Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 1

по курсу «Реализация однослойного перцептрона»

«Теория искусственных нейронных сетей»

Студент группы ИУ9-71Б Баев Д.А

Преподаватель Каганов Ю. Т.

Москва 2023

1 Задание

1. Реализовать на языке высокого уровня однослойный персептрон и проверить его работоспособность на примере искусственных данных типа цифр от 0 до 9 и букв русского алфавита. Размер поля 5x4.

2. Исследовать работу персептрона на основе использования различных функций активации. (Линейной, сигмоиды, гиперболического тангенса, ReLU).

2 Исходный код

В рамках выполнения лабораторной работы были поставлены следующие цели:

1) Нужно осуществлять классификацию на 43 класса (33 буквы и 10 цифр).

2) Для этого будет реализовываться однослойный перцептрон с 43 нейронами: каждый нейрон отвечает за свой класс.

3) Эвристика подготовки датасета заключается в том, что метки изображений кодируются с помощью простейшего one-hot (это имеет значение в интерпретации результата).

4) В качестве функции потерь выбрана MSE, в качестве метрики выбрана точность.

Исходный код программы представлен в листингах 1- 2

Листинг 1 — Подготовка датасета

```
def create_image_array(text, font_path, image_size, font_size):
    image = Image.new('1', image_size, color='white')
    draw = ImageDraw.Draw(image)
    font = ImageFont.truetype(font_path, font_size)

    text_bbox = draw.textbbox((0, 0), text, font=font)
    text_width = text_bbox[2] - text_bbox[0]
    text_height = text_bbox[3] - text_bbox[1]

    x = (image_size[0] - text_width) / 2
    y = (image_size[1] - text_height) / 2

    draw.text((x, y), text, fill='black', font=font)

    image_array = np.array(image)
    return image_array



def generate_dataset(fonts_folder_train, fonts_folder_test, font_size):
    image_size = (5 * font_size, 4 * font_size)
    images_train = []
    images_test = []
    labels_train = []
    labels_test = []

    for char in alphabet:
        for font_file in os.listdir(fonts_folder_train):
            font_path = os.path.join(fonts_folder_train, font_file)
            if font_file.endswith(('.ttf', '.otf')):
                image_array = create_image_array(char, font_path,
    image_size, font_size)
                images_train.append(image_array)
                labels_train.append(classes_to_idx[char])


        for font_file in os.listdir(fonts_folder_test):
            font_path = os.path.join(fonts_folder_test, font_file)
            if font_file.endswith(('.ttf', '.otf')):
                image_array = create_image_array(char, font_path,
    image_size, font_size)
                images_test.append(image_array)
                labels_test.append(classes_to_idx[char])

    return images_train, images_test, labels_train, labels_test
```

Листинг 2 — Реализация перцептрона

```
class Perceptron:
    def __init__(self, input_size, num_neurons, activation, activation_derivative):
        self.input_size = input_size
        self.activation = activation
        self.activation_derivative = activation_derivative
        self.weights = np.zeros(shape=(num_neurons, input_size + 1))

    def train(self, training_data, test_data, train_labels, test_labels, epochs, learning_rate):
        errors_train = []
        errors_test = []
        accuracy_train = []
        accuracy_test = []

        for j in range(epochs):
            running_error = 0
            run_acc = 0
            for i in range(len(training_data)):
                inputs = np.insert(training_data[i], 0, 1)
                label = train_labels[i]
                prediction = self.predict(inputs)
                error = prediction - label
                delta = error * self.activation_derivative(np.dot(self.weights, inputs))

                pred = np.argmax(prediction)
                if pred == np.argmax(label):
                    run_acc += 1
                running_error += MSE(prediction, label)

                self.weights -= learning_rate * np.outer(delta, inputs)
            running_error /= len(training_data)
            errors_train.append(running_error)
            accuracy_train.append(run_acc / len(training_data))

            running_error = 0
            run_acc = 0
            for i in range(len(test_data)):
                inputs = np.insert(test_data[i], 0, 1)
                label = test_labels[i]
                prediction = self.predict(inputs)

                pred  = np.argmax(prediction)
                if pred == np.argmax(label):
                    run_acc += 1
                running_error += MSE(prediction, label)

            running_error /= len(test_data)
            errors_test.append(running_error)
            accuracy_test.append(run_acc / len(test_data))

        return errors_train, errors_test, accuracy_train, accuracy_test

    def predict(self, inputs):
        summation = np.dot(self.weights, inputs)
        return self.activation(summation)
```

3 Результаты

Результаты приведены на рисунках 1- 8

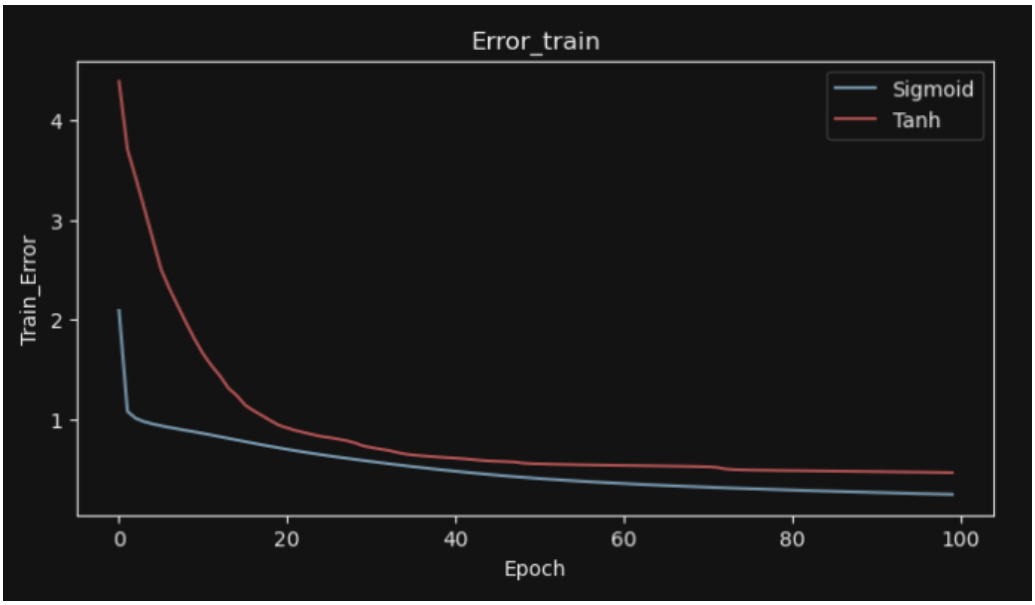

Рис. 1 — График зависимости фунции потерь от количества эпох для сигмоиды и гиперболического тангенса на обучающей выборке

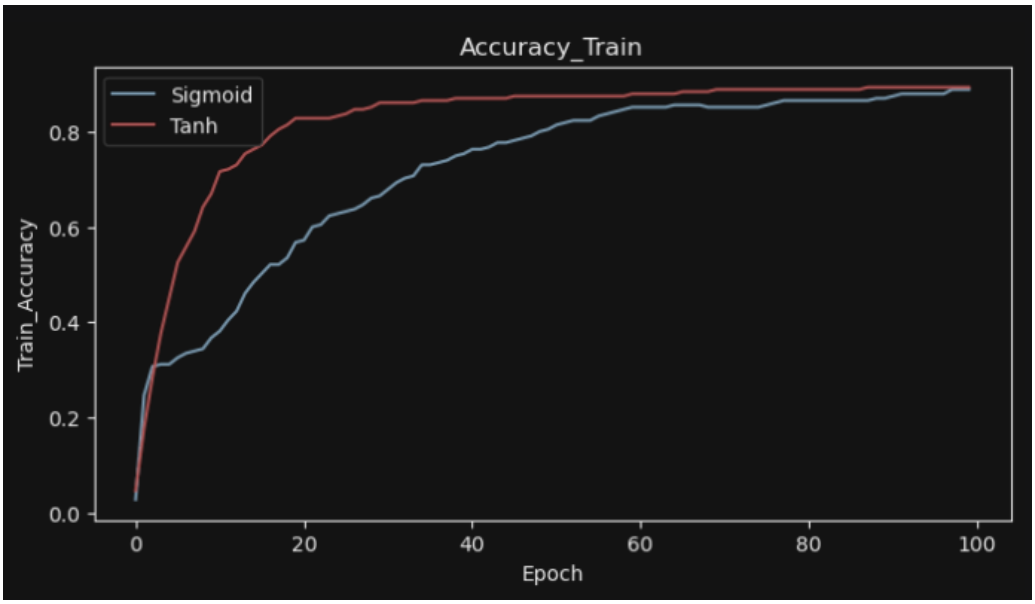

Рис. 2 — График зависимости точности от количества эпох для сигмоиды и гиперболического тангенса на обучающей выборке

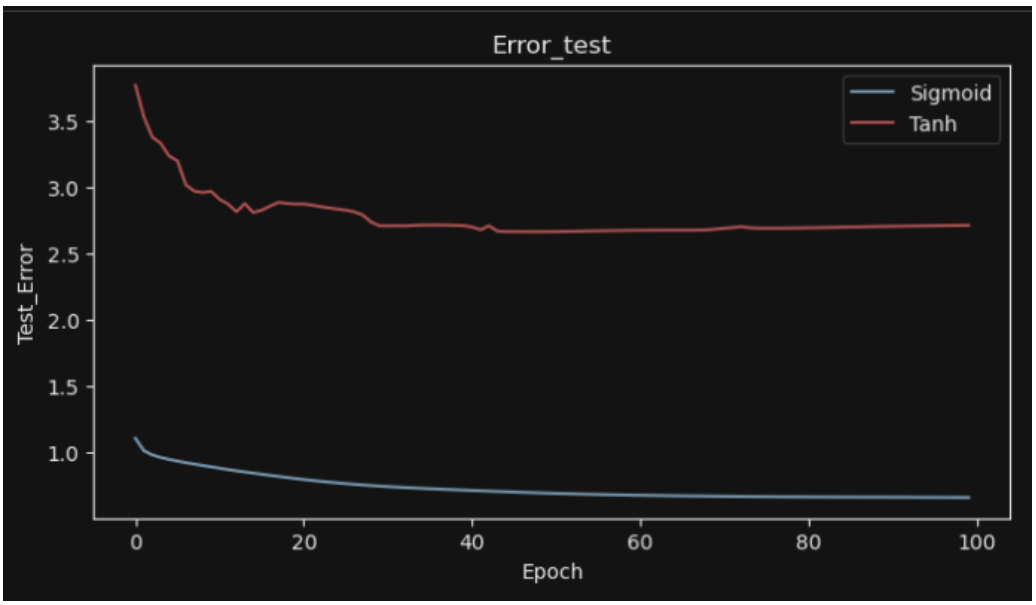

Рис. 3 — График зависимости фунции потерь от количества эпох для сигмоиды и гиперболического тангенса на тестовой выборке

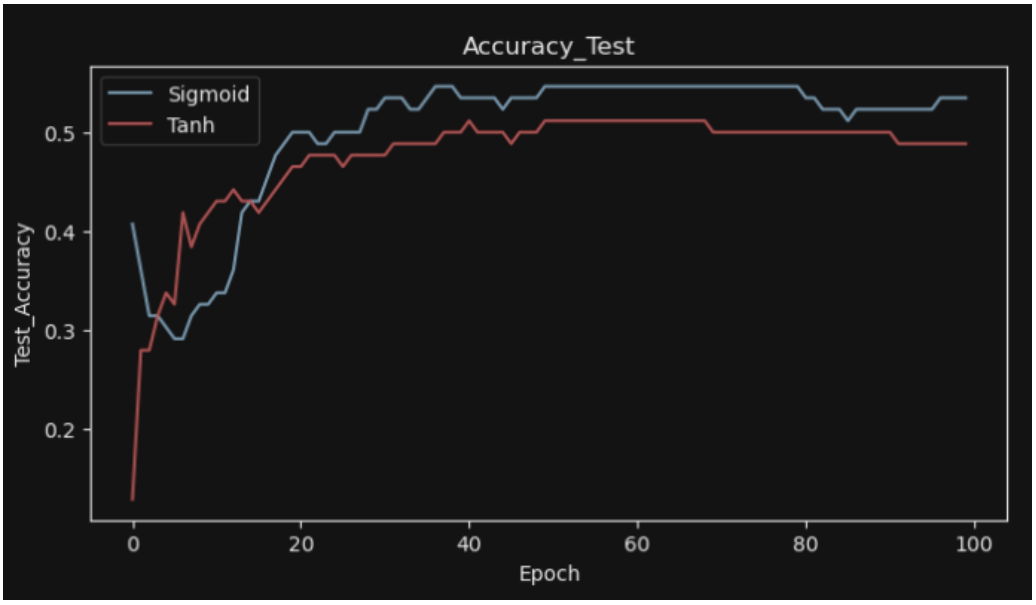

Рис. 4 — График зависимости точности от количества эпох для сигмоиды и гиперболического тангенса на тестовой выборке

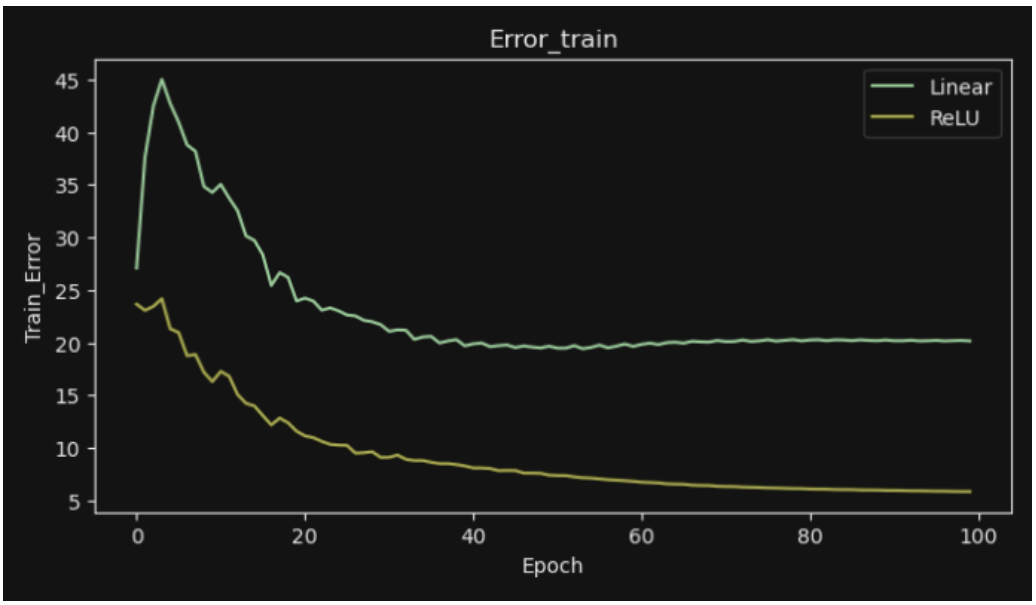

Рис. 5 — График зависимости фунции потерь от количества эпох для линейной функции и ReLU на обучающей выборке

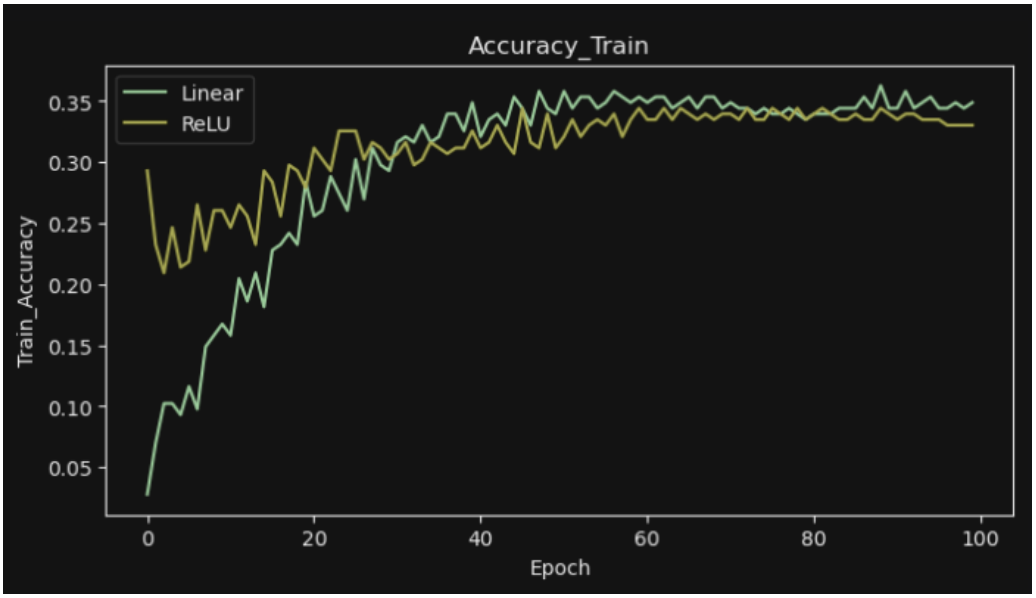

Рис. 6 — График зависимости точности от количества эпох для линейной функции и ReLU на обучающей выборке

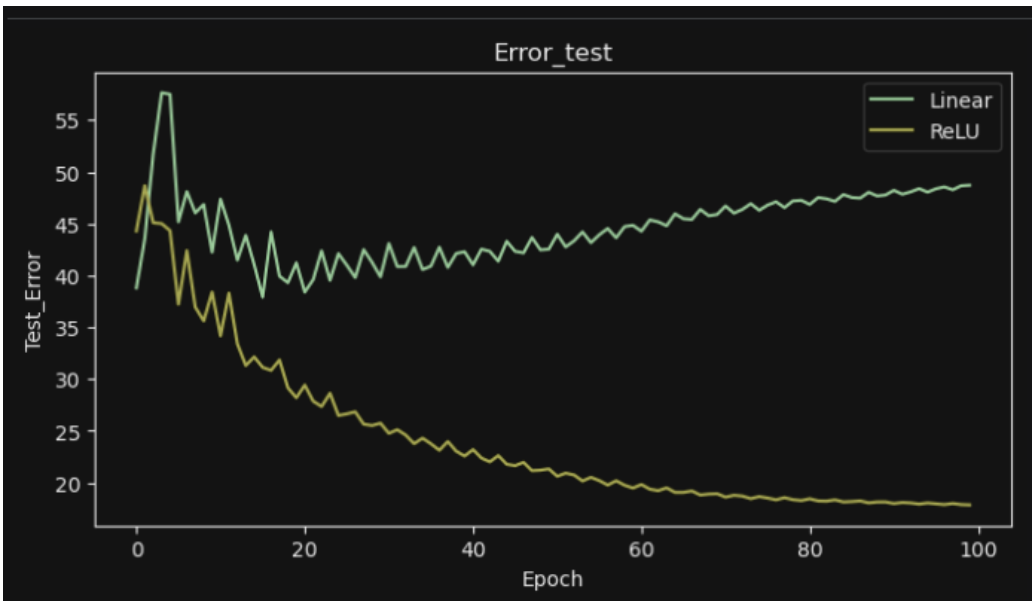

Рис. 7 — График зависимости фунции потерь от количества эпох для линейной функции и ReLU на тестовой выборке

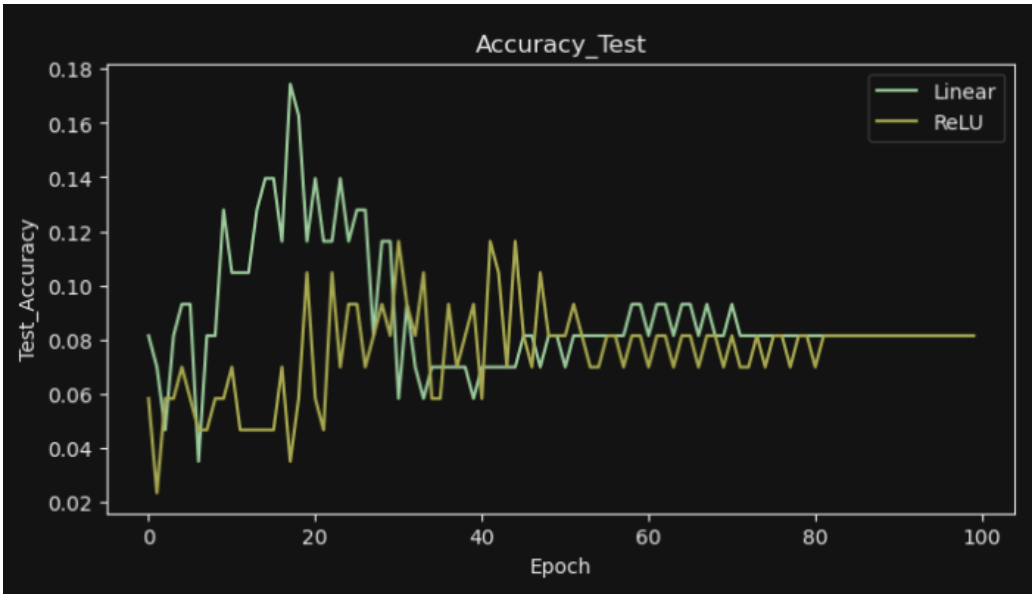

Рис. 8 — График зависимости точности от количества эпох для линейной функции и ReLU на тестовой выборке

По графикам, видно, что при использовании сигмоиды и гиперболического тангенса, модель обучается успешно и показывает удолетворительный результат с учетом простоты своей архитектуры. Но для линейной функции и ReLU результат очень плох. Это связано с тем, что сигмоида и гиперболический тангенс являются ограниченными сверху и снизу, что имеет определяющее значение с учетом введеных эвристик.

4 Выводы

В рамках данной лабораторной работы был вручную реализован однослойный перцептрон, решающий задачу классификации. В процессе интерпретации результатов было установлено, что очень важно, какие именно эвристики вводятся на этапе постановки задачи и подготовки данных, так как это имеет решающее значение в построении архитектуры модели, даже самой простой.