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

ОТЧЕТ

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

Студент гр. 8382 _____ Облизанов А.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель.

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

Требуется:

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

Выполнение работы.

Работа выполнялась на базе операционной системы Windows 10 в среде разработки PyCharm

1. Загрузка данных и создание ИНС.

Был скачан файл iris.data и переименован в iris.csv с исходными данными для анализа. Каждая строка данных содержит 4 параметра (числа с плавающей запятой) и класс цветка (строка). На основе параметров ИНС должна определять класс.

В программе были подключены необходимые библиотеки, загружены данные из файла, параметры цветков были помещены в вектор X, а классы

цветков в вектор Y, который далее был приведен к категориальному виду. Листинг приведен ниже:

```
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 = 500

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)
```

Далее была создана простая модель с двумя слоями: первый имеет 4 нейрона с функцией активации Relu (для обработки 4 входных параметров), второй – 3 нейрона с функцией активации Softmax (каждый нейрон определяет класс цветка на выходе). Далее были инициализированы параметры обучения и выполнено обучение сети. Число эпох установлено в 500. Листинг приведен ниже:

```
model = Sequential()
model.add(Dense(4, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
res = model.fit(X, dummy_y, epochs=epochs, batch_size=64, validation_split=0.1)
```

Для анализа данных и построение графиков была использована библиотека matplotlib. Был реализован вывод 4-х графиков: ошибки во время обучения, точность во время обучения, ошибки на проверяемых данных, точность на проверяемых данных (зависимость показателей от эпох). Листинг приведен ниже:

```
loss_history = numpy.array(res.history["loss"])
val_loss_history = numpy.array(res.history["val_loss"])
accuracy_history = numpy.array(res.history["accuracy"])
val_accuracy_history = numpy.array(res.history["val_accuracy"])
history = [loss_history, accuracy_history, val_loss_history,
val_accuracy_history]
titles = ["Loss", "Accuracy", "Val Loss", "Val Accuracy"]
ylables = ["loss", "accur"]

for i in range(4):
    plt.subplot(2, 2, i + 1)
    plt.title(titles[i])
    plt.xlabel("epoch")
    plt.ylabel(ylables[i % 2])
    axes = plt.gca()
    if i % 2:
        axes.set_ylim([0, 1.1])
    else:
        axes.set_ylim([0, 3])
    plt.grid()
    plt.plot([i for i in range(epochs)], history[i])

plt.show()
```

В дальнейшем программа модифицировалась для анализа других ИНС и их сравнения.

2. Анализ показателей различных ИНС

Слои ИНС №1 представлены в табл. 1.

Таблица 1 – ИНС №1 (2 слоя)

№	Ф-я активации	Кол-во нейронов
1	Relu	4
2	Softmax	3

Результаты тестирования ИНС №1 представлены на рис. 1.

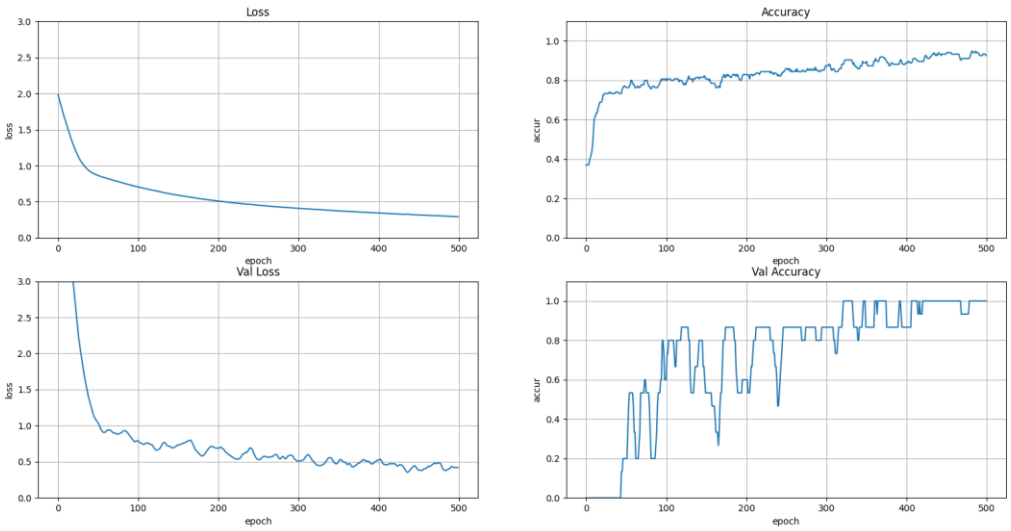

Рисунок 1 – Показатели ИНС №1

Из графиков видно, что в начале обучения ИНС №1 достаточно быстро (~50 эпох) уменьшает показатели ошибок (до 1 на обучаемых данных), однако они остаются достаточно высокими. В то же время точность за 50 эпох достигает 0.75 на обучаемых данных, а на проверочных стабильные результаты достигаются ближе к 300 эпохам.

В целом, ИНС медленно обучается и не достигает высокой точности и малых ошибок.

Было проведено сравнение функций активации Relu и Sigmoid при использовании в скрытом слое в ИНС с тремя слоями.

Слои ИНС №2 и ИНС №3 представлены в табл. 2.

ИНС №2 (синий)	Ф-я акт.	Кол-во нейр.	ИНС №3 (красный)	Ф-я акт.	Кол-во нейр.
	Relu	4		Relu	4
	Relu	64		Sigmoid	64
	Softmax	3		Softmax	3

Результаты тестирования ИНС №2 и №3 представлены на рис. 2 (ИНС №2 – синие линии, ИНС №3 – красные линии).

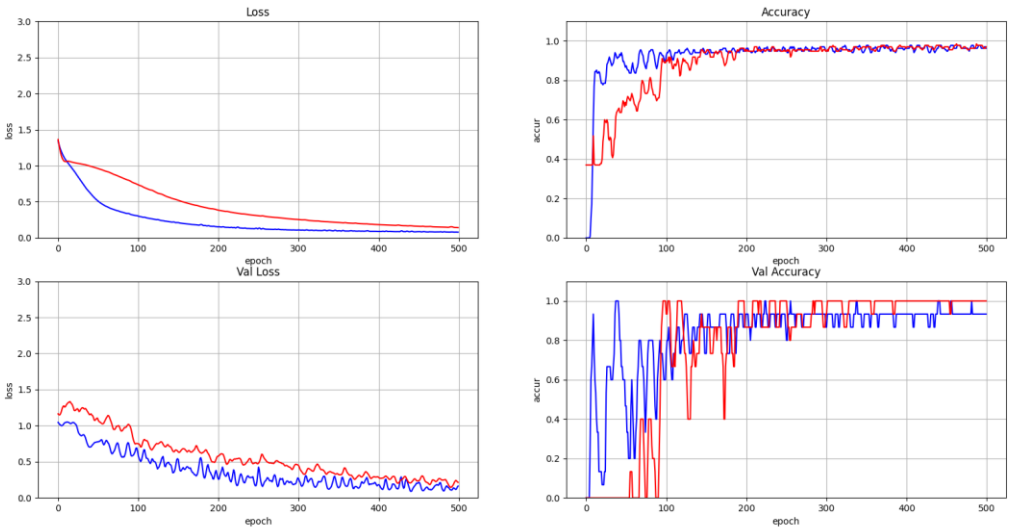

Рисунок 3 – Показатели ИНС №2 и №3

Из графиков видно, что добавление скрытого слоя значительно уменьшило значения ошибок и повысило точность.

ИНС №3 с функцией активации Sigmoid на скрытом слое нейронов обучается медленнее ИНС №2: стабильная точность достигается на 150 эпохе при данных для обучения и на 200 эпохе для данных, которых не было в обучении.

ИНС №2 с функцией активации Relu на скрытом слое нейронов обучается значительно быстрее: уже на 100 эпохе наблюдается точность около 0.95 на данных для обучения. Спустя 300 эпох ошибки на данных для обучения <0.1, для проверки <0.3.

Можно сделать вывод о том, что функция активации Relu является более оптимальной для данной задачи при использовании в скрытом слое в ИНС с тремя слоями, чем функция Sigmoid.

Было проведено то же сравнение, но с большим количеством нейронов в скрытом слое.

Слои ИНС №4 и ИНС №5 представлены в табл. 3.

Таблица 3 – ИНС №4 и №5

ИНС №4 (синий)	Ф-я акт.	Кол-во нейр.	ИНС №5 (красный)	Ф-я акт.	Кол-во нейр.
	Relu	4		Relu	4
	Relu	192		Sigmoid	192
	Softmax	3		Softmax	3

Результаты тестирования ИНС №4 и №5 представлены на рис. 3 (ИНС №4 – синие линии, ИНС №5 – красные линии).

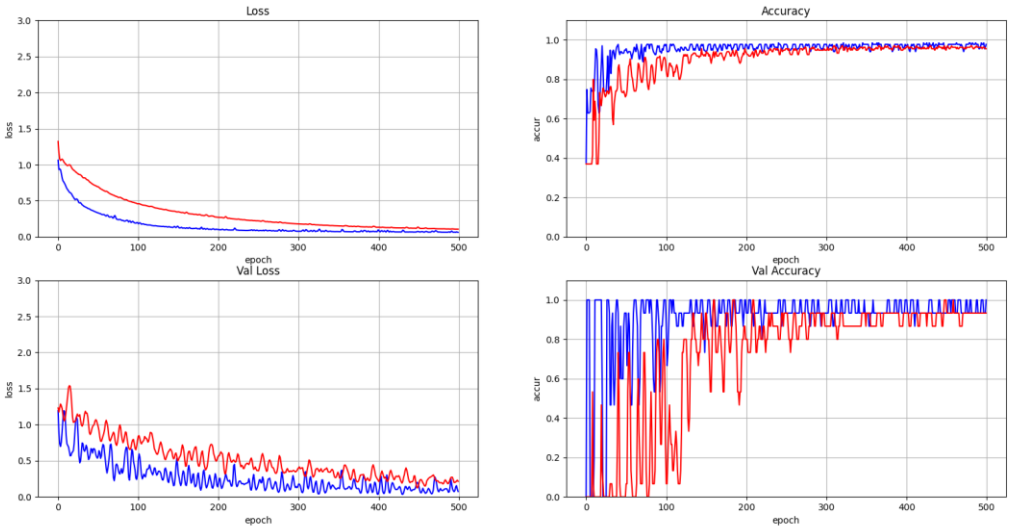

Рисунок 3 – Показатели ИНС №4 и №5

Из рисунка видно, что увеличение числа нейронов положительно повлияло на точность и уменьшило ошибки. В целом выводы сравнения те же: ИНС №4 с функцией активации Relu в скрытом слое показывает более быстрое обучение (на этот раз точность при данных для обучения значительно повышается и стабилизируется уже около 50 эпохи), меньшие ошибки и высокую точность, чем ИНС №5 с функцией активации Sigmoid в скрытом слое.

Был проведен анализ ИНС с двумя скрытыми слоями, число эпох было уменьшено до 200.

Слои ИНС №6 и ИНС №7 представлены в табл. 4.

Таблица 4 – ИНС №6 и №7

ИНС №6 (синий)	Ф-я акт.	Кол-во нейр.	ИНС №7 (красный)	Ф-я акт.	Кол-во нейр.
	Relu	4		Relu	4
	Relu	192		Relu	192
	Sigmoid	192		Relu	192
	Softmax	3		Softmax	3

Результаты тестирования ИНС №6 и №7 представлены на рис. 4 (ИНС №6 – синие линии, ИНС №7 – красные линии).

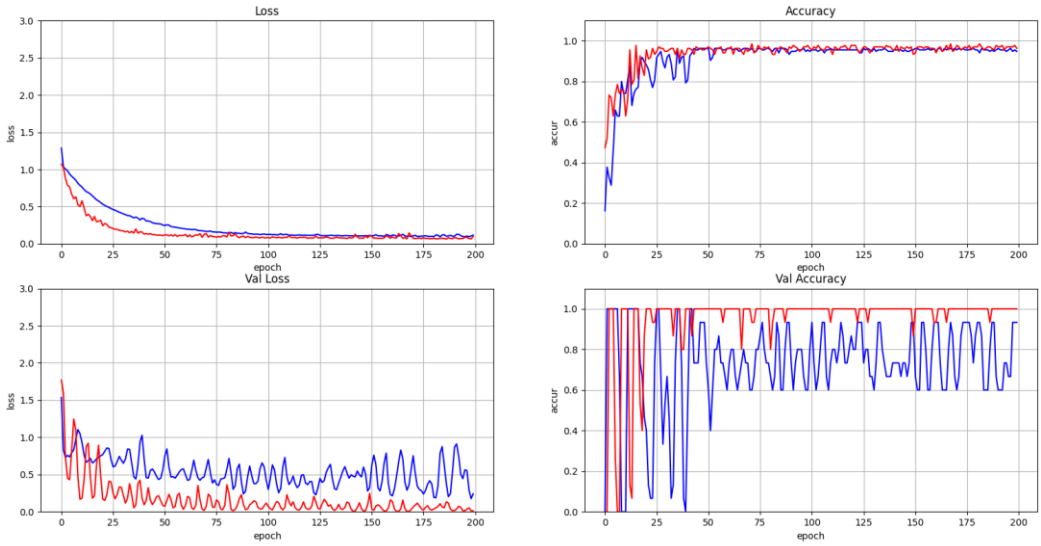

Рисунок 4 – Показатели ИНС №6 и №7

Из графиков видно преимущество ИНС №7 с двумя скрытыми слоями по 192 нейрона с функцией активации Relu. ИНС обучается быстро, уже около 30-й эпохи показывает стабильно высокую точность. Ошибки стабилизируются после 50-й эпохи. На проверочных данных ИНС показывает стабильный результат после 50-й эпохи.

Анализ показал, что функция активации Relu для скрытых слоев является оптимальной для данных и задачи, поэтому в дальнейшем другие функции активации рассматриваться не будут. Необходимо проверить, приведет ли дальнейшее увеличение числа слоев к улучшению результатов. Количество эпох было изменено до 100.

Слои ИНС №8 и ИНС №9 представлены в табл. 5.

Таблица 5 – Слои ИНС №8 и №9

ИНС №8 (синий)	Ф-я акт.	Кол-во нейр.	ИНС №9 (красный)	Ф-я акт.	Кол-во нейр.
	Relu	4		Relu	4
	Relu	192		Relu	192
	Relu	192		Relu	192
	Softmax	3		Relu	192
				Softmax	3

Результаты тестирования ИНС №8 и №9 представлены на рис. 5 (ИНС №8 – синие линии, ИНС №9 – красные линии).

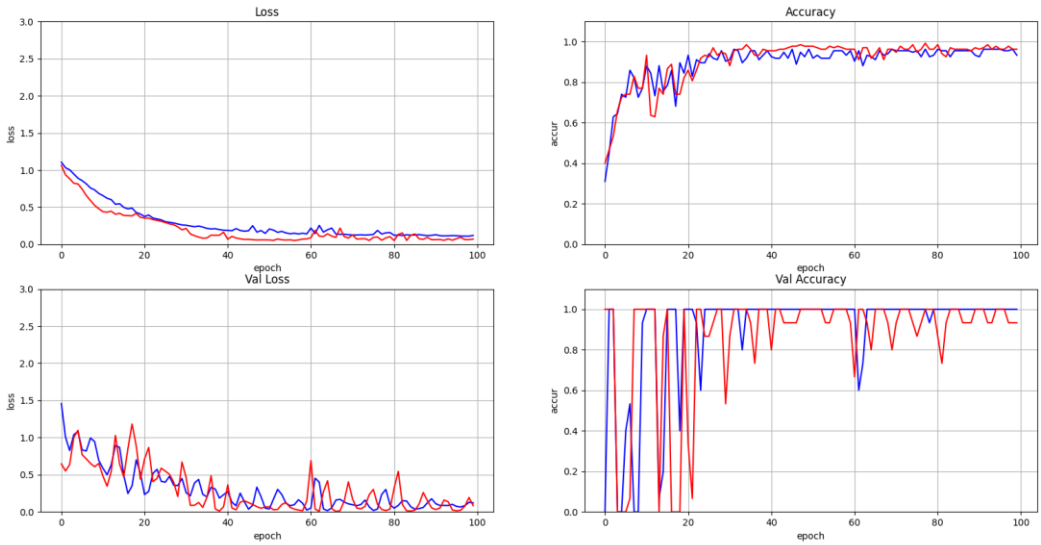

Рисунок 5 – Показатели ИНС №8 и №9

Из графиков видно, что ИНС №9 с тремя скрытыми слоями не имеет значительного преимущества.

Из анализа можно сделать вывод, что наиболее оптимальной для данной задачи можно считать ИНС №8. В дальнейших исследованиях будет использоваться эта ИНС.

Было проведено сравнение обучения модели при разных значениях количества тренировочных объектов. На рис. 6 представлен результат эксперимента, на графиках синяя линяя – batch_size = 64, красная линия – batch_size = 20.

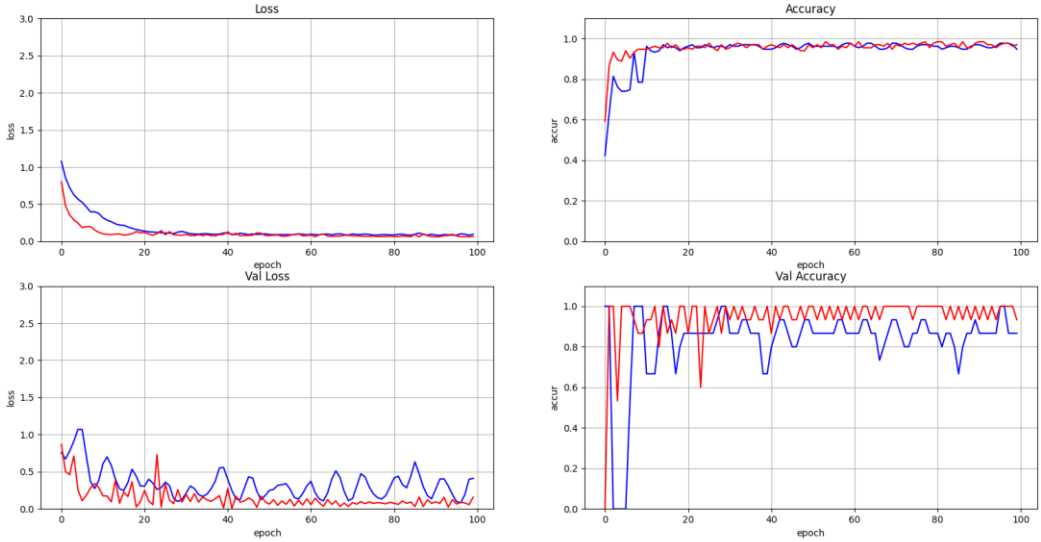

Рисунок 6 – Сравнение обучения при разном batch_size

На графике видно, что при меньшем значении числа тренировочных объектов результаты обучения улучшаются. Скорее всего это связано с тем, что при уменьшении batch_size возрастает число итераций (градиентных обновлений) в каждой эпохе.

Было проведено сравнение обучения модели при разном коэффициенте разделения данных на данные для обучения и данные для проверки (validation_split). На рис. 7 представлен результат эксперимента, на графиках синяя линия – validation_split = 0.1, красная линия – validation_split = 0.4.

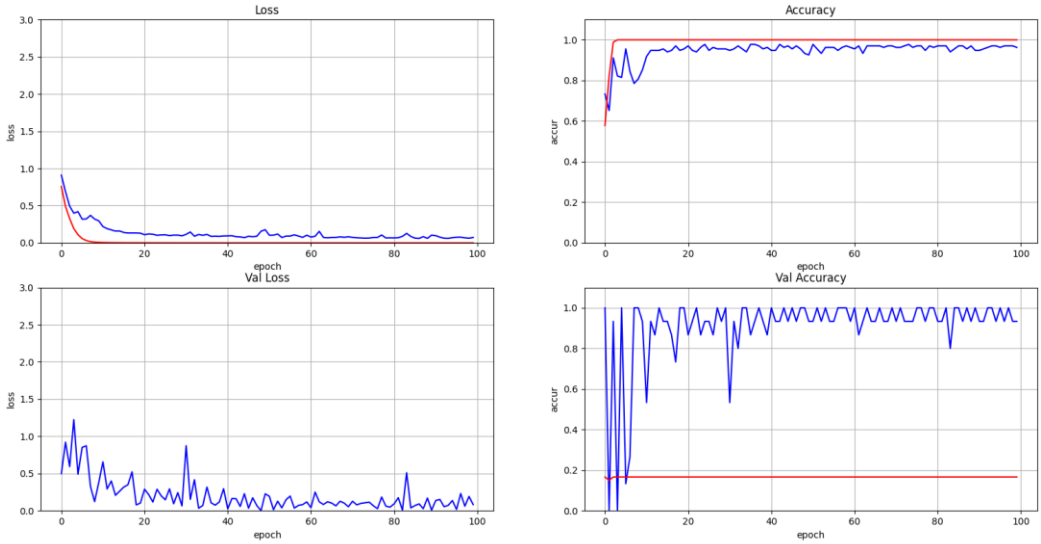

Рисунок 7 – Сравнение обучения при разном validation_split

Из графиков видно, что при значении validation_split = 0.4 ИНС не хватает данных для обучения. Точность на данных для обучения достигает 1, в то время как точность на данных для проверки не превышает 0.17, а потери (не отображаются на графике) равны 13.

При значении validation_split = 0.1 ИНС медленнее обучается, но зато на данных для проверки показывает хорошие результаты.

Можно сделать вывод, что при изменении отношения данных для обучения и данных для проверки (validation_split) изменяется скорость обучения модели, но при больших значениях параметра модель не будет корректно работать на реальных данных, так как ей не хватает данных для обучения.

3. Выбор ИНС, параметров обучения для данных и задачи

Итак, в результате анализа была выбрана модель ИНС №8, структура которой представлена в табл. 6, с особыми параметрами обучения.

Таблица 6 – Структура ИНС №8

Слои	**Ф-я акт.**	**Кол-во нейр.**
	Relu	4
	Relu	192
	Relu	192
	Softmax	3
Параметры обучения		
Validation_split	**Batch_size**	**Epochs**
0.1	20	100

Результаты нескольких тестирований ИНС №8 с выбранными параметрами обучения представлены на рис. 8. На графике 4 линии – 4 последовательных запуска аналогичных моделей ИНС №8 с одинаковыми параметрами тестирования.

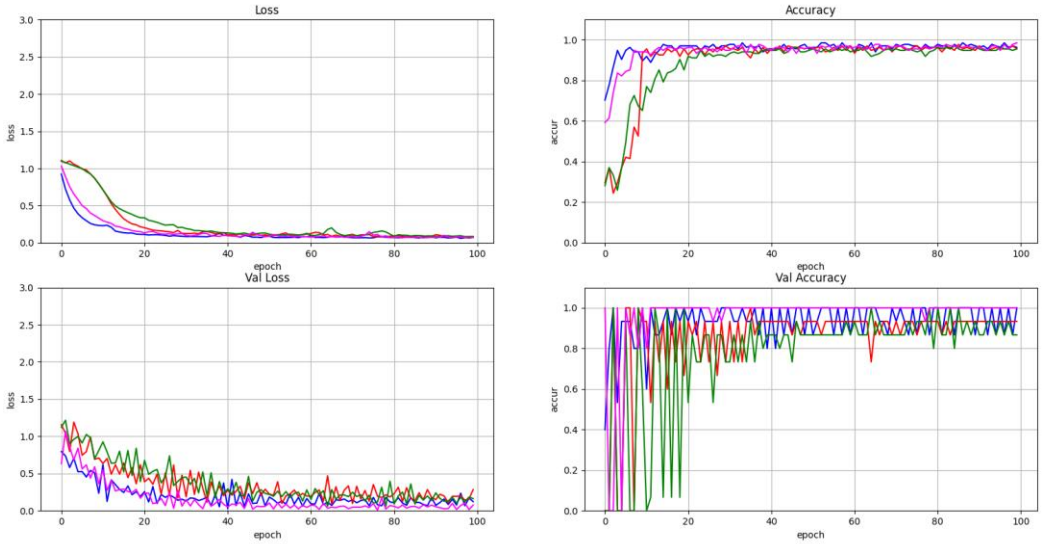

Рисунок 8 – Тестирование ИНС №8 с подобранными параметрами

В лучшем случае (при удачных начальных весах), ИНС достаточно 25-ти эпох для обучения (синяя и розовая линии). В среднем случае модели может потребоваться около 40-ти эпох (красная и зеленая линии) для достижения хороших результатов. Код программы с реализацией данной ИНС приведен в Приложении А. В среднем, показатели таковы:

- Обучение в среднем за 30 эпох
- Потери на данных для обучения: < 0.1
- Точность на данных для обучения: > 0.96
- Потери на данных для проверки: < 0.1
- Точность на данных для проверки: > 0.9333

Выводы.

В результате выполнения лабораторной работы была изучена структура искусственной нейронной сети, такие параметры обучения как число тренировочных объектов и разделение данных для проверки и для обучения. Было проведено сравнение ИНС с различными функциями активации нейронов на слоях, разным количеством слоев, нейронов в слоях, разными параметрами обучения. В результате были определены структура и параметры обучения ИНС для решения задачи классификации цветков между 3-мя классами по 4-м численным параметрам.

ПРИЛОЖЕНИЕ А.

Исходный код программы. Файл main.py.

```
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

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(192, activation='relu'))
model.add(Dense(192, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
res = model.fit(X, dummy_y, epochs=epochs, batch_size=20, validation_split=0.1)

loss_history = numpy.array(res.history["loss"])
val_loss_history = numpy.array(res.history["val_loss"])
accuracy_history = numpy.array(res.history["accuracy"])
val_accuracy_history = numpy.array(res.history["val_accuracy"])

# graph maker
history = [loss_history, accuracy_history, val_loss_history,
val_accuracy_history]
titles = ["Loss", "Accuracy", "Val Loss", "Val Accuracy"]
ylables = ["loss", "accur"]

for i in range(4):
      plt.subplot(2, 2, i + 1)
      plt.title(titles[i])
      plt.xlabel("epoch")
      plt.ylabel(ylables[i % 2])
      axes = plt.gca()
```

```
      if i % 2:
            axes.set_ylim([0, 1.1])
      else:
            axes.set_ylim([0, 3])
      plt.grid()
      plt.plot([i for i in range(epochs)], history[i], color="blue")
plt.show()
```