МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Искусственные нейронные сети»
Тема: Многоклассовая классификация цветов

Студент гр. 8383 _____ Сосновский Д. Н.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2021

Цель работы.

Реализовать классификацию сортов растения ирис (Iris Setosa - 0, Iris Versicolour - 1, Iris Virginica - 2) по четырем признакам: размерам пестиков и тычинок его цветков.

Задачи.

- Ознакомиться с задачей классификации
- Загрузить данные
- Создать модель ИНС в Keras
- Настроить параметры обучения
- Обучить и оценить модель

Ход работы.

1. Загрузка файла с данными

Согласно методическим указаниям был скачан файл с данными iris.data и переименован в файл iris.csv

2. Написание программы

Была написана программа, которая создаёт модель, обучает её, и рисует графики ошибок и точности. Исходный код программы приведён в приложении. Графики ошибок и точности приведены на рис. 1 и рис.2

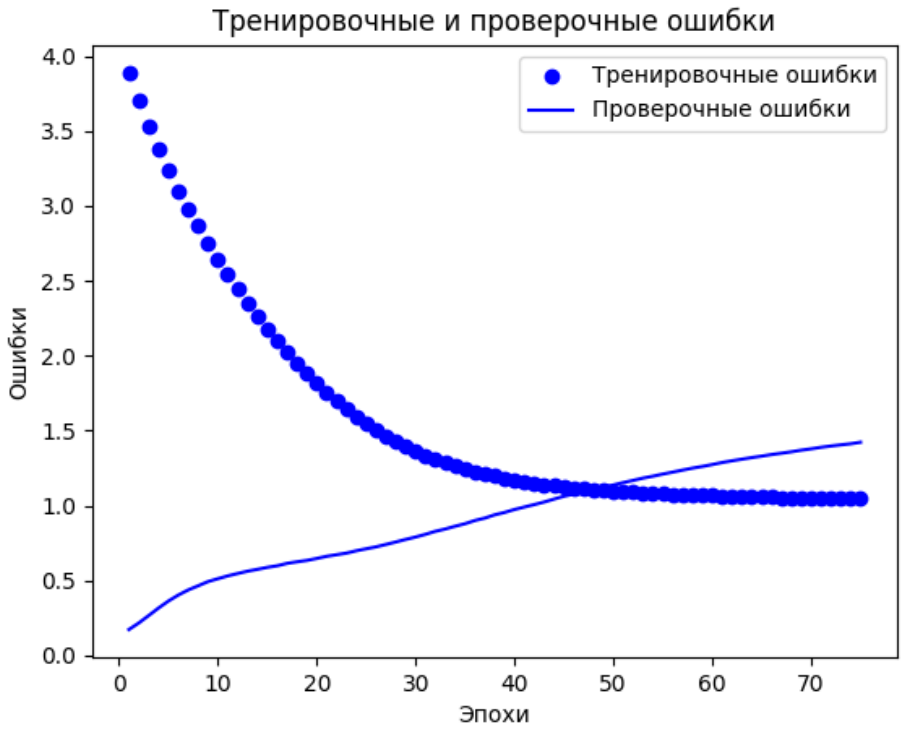

Рисунок 1 - тренировочные и проверочные ошибки

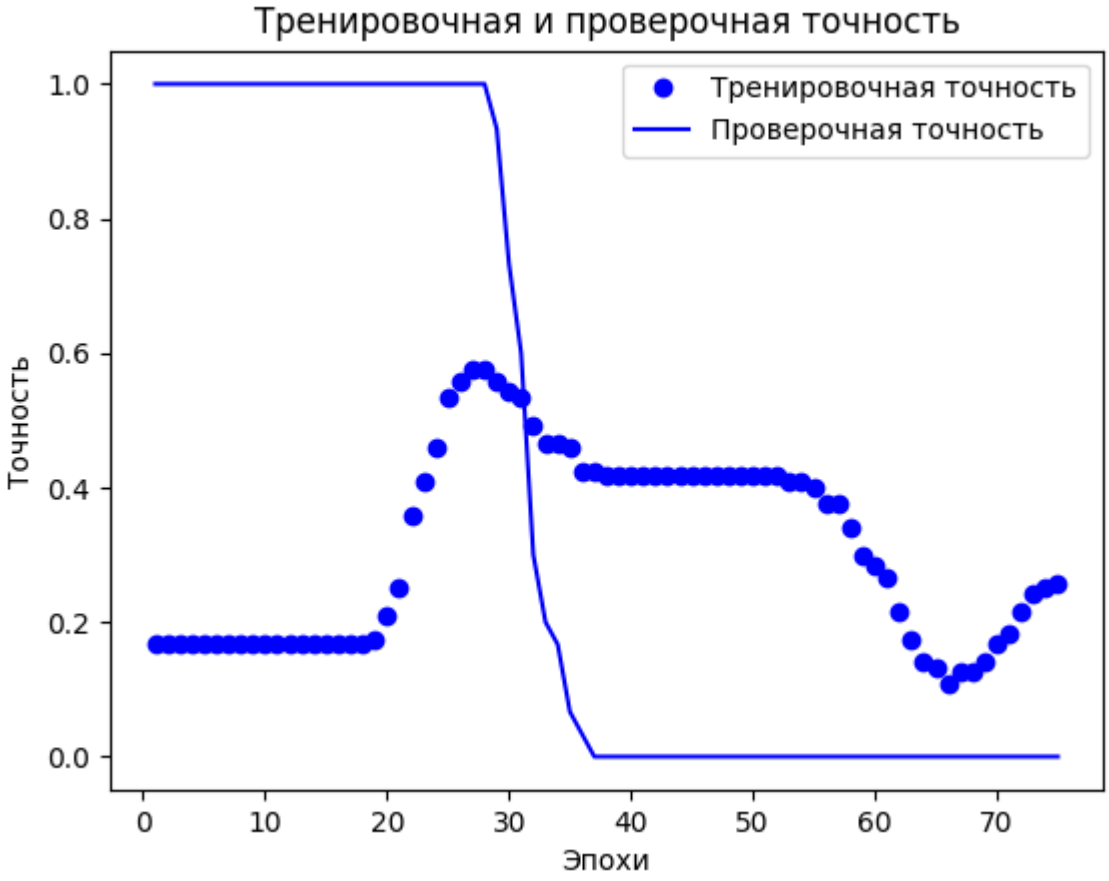

Рисунок 2 - тренировочная и проверочная точность

3. Изучение различных архитектур ИНС

Далее я изменил архитектуру нейронной сети. Сначала я увеличил количество слоёв до 3, добавив ещё один слой с 4-мя нейронами.

Листинг 1 – код с добавленным слоем

```
# Создание модели
model = Sequential()
model.add(Dense(4, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

Тренировочные и проверочные ошибки и точность при работе изменённой нейросети представлены на рисунках 3 и 4.

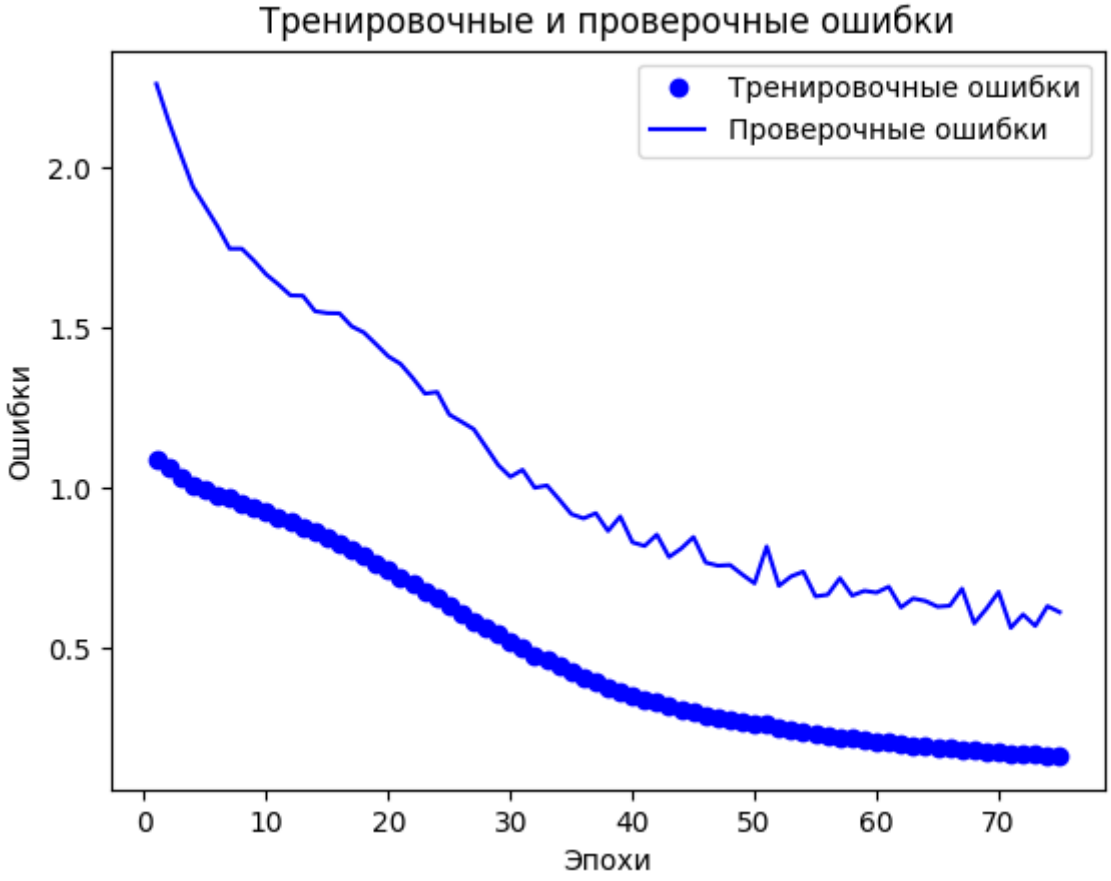

Рисунок 3 - тренировочные и проверочные ошибки

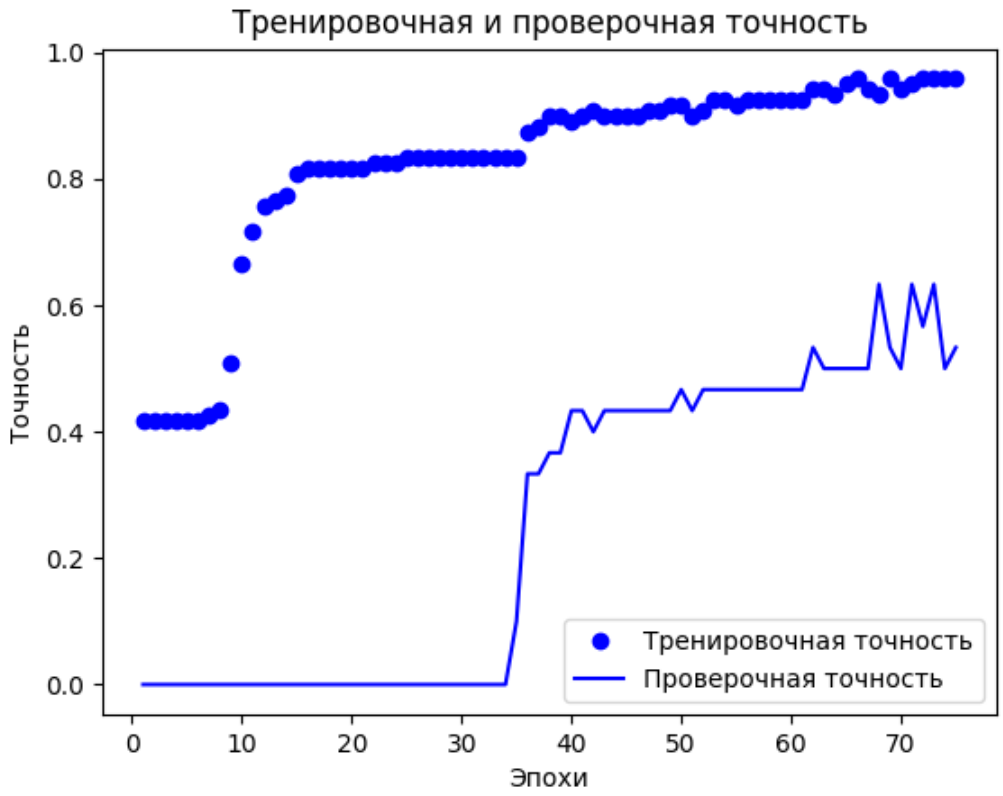

Рисунок 4 - тренировочная и проверочная точность

Видно, что ошибки уменьшились, а точность возросла. Далее было увеличено число нейронов во втором слое до 64.

Листинг 2 – измененное количество нейронов

```
# Создание модели
model = Sequential()
model.add(Dense(4, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

Графики ошибок и точности приведены на рисунках 5 и 6.

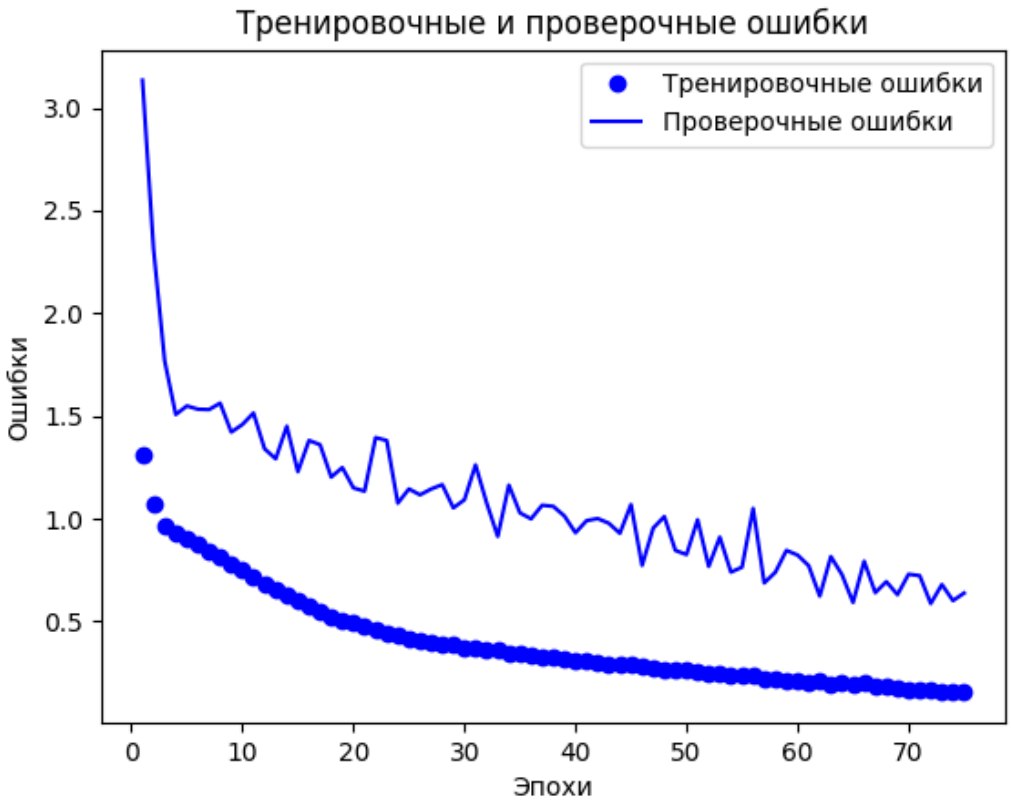

Рисунок 5 - тренировочные и проверочные ошибки

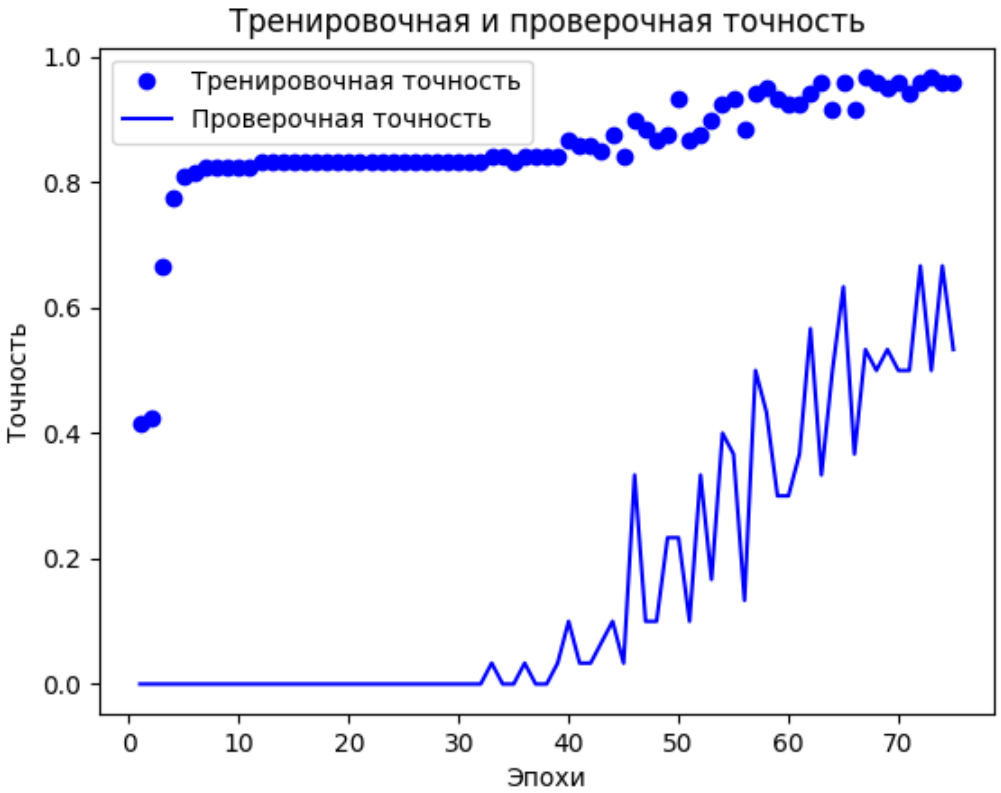

Рисунок 6 - тренировочная и проверочная точность

4. Изучение обучения при различных параметрах обучения (параметров функции fit)

Далее были изменены некоторые параметры функции fit. Для начала попробуем изменить число эпох с 75 до 100. Графики ошибок и точности приведены на рисунках 7 и 8.

Листинг 3 – измененное количество эпох

```
# Обучение сети
history = model.fit(X, dummy_y, epochs=100, batch_size=10, validation_split=0.1, verbose=False)
```

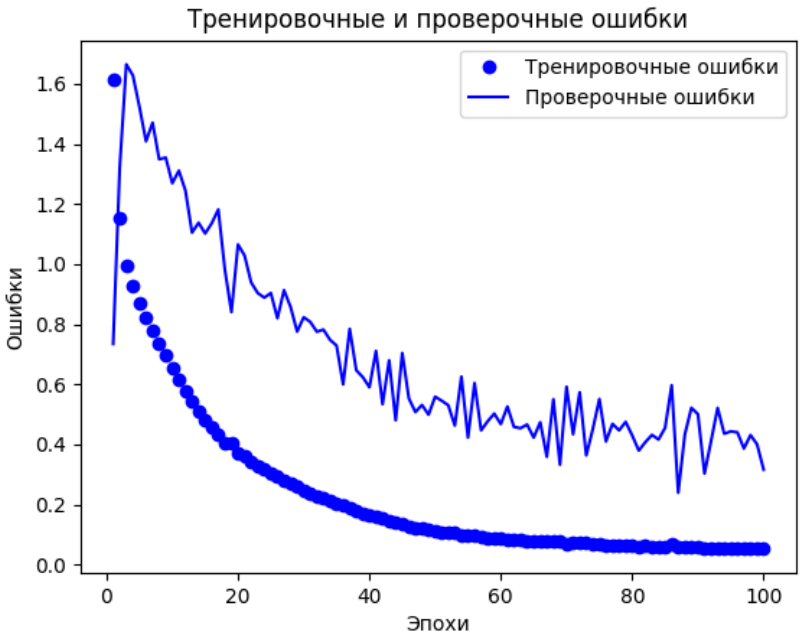

Рисунок 7 – тренировочные и проверочные ошибки

Рисунок 8 - тренировочная и проверочная точность

Как видно из графиков, проверочная точность растёт быстрее и сильнее. Далее изменим параметр batch_size с 10 на 20.

Листинг 4 – измененный batch_size

```
# Обучение сети
history = model.fit(X, dummy_y, epochs=100, batch_size=20, validation_split=0.1, verbose=False)
```

Графики ошибок и точности приведены на рисунках 9 и 10.

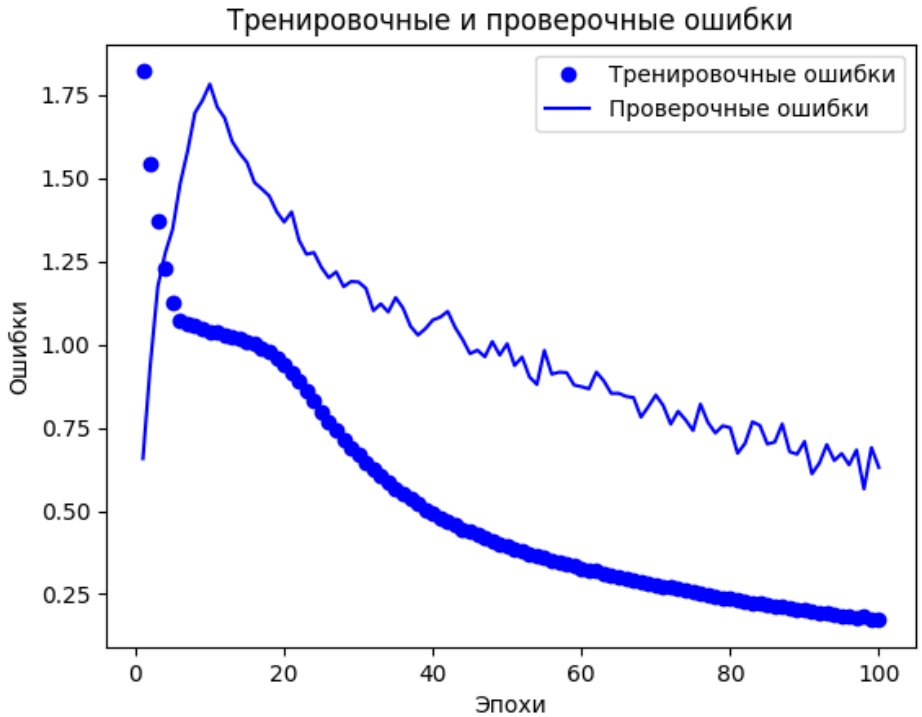

Рисунок 9 – тренировочные и проверочные ошибки

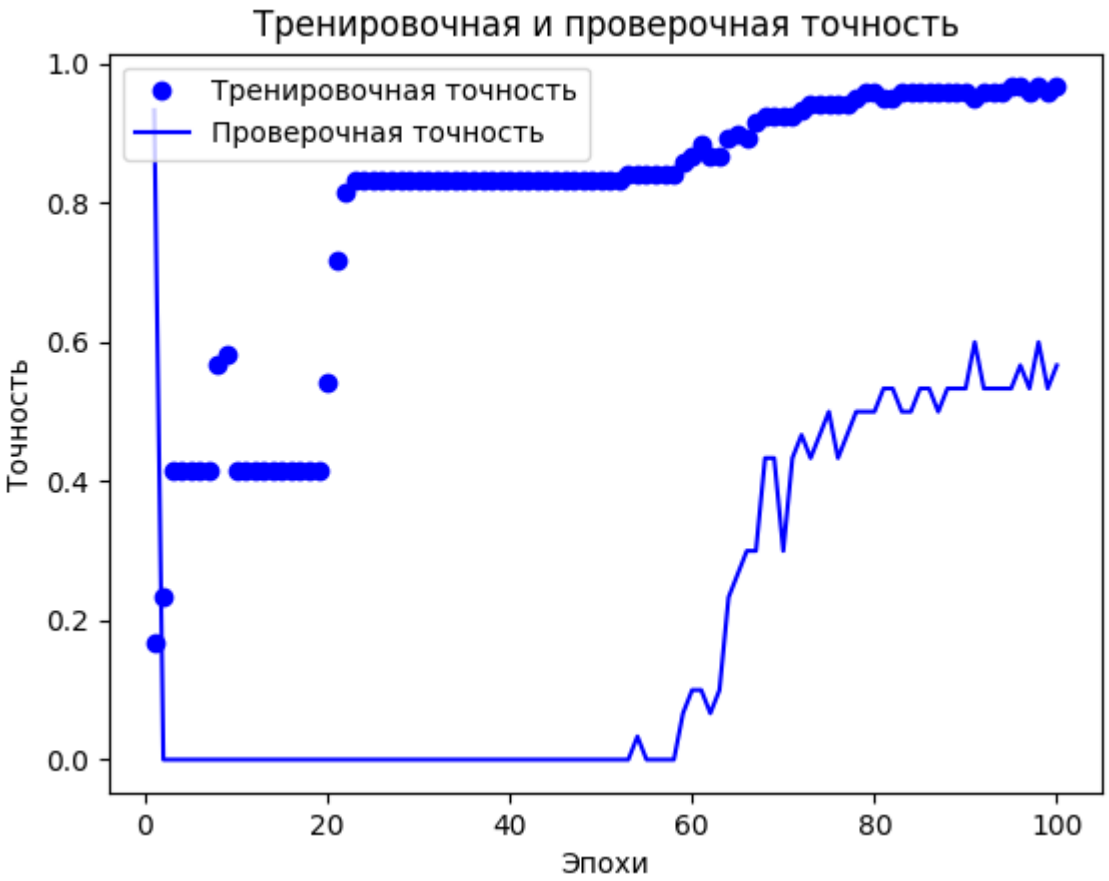

Рисунок 10 – тренировочная и проверочная точность

Из графиков видно, что результаты ухудшились т.к. размер данных для перерасчёта весов синапсов увеличился. Далее изменим параметр validation_split на 0.4. Графики ошибок и точности приведены на рисунках 11 и 12.

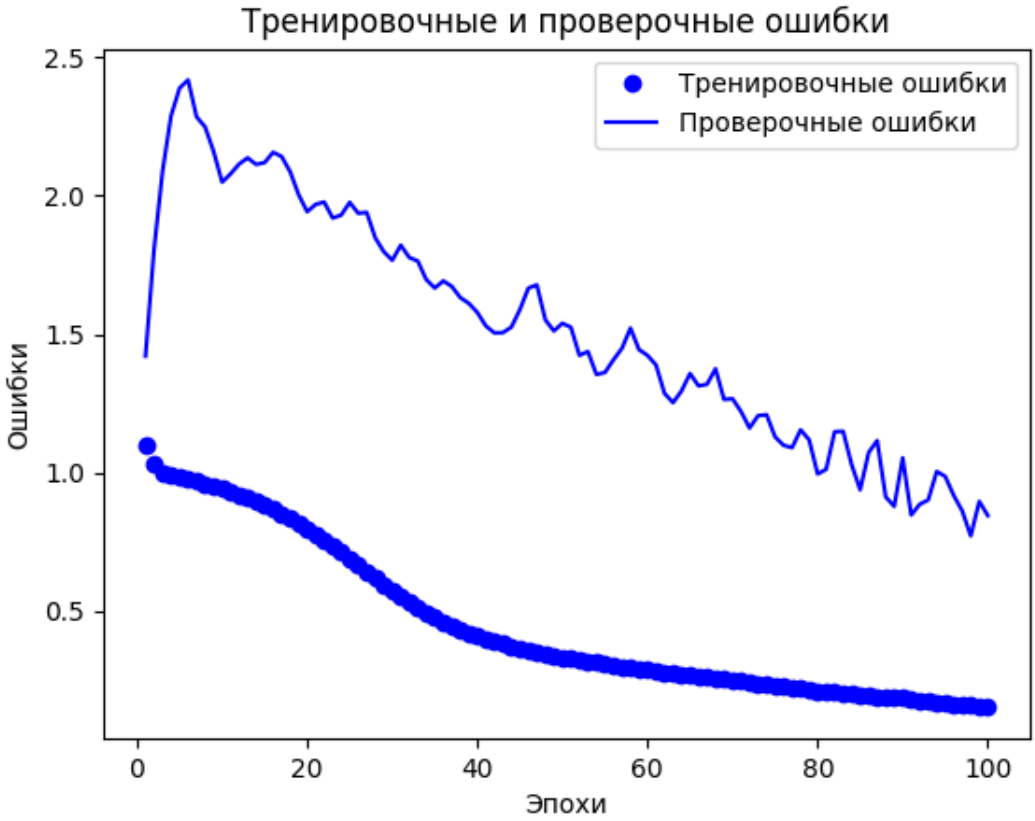

Рисунок 11 – тренировочные и проверочные ошибки

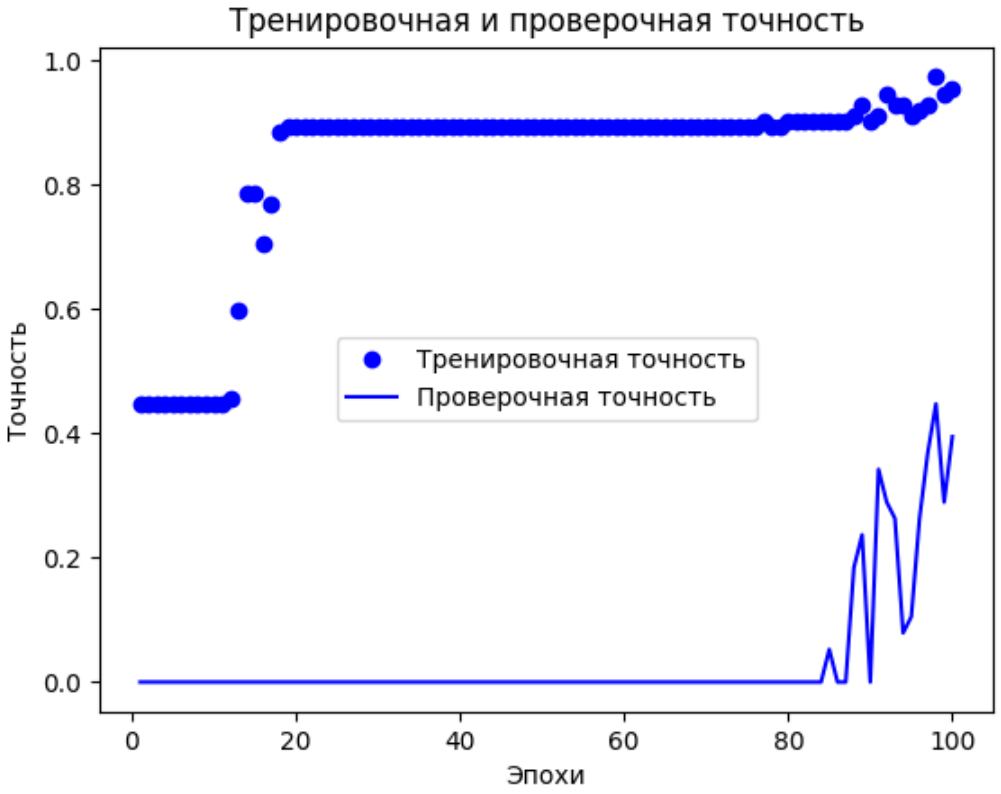

Рисунок 12 – тренировочная и проверочная точность

Видно, что тренировочные данные улучшились, а проверочные ухудшились.

5. Выбор наилучшей модели

После изучения параметров работы ИНС, были выведены следующие данные для наилучшей модели:

batch_size = 10

epochs = 75

validation_split = 0.1

В которой три слоя с количествами нейронов 4, 24, 3.

Вывод

В ходе выполнения данной лабораторной работы была реализована ИНС, которая выполняет классификацию растений по признакам. Были исследованы различные параметры построения архитектуры ИНС.

ПРИЛОЖЕНИЕ

Листинг – исходный код программы

```
import pandas
import numpy
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

epochs = 100

# Загрузка данных
dataframe = pandas.read_csv("iris.csv", header=None)
dataset = dataframe.values
X = dataset[:, 0:4].astype(float)
Y = dataset[:, 4]

# Переход от текстовых меток к категориальному вектору
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_y = to_categorical(encoded_Y)

# Создание модели
model = Sequential()
model.add(Dense(4, activation='relu'))
model.add(Dense(3, activation='softmax'))

# Инициализация параметров обучения
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Обучение сети
history = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1,
verbose=False)

loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'bo', label='Тренировочные ошибки')
plt.plot(epochs, val_loss, 'b', label='Проверочные ошибки')
plt.title('Тренировочные и проверочные ошибки')
plt.xlabel('Эпохи')
plt.ylabel('Ошибки')
plt.legend()
plt.show()

plt.clf()
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
plt.plot(epochs, acc, 'bo', label='Тренировочная точность')
plt.plot(epochs, val_acc, 'b', label='Проверочная точность')
plt.title('Тренировочная и проверочная точность')
plt.xlabel('Эпохи')
plt.ylabel('Точность')
plt.legend()
plt.show()
```