Санкт-Петербургский политехнический университет Петра Великого

Машинное обучение на языке Python  
“Распознавание цифр”

Выполнил : Задорожний Антон 219/5

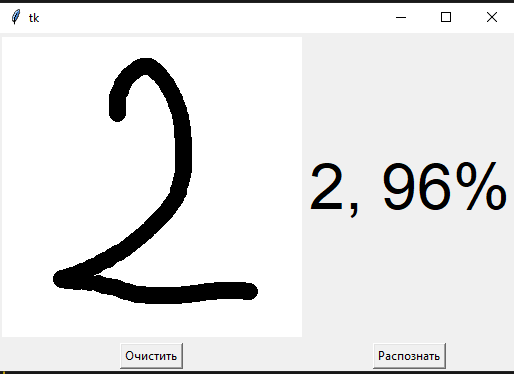
Проверила : Кудрявцева А.С.

Санкт-Петербург

2020 год

О проекте

В этом материале мы реализуем приложение для распознавания написанных от руки цифр с помощью набора данных MNIST. Используем специальный тип глубокой нейронной сети, которая называется сверточной нейронной сетью. А в конце создадим графический интерфейс, в котором можно будет рисовать цифру и тут же ее узнавать.



Установка нужных библиотек

Для этого проекта нам понадобятся библиотеки:

* Tkinter
* Numpy
* Tensorflow
* Keras
* Pillow

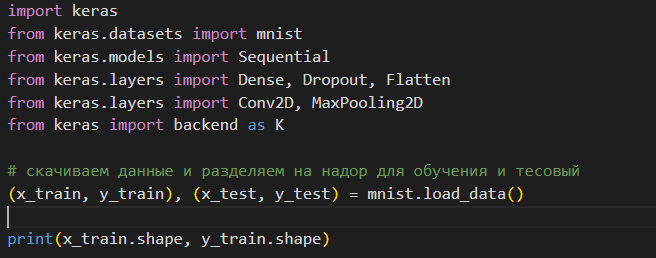
Набор данных MNIST

Этот набор содержит 60 000 тренировочных изображений написанных от руки цифр от 0 до 9, а также 10 000 картинок для тестирования. В наборе есть 10 разных классов. Изображения с цифрами представлены в виде матриц 28 х 28, где каждая ячейка содержит определенный оттенок серого. Для проекта устанавливаем его с официального сайта.



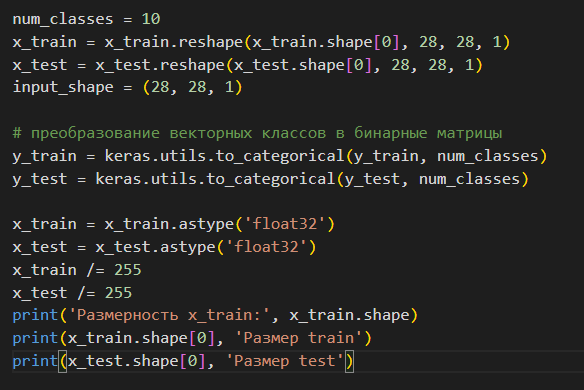
Создание проекта

Для начала в нашем проекте мы должны импортировать все необходимые модули.



В консоль должно вывести : (60000, 28, 28) (60000,).

Суть в том, что нельзя просто так передать данные в модель, поэтому сперва сделать определенные операции. Изначально размер тренировочных данных составляет - (60000, 28, 28), а модель сверточной нейронной сети требует одну размерность, поэтому потребуется перестроить форму (60000, 28, 28, 1).



В консоль должно вывести : Размерность x\_train: (60000, 28, 28, 1)

60000 Размер train

10000 Размер test

Создание модели

Далее мы должны создать модель сверточной нейронной сети. Такая модель лучше работает с данными, предоставленными в качестве сеточных структур.

batch\_size = 128

epochs = 10

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3),activation='relu',input\_shape=input\_shape))

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

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(256, activation='relu'))

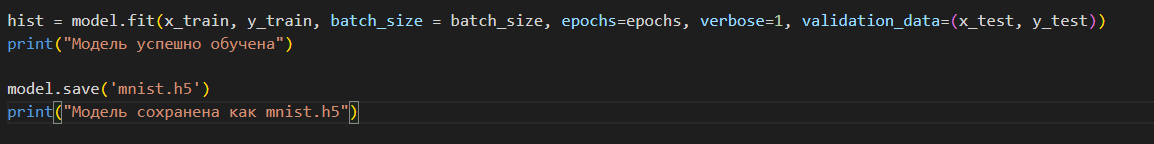
model.add(Dropout(0.5))

model.add(Dense(num\_classes, activation='softmax'))

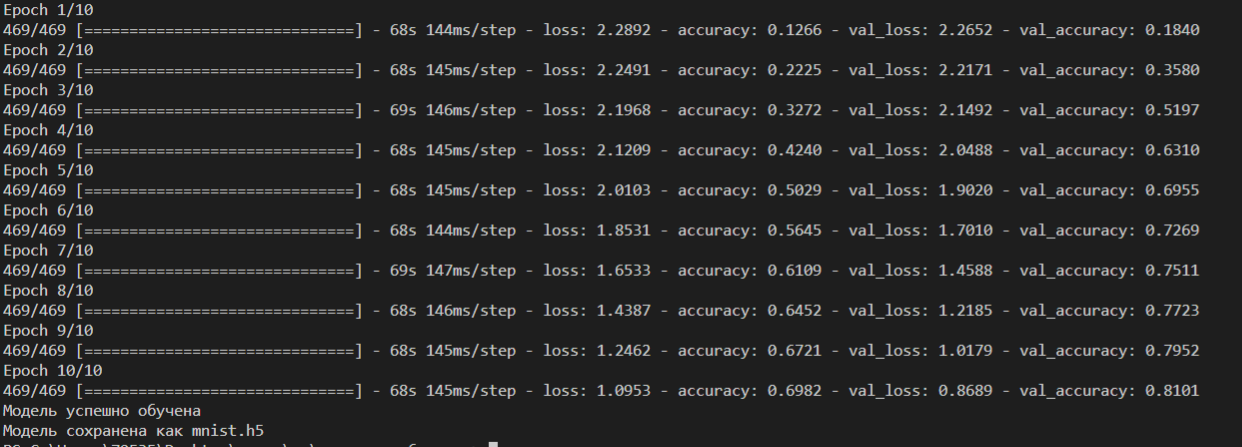
model.compile(loss=keras.losses.categorical\_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])

Тренировка модели

После создания модели нам нужно натренировать ее, чтобы она смогла распознавать рукописные цифры. Функция model.fit() в Keras начнет тренировку модели. Она принимает тренировочные, валидационные данные, эпохи (epoch) и размер батча (batch). Данная тренировка займет некоторое время ( зависит от мощности ПК ).



Запускаем код и наблюдаем тренировку модели



После окончания тренировки у нас появляется файл mnist.h5



Создание графического интерфейса для предсказания цифр

Теперь, когда у нас есть натренированная модель, то мы можем создать уже новый файл с графическим интерфейсом. Для этого нам понадобиться библиотека Tkinter. Функция predict\_digit() принимает входящее изображение и затем использует натренированную сеть для предсказания. Затем создаем класс App, который будет отвечать за построение графического интерфейса приложения. Создаем полотно, на котором можно рисовать, захватывая события мыши. Кнопка же будет активировать функцию predict\_digit() и отображать результат.

from keras.models import load\_model

from tkinter import \*

import tkinter as tk

import win32gui

from PIL import ImageGrab, Image

import numpy as np

model = load\_model('mnist.h5')

def predict\_digit(img):

    # изменение рзмера изобржений на 28x28

    img = img.resize((28,28))

    # конвертируем rgb в grayscale

    img = img.convert('L')

    img = np.array(img)

    # изменение размерности для поддержки модели ввода и нормализации

    img = img.reshape(1,28,28,1)

    img = img/255.0

    # предстказание цифры

    res = model.predict([img])[0]

    return np.argmax(res), max(res)

class App(tk.Tk):

    def \_\_init\_\_(self):

        tk.Tk.\_\_init\_\_(self)

        self.x = self.y = 0

        # Создание элементов

        self.canvas = tk.Canvas(self, width=300, height=300, bg = "white", cursor="cross")

        self.label = tk.Label(self, text="Думаю..", font=("Helvetica", 48))

        self.classify\_btn = tk.Button(self, text = "Распознать", command =         self.classify\_handwriting)

        self.button\_clear = tk.Button(self, text = "Очистить", command = self.clear\_all)

        # Сетка окна

        self.canvas.grid(row=0, column=0, pady=2, sticky=W, )

        self.label.grid(row=0, column=1,pady=2, padx=2)

        self.classify\_btn.grid(row=1, column=1, pady=2, padx=2)

        self.button\_clear.grid(row=1, column=0, pady=2)

        # self.canvas.bind("<Motion>", self.start\_pos)

        self.canvas.bind("<B1-Motion>", self.draw\_lines)

    def clear\_all(self):

        self.canvas.delete("all")

    def classify\_handwriting(self):

        HWND = self.canvas.winfo\_id()

        rect = win32gui.GetWindowRect(HWND) # получаем координату холста

        im = ImageGrab.grab(rect)

        digit, acc = predict\_digit(im)

        self.label.configure(text= str(digit)+', '+ str(int(acc\*100))+'%')

    def draw\_lines(self, event):

        self.x = event.x

        self.y = event.y

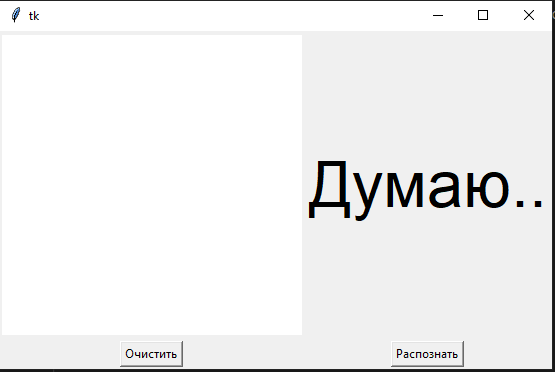
        r=8

        self.canvas.create\_oval(self.x-r, self.y-r, self.x + r, self.y + r, fill='black')

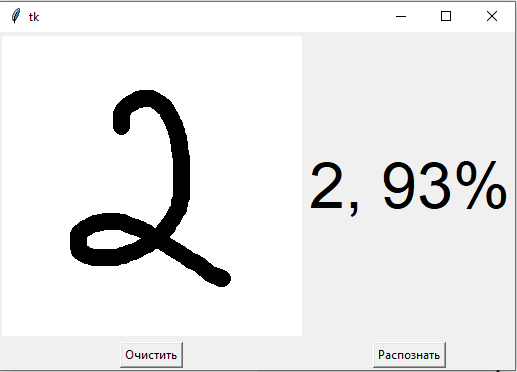
app = App()

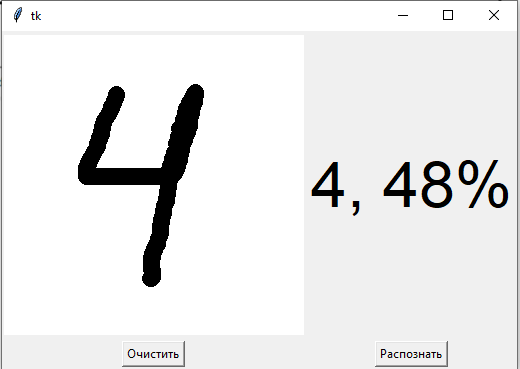
mainloop()

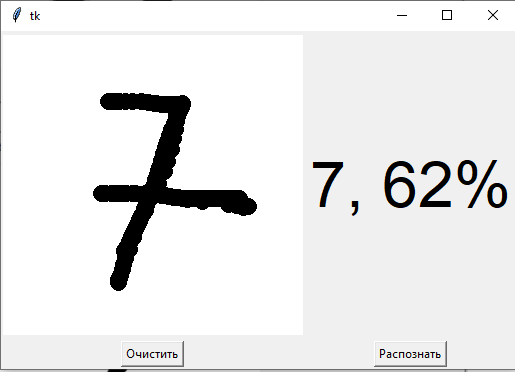
После запуска получаем такое окно

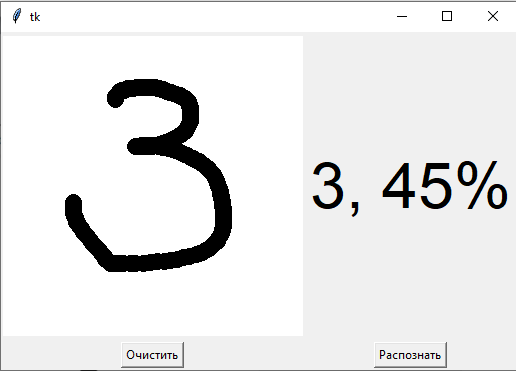


Проверка









Выводы

Проект для распознавания рукописных цифр на Python готов. Была создана и натренирована сверточная нейронная сеть, которая идеально подходит для классификации изображений. Наконец, был реализован графический интерфейс, который используется для рисования и представления результата предсказания цифры.