ass3

November 1, 2019

[3]:
```
!pip install tensorflow
```

```
Collecting tensorflow
  Downloading https://files.pythonhosted.org/packages/2a/5c/f1d66de5dde6f3
ff528f6ea1fd0757a0e594d17debb3ec7f82daa967ea9a/tensorflow-2.0.0-cp37-cp37m-manyl
inux2010_x86_64.whl (86.3MB)
     |                 | 86.3MB 9.0MB/s eta 0:00:014   |
| 6.1MB 1.7MB/s eta 0:00:48     |                               | 7.9MB 3.6MB/s
eta 0:00:22     |                               | 9.5MB 3.6MB/s eta 0:00:22
|                             | 15.6MB 8.6MB/s eta 0:00:09     |
| 16.2MB 8.6MB/s eta 0:00:09     |                             | 17.9MB
8.6MB/s eta 0:00:08     |                           | 20.2MB 10.5MB/s eta
0:00:07               | 50.8MB 1.2MB/s eta 0:00:29     |
| 51.4MB 1.2MB/s eta 0:00:29     |                       | 52.1MB
1.2MB/s eta 0:00:28     |                       | 54.4MB 7.2MB/s eta
0:00:05     |                       | 58.3MB 7.2MB/s eta 0:00:04
|                     | 60.4MB 6.0MB/s eta 0:00:05
|                   | 74.2MB 7.6MB/s eta 0:00:02
Collecting tensorflow-estimator<2.1.0,>=2.0.0 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/fc/08/8b927337b7019c
374719145d1dceba21a8bb909b93b1ad6f8fb7d22c1ca1/tensorflow_estimator-2.0.1-py2.py
3-none-any.whl (449kB)
     |                 | 450kB 3.1MB/s eta 0:00:01
Collecting tensorboard<2.1.0,>=2.0.0 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/d3/9e/a48cd34dd7b672
ffc227b566f7d16d63c62c58b542d54efa45848c395dd4/tensorboard-2.0.1-py3-none-
any.whl (3.8MB)
     |                 | 3.8MB 5.6MB/s eta 0:00:01
|                           | 952kB 5.6MB/s eta 0:00:01     |
| 1.4MB 5.6MB/s eta 0:00:01
Collecting grpcio>=1.8.6 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/b9/ba/254011b066e667
5411ba913dafd6e40ce8d3235bebf64fb226a7305f29ac/grpcio-1.24.3-cp37-cp37m-manylinu
x2010_x86_64.whl (2.2MB)
     |                 | 2.2MB 6.7MB/s eta 0:00:01
Requirement already satisfied: wheel>=0.26 in
/opt/conda/lib/python3.7/site-packages (from tensorflow) (0.33.4)
Collecting termcolor>=1.1.0 (from tensorflow)
```

```
Installing collected packages: tensorflow-estimator, pyasn1, pyasn1-modules,
cachetools, rsa, google-auth, absl-py, grpcio, werkzeug, markdown, requests-
oauthlib, google-auth-oauthlib, tensorboard, termcolor, opt-einsum, gast, wrapt,
google-pasta, astor, keras-preprocessing, keras-applications, tensorflow
Successfully installed absl-py-0.8.1 astor-0.8.0 cachetools-3.1.1 gast-0.2.2
google-auth-1.6.3 google-auth-oauthlib-0.4.1 google-pasta-0.1.7 grpcio-1.24.3
keras-applications-1.0.8 keras-preprocessing-1.1.0 markdown-3.1.1 opt-
einsum-3.1.0 pyasn1-0.4.7 pyasn1-modules-0.2.7 requests-oauthlib-1.2.0 rsa-4.0
tensorboard-2.0.1 tensorflow-2.0.0 tensorflow-estimator-2.0.1 termcolor-1.1.0
werkzeug-0.16.0 wrapt-1.11.2
```

[5]:
```
from __future__ import absolute_import, division, print_function,␣
 ↪unicode_literals
import os, os.path, PIL
import numpy as np
from PIL import Image
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
DIR = './data/Cyrillic'
chars=os.listdir(DIR)
count=dict()
for char in chars:
    count.update({char:len(os.listdir(DIR+'/'+char))})
print(count)
```

```
{' ': 444, ' ': 463, ' ': 462, ' ': 551, ' ': 459, ' ': 474, ' ': 458, ' ': 447,
' ': 427, ' ': 415, ' ': 465, ' ': 431, ' ': 456, ' ': 469, ' ': 486, ' ': 344,
' ': 466, 'I': 247, ' ': 576, ' ': 493, ' ': 487, ' ': 464, ' ': 461, ' ': 433,
' ': 446, ' ': 470, ' ': 431, ' ': 508, ' ': 438, ' ': 459, ' ': 424, ' ': 448,
' ': 465, ' ': 513}
```

[6]:
```
from skimage import io
from skimage import img_as_float
from skimage.transform import resize
from PIL import Image

img_collection_arr=dict()
img_collection=dict()
for key,value in count.items():
    folder=os.path.join('./data/Cyrillic/',key)
    allfiles=os.listdir(folder)
    imlist=[filename for filename in allfiles if filename[-4:] in [".png",".
 ↪PNG"]]
    # Assuming all images are the same size, get dimensions of first image
    for image in imlist:
        filename=os.path.join(folder,image)
```

```
        camera = io.imread(filename,-1)
        image_file = Image.open(filename) # opens image
        img = image_file.convert('LA').resize((28,28)) # converts to grayscale␣
 ↪w/ alpha
        image_1=np.array(img)
#         print(image_1.shape)
#         MPL.imshow(img)
#         MPL.show()
        if(key not in img_collection_arr.keys()):
            img_collection_arr.update({key:[image_1]})
        else:
            if(len(img_collection_arr[key])<200):
                img_collection_arr[key].append(image_1)
    print(img_collection_arr.keys())
```

```
dict_keys([' '])
dict_keys([' ', ' '])
dict_keys([' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I'])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
```

```
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' '])
dict_keys([' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' '])
```

[12]:
```
count=dict()
for key,val in img_collection_arr.items():
    count.update({key:len(val)})
print(count)
```

```
{' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200,
' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200,
' ': 200, 'I': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200,
' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200, ' ': 200,
' ': 200, ' ': 200}
```

[7]:
```
import matplotlib.pyplot as MPL
from sklearn.model_selection import train_test_split
X=[];y=[]
class_names = list(img_collection_arr.keys())
for key,imgs in img_collection_arr.items():
    [X.append(img) for img in imgs]
    [y.append([class_names.index(key)]) for img in imgs]
#     for img in imgs:
```

```
#         print(img.shape)
#     print(img)
#     MPL.imshow(img)
#     MPL.show()
train_images, test_images, train_labels, test_labels = train_test_split(np.
 ↪array(X), np.array(y),
                                                    test_size=0.2,
                                                    random_state=0)
```

[21]:
```
print('classes:',class_names)
print('number of category:',len(class_names))
print("train image collection:",type(train_images),
      "\ntrain image:",type(train_images[0]),
      '\ntraining label collection:',type(train_labels),
      '\ntraining label:',type(train_labels[0]))
print('shape of training image:',train_images[0].shape)
```

```
classes: [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ']
number of category: 34
train image collection: <class 'numpy.ndarray'>
train image: <class 'numpy.ndarray'>
training label collection: <class 'numpy.ndarray'>
training label: <class 'numpy.ndarray'>
shape of training image: (28, 28, 2)
```

[9]:
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 2)))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(len(class_names), activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        608
_________________________________________________________________
```

```
batch_normalization (BatchNo (None, 26, 26, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
batch_normalization_1 (Batch (None, 11, 11, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
batch_normalization_2 (Batch (None, 3, 3, 64)          256
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 64)                36928
_________________________________________________________________
dense_1 (Dense)              (None, 34)                2210
=================================================================
Total params: 95,810
Trainable params: 95,490
Non-trainable params: 320
_________________________________________________________________
```

[10]:
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
```

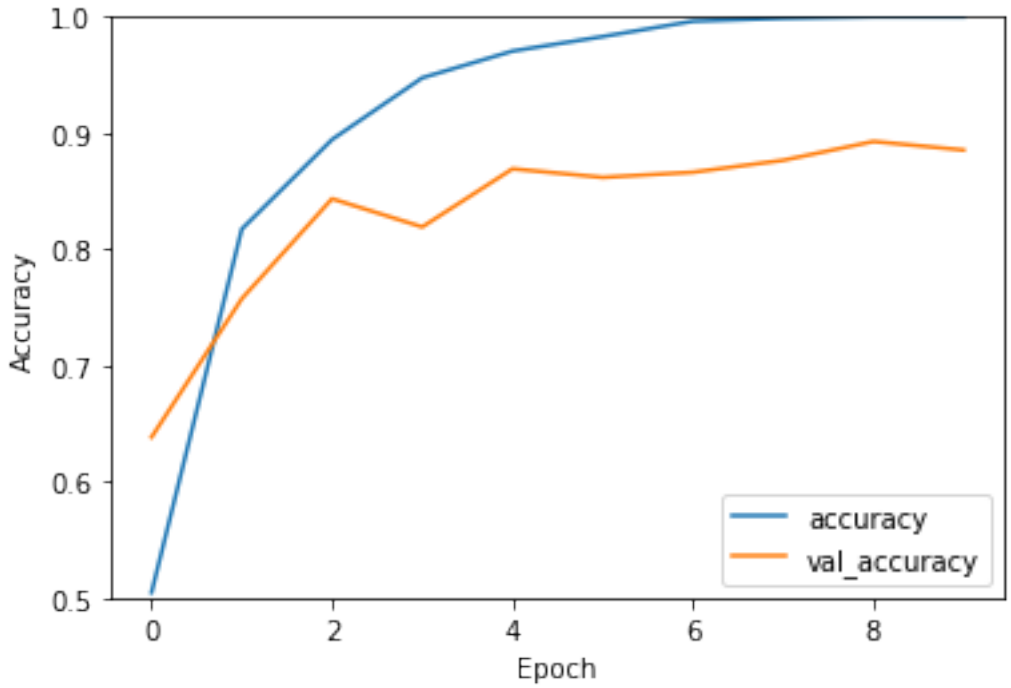
```
Train on 5440 samples, validate on 1360 samples
Epoch 1/10
5440/5440 [==============================] - 12s 2ms/sample - loss: 1.8315 -
accuracy: 0.5046 - val_loss: 1.2628 - val_accuracy: 0.6382
Epoch 2/10
5440/5440 [==============================] - 11s 2ms/sample - loss: 0.6409 -
accuracy: 0.8173 - val_loss: 0.8388 - val_accuracy: 0.7574
Epoch 3/10
5440/5440 [==============================] - 10s 2ms/sample - loss: 0.3661 -
accuracy: 0.8943 - val_loss: 0.5372 - val_accuracy: 0.8434
Epoch 4/10
5440/5440 [==============================] - 10s 2ms/sample - loss: 0.2052 -
accuracy: 0.9474 - val_loss: 0.6008 - val_accuracy: 0.8191
Epoch 5/10
5440/5440 [==============================] - 11s 2ms/sample - loss: 0.1234 -
accuracy: 0.9704 - val_loss: 0.4653 - val_accuracy: 0.8691
```

```
Epoch 6/10
5440/5440 [==============================] - 10s 2ms/sample - loss: 0.0783 -
accuracy: 0.9829 - val_loss: 0.4719 - val_accuracy: 0.8618
Epoch 7/10
5440/5440 [==============================] - 11s 2ms/sample - loss: 0.0367 -
accuracy: 0.9961 - val_loss: 0.5007 - val_accuracy: 0.8662
Epoch 8/10
5440/5440 [==============================] - 11s 2ms/sample - loss: 0.0177 -
accuracy: 0.9987 - val_loss: 0.4323 - val_accuracy: 0.8765
Epoch 9/10
5440/5440 [==============================] - 10s 2ms/sample - loss: 0.0097 -
accuracy: 1.0000 - val_loss: 0.4000 - val_accuracy: 0.8926
Epoch 10/10
5440/5440 [==============================] - 10s 2ms/sample - loss: 0.0057 -
accuracy: 1.0000 - val_loss: 0.4047 - val_accuracy: 0.8853
```

[11]:
```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print("test accuracy:",test_acc)
```

```
1360/1 - 1s - loss: 0.3869 - accuracy: 0.8853
test accuracy: 0.88529414
```


1.0
0.9
0.8
0.7
0.6
0.5
Accuracy
0
2
4
6
8
Epoch
accuracy
val_accuracy

[ ]:

[ ]:

[ ]: