Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



**Домашнее задание №1, 2 по дисциплине**

«Проектирование интеллектуальных систем»

**ИСПОЛНИТЕЛЬ:**

Березин И. С.

Группа ИУ5-23М

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"\_\_"\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Москва 2019

# Цель работы

Научиться использовать предобученные модели распознавания и преобразовывать датасеты в TFRecord.

# Часть 1

### Разворачивание необходимых компонентов

|  |  |
| --- | --- |
|  | from tqdm import tqdm  from numpy.random import randn  import pathlib  import random  import matplotlib.pyplot as plt  import tensorflow as tf  import numpy as np  AUTOTUNE = tf.data.experimental.AUTOTUNE |

## Загрузка датасета

|  |  |
| --- | --- |
|  | import pathlib  data\_root\_orig = tf.keras.utils.get\_file('flower\_photos',  'https://storage.googleapis.com/download.tensorflow.org/example\_images/flower\_photos.tgz',  untar=True)  data\_root = pathlib.Path(data\_root\_orig)  print(data\_root) |

C:\Users\uivan\_000\.keras\datasets\flower\_photos

for item in data\_root.iterdir():

print(item)

C:\Users\uivan\_000\.keras\datasets\flower\_photos\daisy

C:\Users\uivan\_000\.keras\datasets\flower\_photos\dandelion

C:\Users\uivan\_000\.keras\datasets\flower\_photos\LICENSE.txt

C:\Users\uivan\_000\.keras\datasets\flower\_photos\roses

C:\Users\uivan\_000\.keras\datasets\flower\_photos\sunflowers

C:\Users\uivan\_000\.keras\datasets\flower\_photos\tulips

import random

all\_image\_paths = list(data\_root.glob('\*/\*'))

all\_image\_paths = [str(path) for path in all\_image\_paths]

random.shuffle(all\_image\_paths)

image\_count = len(all\_image\_paths)

image\_count

3670

### Изучение данных

import os

attributions = (data\_root/"LICENSE.txt").open(encoding='utf-8').readlines()[4:]

attributions = [line.split(' CC-BY') for line in attributions]

attributions = dict(attributions)

import IPython.display as display

for n in range(2):

image\_path = random.choice(all\_image\_paths)

display.display(display.Image(image\_path))

### Определение меток для каждого изображения

label\_names = sorted(item.name for item in data\_root.glob('\*/') if item.is\_dir())

label\_names

['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']

label\_to\_index = dict((name, index) for index,name in enumerate(label\_names))

label\_to\_index

{'daisy': 0, 'dandelion': 1, 'roses': 2, 'sunflowers': 3, 'tulips': 4}

all\_image\_labels = [label\_to\_index[pathlib.Path(path).parent.name]

for path in all\_image\_paths]

print("First 10 labels indices: ", all\_image\_labels[:10])

Первые 10 индексов изображений: [1, 3, 1, 0, 4, 4, 1, 0, 4, 3]

### Загрузка и форматирование изображений

|  |
| --- |
| ## изменение размера изображения  def preprocess\_image(image):  image = tf.image.decode\_jpeg(image, channels=3)  image = tf.image.resize(image, [192, 192])  image /= 255.0 # normalize to [0,1] range  return image  ## Загрузка изображения в raw формате и его нормализация  def load\_and\_preprocess\_image(path):  image = tf.read\_file(path)  return preprocess\_image(image) return preprocess\_image(image) |
| ## пример работы нормализации  import matplotlib.pyplot as plt  image\_path = all\_image\_paths[0]  label = all\_image\_labels[0]  plt.imshow(load\_and\_preprocess\_image(img\_path))  plt.grid(False)  plt.title(label\_names[label].title())  C:\Users\uivan_000\Desktop\загруженное (1).png |

### Сборка tf.data.Dataset

path\_ds = tf.data.Dataset.from\_tensor\_slices(all\_image\_paths)

image\_ds = path\_ds.map(load\_and\_preprocess\_image, num\_parallel\_calls=AUTOTUNE)

import matplotlib.pyplot as plt

plt.figure(figsize=(8,8))

for n,image in enumerate(image\_ds.take(4)):

plt.subplot(2,2,n+1)

plt.imshow(image)

plt.grid(False)

plt.xticks([])

plt.yticks([])

plt.show()

# Набор данных пар (изображение, метка)

label\_ds = tf.data.Dataset.from\_tensor\_slices(tf.cast(all\_image\_labels, tf.int64))print(label\_ds)

<TensorSliceDataset shapes: (), types: tf.int64>

### Создание tfrec

## image\_ds = tf.data.Dataset.from\_tensor\_slices(all\_image\_paths).map(tf.read\_file)

## tfrec = tf.data.experimental.TFRecordWriter('images.tfrec')

## Часть 2. Обучение существующей модели нейронной сети

image\_ds = tf.data.TFRecordDataset('images.tfrec').map(preprocess\_image)

print(image\_ds)

import matplotlib.pyplot as plt

plt.figure(figsize=(8,8))

for n,image in enumerate(image\_ds.take(4)):

plt.subplot(2,2,n+1)

plt.imshow(image)

plt.grid(False)

plt.xticks([])

plt.yticks([])

plt.show()

<DatasetV1Adapter shapes: (), types: tf.int64>

image\_label\_ds = tf.data.Dataset.zip((image\_ds, label\_ds))

print(image\_label\_ds)

<DatasetV1Adapter shapes: ((192, 192, 3), ()), types: (tf.float32, tf.int64)>

BATCH\_SIZE = 32

# Setting a shuffle buffer size as large as the dataset ensures that the data is

# completely shuffled.

ds = image\_label\_ds.shuffle(buffer\_size=image\_count)

ds = ds.repeat()

ds = ds.batch(BATCH\_SIZE)

# `prefetch` lets the dataset fetch batches, in the background while the model is training.

ds = ds.prefetch(buffer\_size=AUTOTUNE)

ds

<DatasetV1Adapter shapes: ((?, 192, 192, 3), (?,)), types: (tf.float32, tf.int64)>

### Передача набора данных в модель

pretrained\_net = tf.keras.applications.mobilenet.MobileNet(input\_shape=(192, 192, 3), include\_top=False)

pretrained\_net.trainable=False

def change\_range(image,label):

return 2\*image-1, label

keras\_ds = ds.map(change\_range)

model = tf.keras.Sequential([

pretrained\_net,

tf.keras.layers.GlobalAveragePooling2D(),

tf.keras.layers.Dense(len(label\_names))])

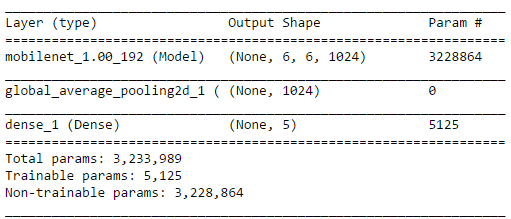
### Компиляция модели

model.compile(optimizer=tf.train.AdamOptimizer(),

loss=tf.keras.losses.sparse\_categorical\_crossentropy,

metrics=["accuracy"])

model.summary()



### Обучение модели

count=1

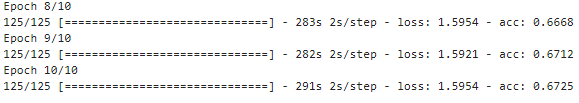
n=1

while count <= n:

print ("Цикл ",count," / ",n)

history = model.fit(ds, epochs=10, steps\_per\_epoch=125)

count=count+1



print(history.history.keys())

dict\_keys(['loss', 'acc'])

plt.figure(1)

# summarize history for accuracy

plt.subplot(211)

plt.plot(history.history['acc'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['acc'])

plt.show()

plt.subplot(212)

plt.plot(history.history['loss'])

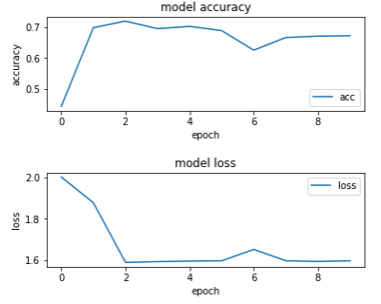
plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['loss'], loc='upper right')

plt.show()



# Список литературы

1. MobileNet: меньше, быстрее, точнее – <https://habr.com/ru/post/352804/>
2. Transfer Learning Using Pretrained ConvNets – <https://www.tensorflow.org/alpha/tutorials/images/transfer_learning>
3. Natural Images – <https://www.kaggle.com/prasunroy/natural-images>
4. Load images with tf.data – <https://www.tensorflow.org/alpha/tutorials/load_data/images>