Министерство науки и высшего образованияРоссийской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический университет

имени В.Ф.Уткина»

Кафедра «Электронные вычислительные машины»

Отчет по практическому занятию №10

на тему

«Алгоритм раскрашивания изображений»

по дисциплине  
**«Машинное обучение»**

Выполнили:

Студенты группы №140

Бригада 7

Сафаров Д. А.

Тимохин Е. М.

Проверила:  
ас. Панина И.С.

**Цель работы**

Познакомиться с алгоритмом раскрашивания черно-белых изображений, теорией цвета и построить нейросеть для раскрашивания незнакомых изображений

**Задание**

1. Получить результат для 3 черно-белых изображений (обучить сеть на 1 изображении).

2. Обучить нейронную сеть не на одном изображении, а на 15. Сравнить результат на тех же изображениях, что были использованы в п. 1.

**Практическая часть**

Код программы представлен в приложении А.

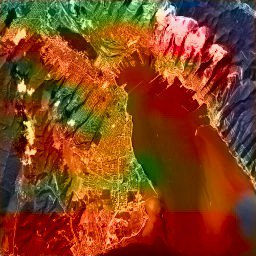
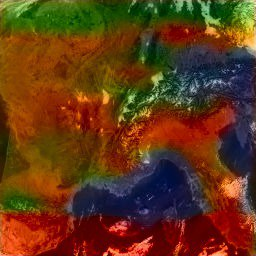
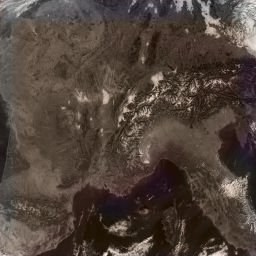
Результат работы после обучения на 1 изображении представлен на рисунке 1.

Рисунок 1 – Результат работы на основе 1 изображения

Результат работы после обучения на 15 изображениях представлен на рисунке 2.

Рисунок 2 – Результат работы на основе 15 изображений

Приложение А. Текст программы

import os

from os.path import isfile, join

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '3' # 0 = all messages are logged, 3 - INFO, WARNING, and ERROR messages are not printed

import tensorflow as tf

from keras.layers import Conv2D, Conv2DTranspose, UpSampling2D

from keras.layers import Activation, Dense, Dropout, Flatten, InputLayer

from keras.layers import BatchNormalization

from keras.callbacks import TensorBoard

from keras.models import Sequential

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from keras.utils import array\_to\_img, img\_to\_array, load\_img

from skimage.color import rgb2lab, lab2rgb, rgb2gray

from skimage.io import imsave

import numpy as np

import random

import cv2

from keras.regularizers import l2

folder\_train = "Faces"

folder\_src = "Input"

folder\_dst = "Output"

model\_file = "ss.weights.h5"

brightness\_corr = 250

do\_train = True

# Get train images

X = []

for filename in os.listdir(folder\_train):

X.append(img\_to\_array(load\_img(folder\_train + os.sep + filename)))

X = np.array([cv2.resize(i, (256, 256)) for i in X], dtype=float)/255.0

# X = np.array(X, dtype=float)

Xtrain = X

# Model

model = Sequential()

model.add(InputLayer(input\_shape=(256, 256, 1)))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same', strides=2))

model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))

model.add(Conv2D(128, (3, 3), activation='relu', padding='same', strides=2))

model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))

model.add(Conv2D(256, (3, 3), activation='relu', padding='same', strides=2))

model.add(Conv2D(512, (3, 3), activation='relu', padding='same'))

model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))

model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))

model.add(UpSampling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))

model.add(UpSampling2D((2, 2)))

model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))

model.add(Conv2D(2, (3, 3), activation='tanh', padding='same'))

model.add(UpSampling2D((2, 2)))

model.compile(optimizer='rmsprop', loss='mse')

# Image transformer

datagen = ImageDataGenerator(shear\_range=0.2, zoom\_range=0.2, rotation\_range=20, horizontal\_flip=True)

# Generate training data

batch\_size = 10

def image\_a\_b\_gen(batch\_size):

for batch in datagen.flow(Xtrain, batch\_size=batch\_size):

lab\_batch = rgb2lab(batch)

X\_batch = lab\_batch[:,:,:,0]

Y\_batch = lab\_batch[:,:,:,1:] / 128

yield (X\_batch.reshape(X\_batch.shape+(1,)), Y\_batch)

if do\_train:

# Train model

model.fit(image\_a\_b\_gen(batch\_size), epochs=2, steps\_per\_epoch=50)

# Save model

model.save\_weights(model\_file)

# Load model

model.load\_weights(model\_file)

# Process images

color\_me = []

onlyfiles = [f for f in os.listdir(folder\_src) if isfile(join(folder\_src, f))]

for filename in onlyfiles:

color\_me.append(img\_to\_array(load\_img(folder\_src + os.sep + filename)))

color\_me = np.array([cv2.resize(i, (256, 256)) for i in color\_me], dtype=float)

# color\_me = np.array(color\_me, dtype=float)

color\_me = rgb2lab(1.0/255\*color\_me)[:,:,:,0]

color\_me = color\_me.reshape(color\_me.shape+(1,))

# Test model

output = model.predict(color\_me)

# Output colorizations

for i in range(len(output)):

cur = np.zeros((256, 256, 3))

cur[:,:,0] = color\_me[i][:,:,0]

cur[:,:,1:] = output[i] \* 128

img\_rgb = lab2rgb(cur)\*brightness\_corr

imsave(folder\_dst + os.sep + "img\_%d.png" % i, img\_rgb.astype(np.uint8))

print("img\_%d.png saved" % i)

**Вывод**

Был реализован алгоритм раскрашивания изображений с помощью нейросетей.