**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №4

По дисциплине: **«Введение в нейронные сети»**

на тему: **Задача классификации**

Выполнил: студент группы ИТП-31

Пронуза М.Ю.

Проверил:

Башаримов Ю.С.

Гомель 2024

**Цель работы**: разработать программное обеспечение для решения задачи распознавания образов и классификации.

**Задание:**

Исходными данными является изображение, содержащее распознаваемый объект. Необходимо выполнить их распознавание (соотнесение с одним из заданных классов). В качестве параметров программа должна принимать: файл, содержащий:

ссылки на файлы с обучающей выборкой;

ссылки на файлы с тестируемыми образцами;

ссылку на файл с результатами проверки образцов (принадлежит какому классу).

Если происходит запуск без параметров, то запускается программа с графическим интерфейсом, ссылки на обучающую выборку берутся из конфигурационного XML-файла аналогичной структуры, как и файл параметров командной строки.

Для классификации использовать персептрон с заданным количеством скрытых слоёв и количеством нейронов на выходном слое. Для обучения использовать метод обратного распространения ошибки с указанных методом минимизации (таблица 4.1).

**Ход выполнения и результаты работы:**

На рисунке 1 представлены обработанные изображения.

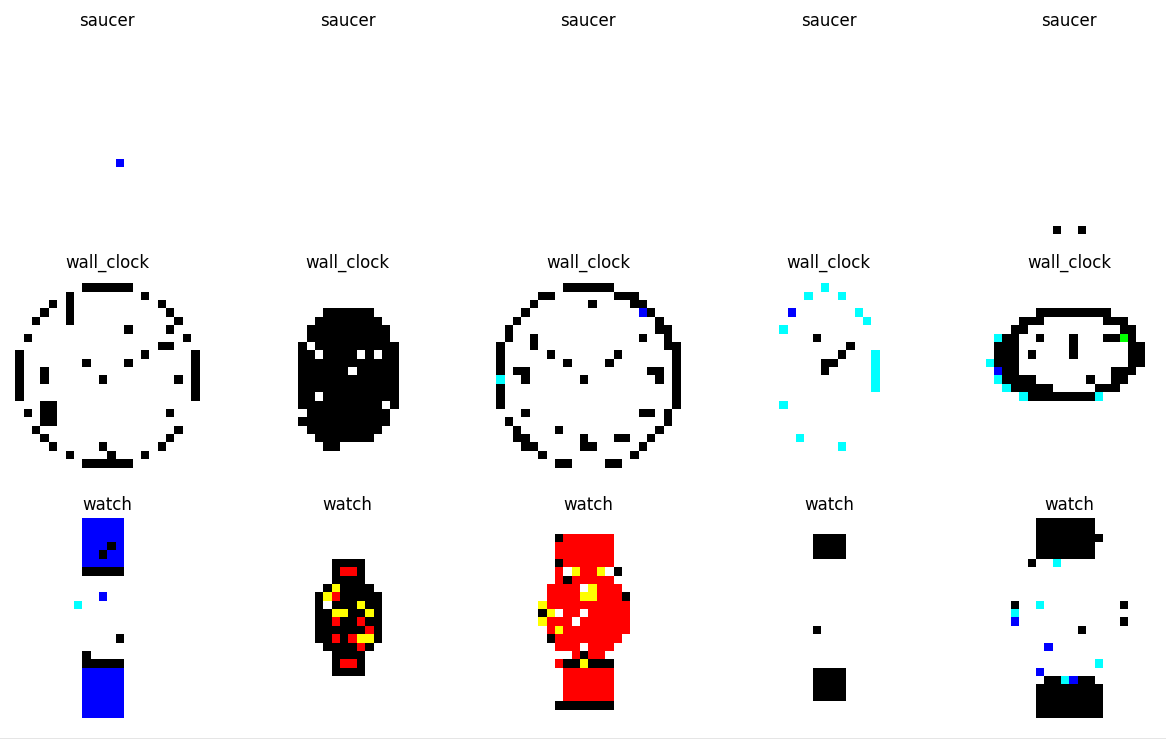


Рисунок 1 – обработанные изображения

На рисунке 2 представлена точность обучения.

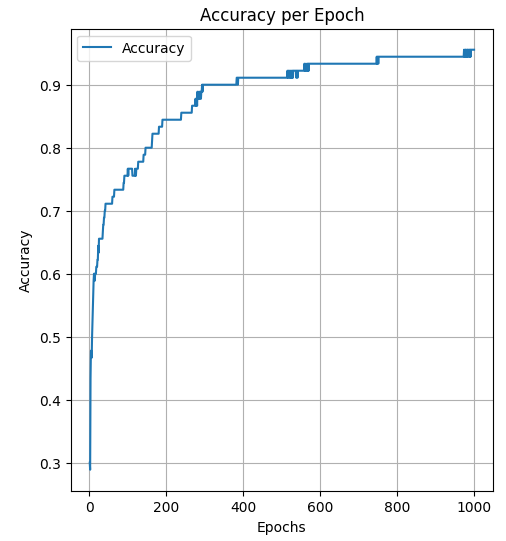


Рисунок 2 – точность обучения

На рисунке 3 представлены предсказания нейронной сети.

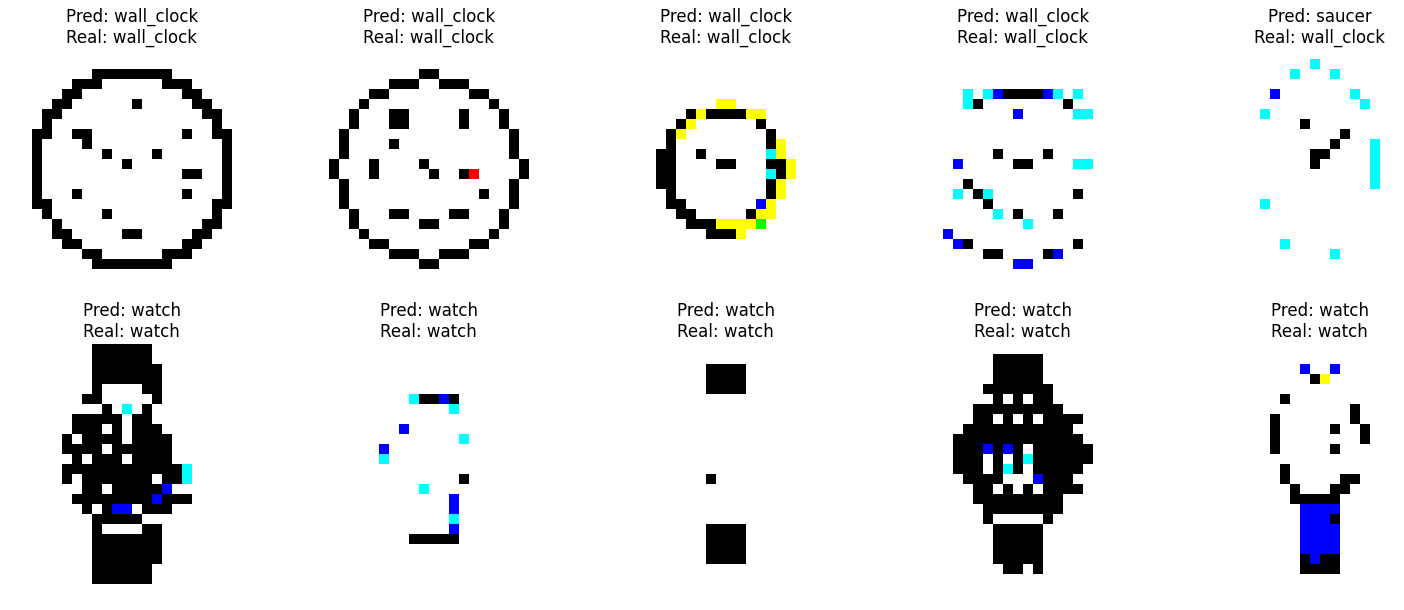


Рисунок 3 – предсказания нейронной сети

Текст обработки входных данных представлен в приложении А.

Текст программы классификации методом градиентного спуска представлен в приложении Б.

Текст графиков сравнения представлен в приложении В.

**Вывод:** в ходе выполнения лабораторной работы было разработано программное обеспечение для решения задачи распознавания образов и классификации.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст обработки входных данных**

import zipfile

import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

import random # Для случайного выбора изображений

# Функция для распаковки архива

def unpack\_archive(archive\_path, extract\_to="./data"):

with zipfile.ZipFile(archive\_path, 'r') as zip\_ref:

zip\_ref.extractall(extract\_to)

# Функция для обработки изображения

def process\_image(image\_path, threshold=128):

# Чтение изображения

image = cv2.imread(image\_path)

# Изменение размера до 24x24

resized\_image = cv2.resize(image, (24, 24))

# Преобразование в бинарный формат на основе порога

binary\_image = (resized\_image > threshold).astype(np.uint8) \* 255 # Преобразование в [0, 255]

return binary\_image

# Распаковка архива

archive\_path = "data.zip"

unpack\_archive(archive\_path)

# Путь к директории с данными

data\_dir = "./data"

categories = ["saucer", "wall\_clock", "watch"]

# Словарь для хранения обработанных данных

processed\_data = {category: [] for category in categories}

# Обработка изображений

for category in categories:

category\_path = os.path.join(data\_dir, category)

for image\_name in os.listdir(category\_path):

image\_path = os.path.join(category\_path, image\_name)

processed\_image = process\_image(image\_path)

processed\_data[category].append(processed\_image)

# Функция для отображения изображений (выбирает 5 случайных изображений)

def plot\_images(data, category):

plt.figure(figsize=(15, 3))

random\_images = random.sample(data, min(len(data), 5)) # Выбирает до 5 случайных изображений

for i, image in enumerate(random\_images):

plt.subplot(1, 5, i + 1)

plt.imshow(image)

plt.title(category)

plt.axis('off')

plt.show()

for category in categories:

plot\_images(processed\_data[category], category)

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Текст метода градиентного спуска**

import numpy as np

import matplotlib.pyplot as plt

# Инициализация весов с использованием He Initialization

def initialize\_weights(input\_size, hidden1\_size, hidden2\_size, output\_size):

W1 = np.random.randn(input\_size, hidden1\_size) \* np.sqrt(2. / input\_size)

b1 = np.zeros((1, hidden1\_size))

W2 = np.random.randn(hidden1\_size, hidden2\_size) \* np.sqrt(2. / hidden1\_size)

b2 = np.zeros((1, hidden2\_size))

W3 = np.random.randn(hidden2\_size, output\_size) \* np.sqrt(2. / hidden2\_size)

b3 = np.zeros((1, output\_size))

return W1, b1, W2, b2, W3, b3

# Подготовка данных

categories = ["saucer", "wall\_clock", "watch"]

X, y = prepare\_dataset(processed\_data, categories)

# Нормализация входных данных (приведение к [0, 1])

X = X / 255.0

y\_encoded = one\_hot\_encode(y, num\_classes=3)

# Параметры модели

input\_size = 24 \* 24 \* 3

hidden1\_size = 128

hidden2\_size = 64

output\_size = 3

learning\_rate = 0.001 # Уменьшен для более стабильного обучения

epochs = 1000 # Увеличено количество эпох

# Инициализация параметров

W1, b1, W2, b2, W3, b3 = initialize\_weights(input\_size, hidden1\_size, hidden2\_size, output\_size)

# Обучение нейросети

losses = []

accuracies = []

for epoch in range(epochs):

# Прямой проход

Z1, A1, Z2, A2, Z3, A3 = forward\_propagation(X, W1, b1, W2, b2, W3, b3)

# Вычисление потерь

loss = compute\_loss(y\_encoded, A3)

losses.append(loss)

# Вычисление точности

accuracy = compute\_accuracy(A3, y\_encoded)

accuracies.append(accuracy)

# Обратное распространение

dW1, db1, dW2, db2, dW3, db3 = backward\_propagation(X, y\_encoded, Z1, A1, Z2, A2, A3, W1, W2, W3)

# Обновление параметров

W1, b1, W2, b2, W3, b3 = update\_parameters(W1, b1, W2, b2, W3, b3, dW1, db1, dW2, db2, dW3, db3, learning\_rate)

# Печать результата на каждой эпохе

if (epoch + 1) % 10 == 0 or epoch == 0:

print(f"Epoch {epoch + 1}/{epochs}, Loss: {loss:.4f}, Accuracy: {accuracy:.4f}")

**ПИЛОЖЕНИЕ В**

(обязательное)

**Текст графиков**

# Построение графиков

plt.figure(figsize=(12, 6))

# График потерь

plt.subplot(1, 2, 1)

plt.plot(range(1, epochs + 1), losses, label="Loss")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.title("Loss per Epoch")

plt.legend()

plt.grid()

# График точности

plt.subplot(1, 2, 2)

plt.plot(range(1, epochs + 1), accuracies, label="Accuracy")

plt.xlabel("Epochs")

plt.ylabel("Accuracy")

plt.title("Accuracy per Epoch")

plt.legend()

plt.grid()

plt.show()