classify_garbage

March 27, 2021

[1]:
```
import os
import matplotlib.pyplot as plt
import tensorflow as tf
```

[2]:
```
gpus = tf.config.list_physical_devices(device_type='GPU')
if len(gpus)>0 :
    tf.config.set_visible_devices(devices=gpus[0], device_type='GPU')
```

[3]:
```
num_epochs = 2
batch_size = 32
learning_rate = 0.01
```

[ ]:
```
# data_dir = './data'
# train_bottle_dir = data_dir + '/train/ /'
# train_chopsticks_dir = data_dir + '/train/  /'
```

[4]:
```
data_dir = './data/garbage_classification/'
battery_dir = data_dir + 'battery'
biological_dir = data_dir + 'biological'
brown_glass_dir = data_dir + 'brown-glass'
cardboard_dir = data_dir + 'cardboard'
clothes_dir = data_dir + 'clothes'
green_glass_dir = data_dir + 'green-glass'
metal_dir = data_dir + 'metal'
paper_dir = data_dir + 'paper'
plastic_dir = data_dir + 'plastic'
shoes_dir = data_dir + 'shoes'
trash_dir = data_dir + 'trash'
white_glass_dir = data_dir + 'white-glass'

dirs = [battery_dir,biological_dir,brown_glass_dir,
        cardboard_dir,clothes_dir,green_glass_dir,
        metal_dir,paper_dir,plastic_dir,
        shoes_dir,trash_dir,white_glass_dir]

classes = {'battery':0, 'biological':1, 'brown-glass':2,
           'cardboard':3, 'clothes':4, 'green-glass':5,
           'metal':6, 'paper':7, 'plastic':8,
```

```
            'shoes':9, 'trash':10, 'white-glass':11}
```

[5]:
```
def load_data():
    images_name = []
    labels = []
    for file_dir in dirs:
        filename = tf.constant([file_dir + '/' + filename for filename in os.
 ↪listdir(file_dir)])
        images_name = tf.concat([images_name,filename],axis=-1)
        labels_index = classes[file_dir.split('/')[-1]]
        labels = tf.concat([labels,tf.constant(labels_index,shape=filename.
 ↪shape[0])],axis=-1)
    print("total:%d" %  images_name.shape[0])
    return images_name,labels
```

[6]:
```
def _decode_and_resize(filename, label):
    image_string = tf.io.read_file(filename)  #
    image_decoded = tf.image.decode_jpeg(image_string)  #  JPEG
    image_resized = tf.image.resize(image_decoded, [224, 224]) / 255.0
    return image_resized, label
```

[7]:
```
train_filenames, train_labels = load_data()
train_dataset = tf.data.Dataset.from_tensor_slices((train_filenames,␣
 ↪train_labels))
train_dataset = train_dataset.map(
    map_func=_decode_and_resize,
    num_parallel_calls=tf.data.experimental.AUTOTUNE)
```

```
total:15515
```

[8]:
```
#
train_dataset = train_dataset.shuffle(buffer_size=10000)
train_dataset = train_dataset.batch(batch_size)
train_dataset = train_dataset.prefetch(tf.data.experimental.AUTOTUNE)
```

[9]:
```
#
model = tf.keras.applications.MobileNet(weights=None,classes=len(dirs))
model.compile(
        optimizer=tf.keras.optimizers.Adam(),
        loss=tf.keras.losses.sparse_categorical_crossentropy,
        metrics=[tf.keras.metrics.sparse_categorical_accuracy]
    )

model.fit(train_dataset,epochs=num_epochs)
```

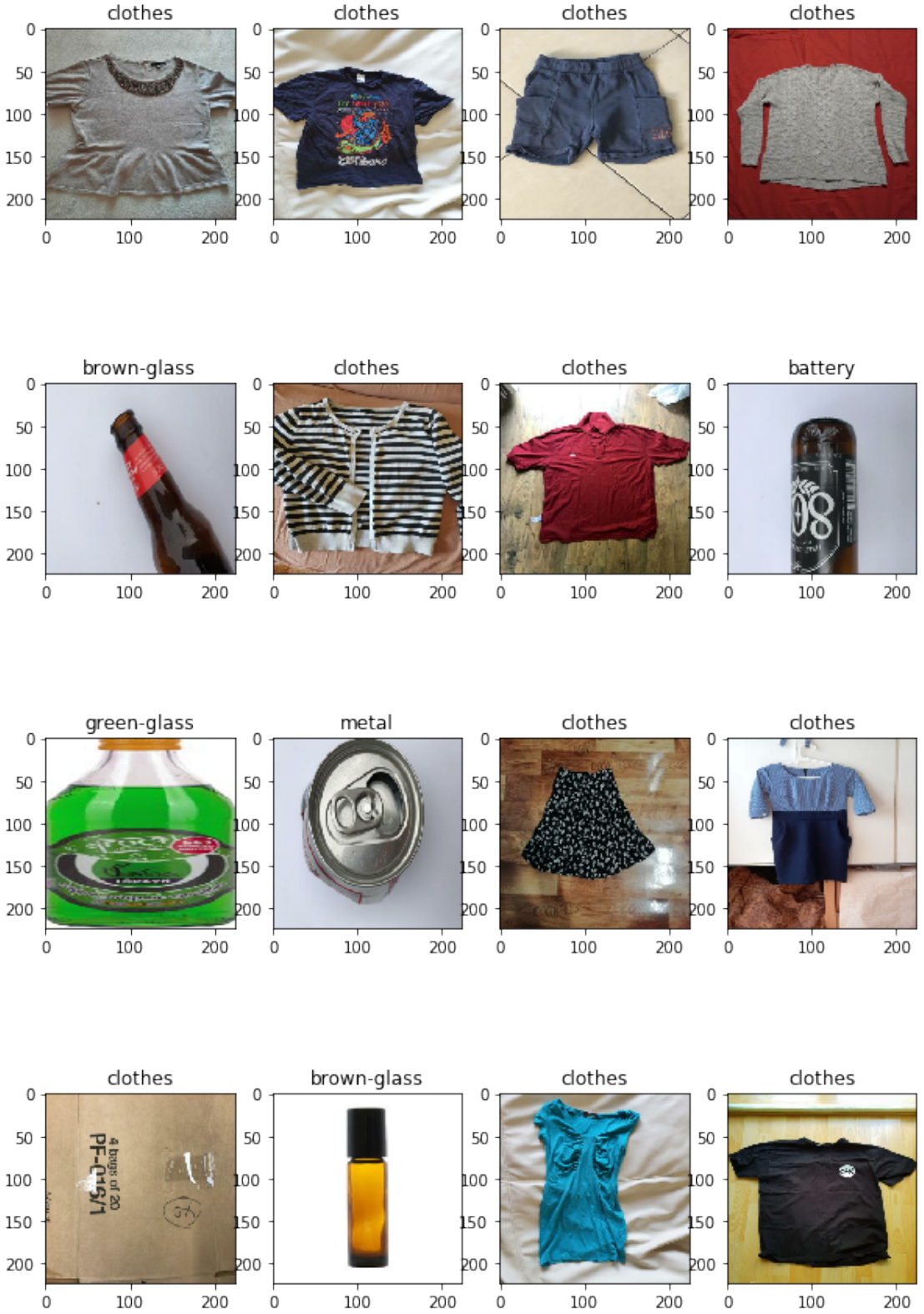
```
Epoch 1/2
485/485 [==============================] - 265s 454ms/step - loss: 1.3845 -
sparse_categorical_accuracy: 0.5499
```

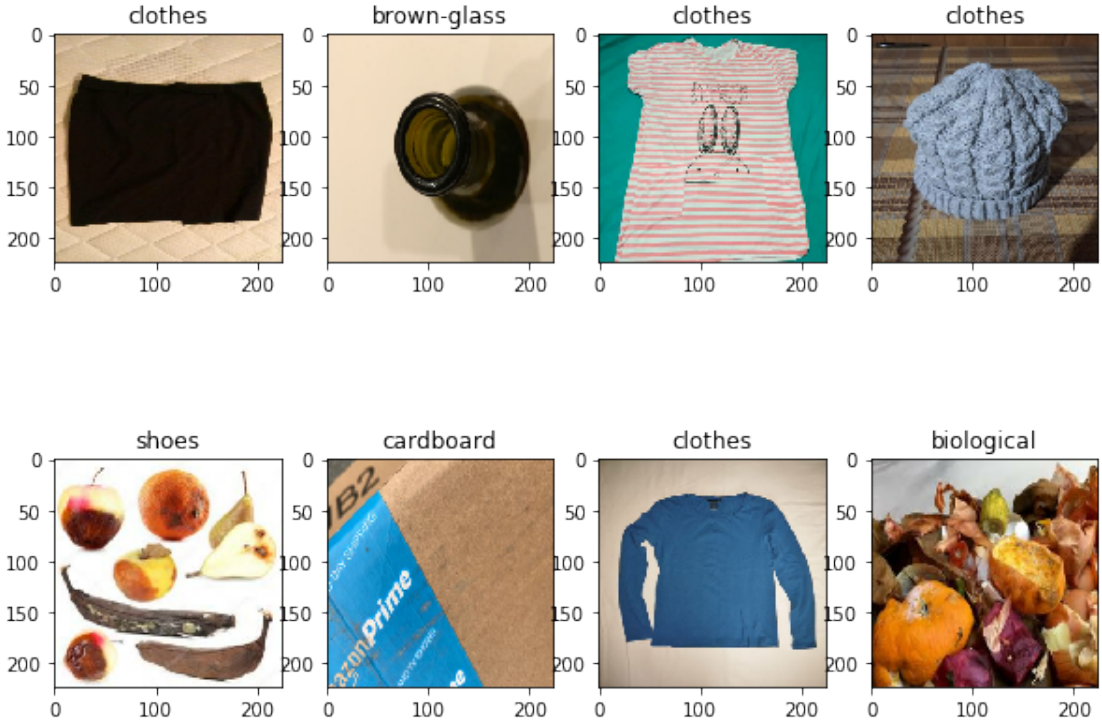
```
Epoch 2/2
485/485 [==============================] - 207s 415ms/step - loss: 0.8562 -
sparse_categorical_accuracy: 0.7292
```

[9]:
```
<tensorflow.python.keras.callbacks.History at 0x174a9b7fb08>
```

[39]:
```
#
for images, labels in train_dataset:
    width,height = 4,8
    for i in range(height):
        plt.figure(figsize=(10, 10))
        for j in range(width):
            plt.subplot(1,width,j+1)
            index = tf.math.argmax(model(tf.
 ↪expand_dims(images[i*width+j],axis=0)),axis=1).numpy()
            for k,v in classes.items():
                if index==v:
                    kind = k
                    break
            plt.title(kind)
            plt.imshow(images[i*width+j].numpy())
        plt.show()
    break
```

```
(32, 224, 224, 3)
```


clothes
clothes
clothes
clothes
brown-glass
clothes
clothes
battery
green-glass
metal
clothes
clothes
clothes
brown-glass
clothes
clothes


clothes
brown-glass
clothes
clothes
shoes
cardboard
clothes
biological

[ ]:

[ ]: