In [1]:
```
import numpy as np
import h5py
import cv2
import os
import pandas as pd

from skimage import io
from skimage.transform import resize

import matplotlib.pyplot as plt
%matplotlib inline

import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.advanced_activations import LeakyReLU
```

```
Using TensorFlow backend.
```

Обработаем данные

In [2]:
```
# перечислим все классы одежды, которые имеются
Y_classes = np.array(os.listdir('img'))
Y_classes
```

Out[2]:
```
array(['Blazer_Пиджак', 'Blouse_Блузка', 'Cardigan_Кофта', 'Coat_Пальто',
       'Dress_Платье', 'Hoodie_Толстовка', 'Jacket_Куртка',
       'Jeans_Джинсы', 'Joggers_Джоггер', 'Jumpsuit_Комбинезон',
       'Romper_Миникомб', 'Shorts_Шорты', 'Skirt_Юбки', 'Sweater_Свитер',
       'Tank_Майка', 'Top_Верх'], dtype='<U19')
```

In [3]:
```
df = pd.DataFrame(Y_classes,columns=['классы'])
df
```

In [5]:
```
X = []
Y = []

index = 0 # начальный номер класса

for c in Y_classes: # бегаем по всем классам = именам корневых папок

    sub_folders = os.listdir('img/'+c) # все папки в классе

    for folder_name in sub_folders[:10]:
        img_names = os.listdir('img/'+c+'/'+folder_name) # имена изобр. в каждой подпапке

        for img_k in img_names: # бегаем по всем именам
            img = io.imread('img/'+c+'/'+folder_name+'/'+ img_k)
            img = cv2.resize(img,(70,70)) # приводим все фотки к одинаковому размеру

            X.append(img)
            Y.append(index)

    index += 1

X = np.array(X)
Y = np.array(Y)
```

In [7]:
```
# let's plot random image
plt.imshow(X[1111])
plt.title(Y_classes[Y[1111]])
```

Out[7]:
```
Text(0.5,1,'Blazer_Пиджак')
```

In [11]:
```
# разобьем данные на обучающ. и тестовую выборки в отношении 70 к 30

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size=0.3,random_state=123)
```

In [12]:
```
print(len(x_train),"+",len(x_test),'=',len(X))
```

```
5985 + 2566 = 8551
```

In [13]:
```
# one-hot кодирование ответов
# было
y_train[100]
```

Out[13]:
```
array([10])
```

In [14]:
```
# стало
y_train_2 = keras.utils.to_categorical(y_train, len(Y_classes))
y_test_2 = keras.utils.to_categorical(y_test, len(Y_classes))

y_train_2[100]
# на втором месте один, все остальное нули
```

```
Out[14]: array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
               dtype=float32)
```

```
In [19]: # очистим
         X = None
         Y = None
         del X
         del Y

         y_train = None
         y_test = None
         del y_train
         del y_test
```

```
In [20]: # отнормируем данные
         x_train = x_train.astype('float32')
         x_test  = x_test.astype('float32')
```

```
In [21]: x_train_2 = x_train/255 - 0.5
```

```
In [22]: x_test_2 = x_test/255 - 0.5
```

```
In [24]: x_train = None
         x_test  = None
         del x_train
         del x_test
```

Сетка Convolutional Neural Network (CNN)

```
In [32]: model = Sequential()

         model.add(Conv2D(32,kernel_size=(3,3),padding='same',input_shape=(70,70,3)))
         model.add(LeakyReLU(0.1))

         model.add(Conv2D(32, (3, 3)))
         model.add(LeakyReLU(0.1))

         model.add(MaxPooling2D(pool_size=(2, 2)))
         model.add(Dropout(0.25))

         model.add(Conv2D(64,(3, 3), padding='same'))
         model.add(LeakyReLU(0.1))
         model.add(Conv2D(64, (3, 3)))
         model.add(LeakyReLU(0.1))

         model.add(MaxPooling2D(pool_size=(2, 2)))
         model.add(Dropout(0.25))

         model.add(Flatten())
         model.add(Dense(512))
         model.add(LeakyReLU(0.1))
         model.add(Dropout(0.5))

         model.add(Dense(len(Y_classes)))
         model.add(Activation('softmax'))
```

```
In [33]: from keras.optimizers import SGD

         sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
         model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
```

```
In [34]: model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_5 (Conv2D)            (None, 70, 70, 32)        896       
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 70, 70, 32)        0         
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 68, 68, 32)        9248      
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 68, 68, 32)        0         
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 34, 34, 32)        0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 34, 34, 32)        0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 34, 34, 64)        18496     
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 34, 34, 64)        0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 32, 32, 64)        36928     
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 32, 32, 64)        0         
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 16, 16, 64)        0         
dropout_5 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 16384)             0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               8389120
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 512)               0
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 16)                8208
_________________________________________________________________
activation_2 (Activation)    (None, 16)                0
```

In [37]:
```
# save weights to file
model.save_weights("weights_VM.h5")
```

In [47]:
```
img_for_test = io.imread('test_image.png')
img_for_test = cv2.resize(img_for_test,(70,70))
plt.imshow(img_for_test)
```

In [48]:
```
img_for_test = img_for_test.astype('float32')
img_for_test = img_for_test/255 -0.5
img_for_test = np.expand_dims(img_for_test,axis=0)
```

In [67]:
```
model.predict(img_for_test)
```

Out[67]:
```
array([[5.5585649e-12, 4.6862926e-12, 1.0996182e-05, 6.7799049e-13,
        9.9980801e-01, 1.2770577e-08, 1.7296991e-04, 1.6277980e-13,
        2.6961220e-07, 1.0164332e-06, 2.6753602e-07, 3.0555275e-10,
        6.5364811e-06, 7.4434077e-11, 1.0221152e-11, 4.2707327e-10]],
      dtype=float32)
```

In [65]:
```
np.argmax(model.predict(img_for_test)) # Значение, которое предсказала сетка
```

Out[65]:
```
4
```

In [66]:
```
plt.imshow(io.imread('test_image.png'))
plt.title(Y_classes[np.argmax(model.predict(img_for_test))])
```

Out[66]:
```
Text(0.5,1,'Dress_Платье')
```

In [1]:
```
import numpy as np
import h5py
import cv2
import os
import pandas as pd

from skimage import io
from skimage.transform import resize

import matplotlib.pyplot as plt
%matplotlib inline

import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.advanced_activations import LeakyReLU
```

```
Using TensorFlow backend.
```

In [3]:
```
# перечислим все классы одежды, которые имеются
Y_classes = np.array(os.listdir('img'))
Y_classes
```

Out[3]:
```
array(['Blazer_Пиджак', 'Blouse_Блузка', 'Cardigan_Кофта', 'Coat_Пальто',
       'Dress_Платье', 'Hoodie_Толстовка', 'Jacket_Куртка',
       'Jeans_Джинсы', 'Joggers_Джоггер', 'Jumpsuit_Комбинезон',
       'Romper_Миникомб', 'Shorts_Шорты', 'Skirt_Юбки', 'Sweater_Свитер',
       'Tank_Майка', 'Top_Верх'], dtype='<U19')
```

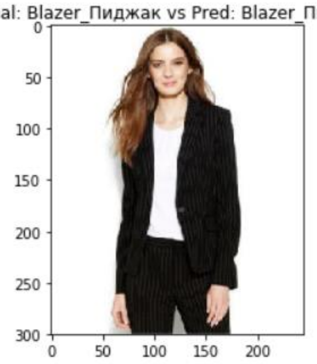
In [4]:
```
# Сетка
model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='same',input_shape=(70,70,3)))
model.add(LeakyReLU(0.1))

model.add(Conv2D(32, (3, 3)))
model.add(LeakyReLU(0.1))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(64,(3, 3), padding='same'))
model.add(LeakyReLU(0.1))
model.add(Conv2D(64, (3, 3)))
model.add(LeakyReLU(0.1))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(LeakyReLU(0.1))
model.add(Dropout(0.5))

model.add(Dense(len(Y_classes)))
model.add(Activation('softmax'))
```

In [5]:

```
# load weights from file (can call without model.fit)
model.load_weights("weights_VM.h5")
```

```
for i in range(20):
    plt.figure()
    plt.title('Real: '+Y_classes[int(img_names_1[i][:1])]+' vs '+'Pred: '+Y_classes[pred_class_1[i]])
    plt.imshow(io.imread('test_images_1/'+img_names_1[i]))
```

Real: Blazer_Пиджак vs Pred: Blazer_Пиджак



Real: Blazer_Пиджак vs Pred: Blazer_Пиджак