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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Искусственные нейронные сети»

Тема: Многоклассовая классификация цветов

Студент гр. 8383 _____ Дейнега В. Е.

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.

Задание.

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

Требования.

- Изучить различные архитектуры ИНС (Разное кол-во слоев, разное кол-во нейронов на слоях)
- Изучить обучение при различных параметрах обучения (параметры ф-ций fit)
- Построить графики ошибок и точности в ходе обучения
- Выбрать наилучшую модель

Выполнение работы

1) Для выполнения лабораторной работы были импортированы все необходимые библиотеки, скачан и открыт файл с данными для анализа iris.data(csv).

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

dataframe = pandas.read_csv("iris.csv", header=None)
dataset = dataframe.values
```

Далее атрибуты данных были разделены на входные данные X и выходные данные Y, также выходные атрибуты были преобразованы в матрицу:

```
X = dataset[:,0:4].astype(float)
Y = dataset[:,4]
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_y = to_categorical(encoded_Y)
```

Была создана модель с 2 слоями, определена функция потерь, оптимизатор и метрика:

```
model = Sequential()
model.add(Dense(4, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
hist = model.fit(X, dummy_y, epochs=75, batch_size=10,
validation_split=0.1)
```

Для наглядности были построены графики ошибок и точности в ходе обучения:

```
hist_dict = hist.history
loss_values = hist_dict['loss']
val_loss_values = hist_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation
loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()


plt.clf()
acc_values = hist_dict['accuracy']
val_acc_values = hist_dict['val_accuracy']
plt.plot(epochs, acc_values, 'bo', label='Training acc')
plt.plot(epochs, val_acc_values, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Результаты тестирования ИНС1 представлены на рис. 1 и 2.

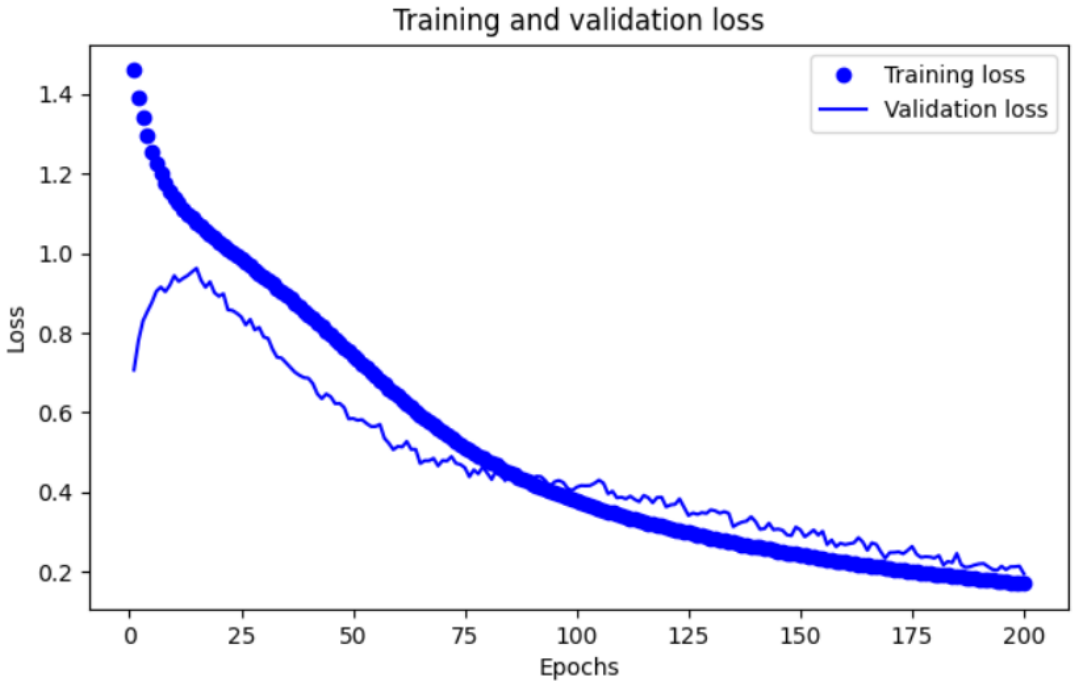

Рисунок 1 – График ошибки ИНС1

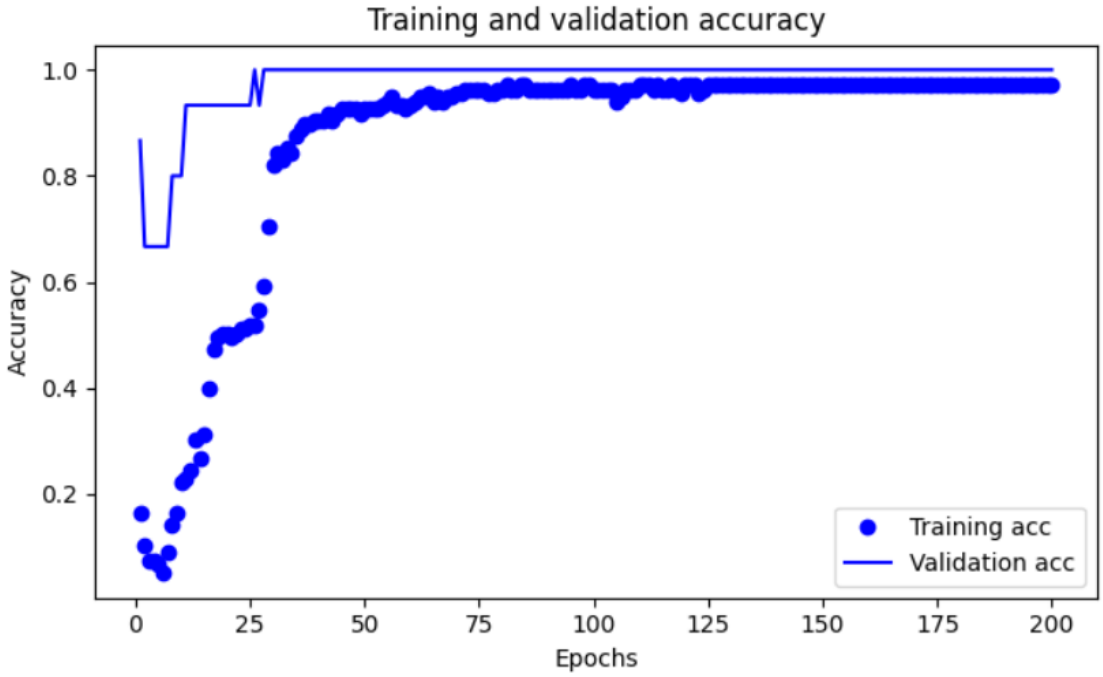

Рисунок 2 – График точности ИНС1

Стоит заметить, что от запуска к запуску графики значительно изменяются, что затрудняет сделать общий вывод о преимуществах и недостатках этой ИНС.

2) Увеличим количество нейронов во входном слое.

ИНС2 с 8 нейронами во входном слое:

```
model.add(Dense(8, activation='relu', input_shape=(4,)))
model.add(Dense(3, activation='softmax'))
```

Результаты тестирования ИНС2 представлены на рис. 3, 4.

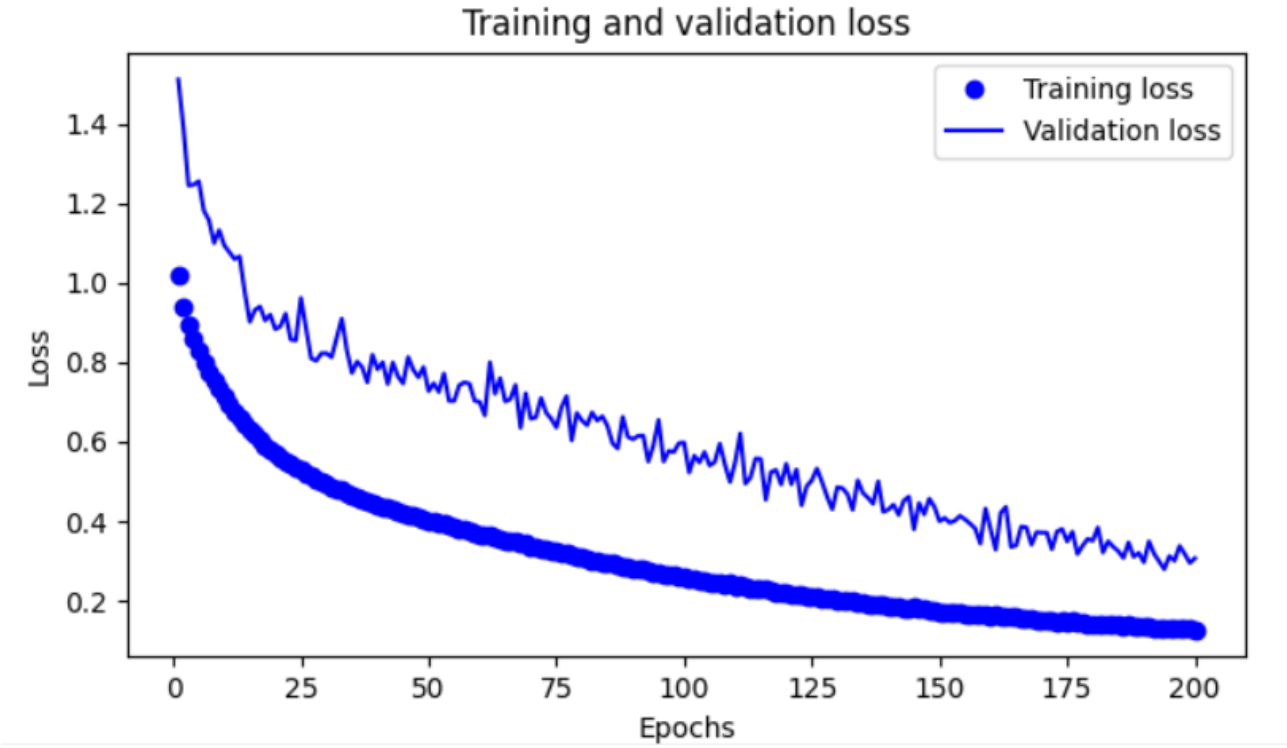

Рисунок 3 – График ошибки ИНС2

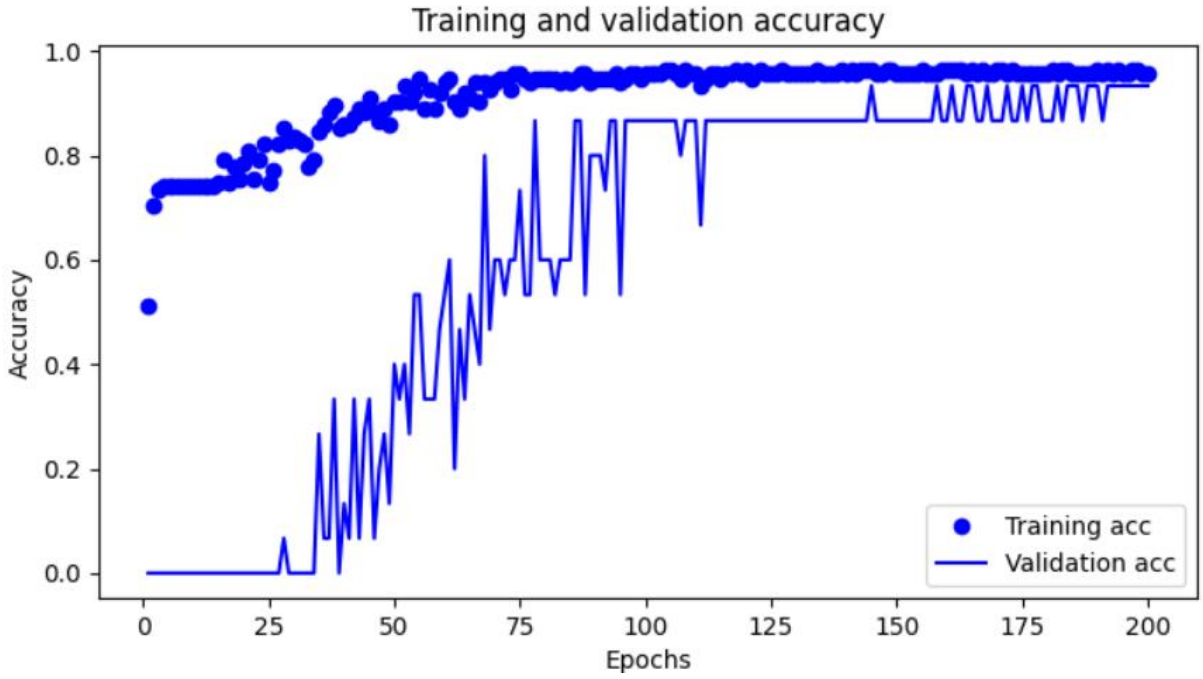

Рисунок 4 – График точности ИНС2

Полученные графики выглядят похоже на графики для ИНС1, ошибки на тренировочных данных уменьшились, на проверочных увеличились, та же картина наблюдается для точности.

3) Увеличим количество слоев в ИНС до трех.

На первом слое будет 4 нейрона, на втором 3 и на третьем тоже 3. Результат представлен на рис. 5 и 6.

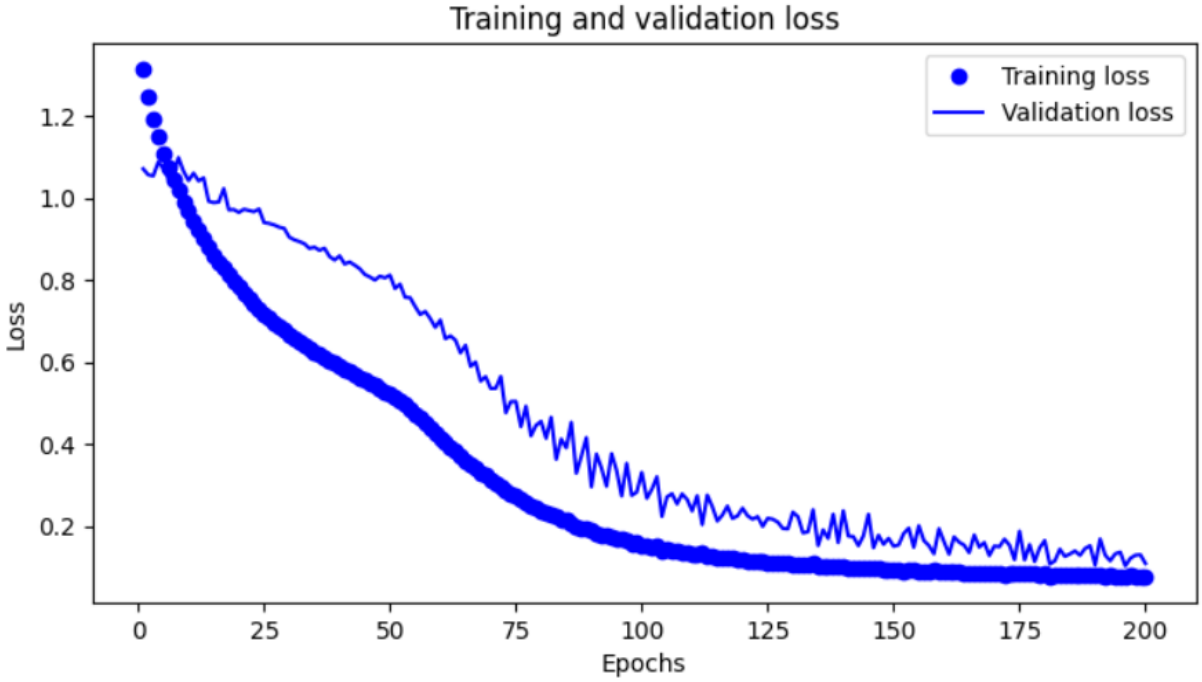

Рисунок 5 – График ошибки ИНС3

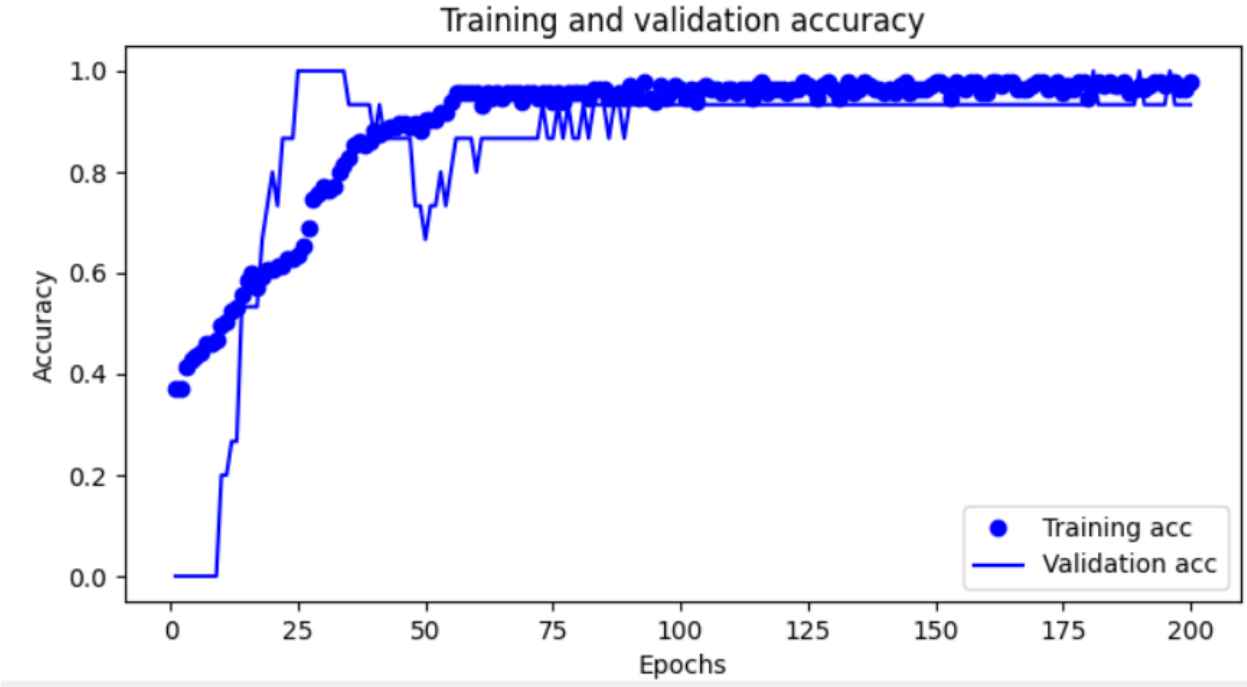

Рисунок 6 – График точности ИНС3

На графиках видно, что заметно уменьшилось число ошибок и увеличилась точность, также обучение ИНС происходит быстрее

Увеличим количество нейронов на входном уровне до 8 штук, на скрытом уровне до 5. Результаты работы ИНС4 представлены на рис. 7, 8.

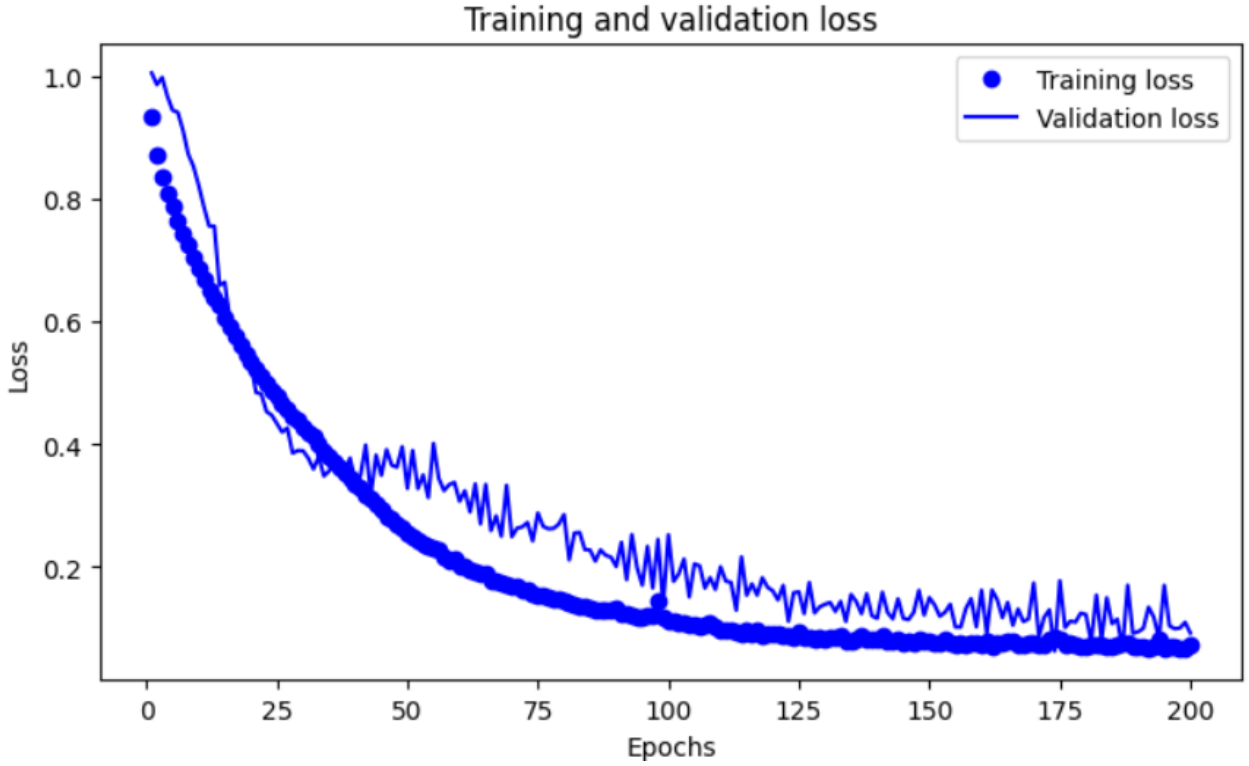

Рисунок 7 – График ошибки ИНС4

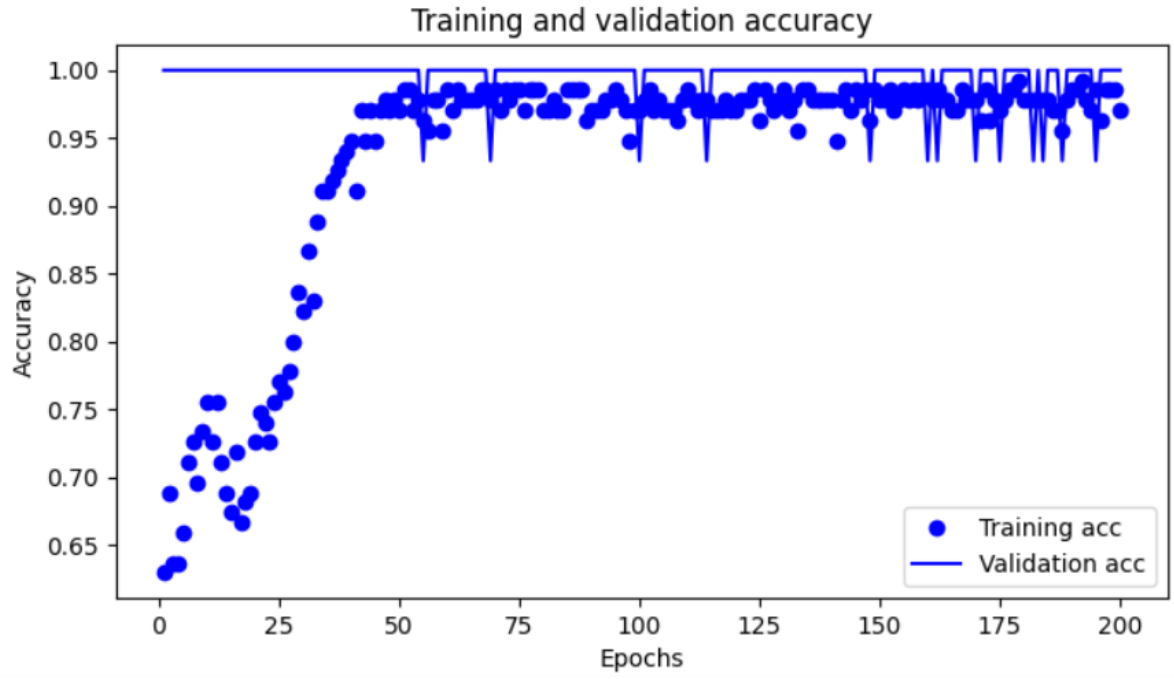

Рисунок 8 – График точности ИНС4

На графиках наблюдается уменьшение ошибок и увеличение точности. ИНС4 обучается быстрее ИНС3, высокая точность наступает раньше.

Увеличим количество нейронов на входном уровне до 16 штук, на скрытом уровне до 9. Результаты работы ИНС5 представлены на рис. 9, 10.

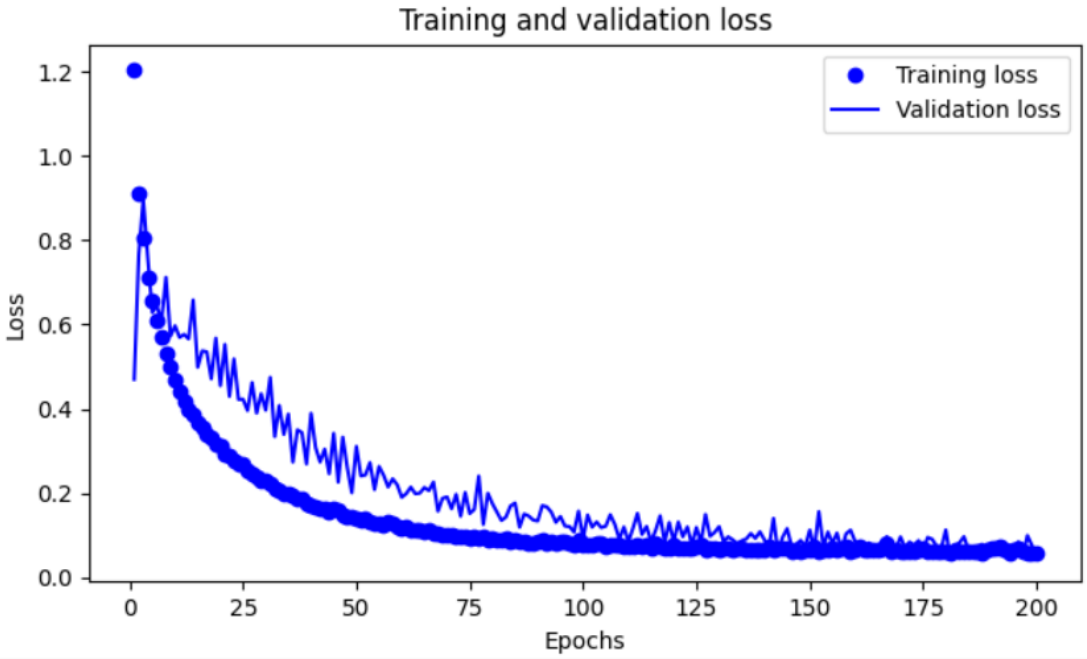

Рисунок 9 – График ошибки ИНС5

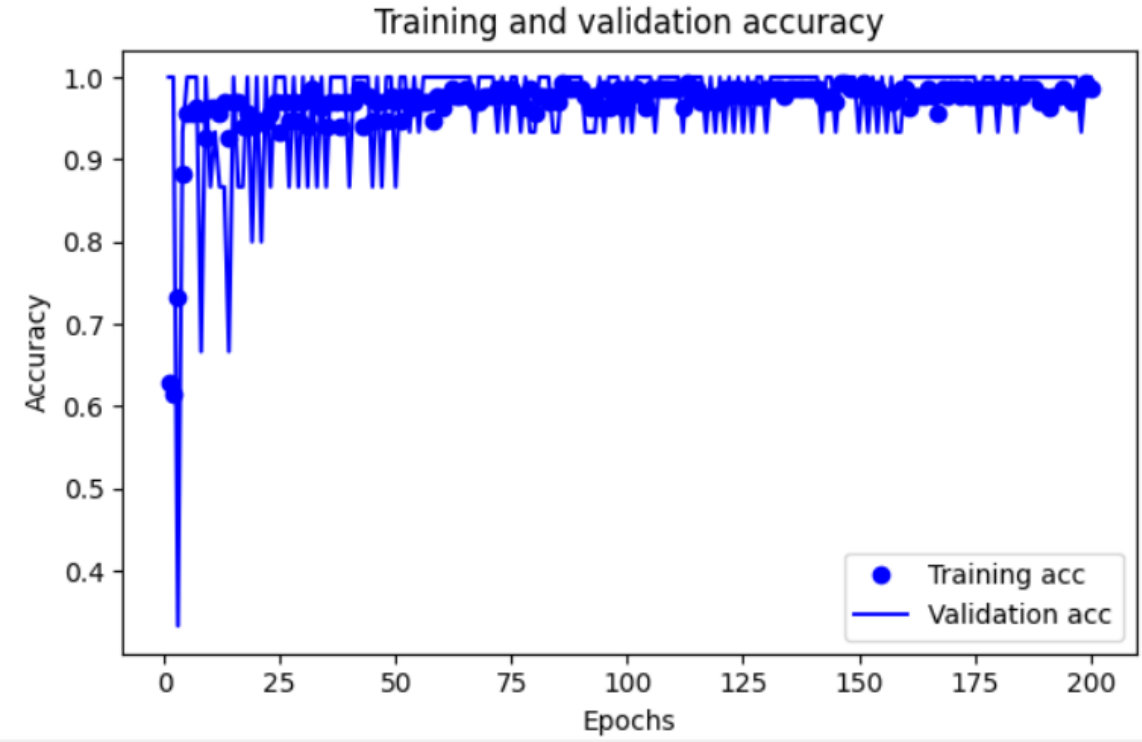

Рисунок 10 – График точности ИНС5

На графиках заметно, что ИНС5 обучается заметно быстрее ИНС4, однако максимальная точность ИНС5 не сильно отличается от ИНС4.

3) Рассмотрим различные параметры функции fit.

Установим параметр validation_split = 0.2 т.е. отношение данных обучения к данным тестирования будет 80 к 20. Результат представлен на рис. 11 и 12.

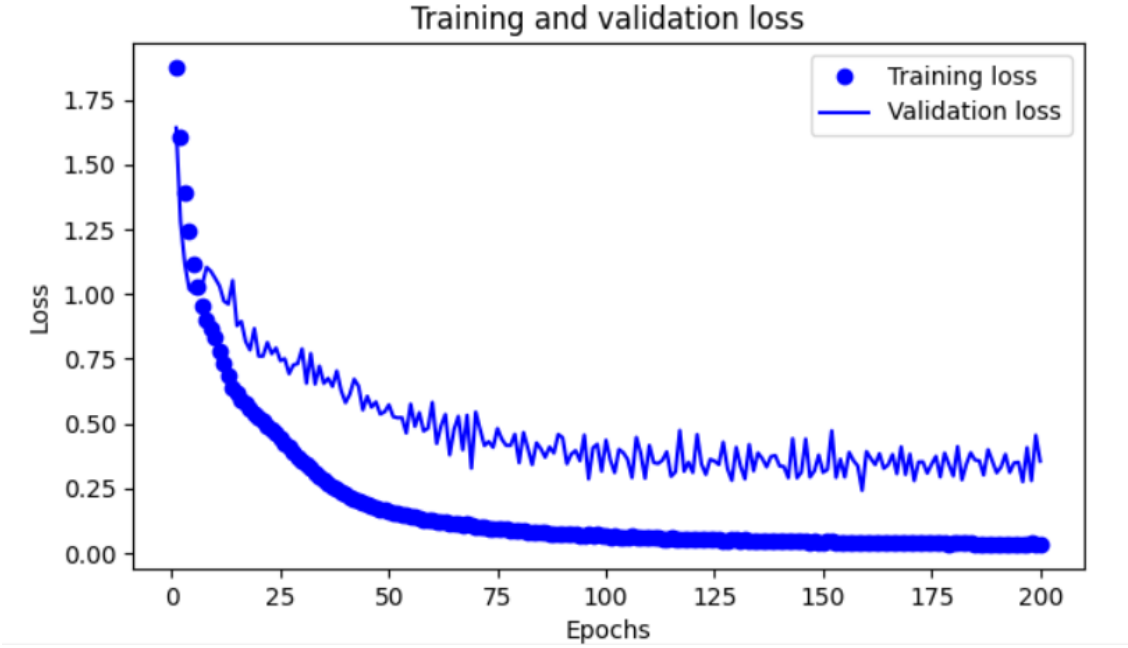

Рисунок 11 – График ошибки ИНС5 при validation_split = 0.2

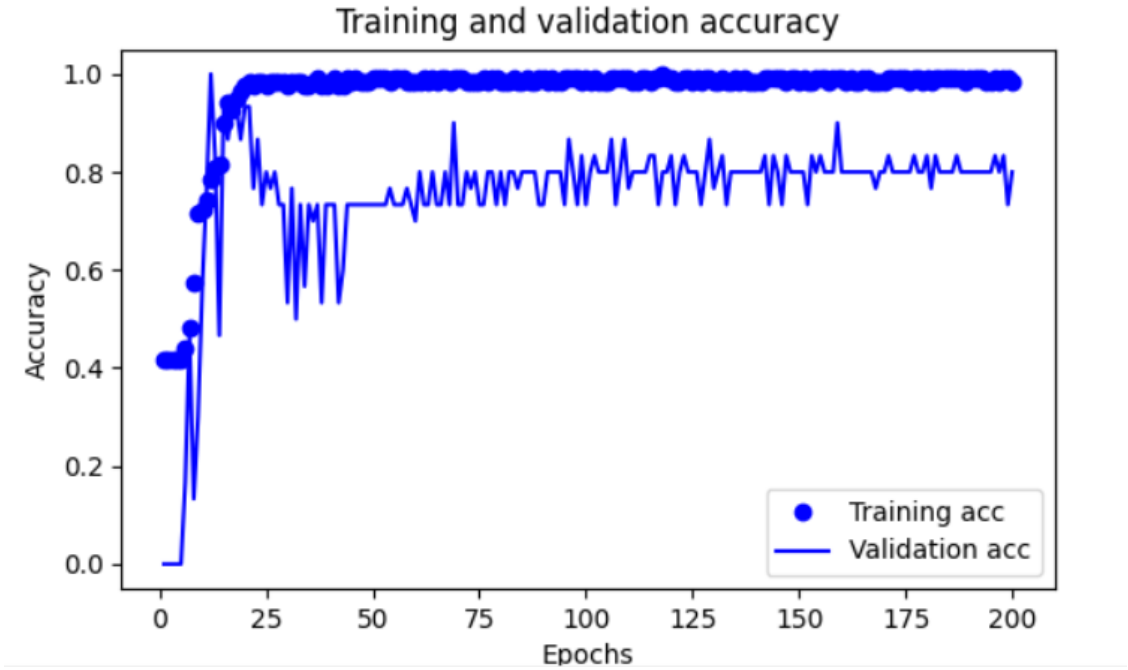

Рисунок 12 – График точности ИНС5 при validation_split = 0.2

На графиках видно, что ошибка для на данных для обучения не изменилась, однако сильно возросла на тестовых данных. Точность данных обучения осталась неизменной, но для тестовых данных заметно уменьшилась. Из этого можно сделать вывод, что оптимальное значение validation_split = 0.1

Установим batch_size в 5, batch_size – количество тренировочных объектов в батче, батч – некоторая часть всех данных для обучения. Результат работы представлен на рис. 13, 14.

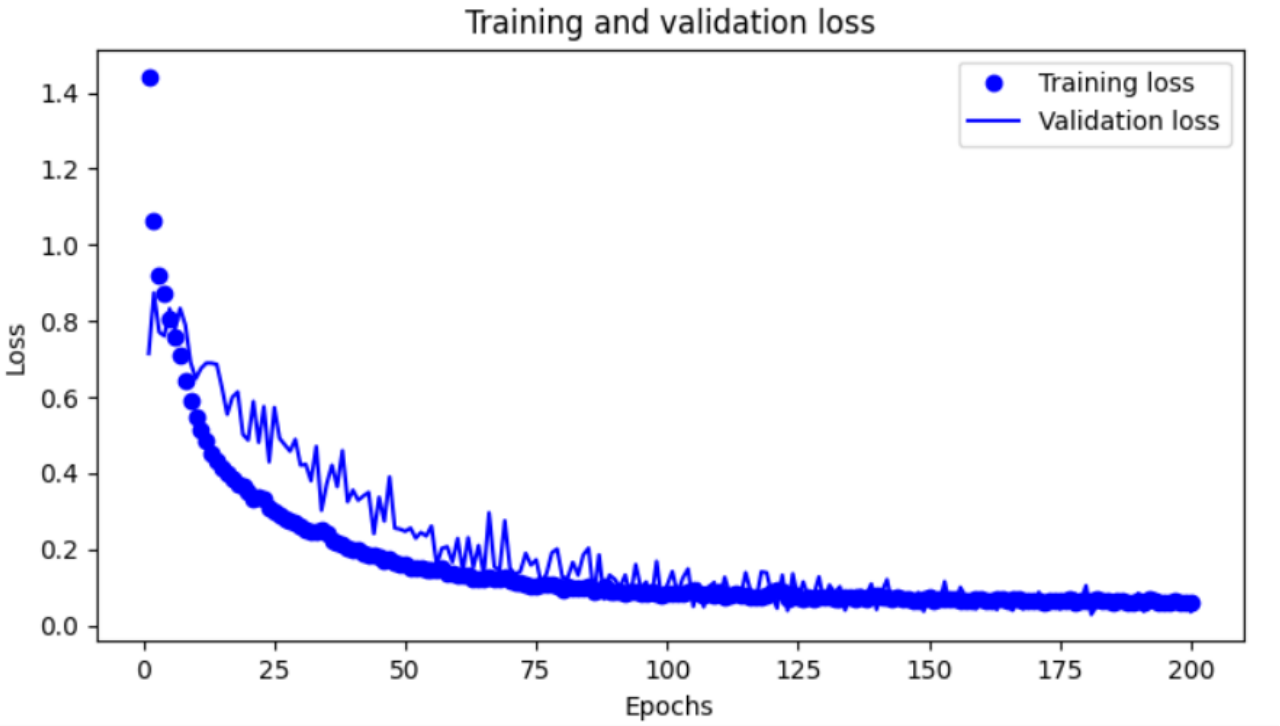

Рисунок 13 – График ошибки ИНС5 при batch_size = 5

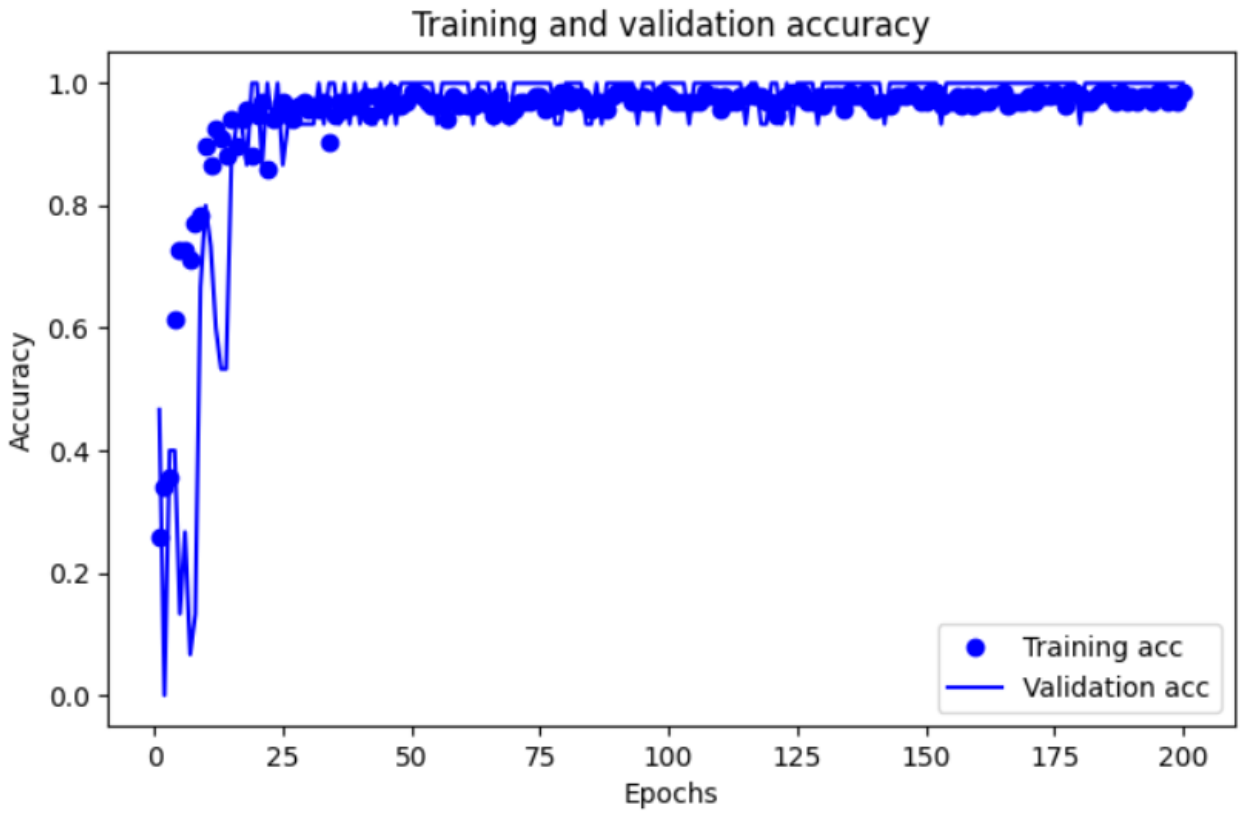

Рисунок 14 – График точности ИНС5 при batch_size = 5

Видимых изменений в графике ошибок (по сравнению с рис. 9) не произошло, однако график для точности стал более гладким, скачки точности обладают меньшей амплитудой. Оставим batch_size равным 5.

Установим количество эпох равным 1000. Прошла 1 эпоха означает, что все данные для обучения прошли черен нейронную сеть. Результаты работы ИНС5 с epochs = 1000 представлены на рис. 15 и 16.

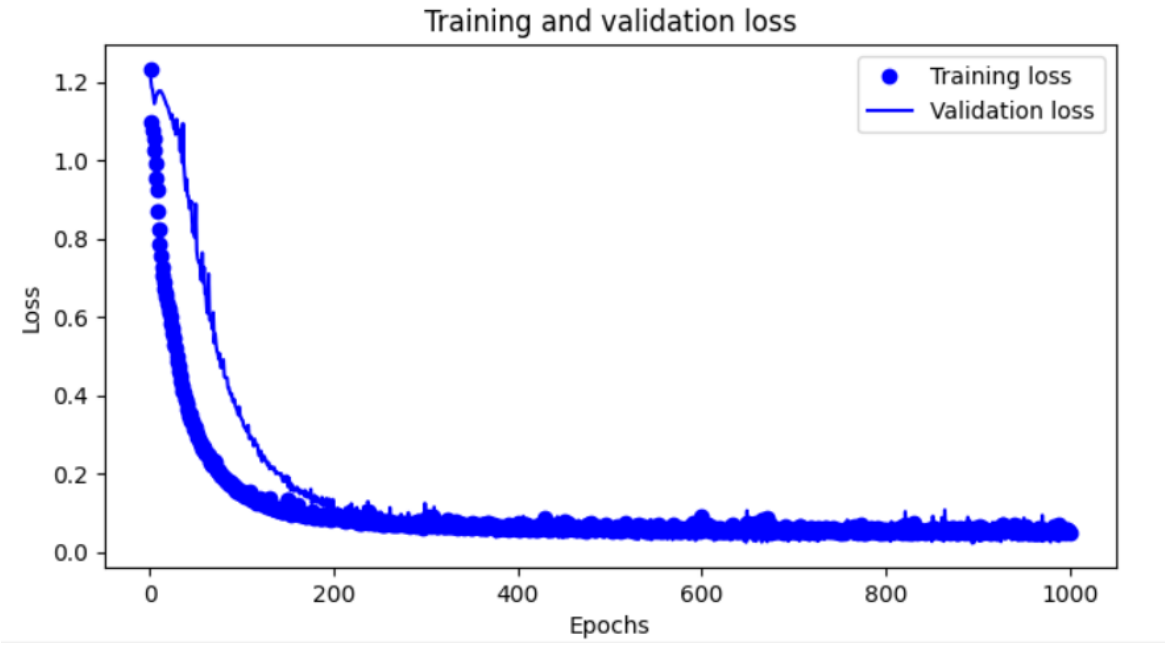

Рисунок 15 – График ошибки ИНС5 при epoch = 1000

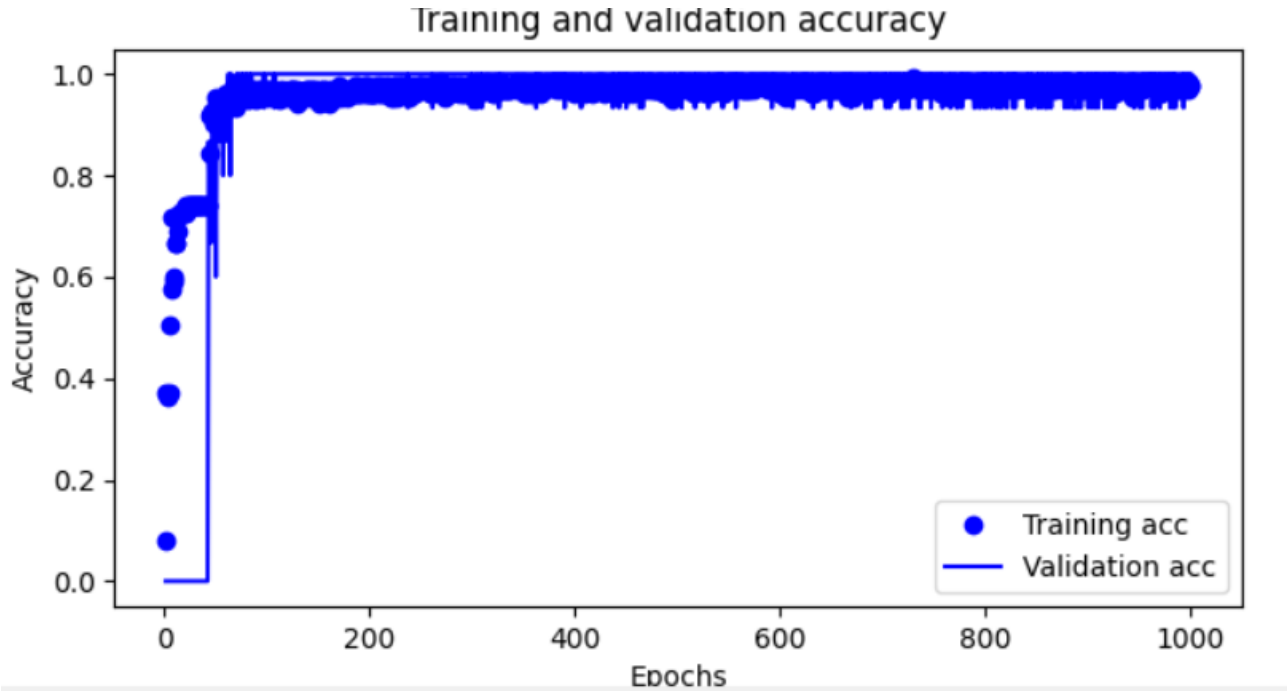

Рисунок 16 – График точности ИНС5 при epoch =1000

На графиках видно, что роста скорости обучения не произошло. За 1000 эпох точность возросла, а ошибка уменьшилась. Однако если поставить слишком большое количество эпох, может произойти переобучение и графики начнут расходится.

Оптимальная конфигурация ИНС – ИНС5 при batch_size = 5, epoch = 1000, validation_split = 0.1. Код представлен в приложении А.

Выводы.

В ходе лабораторной работы была реализована классификацию сортов растения ирис (Iris Setosa - 0, Iris Versicolour - 1, Iris Virginica - 2) по четырем признакам: размерам пестиков и тычинок его цветков. Также была определена оптимальная конфигурация ИНС для выполнения данной классификации.

Приложение А.

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

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

encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_y = to_categorical(encoded_Y)

model = Sequential()
#model.add(Dense(4, activation='relu'))
model.add(Dense(16, activation='relu', input_shape=(4,)))
model.add(Dense(9, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
hist = model.fit(X, dummy_y, epochs=1000, batch_size=5,
validation_split=0.1)

hist_dict = hist.history
loss_values = hist_dict['loss']
val_loss_values = hist_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation
loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()


plt.clf()
acc_values = hist_dict['accuracy']
val_acc_values = hist_dict['val_accuracy']
plt.plot(epochs, acc_values, 'bo', label='Training acc')
plt.plot(epochs, val_acc_values, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```