Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИУК «Информатика и управление»**

КАФЕДРА **ИУК5 «Информатика и вычислительная техника»**

Лабораторная работа №4

«Нейронные сети»

ДИСЦИПЛИНА: «Проектирование интеллектуальных систем»

Выполнил: студент гр. ИУК4-11М _____ (Сафронов Н.С.)
(подпись) (Ф.И.О.)

Проверил: _____ (Потапов А.Е.)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2024

Цель работы: формирование практических навыков построения, обучения и анализа моделей на базе нейронных сетей.

Задачи:

Спроектировать входы, выходы и целевую функцию нейронной сети. Выполнить обучение сетей с различными гиперпараметрами. Проиллюстрировать переобучение и недообучение модели.

Результаты выполнения работы

Листинг программы

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam


X, y = make_moons(n_samples=1000, noise=0.2, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)


def train_model(layers, epochs, batch_size, dropout_rate=0):
    model = Sequential()
    for i, units in enumerate(layers):
        if i == 0:
            model.add(Dense(units, activation='relu',
input_shape=(2,)))
        else:
            model.add(Dense(units, activation='relu'))
        if dropout_rate > 0:
            model.add(Dropout(dropout_rate))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(optimizer=Adam(learning_rate=0.01),
loss='binary_crossentropy', metrics=['accuracy'])
    history = model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_size, validation_data=(X_test, y_test), verbose=0)
    return model, history

model_under, history_under = train_model(layers=[2], epochs=10,
batch_size=32)

model_over, history_over = train_model(layers=[128, 128, 128],
epochs=300, batch_size=32)
```

```
model_optimal, history_optimal = train_model(layers=[64, 64],
epochs=100, batch_size=32, dropout_rate=0.3)

def plot_individual_history(history, title):
    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title(f'{title} - Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(history.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation
Accuracy')
    plt.title(f'{title} - Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.suptitle(title)
    plt.show()

plot_individual_history(history_under, 'Underfitting')
plot_individual_history(history_over, 'Overfitting')
plot_individual_history(history_optimal, 'Optimal')
```

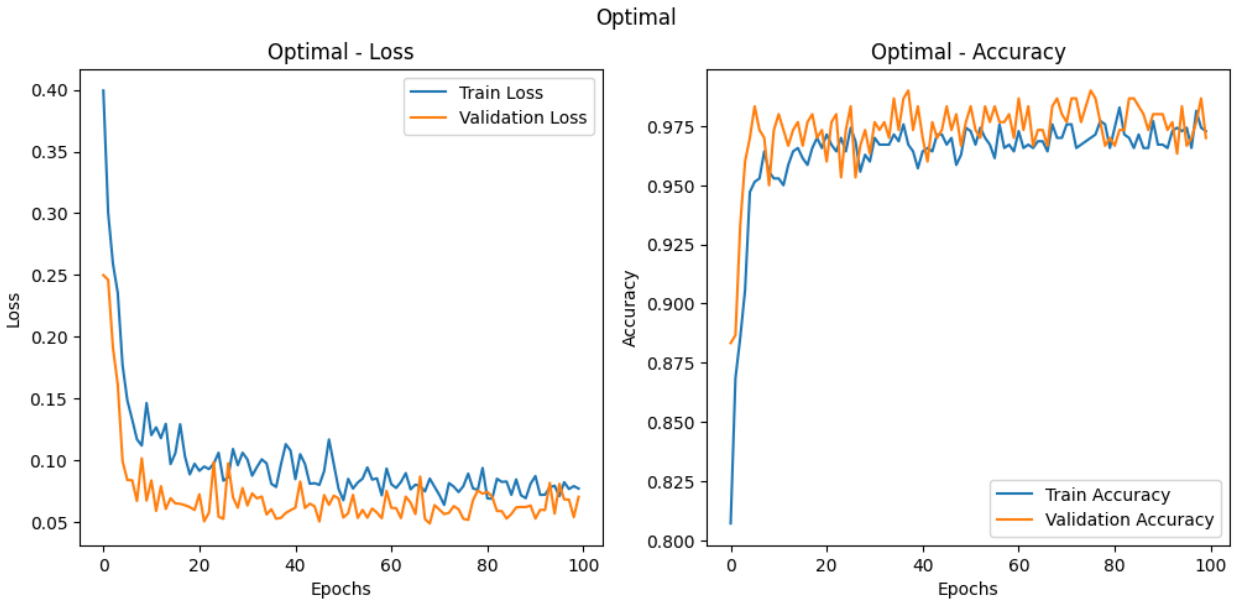

Рисунок 1 – Результирующая ошибка и точность для обученной модели

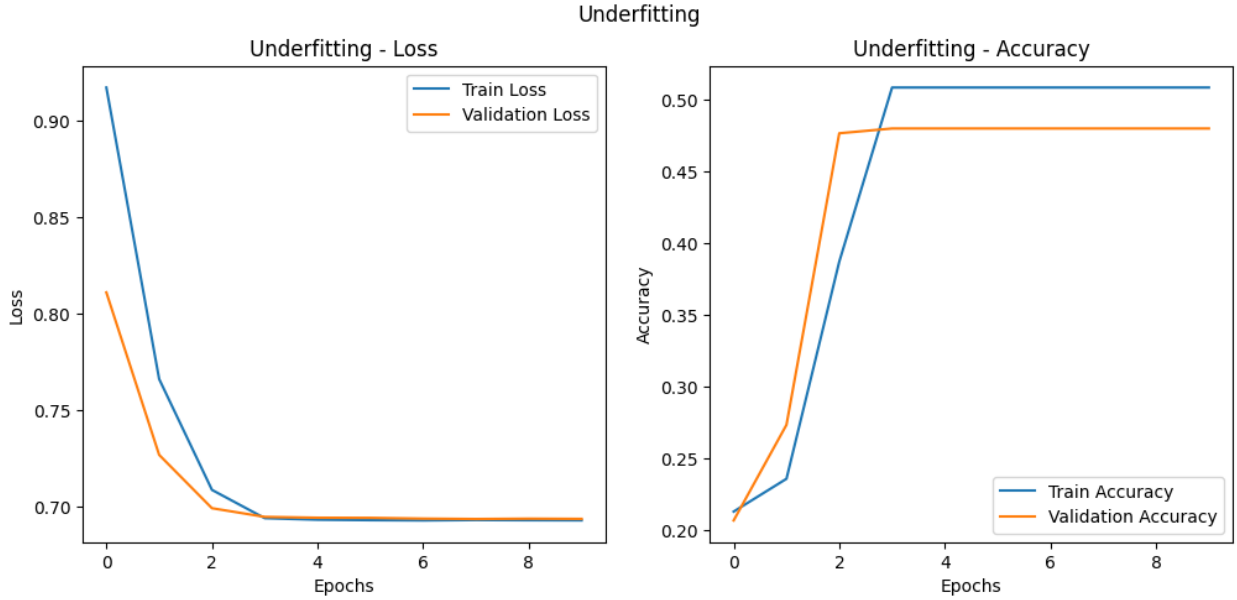

Рисунок 2 – Результирующая ошибка и точность для недообученной модели

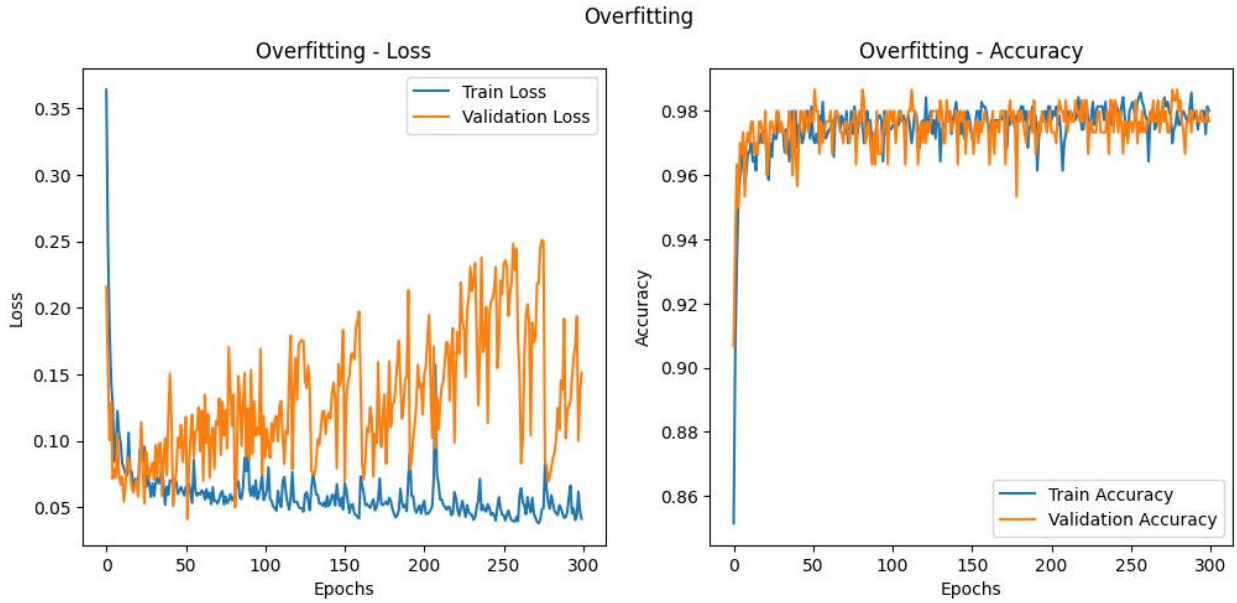

Рисунок 3 – Результирующая ошибка и точность для переобученной модели

Проектирование архитектуры нейронной сети:

Архитектура сети определяется типом задачи: классификация, регрессия, обработка текста или изображений. Например, в задаче классификации точек (как в нашем примере) достаточно нескольких плотных (Dense) слоев.

Выбор слоев и нейронов:

Для простых задач, как правило, достаточно небольшого числа слоев и нейронов, чтобы избежать переобучения. Увеличение числа слоев и нейронов позволяет сети извлекать сложные зависимости, но при этом повышает риск переобучения и требует больше данных.

Активационные функции:

Выбор активационной функции (например, ReLU или Sigmoid) важен для корректного обучения. ReLU часто используется в скрытых слоях, так как она помогает справляться с проблемой затухающих градиентов.

Подбор гиперпараметров:

Оптимальные гиперпараметры (learning rate, размер батча, количество эпох) нельзя предсказать заранее. Их подбор требует экспериментов.

Слишком большие значения (например, learning rate) могут привести к нестабильному обучению, а слишком маленькие — к долгому обучению или застреванию в локальных минимумах.

Регуляризация:

Добавление Dropout или L2-регуляризации помогает бороться с переобучением, особенно на малых наборах данных.

Эпохи:

Количество эпох должно быть сбалансировано: недостаток эпох приведет к недообучению, а их избыток — к переобучению.

Наблюдения за процессом обучения:

Недообучение наблюдалось, когда сеть имела слишком простую архитектуру (например, один слой с 2 нейронами) или обучалась на малом числе эпох**.** На графиках точность на обучающей и тестовой выборках была низкой, а ошибка почти не уменьшалась.

Переобучение возникало, когда сеть имела избыточную сложность (например, два скрытых слоя по 128 нейронов) или обучалась слишком долго.

На графиках точность на обучающей выборке была высокой, но на тестовой начинала снижаться, а ошибка на тестовой выборке росла.

Выводы:

Архитектура и гиперпараметры сети должны соответствовать сложности задачи и объему данных. Простая задача не требует сложной сети, и наоборот**.** Наблюдение за графиками точности и потерь во время обучения позволяет вовремя заметить признаки недообучения или переобучения и скорректировать гиперпараметры. Подбор гиперпараметров и архитектуры — это итеративный процесс, который требует понимания задачи, экспериментов и анализа результатов.

Вывод: в ходе работы сформированы практические навыки построения, обучения и анализа моделей на базе нейронных сетей.