**Лабораторная работа №1****Нейросеть по распознаванию цифр**

**Задача:** Описание задач. Требуется распознать черно-белые цифры от 0 до 9, представленные в виде черных квадратиков в табличке 3×5 квадратов.

Каждая цифра представляет собой всего пятнадцать квадратиков, причем только двух возможных цветов. За белый квадратик отвечает 0, а черный квадратик – 1. Можно представить цифры в виде последовательности нулей и единиц

**Результаты работы программы:**

Пример что определяет значение:



Пример что может не определить значение:

****

**Код программы (на языке Python):**

import tkinter as tk  
import random  
  
zero = list("111101101101111")  
one = list("001001001001001")  
two = list("111001111100111")  
three = list("111001111001111")  
four = list("101101111001001")  
five = list("111100111001111")  
six = list("111100111101111")  
seven = list("111001001001001")  
eight = list("111101111101111")  
nine = list("111101111001111")  
  
digits = [zero, one, two, three, four, five, six, seven, eight, nine]  
  
weights = [[0 for \_ in range(15)] for \_ in range(10)]  
bias = 7  
  
def proceed(number, correct\_num=5):  
 net = sum(int(number[i]) \* weights[correct\_num][i] for i in range(15))  
 return net >= bias  
  
def decrease(number, correct\_num=5):  
 for i in range(15):  
 if int(number[i]) == 1:  
 weights[correct\_num][i] -= 1  
  
def increase(number, correct\_num=5):  
 for i in range(15):  
 if int(number[i]) == 1:  
 weights[correct\_num][i] += 1  
  
def train():  
 for correct\_num in range(10):  
 for \_ in range(10000):  
 option = random.randint(0, 9)  
 if option != correct\_num:  
 if proceed(digits[option], correct\_num):  
 decrease(digits[option], correct\_num)  
 else:  
 if not proceed(digits[correct\_num], correct\_num):  
 increase(digits[correct\_num], correct\_num)  
  
def recognize(num):  
 scores = []  
 for i in range(10):  
 score = sum(int(num[j]) == int(digits[i][j]) for j in range(15))  
 scores.append((score, str(i)))  
  
 scores.sort(reverse=True, key=lambda x: x[0])  
  
 if scores[0][0] == 15:  
 return scores[0][1]  
  
 if scores[0][0] >= 5:  
 if len(scores) > 1 and scores[1][0] >= 5:  
 return f"Скорее {scores[0][1]}"  
 return scores[0][1]  
  
 return "Не знаю, что за число"  
  
class RecognizerApp:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Распознавание цифр 3x5")  
  
 self.exit\_button = tk.Button(root, text="Выход", command=root.quit)  
 self.exit\_button.grid(row=0, column=0)  
  
 self.matrices = []  
  
 self.create\_matrices()  
  
 def create\_matrices(self):  
 for matrix in self.matrices:  
 for widget in matrix["widgets"]:  
 widget.destroy()  
 self.matrices.clear()  
  
 for m in range(10):  
 matrix\_frame = tk.Frame(self.root)  
 matrix\_frame.grid(row=1, column=m, padx=5, pady=10)  
  
 grid = [[0 for \_ in range(3)] for \_ in range(5)]  
 buttons = []  
 for row in range(5):  
 button\_row = []  
 for col in range(3):  
 btn = tk.Button(matrix\_frame, bg="white", width=2, height=1, command=lambda r=row, c=col, m=m: self.toggle(m, r, c))  
 btn.grid(row=row, column=col, padx=1, pady=1)  
 button\_row.append(btn)  
 buttons.append(button\_row)  
  
 recognize\_button = tk.Button(matrix\_frame, text="Узнать число", command=lambda m=m: self.on\_recognize(m))  
 recognize\_button.grid(row=5, column=0, columnspan=3, pady=5)  
  
 result\_label = tk.Label(matrix\_frame, text="Результат:", font=("Arial", 10), width=20, anchor="w")  
 result\_label.grid(row=6, column=0, columnspan=3, pady=5)  
  
 self.matrices.append({  
 "frame": matrix\_frame,  
 "grid": grid,  
 "buttons": buttons,  
 "recognize\_button": recognize\_button,  
 "result\_label": result\_label,  
 "widgets": buttons + [recognize\_button, result\_label]  
 })  
  
 def toggle(self, m, row, col):  
 current\_color = self.matrices[m]["buttons"][row][col].cget("bg")  
 new\_color = "black" if current\_color == "white" else "white"  
 self.matrices[m]["buttons"][row][col].config(bg=new\_color)  
 self.matrices[m]["grid"][row][col] = 1 if new\_color == "black" else 0  
  
 def on\_recognize(self, m):  
 num = "".join(str(self.matrices[m]["grid"][row][col]) for row in range(5) for col in range(3))  
 result = recognize(num)  
 self.matrices[m]["result\_label"].config(text=f"Результат:\n{result}")  
  
train()  
root = tk.Tk()  
root.geometry("1550x350")  
app = RecognizerApp(root)  
root.mainloop()

**Вывод:** Разработанная нейросеть показала невысокую точность распознавания цифр при неполном заполнении клеток, в некоторых случаях она всё же демонстрировала результаты. Проведение этого лабораторного эксперимента позволило мне получить начальные навыки и знания в области разработки нейронных сетей.