Квасов Андрей Николаевич, ИТУ (маг. 1к.)
Лабораторная работа №4
Вариант - 11

Задание:
Решить задачу классификации исходного изображения с помощью метода k ближайших соседей (римские цифры). Оценить точность полученной модели. Возможно использование преобученной нейронной сети.

датасет для выполнения был взят здесь.

Код программы:

```
import glob
import numpy as np
import matplotlib.pyplot as plt
from pylab import rcParams
import PIL.Image as img
# Uploading dataset
def img_array(path):
    image = img.open(path)
    tmp = np.array(image)
    image.close()
    return tmp

globs =
glob.glob('/content/drive/MyDrive/dataset/roman/dataset/1/*.png')
dataset = np.array(list(map(img_array, globs)))
labels = np.ones(len(dataset))-1

for i in range(2,9):
    globs =
glob.glob('/content/drive/MyDrive/dataset/roman/dataset/'+str(i)+'/*.pn
g')
    dataset = np.concatenate([dataset,np.array(list(map(img_array,
globs)))])
    labels = np.concatenate([labels,
np.ones(len(np.array(list(map(img_array, globs)))))*(i-1)])
dataset = np.mean(dataset, axis=3)
# Checking shapes
print(dataset.shape)
print(labels.shape)
rcParams['figure.figsize'] = 5, 10
```

```
plt.title('Picture example')
plt.imshow(dataset[1], cmap='Greys')
plt.show()
def gallery(array, ncols=20):
    nindex, height, width = array.shape
    nrows = nindex//ncols
    assert nindex == nrows*ncols
    # want result.shape = (height*nrows, width*ncols, intensity)
    result = (array.reshape(nrows, ncols, height, width)
              .swapaxes(1,2)
              .reshape(height*nrows, width*ncols))
    return result

rcParams['figure.figsize'] = 15, 30
result = gallery(dataset)
plt.title('Raw dataset of Roman Numerals')
plt.imshow(result, cmap='Greys')
plt.show()
!pip install Augmentor
import Augmentor
from Augmentor import Pipeline

def augmentation(path):
    p = Augmentor.Pipeline(path) # ensure you press enter after this,
don't just c&p this code.
    Pipeline.set_seed(100)
    p.rotate(probability=0.3, max_left_rotation=3,
max_right_rotation=3)
    p.random_distortion(probability=0.7, grid_width=4, grid_height=4,
magnitude=2)
    p.random_erasing(probability=0.2, rectangle_area=0.2)
    p.sample(200)

path = '/content/drive/MyDrive/dataset/roman/dataset/'
for i in range(1,9):
    augmentation(path+str(i)+'/')
globs =
glob.glob('/content/drive/MyDrive/dataset/roman/dataset/1/output/*.png'
)
aug_dataset = np.array(list(map(img_array, globs)))
aug_labels = np.ones(len(aug_dataset))-1

for i in range(2,9):
```

```
    globs =
glob.glob('/content/drive/MyDrive/dataset/roman/dataset/'+str(i)+'/outp
ut/*.png')
    aug_dataset =
np.concatenate([aug_dataset,np.array(list(map(img_array, globs)))])
    aug_labels = np.concatenate([aug_labels,
np.ones(len(np.array(list(map(img_array, globs))))*(i-1)])

# Making them all in one chanel
aug_dataset = np.mean(aug_dataset, axis=3)
result = gallery(aug_dataset, 32)
plt.title('Augmented part of dataset of Roman Numerals')
plt.imshow(result, cmap='Greys')
plt.show()
from sklearn.model_selection import train_test_split
from sklearn import neighbors

X = np.concatenate([dataset, aug_dataset])
# X = dataset
# y = labels
X = X/255.
X = X.reshape((len(X), 28*28))
y = np.concatenate([labels, aug_labels])
# X,y = shuffle(X, y, random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

model = neighbors.KNeighborsClassifier(n_neighbors = 3)
model.fit(X_train, y_train)
y_test_pred = model.predict(X_test)

N_test = X_test.shape[0]
N_test
from sklearn import metrics
print(metrics.accuracy_score(y_test, y_test_pred))
print(np.mean(y_test != y_test_pred))
M = metrics.confusion_matrix(y_test, y_test_pred)
#M = M - np.diag(np.diag(M))
M = np.sqrt(M)
plt.imshow(M, interpolation = 'nearest')
plt.set_cmap('binary')
plt.grid(False)
plt.xticks(range(10))
```

```
plt.yticks(range(10))
plt.xlabel("predicted label")
plt.ylabel("true label")
plt.colorbar()
kk = range(1, 30, 2)
err_train = []
err_test = []
for k in kk:
    model = neighbors.KNeighborsClassifier(n_neighbors = k)
    model.fit(X_train, y_train)
    err_train.append(np.mean(model.predict(X_train) != y_train))
    err_test.append(np.mean(model.predict(X_test) != y_test))
plt.plot(kk, err_train, '.-r', label = 'Train error')
plt.plot(kk, err_test,  '.-b', label = 'Test error')
plt.legend(loc = 2)
print(min(err_test))
print(kk[err_test.index(min(err_test))])
plt.plot(kk, err_train, '.-r', label = 'Train error')
plt.plot(kk, err_test,  '.-b', label = 'Test error')
plt.legend(loc = 1)
plt.xlim([30, 0])
model = neighbors.KNeighborsClassifier(n_neighbors = 3)
model.fit(X_train, y_train)
i=1
plt.figure(figsize = (10, 10)) # Размер окна в дюймах
i_subplot = 1
yi_test_pred =model.predict(X_test)
for i in range(N_test):
    if yi_test_pred[i]!= y_test[i]:
        plt.subplot(8, 8, i_subplot)
        i_subplot += 1
        plt.xticks([])
        plt.yticks([])
        #plt.imshow(np.reshape(X_test[i, :], [28, 28]), cmap='gray')
        plt.imshow(np.reshape(X_test[i, :], (28, 28)), cmap =
plt.cm.binary)
        plt.text(0, 7, str(y_test[i]),    color = 'b')
        plt.text(0, 1, str(yi_test_pred[i]), color = 'r')
```

Результаты:

```
0.0014534883720930232
9
```

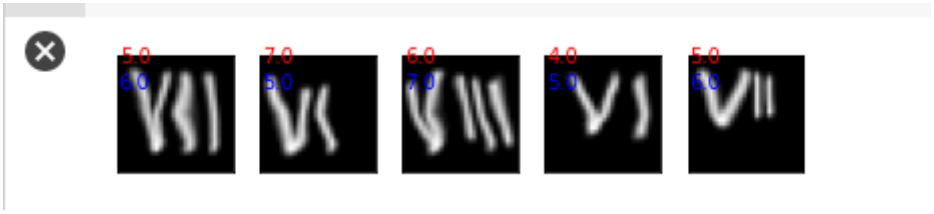

Рис. 1. - Распознавание римских цифр

Вывод:

В ходе выполнения данной работы, было осуществлено распознавание римских цифр при помощи метода к-ближайших соседей. Судя по метрикам ошибка, метод показал достаточно неплохие результаты.