برنامه ایجاد دیتا ست عکس برای استفاده در شبکه عصبی

ایجاد دیتاست در ۵ کلاس هر کلاس به تعداد اختیاری عکس داشته باشند و لیبل گذاری شده باشند و در برنامه ذیل به جای دیتای fashion_mnist جایگزین شده و در محیط گوگل کولب اجرا شده و برنامه همراه با یک توضیح در مورد روش کار ارائه گردد.همچنین تصاویر سه کاناله یعنی رنگی و با رزولوشن 240X240 پیکسل در نظر گرفته شود.

```
# Load the TensorBoard notebook extension

%load_ext tensorboard


import tensorflow as tf

import datetime

from tensorflow import keras

import numpy as np

#import sklearn as sklearn

#from sklearn import metrics

#from matplotlib import pyplot as plt

import itertools



#from datetime import datetime

import sklearn

import sklearn.metrics

from packaging import version

import matplotlib.pyplot as plt

import io



# Clear any logs from previous runs
```

```
!rm -rf ./logs/


#Importing Dataset
# downloading the dataset
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

# all the classes
class_names = ('T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
       'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle Boot')

#train model
model = keras.models.Sequential([
  keras.layers.Flatten(input_shape=(28, 28)),
  keras.layers.Dense(512, activation='relu'),
  keras.layers.Dense(256, activation='relu'),
  keras.layers.Dense(128, activation='relu'),
  keras.layers.Dense(64, activation='relu'),
  keras.layers.Dense(32, activation='relu'),
  keras.layers.Dense(10, activation='softmax')
])
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])



# Clearing out prior logging data.
!rm -rf logs/image
def plot_confusion_matrix(cm, class_names):
   figure = plt.figure(figsize=(8, 8))
   plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
```

```
    plt.title("Confusion Matrix of the Results")
    plt.colorbar()
    tick_marks = np.arange(len(class_names))
    plt.xticks(tick_marks, class_names, rotation=90)
    plt.yticks(tick_marks, class_names)

    labels = np.around(cm.astype('float') / cm.sum(axis=1)[:, np.newaxis], decimals=2)

    threshold = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        color = "white" if cm[i, j] > threshold else "black"
        plt.text(j, i, labels[i, j], horizontalalignment="center", color=color)

    plt.tight_layout()
    plt.ylabel('Real Class')
    plt.xlabel('Predicted Class')
    return figure

logdir = "logs/image/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
# Defining the basic TensorBoard callback.
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logdir)
file_writer_cm = tf.summary.create_file_writer(logdir + '/cm')
```