
from matplotlib import pyplot as plt
from keras.preprocessing import image
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
from keras import backend as K

%matplotlib inline
```

```
Using TensorFlow backend.
```

Importing Data

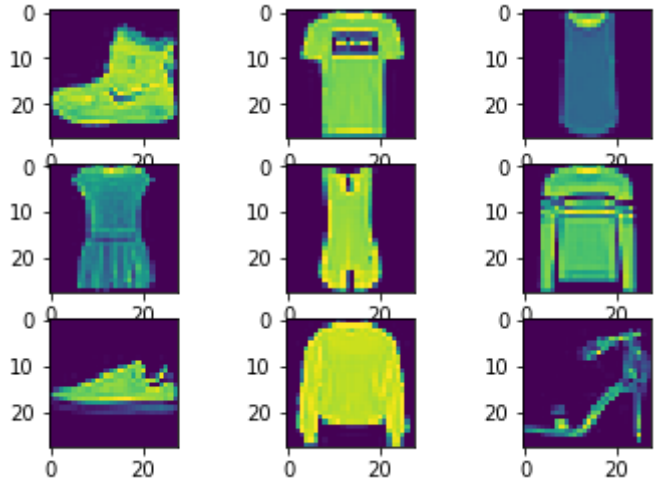
```
fashion_mnist = keras.datasets.fashion_mnist

(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()
```

```
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.c
32768/29515 [==============================] - 0s 3us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.c
26427392/26421880 [==============================] - 2s 0us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.c
8192/5148 [===============================================] - 0s 0us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.c
4423680/4422102 [==============================] - 1s 0us/step
```

```
for i in range(0, 9):
  pyplot.subplot(330 + 1 + i)
```

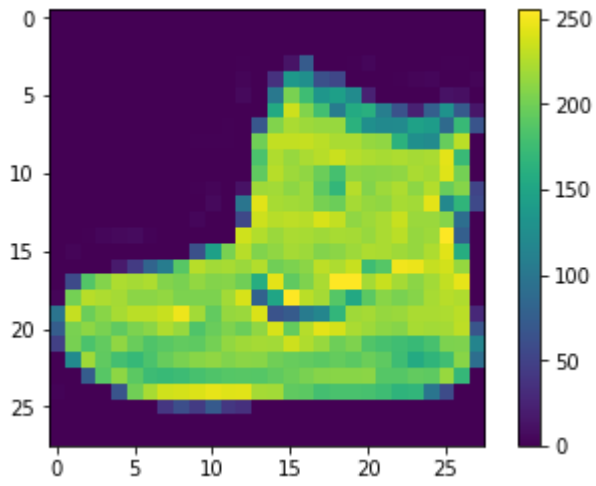
```
  pyplot.imshow(toimage(X_train[i]))
# show the plot

pyplot.show()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: DeprecationWar
`toimage` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use Pillow's ``Image.fromarray`` directly instead.
  This is separate from the ipykernel package so we can avoid doing imports ur
```



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

```
pyplot.figure()
pyplot.imshow(X_train[0])
pyplot.colorbar()
pyplot.grid(False)
pyplot.show()
```



```
pyplot.figure(figsize=(10,10))
for i in range(25):
    pyplot.subplot(5,5,i+1)
    pyplot.xticks([])
    pyplot.yticks([])
    pyplot.grid(False)
    pyplot.imshow(X_train[i], cmap=pyplot.cm.binary)
    pyplot.xlabel(class_names[y_train[i]])
```

```
pyplot.show()
```

Data Wrangling

```
print('train data shape : {}'.format(X_train.shape))
print('test data shape : {}'.format(X_test.shape))
```

```
train data shape : (60000, 28, 28)
test data shape : (10000, 28, 28)
```

```
X_train = X_train.reshape(-1, 28, 28, 1)
X_test = X_test.reshape(-1, 28, 28, 1)

X_train = X_train / 255.0
X_test = X_test / 255.0
```

```
print('train data shape : {}'.format(X_train.shape))
print('test data shape : {}'.format(X_test.shape))
```

```
train data shape : (60000, 28, 28, 1)
test data shape : (10000, 28, 28, 1)
```

Building the Model

```
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]


# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(28, 28, 1), activation='relu', padding='sa
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

print(model.summary())
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/pyth
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/t
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - ke
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
dropout_1 (Dropout)          (None, 28, 28, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
dropout_2 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 128)         73856
_________________________________________________________________
dropout_3 (Dropout)          (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 7, 7, 128)         147584
```

```
epochs = 50
batch_size=32
model.compile(loss='categorical_crossentropy',
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])

history = model.fit(X_train, y_train,
                    validation_data=(X_test, y_test),
                    epochs=epochs,
                    batch_size=batch_size)

scores = model.evaluate(X_test, y_test, verbose=0)

print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/pyth
Instructions for updating:
Use tf.cast instead.
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 23s 383us/step - loss: 0.5070 -
Epoch 2/50
60000/60000 [==============================] - 17s 281us/step - loss: 0.3257 -
Epoch 3/50
60000/60000 [==============================] - 17s 281us/step - loss: 0.2868 -
Epoch 4/50
60000/60000 [==============================] - 17s 289us/step - loss: 0.2626 -
Epoch 5/50
60000/60000 [==============================] - 17s 280us/step - loss: 0.2528 -
Epoch 6/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.2385 -
Epoch 7/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.2316 -
Epoch 8/50
60000/60000 [==============================] - 17s 285us/step - loss: 0.2255 -
Epoch 9/50
60000/60000 [==============================] - 17s 284us/step - loss: 0.2206 -
Epoch 10/50
60000/60000 [==============================] - 18s 294us/step - loss: 0.2170 -
Epoch 11/50
60000/60000 [==============================] - 17s 280us/step - loss: 0.2094 -
Epoch 12/50
60000/60000 [==============================] - 17s 281us/step - loss: 0.2137 -
Epoch 13/50
60000/60000 [==============================] - 17s 289us/step - loss: 0.2118 -
Epoch 14/50
60000/60000 [==============================] - 17s 285us/step - loss: 0.2061 -
Epoch 15/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.2051 -
Epoch 16/50
60000/60000 [==============================] - 17s 280us/step - loss: 0.2094 -
Epoch 17/50
60000/60000 [==============================] - 17s 280us/step - loss: 0.2049 -
Epoch 18/50
60000/60000 [==============================] - 17s 289us/step - loss: 0.2013 -
Epoch 19/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.2057 -
Epoch 20/50
60000/60000 [==============================] - 17s 280us/step - loss: 0.1994 -
Epoch 21/50
60000/60000 [==============================] - 17s 280us/step - loss: 0.2031 -
Epoch 22/50
60000/60000 [==============================] - 17s 281us/step - loss: 0.1980 -
Epoch 23/50
60000/60000 [==============================] - 17s 287us/step - loss: 0.1995 -
Epoch 24/50
60000/60000 [==============================] - 17s 280us/step - loss: 0.1995 -
Epoch 25/50
60000/60000 [==============================] - 17s 280us/step - loss: 0.2004 -
Epoch 26/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.2005 -
Epoch 27/50
```

```
60000/60000 [==============================] - 17s 285us/step - loss: 0.1967 -
Epoch 28/50
60000/60000 [==============================] - 18s 297us/step - loss: 0.1996 -
Epoch 29/50
60000/60000 [==============================] - 17s 280us/step - loss: 0.1957 -
Epoch 30/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.1949 -
Epoch 31/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.1946 -
Epoch 32/50
60000/60000 [==============================] - 18s 294us/step - loss: 0.1960 -
Epoch 33/50
60000/60000 [==============================] - 17s 280us/step - loss: 0.1951 -
Epoch 34/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.1968 -
Epoch 35/50
60000/60000 [==============================] - 17s 278us/step - loss: 0.1895 -
Epoch 36/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.1940 -
Epoch 37/50
60000/60000 [==============================] - 17s 289us/step - loss: 0.1979 -
Epoch 38/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.2107 -
Epoch 39/50
60000/60000 [==============================] - 17s 278us/step - loss: 0.2026 -
Epoch 40/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.1919 -
Epoch 41/50
60000/60000 [==============================] - 17s 280us/step - loss: 0.1901 -
Epoch 42/50
60000/60000 [==============================] - 17s 288us/step - loss: 0.1949 -
Epoch 43/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.1905 -
Epoch 44/50
60000/60000 [==============================] - 17s 279us/step - loss: 0.1876 -
Epoch 45/50
60000/60000 [==============================] - 17s 277us/step - loss: 0.1878 -
Epoch 46/50
60000/60000 [==============================] - 17s 283us/step - loss: 0.1927 -
Epoch 47/50
60000/60000 [==============================] - 17s 281us/step - loss: 0.1914 -
```

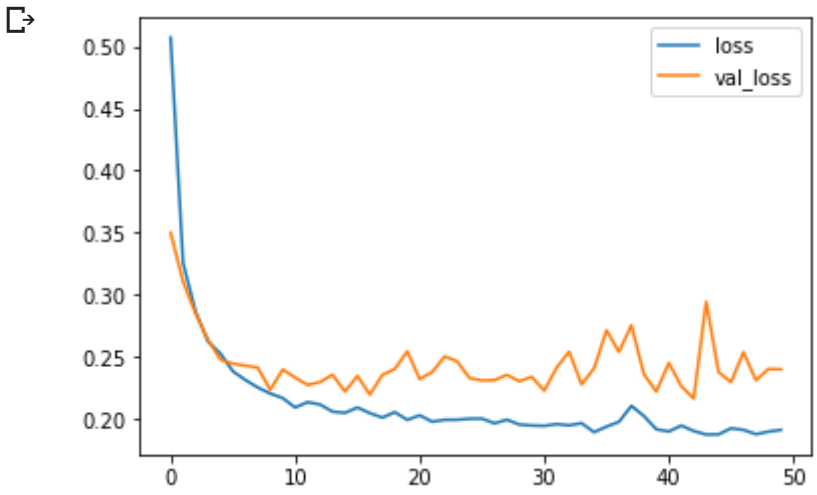
```
predictions = model.predict(X_test)
print(predictions)
```

```
[[1.5824467e-38 0.0000000e+00 2.2892394e-38 ... 4.2318698e-07
```

Plot the training and validation loss

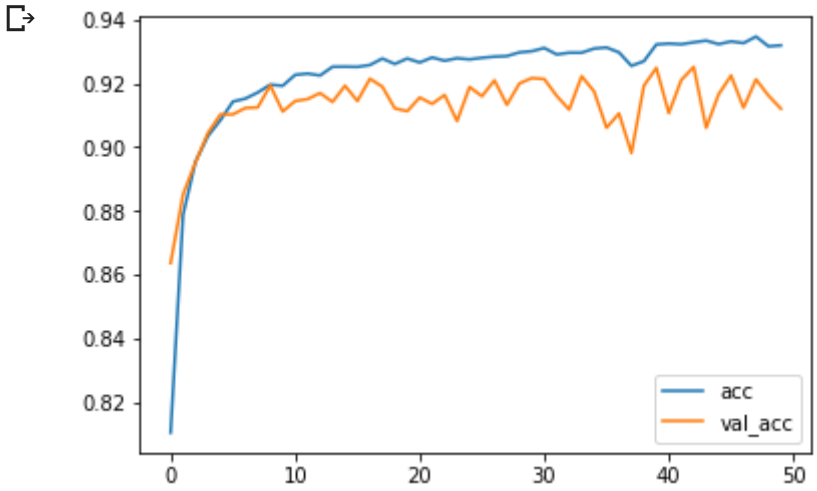
```
 4.0742539e-22 0.0000000e+00]
```

```
pyplot.plot(history.history['loss'], label='loss')
pyplot.plot(history.history['val_loss'], label='val_loss')
pyplot.legend()
pyplot.show()
```



Plot the training and validation accuracy

```
pyplot.plot(history.history['acc'], label='acc')
pyplot.plot(history.history['val_acc'], label='val_acc')
pyplot.legend()
pyplot.show()
```



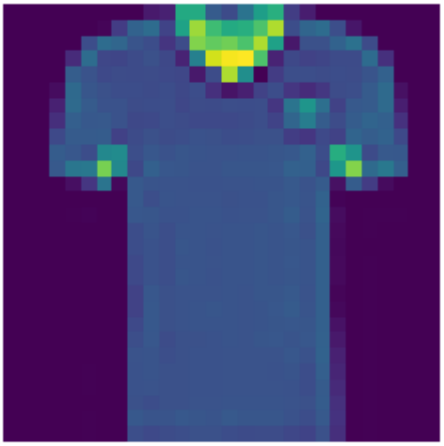

```
mapping = {
    "0" : "T-shirt/top",
    "1" : "Trouser",
    "2" : "Pullover",
    "3" : "Dress",
    "4" : "Coat",
    "5" : "Sandal",
    "6" : "Shirt",
```

```
    "7" : "Sneaker",
    "8" : "Bag",
    "9" : "Ankle boot"
}
```

Predicting / Visualizing the Results

```
pyplot.imshow(X_test[59].reshape(28,28))
pyplot.axis('off')
pyplot.figure(figsize=(28,28))
class_val = np.argmax(model.predict(X_test[59].reshape(-1,28,28,1)))
print("Predicted as {}".format(mapping[str(class_val)]))
```

```
Predicted as T-shirt/top
```



```
<Figure size 2016x2016 with 0 Axes>
```

Predicting from the test set

```
pyplot.figure(figsize=(10,10))
for i in range(25):
    pyplot.subplot(5,5,i+1)
    pyplot.xticks([])
    pyplot.yticks([])
    pyplot.grid(False)
    pyplot.imshow(X_test[i].reshape(28,28), cmap=pyplot.cm.binary)
    class_val = np.argmax(model.predict(X_test[i].reshape(-1,28,28,1)))
    pyplot.xlabel(mapping[str(class_val)])
pyplot.show()
```

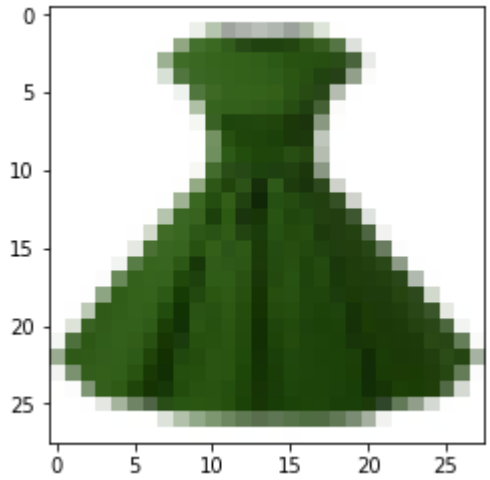
Predicting an image outside the dataset

```
!wget 'https://images-na.ssl-images-amazon.com/images/I/412V6B5Fo4L._SY445_QL70_.jpg

img = cv2.imread("412V6B5Fo4L._SY445_QL70_.jpg")
img_cvt=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img_cvt)
plt.show()


dim = (28, 28)
resized_img = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)

img_tensor = image.img_to_array(resized_img)
img_tensor = np.expand_dims(img_tensor, axis=0)
# Remember that the model was trained on inputs
# that were preprocessed in the following way:
img_tensor /= 255.

print(img_tensor.shape)
img_tensor

plt.imshow(img_tensor[0])
plt.show()

class_val = np.argmax(model.predict(img_tensor.reshape(-1,28,28,1)))
print("Predicted as {}".format(mapping[str(class_val)]))


!wget 'https://dimg.dillards.com/is/image/DillardsZoom/nav2/brahmin-melbourne-collec

img = cv2.imread("04774914_zi_sunflower.jpg")
img_cvt=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img_cvt)
plt.show()


dim = (28, 28)
resized_img = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
```

```
img_tensor = image.img_to_array(resized_img)
img_tensor = np.expand_dims(img_tensor, axis=0)
# Remember that the model was trained on inputs
# that were preprocessed in the following way:
img_tensor /= 255.

print(img_tensor.shape)
img_tensor

plt.imshow(img_tensor[0])
plt.show()

class_val = np.argmax(model.predict(img_tensor.reshape(-1,28,28,1)))
print("Predicted as {}".format(mapping[str(class_val)]))
```

```
--2019-05-14 18:39:03--  https://images-na.ssl-images-amazon.com/images/I/412V
Resolving images-na.ssl-images-amazon.com (images-na.ssl-images-amazon.com)...
Connecting to images-na.ssl-images-amazon.com (images-na.ssl-images-amazon.com
HTTP request sent, awaiting response... 200 OK
Length: 12011 (12K) [image/jpeg]
Saving to: '412V6B5Fo4L._SY445_QL70_.jpg.2'

412V6B5Fo4L._SY445_ 100%[===================>]  11.73K  --.-KB/s    in 0s

2019-05-14 18:39:03 (210 MB/s) - '412V6B5Fo4L._SY445_QL70_.jpg.2' saved [12011
```


```
(1, 28, 28, 3)
```




```
Predicted as Dress
--2019-05-14 18:39:04--  https://dimg.dillards.com/is/image/DillardsZoom/nav2/
```

Confusion Matrix

```
Length: 40268 (39K) [image/jpeg]
```

```
#http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
```

```
        if normalize:
            cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
            print("Normalized confusion matrix")
        else:
            print('Confusion matrix, without normalization')

        plt.imshow(cm, interpolation='nearest', cmap=cmap)
        plt.title(title)
        plt.colorbar()
        tick_marks = np.arange(len(classes))
        plt.xticks(tick_marks, classes, rotation=45)
        plt.yticks(tick_marks, classes)

        fmt = '.2f' if normalize else 'd'
        thresh = cm.max() / 2.
        for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
            plt.text(j, i, format(cm[i, j], fmt),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")

        plt.tight_layout()
        plt.ylabel('True label')
        plt.xlabel('Predicted label')


    preds = np.argmax(model.predict(X_test), axis = 1)
    y_orig = np.argmax(y_test, axis = 1)
    cm = confusion_matrix(preds, y_orig)

    plt.figure(figsize=(8,8))
    plot_confusion_matrix(cm, class_names, normalize=True)
```

```
Normalized confusion matrix
```