

פעילות 17 - סיווג פריטי לבוש בתמונות תוך שימוש keras

קישורים:

<https://keras.io/models/model/>

<https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/2.1-a-first-look-at-a-neural-network.ipynb>

<https://www.tensorflow.org/tutorials/keras/classification>

https://www.manning.com/books/deep-learning-with-python?a_aid=keras&a_bid=76564dff

<https://livebook.manning.com/book/deep-learning-with-python/chapter-3/8>

בפעילות זו נעשה שימוש במסד נתונים בשם Fashion MNIST dataset הכולל אלפי תמונות של פריטי לבוש מתוייגים. להלן דוגמה לחלק מהתמונות במסד הנתונים:



נעזר בספריה keras היושבת כמעטפת מעל TensorFlow כך שתעזור לנו לבנות מכונה לומדת המבוססת על רשת נוירונים מלאכותית ANN המסוגלת לזהות פריטי לבוש.

מקורות:

<https://github.com/tensorflow/docs/blob/master/site/en/tutorials/keras/classification.ipynb>

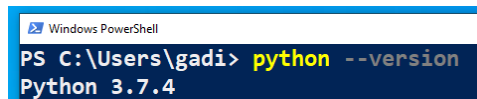
<https://www.tensorflow.org/tutorials/keras/classification>

<https://www.tensorflow.org/install/pip?lang=python3>

התקנת הספרייה

נפתח את חלון הפקודות Windows PowerShell ובבדוק תחילה שהמפרש של Python מותקן במחשב, נעשה זאת על ידי ההוראה:

```
> python --version
```

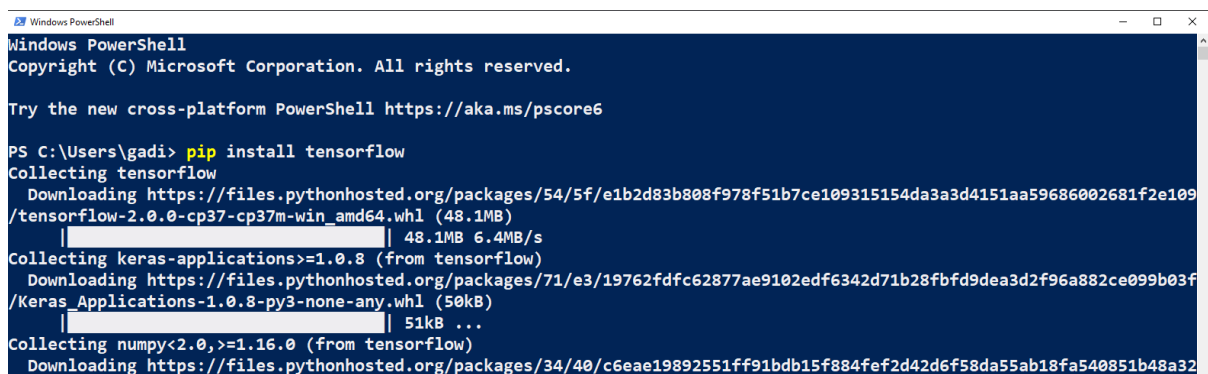


```
Windows PowerShell
PS C:\Users\gadi> python --version
Python 3.7.4
```

התקנת הספרייה תתבצע על ידי ההוראה הבאה:

```
> python -m pip install -U pip
> python -m pip install -U tensorflow
> python -m pip install -U keras
```

נקבל כפלט את המסך הבא:



```
Windows PowerShell
Copyright (c) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\gadi> pip install tensorflow
Collecting tensorflow
  Downloading https://files.pythonhosted.org/packages/54/5f/e1b2d83b808f978f51b7ce109315154da3a3d4151aa59686002681f2e1099/tensorflow-2.0.0-cp37-cp37m-win_amd64.whl (48.1MB)
    | 48.1MB 6.4MB/s
Collecting keras-applications>=1.0.8 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/71/e3/19762fdcf62877ae9102edf6342d71b28fbfd9dea3d2f96a882ce099b03f/Keras_Applications-1.0.8-py3-none-any.whl (50kB)
    | 51kB ...
Collecting numpy<2.0,>=1.16.0 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/34/40/c6eae19892551ff91bdb15f884fef2d42d6f58da55ab18fa540851b48a32/
```

קחו בחשבון שלא ניתן להתקין את הספרייה TensorFlow על כל פלטפורמה של מערכת הפעלה וגם לא על כל גרסה של python להלן רשימה נכונה אפריל 2020

Windows	
Python 3.5 GPU support	https://storage.googleapis.com/tensorflow/windows/gpu/tensorflow_gpu-2.1.0-cp35-cp35m-win_amd64.whl
Python 3.5 CPU-only	https://storage.googleapis.com/tensorflow/windows/cpu/tensorflow_cpu-2.1.0-cp35-cp35m-win_amd64.whl
Python 3.6 GPU support	https://storage.googleapis.com/tensorflow/windows/gpu/tensorflow_gpu-2.1.0-cp36-cp36m-win_amd64.whl
Python 3.6 CPU-only	https://storage.googleapis.com/tensorflow/windows/cpu/tensorflow_cpu-2.1.0-cp36-cp36m-win_amd64.whl
Python 3.7 GPU support	https://storage.googleapis.com/tensorflow/windows/gpu/tensorflow_gpu-2.1.0-cp37-cp37m-win_amd64.whl
Python 3.7 CPU-only	https://storage.googleapis.com/tensorflow/windows/cpu/tensorflow_cpu-2.1.0-cp37-cp37m-win_amd64.whl
Raspberry PI (CPU-only)	
Python 3, Pi0 or Pi1	https://storage.googleapis.com/tensorflow/raspberrypi/tensorflow-2.1.0-cp35-none-linux_armv6l.whl
Python 3, Pi2 or Pi3	https://storage.googleapis.com/tensorflow/raspberrypi/tensorflow-2.1.0-cp35-none-linux_armv7l.whl

<https://www.tensorflow.org/install/pip?lang=python3>

משימה 1: בדיקת ההתקנה

נכתוב את הקוד הבא:

```
import tensorflow as tf
import keras
print("tensorflow version: ", tf.version.GIT_VERSION, tf.version.VERSION)
print("keras version:", keras.__version__)
```

נקבל כפלט את המסך הבא:

```
Using TensorFlow backend.
tensorflow version: v2.1.0-rc2-17-ge5bf8de410 2.1.0
keras version: 2.3.1
```

משימה 2: יבוא מסד הנתונים Fashion MNIST dataset

נעזר בפעולה `load_data` כדי להוריד למחשב את מסד הנתונים Fashion MNIST dataset. קובץ המסד מכיל 70000 תמונות בגודל של 28 פיקסלים על 28 פיקסלים לכל תמונה. כמו כן קובץ תמונות מתיוגות ל-10 קטגוריות שונות והם:

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

ניתן להוריד את קבצי התמונות באופן ידני דרך הקישור הבא:

<https://github.com/zalandoresearch/fashion-mnist>

Name	Content	Examples	Size	Link	MD5 Checksum
train-images-idx3-ubyte.gz	training set images	60,000	26 MBytes	Download	8d4fb7e6c68d591d4c3dfe9ec88bf0d
train-labels-idx1-ubyte.gz	training set labels	60,000	29 KBytes	Download	25c81989df183df01b3e8a0aad5dffbe
t10k-images-idx3-ubyte.gz	test set images	10,000	4.3 MBytes	Download	bef4ecab320f06d8554ea6380940ec79
t10k-labels-idx1-ubyte.gz	test set labels	10,000	5.1 KBytes	Download	bb300cfdad3c16e7a12a480ee83cd310

קבצי הנתונים מחולקים ל- 60000 תמונות מתיוגות שימשו לאימון הרשת ועוד 10000 תמונות מתיוגות לבדיקה לאחר האימון.

הקוד הבא מוריד את קבצי הנתונים ישירות מהאינטרנט למחשב ומציג את התמונה הראשונה במערך תמונות:

```
from tensorflow import keras
import numpy as np

fashion_mnist = keras.datasets.fashion_mnist

(train_data, train_lbl), (test_data, test_lbl) = fashion_mnist.load_data()

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

print("\n", np.where(train_data[0] > 128, 8, 1))
print(lbls[train_lbl[0]])
```

פלט התוכנית יראה כך:



השתמשנו בהוראה הבא כדי לשרטט את התמונה ישירות על גבי מסך הטרמינל:

```
print("\n", np.where(train_data[0] > 128, 8, 1))
```

כפי שלמדנו בפעילות 2 בנושא מערכים תחת NumPy Arrays שניתן לבצע פעולות לוגיות מתמטיות ישירות על מבני נתונים שלמים. ההוראה המתוארת ממירה את הערכים המרכיבים את התמונה הראשונה במערך כאשר כל ערך הגדול מ-128 יקבל את הערך 8 וכל ערך קטן או שווה ל-128 יקבל את הערך 0. זאת במטרה ליצור דמות יחסית ברורה על מסך הטרמינל.

כמובן שניתן להיעזר בספריה matplotlib כדי להציג את התמונה בכלי גרפי. להלן הקוד:

```
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt

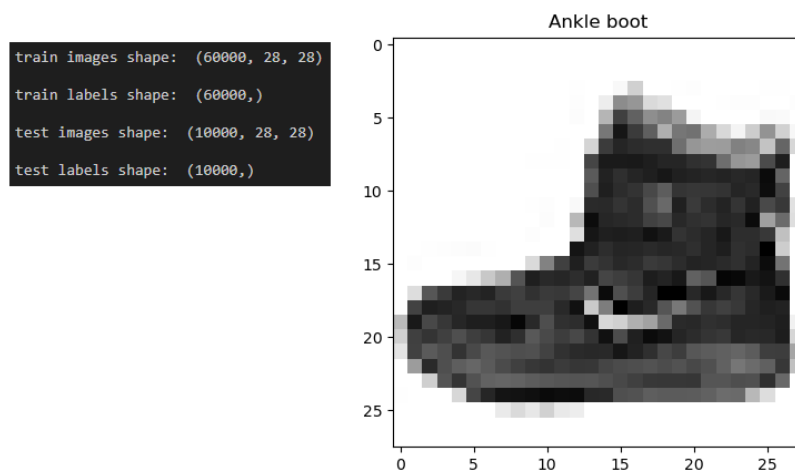
fashion_mnist = keras.datasets.fashion_mnist

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

print("\ntrain images shape: ",train_data.shape)
print("\ntrain labels shape: ",train_lbl.shape)
print("\ntest images shape: ",test_data.shape)
print("\ntest labels shape: ",test_lbl.shape)

plt.figure()
plt.imshow(train_data[0], cmap='Greys')
plt.title(lbls[train_lbl[0]])
plt.show()
```

פלט התוכנית יראה כך:



מפלט התוכנית ניתן ללמוד את הדברים הבאים:

1. גודל כל תמונה הוא 28X28 פיקסלים.
2. קובץ הנתונים לאימון מכיל 60000 תמונות.
3. קובץ הבדיקה מכיל 10000 תמונות.
4. קובץ התגים מכיל מספרים בין 0 ל-9 בהתאם למתואר בתמונה על פי המפתח הבא:
5. אכן התמונה הראשונה המציגה מגף מתויגת כ-9 כלומר Ankle boot.

משימה 3: הצגת מערך התמונות כולל התיג שלם

קובץ המסד מכיל 70000 תמונות בגודל של 28 פיקסלים על 28 פיקסלים לכל תמונה. כמובן שלא ניתן להציג על המסך את כל התמונות אך ניתן להציג לפחות חלק מהם. בקוד הבא נציג את 49 התמונות הראשונות במאגר כולל התיג של כל תמונה:

```
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt

fashion_mnist = keras.datasets.fashion_mnist
(train_data, train_lbl), (test_data, test_lbl) = fashion_mnist.load_data()

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

train_data = train_data / 255.0

plt.figure()
for i in range(70):
    plt.subplot(7,10,i+1)
    plt.axis("off")
    plt.imshow(train_data[i], cmap='Greys')
    plt.title(lbls[train_lbl[i]], fontsize=10)
plt.show()
```

כפלט נקבל את המסך הבא:



משימה 4: התאמת מערך התמונות למבואות המכונה הלומדת

אחת המשימות החשובות בעבודה עם מכונות לומדות הוא להתאים את נתוני האימון והבדיקה לערכים המתאימים להיקלט במבואות של המכונה.

במכונה שלנו אנו צריכים לבצע 2 מטלות:

1. להמיר את הצבעים המרכיבים כל פיקסל בתמונה מטווח בין 0 ל-255 לערכים בין 0 ל-1 כדי שלמדנו בפעילויות הקודמות.

2. המרת מטריצה דו-מימדית של כל תמונות (28X28) למערך חד ממדי הכולל 784 ערכים.

נעשה זאת על ידי חלוקת כל ערכי הפיקסלים ב-255 על ידי ההוראות הבאות:

```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

שאלה:

כיצד 2 השורות הבאות מצליחות להמיר כל אחד מ-784 הפיקסלים שבכל אחת מ-70000 התמונות?

תשובה:

כדי שלמדנו על מערכים מסוג `numpy.ndarray` אנו יכולים לבצע פעולות ישירות על כל מבנה הנתונים. לדוגמה, הקוד הבא מראה כיצד מחלקים ערכים בתוך מטריצה מסוג `numpy.ndarray`:

```
1 import numpy as np
2 a = np.array([[4, 6], [8, 10]])
3 print(a)
4 print(a.dtype.name)
5 print(type(a))
6 a=a/2.0
7 print(a)
8 print(a.dtype.name)
```

[[4 6]
[8 10]]
int32
<class 'numpy.ndarray'>
[[2. 3.]
[4. 5.]]
float64

נבחן את תמונת המגף לאחר המרת הערכים מ 0 ל-255 לערכים מ 0 ל-1 להלן קוד הבדיקה ופלט התוכנית:

```
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt

fashion_mnist = keras.datasets.fashion_mnist
```

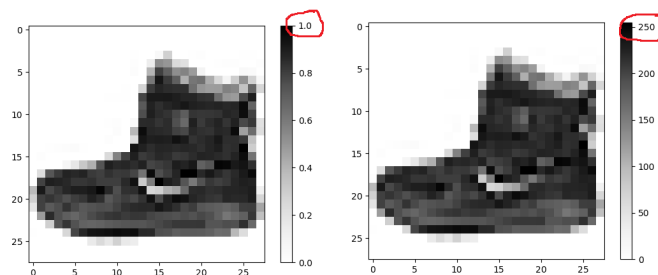
```
(train_data, train_lbl), (test_data, test_lbl) = fashion_mnist.load_data()
```

```
plt.figure()  
plt.imshow(train_data[0], cmap='Greys')  
plt.colorbar()  
plt.grid(False)  
plt.show()
```

```
train_data = train_data / 255.0
```

```
plt.figure()  
plt.imshow(train_data[0], cmap='Greys')  
plt.colorbar()  
plt.grid(False)  
plt.show()
```

נקבל את התמונות הבאות:



המרת מטריצה דו-מימדית של כל תמונות (28X28) למערך חד ממדי הכולל 784 ערכים נעשית תוך כדי בניית רשת נוירונים האופן הבא:

```
model = keras.Sequential([  
    keras.layers.Flatten(input_shape=(28, 28)),  
    keras.layers.Dense(128, activation='relu'),  
    keras.layers.Dense(10, activation='softmax')  
])
```

רשת הנוירונים לניתוח התמונות המורכבות מ- 28X28 פיקסלים ומתוייגות ל- 10 קטגוריות שונות בנוייה באופן הבא:

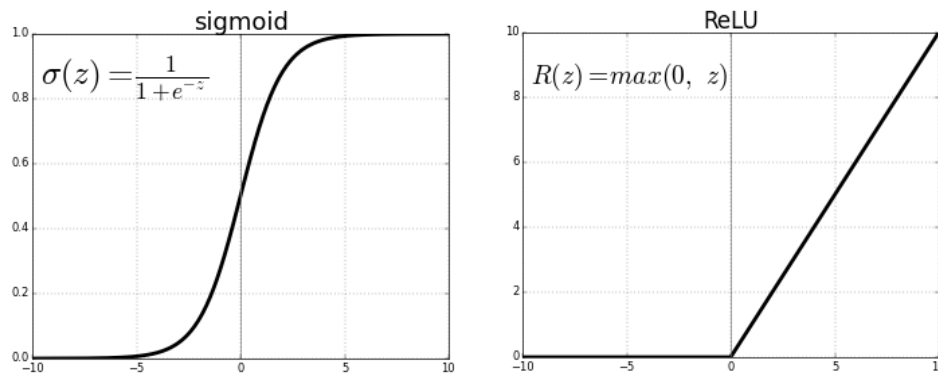
שכבת מבוא הכולל 784 כאשר הפעולה:

```
Flatten(input_shape=(28, 28))
```

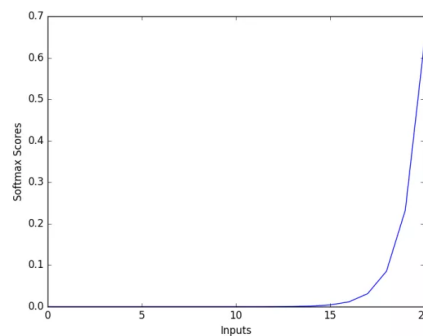

ממירה את הפיקסלים בתמונה למערך חד מימדי.

בנוסף הרשת מורכבת מ- 128 ניוירונים בשכבה שנייה ועוד 10 נוריות בשכבת המוצא.

פונקציית האקטיבציה של 128 הנורונים בשכבה השנייה היא relu המתוארת באיור הבא תוך השוואה לפונקציה sigmoid שהכרנו כבר מהפעילויות הקודמות



פונקציית האקטיבציה של 10 הנורונים בשכבה המוצא היא softmax המתוארת באיור הבא:



פונקציה זו שימושית בעיקר במערך נירוני המוצא. ניתן לממש את פונקציית האקטיבציה softmax על ידי הקוד הבא:

```
def softmax(x):  
    e_x = np.exp(x - np.max(x))  
    return e_x / e_x.sum()
```

$$\text{Softmax}(z_j) = \frac{\sum e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$

לאחר הגדרת רשת הנורונים נעזר בפעולה compile כדי ליצור את הרשת. הפעולה מקבלת מספר פרמטרים הקובעים את האלגוריתמים ליישום הרשת:

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

לאחר בניית המודל נעזר בפעולה fit כדי לאמן את רשת הנוירונים. הפעולה fit מקבלת כקלט שלוש פרמטרים.

test_data - מערך התמונות לאימון

test_lbl - רשימת התגיות מתאימה לתמונות

epochs - מספר הפעמים שמעבירים את מערך התמונות דרך הרשת.

```
model.fit(train_data, train_lbl, epochs=10)
```

להלן ריכוז כל הקוד למדנו על עכשיו:

```
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt

fashion_mnist = keras.datasets.fashion_mnist
(train_data, train_lbl), (test_data, test_lbl) = fashion_mnist.load_data()

train_data = train_data / 255.0
test_data = test_data / 255.0

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

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

model.fit(train_data, train_lbl, epochs=10)
```

לאחר הרמת הקוד נקבל את הפלט הבא:

```

Train on 60000 samples
Epoch 1/10
60000/60000 [=====] - 3s 48us/sample - loss: 0.5008 - accuracy: 0.8232
Epoch 2/10
60000/60000 [=====] - 2s 40us/sample - loss: 0.3773 - accuracy: 0.8626
Epoch 3/10
60000/60000 [=====] - 2s 40us/sample - loss: 0.3383 - accuracy: 0.8763
Epoch 4/10
60000/60000 [=====] - 2s 40us/sample - loss: 0.3145 - accuracy: 0.8838
Epoch 5/10
60000/60000 [=====] - 2s 40us/sample - loss: 0.2957 - accuracy: 0.8902
Epoch 6/10
60000/60000 [=====] - 2s 41us/sample - loss: 0.2826 - accuracy: 0.8950
Epoch 7/10
60000/60000 [=====] - 3s 53us/sample - loss: 0.2703 - accuracy: 0.8994
Epoch 8/10
60000/60000 [=====] - 3s 51us/sample - loss: 0.2576 - accuracy: 0.9047
Epoch 9/10
60000/60000 [=====] - 3s 49us/sample - loss: 0.2487 - accuracy: 0.9072
Epoch 10/10
60000/60000 [=====] - 3s 51us/sample - loss: 0.2397 - accuracy: 0.9090

```

ניתן לראות שהמחשב עבר 10 פעמים על 60000 תמונות ובכל פעל עלתה רמת הדיוק של הרשת.

משימה 5: בדיקת הרשת לאחר שלב האימון

לאחר ביצוע האימון הרשת נבדוק את יכולת הרשת על ידי כך שנציג לה תמונות שלא לקחו חלק בשלב האימון, כלומר תמונות שהרשת לא הראתה לפני כן.

כדי לעשות את השלב הזה השארנו לנו 10000 תמונות מתויוגות ברשימה בשם `test_images`. מעזר בפעולה `evaluate` כדי לבדוק את יעילות הרשת. להלן ההוראה:

```

test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=0)
print("\nTest accuracy:", test_acc)
print("\nTest loss:", test_loss)

```

נקבל סיכום של הבדיקה באופן הבא:

```

Test accuracy: 0.8909
Test loss: 0.3177053982079029

```

משמעות הדברים היא שהמדד לשגיאות הוא 0.317, ככל שערך זה קטן יותר משמעות דברים שהערכים במוצא הרשת קרובים יותר לערכים הנכונים בהשוואה לתוויות.

לאחר שקיבלנו את התוצאות הסטטיסטיות של ביצועי הרשת, נבחן עכשיו את המערכת על ידי כך שנבדוק את יכולות הסיווג שלה עבור תמונה בודדת ממאגר תמונות הבדיקה. נעשה זאת על ידי הקוד הבא:

```

from tensorflow import keras
import numpy as np

```

```

import matplotlib.pyplot as plt

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

fashion_mnist = keras.datasets.fashion_mnist
(train_data, train_lbl), (test_data, test_lbl) = fashion_mnist.load_data()

train_data = train_data / 255.0
test_data = test_data / 255.0

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

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

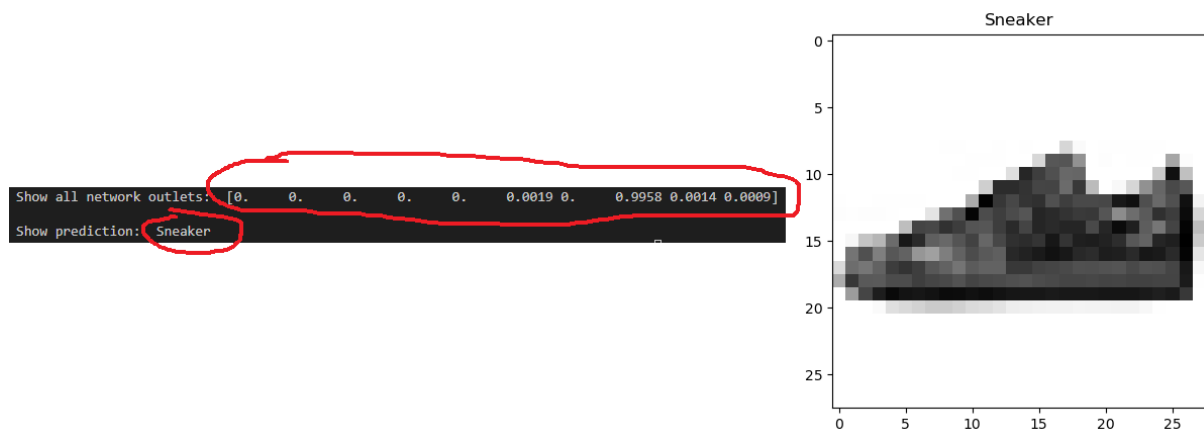
model.fit(train_data, train_lbl, epochs=3)

predictions = model.predict(test_data)
np.set_printoptions(precision=4, suppress=True)
print("\n Show all network outlets: ", predictions[22])
predic = np.argmax(predictions[22])
print("\n Show prediction: ", lbls[predic])

plt.figure()
plt.imshow(test_data[22], cmap='Greys')
plt.title(lbls[predic])
plt.grid(False)
plt.show()

```

נקבל את הקוד הבא:



ניתן לראות תמונה בודדת מתוך התמונות. מדובר בתמונה של נעל ואכן המכונה סיווגה אותה כנעל ספורט.

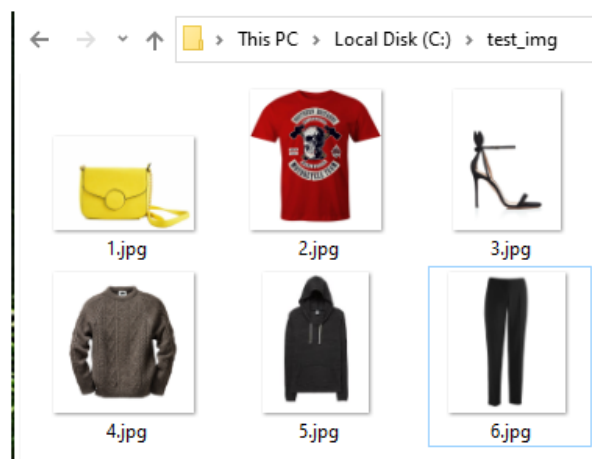
הרשת כוללת 10 מוצאים, אחד לכל קטגוריה, ניתן לראות שקיבלנו בכל אחד מהמוצאים את הפלט הבא:

0	T-shirt/top
0	Trouser
0	Pullover
0	Dress
0	Coat
0.0019	Sandal
0	Shirt
0.9958	Sneaker
0.0014	Bag
0.0009	Ankle boot

ניתן לראות שיש לנו זיהוי מושלם ברמת דיוק גבוהה מאוד ביחס לקטגוריות האחרות.

משימה 6: בדיקת המכונה על ידי תמונות חדשות שלא מתוך מאגר התמונות

נבדוק את יכולות המכונה לסווג תמונות חדשות מהאינטרנט. נוריד תחילה מספר תמונות של פריטי לבוש לתוך תיקייה ייעודית (קחו בחשבון שלא גודל התמונה אך כן חשוב סוג הקובץ).



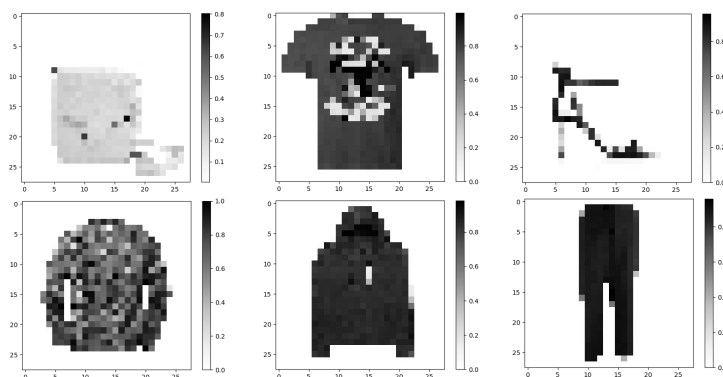
נכתוב תוכנית בדיקה העוברת בלולאה על כל התמונות שבתיקייה ומבצעת את הפעולות הבאות:

1. קוראת את קובץ התמונה המקורי וממירה אותה לגודל 28 על 28 פיקסלים.
2. ממירה את התמונה לגווני אפור (בטווח שבין 0 ל-1)
3. ממירה את התמונה למערך מספרים ושומרת אותה בתוך מערך NumPy
4. מציגה את התמונה לבדיקה

להלן קוד התוכנית:

```
import numpy as np
import matplotlib.pyplot as plt
from keras.preprocessing import image
import glob
file_list = glob.glob("C:/test_img/*.jpg")
x=[]
for each in file_list:
    img = image.load_img(each, color_mode = "grayscale", target_size=(28, 28))
    img = image.img_to_array(img)
    img = img.reshape(28, 28)
    img = img.astype('float32')
    img = (255-img)/255.0
    x.append(img)
    plt.figure()
    plt.imshow(img , cmap='Greys')
    plt.grid(False)
    plt.colorbar()
    plt.show()
x=np.array(x)
```

פלט התוכנית יהיה כך



נשלב את הקוד שכתבנו זה עתה יחד עם המכונה הלומדת שכבר כתבנו בתחילת הפעילות ונקבל את הקוד הבא:

```
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from keras.preprocessing import image
import glob

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

fashion_mnist = keras.datasets.fashion_mnist
(train_data, train_lbl), (test_data, test_lbl) = fashion_mnist.load_data()
train_data = train_data / 255.0
test_data = test_data / 255.0

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_data, train_lbl, epochs=50)

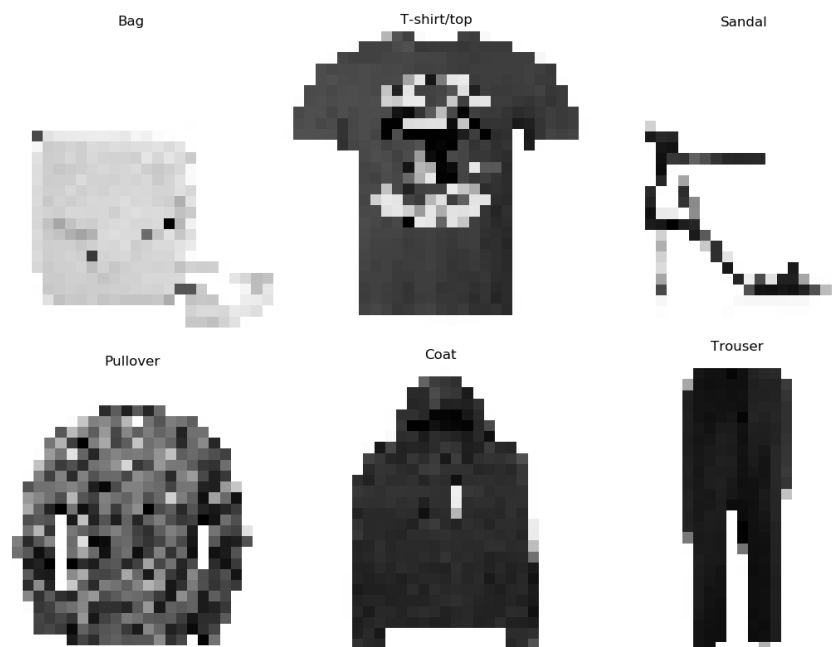
file_list = glob.glob("C:/test_img/*.jpg")
x=[]
for each in file_list:
    img = image.load_img(each, color_mode = "grayscale", target_size=(28, 28))
    img = image.img_to_array(img)
    img = img.reshape(28, 28)
    img = img.astype('float32')
    img = (255-img)/255.0
```

```

x.append(img)
x=np.array(x)
result = model.predict_classes(x)
for i in range(len(result)):
    print(lbls[result[i]], result[i])
print(result)
for i in range(len(result)):
    plt.figure()
    plt.imshow(x[i] , cmap='Greys')
    plt.grid(False)
    plt.axis('off')
    plt.colorbar()
    plt.title(lbls[result[i]])
    plt.show()

```

נקבל את הפלט הבא:



הצלחנו לייצר מכונה לומדת מבוססת על רשת נוירונים מלאכותיים ANN המסוגלת לסווג תמונות של פריטי לבוש.

תנאי השימוש

תנאי השימוש במסמך זה הם לפי הסטנדרט הבא:

You are free:

to Share – to copy, distribute and transmit the material
to Remix – to adapt the material

Under the following conditions:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.