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»

Студентка гр. 8382 _____ Бердникова А.А.

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 функций 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.

Импортируем необходимые для работы классы и функции. Кроме Keras понадобится Pandas для загрузки данных и scikit-learn для подготовки данных и оценки модели.

```
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
```

Набор данных загружается напрямую с помощью pandas. Затем необходимо разделить атрибуты (столбцы) на входные данные (X) и выходные данные (Y).

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

Далее происходит переход от текстовых меток к категориальному вектору.

```
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).

```
# 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
H = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1)
```

Был реализован вывод 4-х графиков: ошибки во время обучения, точность во время обучения, ошибки на проверяемых данных, точность на проверяемых данных.

```
# Построить графики ошибок и точности в ходе обучения
#Получение ошибки и точности в процессе обучения
loss = H.history['loss']
val_loss = H.history['val_loss']
acc = H.history['accuracy']
val_acc = H.history['val_accuracy']
epochs = range(1, len(loss) + 1)
#Построение графика ошибки
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
#Построение графика точности
plt.clf()
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

2. Анализ показателей различных ИНС

Модель №1.

```
# 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
H = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1)
```

Результаты тестирования модели №1 представлены на рис. 1.

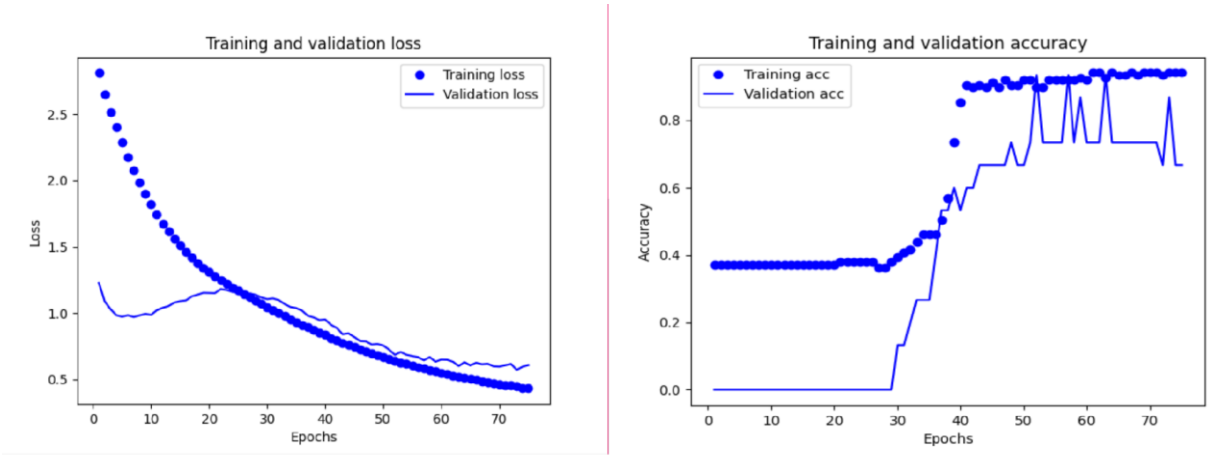

Рисунок 1 – Показатели ИНС №1

Из графиков видно, что в начале обучения ИНС №1 достаточно быстро уменьшает показатели ошибок. В то же время точность за 40 эпох достигает ~0.7-0.9 на обучаемых данных, а на проверочных результат в ~0.9.

Значения при каждом запуске очень сильно отличаются друг от друга.

Модель №2.

```
# Создание модели
model = Sequential()
model.add(Dense(4, activation="relu"))
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
H = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1)
```

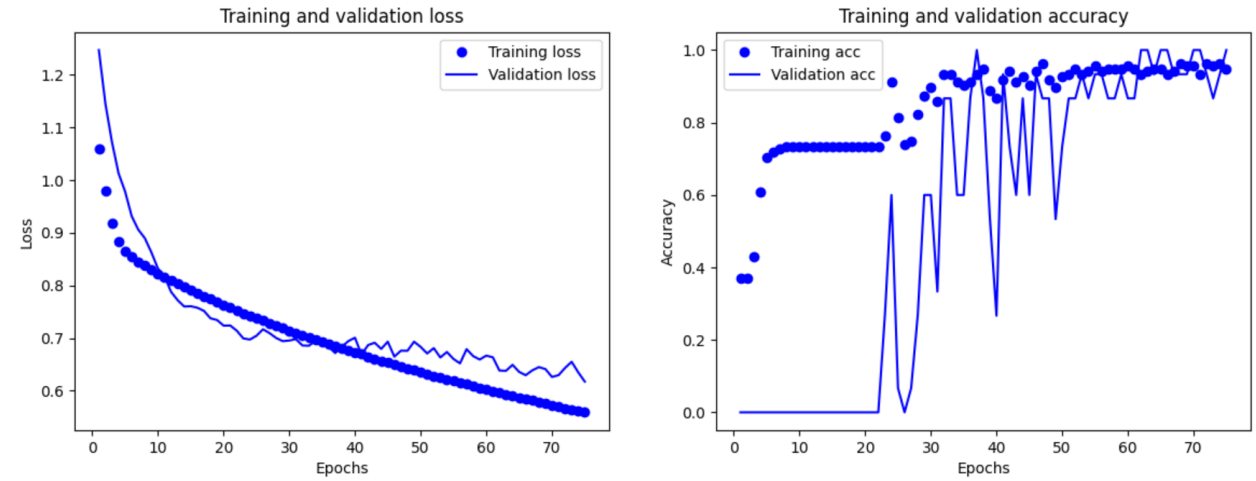

Рисунок 2 – Показатели ИНС №2

Из графиков видно, что добавление скрытого слоя с 4 нейронами ухудшило точность результатов. Стабильная ситуация начинается после 50 эпохи.

Модель №3.

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

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

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

# Обучение сети
H = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1)
```

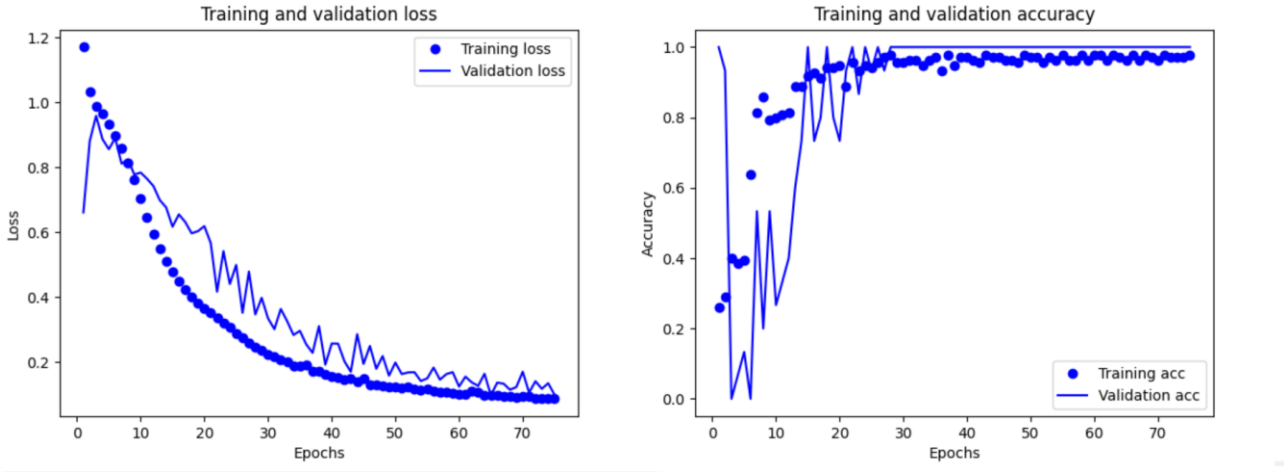

Рисунок 3 – Показатели ИНС №3

Из рисунка видно, что увеличение числа нейронов в скрытом слое положительно повлияло на точность и уменьшило ошибки.

В начале обучения ИНС №3 достаточно быстро уменьшает показатели ошибок. В то же время точность за 30 эпох достигает ~0.9 на обучаемых данных, а на проверочных результат практически ~1.

Но значения при каждом запуске еще довольно отличаются друг от друга.

Модель №4.

Было проведено то же сравнение, но с еще большим количеством нейронов в скрытом слое с функцией активации Relu.

```
# Создание модели
model = Sequential()
model.add(Dense(4, activation="relu"))
model.add(Dense(200, activation="relu"))
model.add(Dense(3, activation="softmax"))

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

# Обучение сети
H = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1)
```

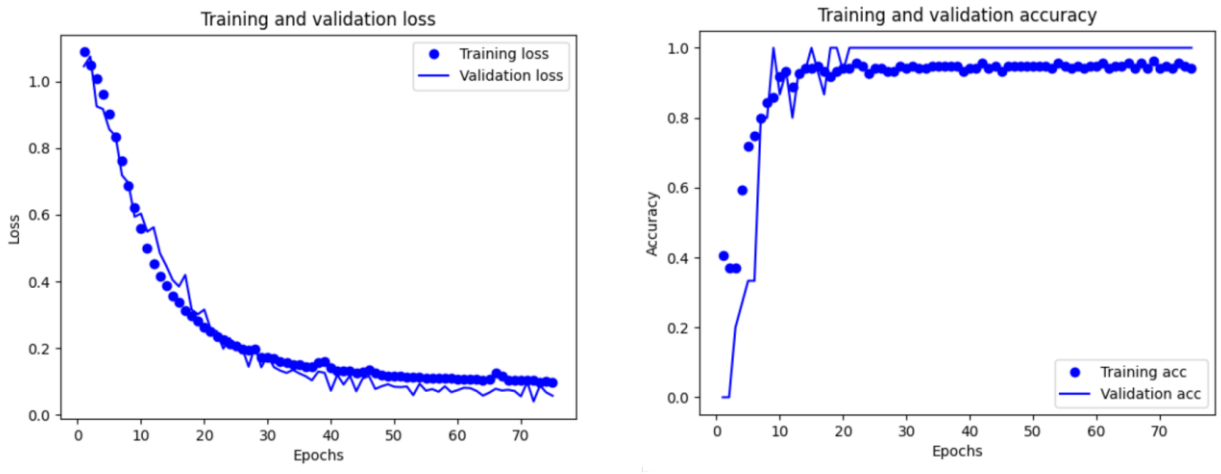

Рисунок 4 – Показатели ИНС №4

Из рисунка видно, что увеличение числа нейронов в скрытом слое положительно повлияло на точность и уменьшило ошибки. Результат стал более стабилен.

В начале обучения ИНС №4 быстро уменьшает показатели ошибок. В то же время точность за 20 эпох достигает ~0.9 на обучаемых данных, а на проверочных результат практически ~1.

Модель №5.

Был добавлен еще один скрытый слой с функцией активации Relu. Кол-во нейронов – 64 в каждом.

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

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

# Обучение сети
H = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1)
```

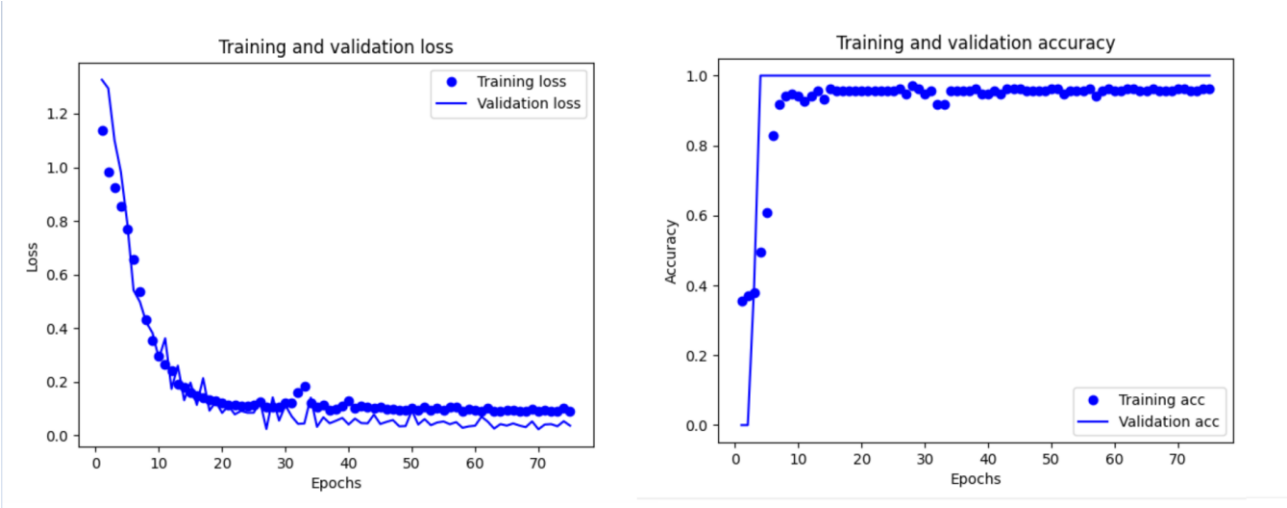

Рисунок 5 – Показатели ИНС №5

Из графиков видно преимущество ИНС №5 с двумя скрытыми слоями по 64 нейрона с функцией активации Relu. ИНС обучается быстро, уже на 10-й эпохе показывает стабильно высокую точность. Ошибки стабилизируются после 20-й эпохи.

Результат достаточно стабилен при различных запусках.

При увеличении нейронов и скрытых слоев результат достаточно сильно скачет, поэтому приходим к выводу, что модель №5 показала самые лучшие результаты.

Выводы.

В результате выполнения лабораторной работы была изучена структура искусственной нейронной сети. Ознакомились с задачей классификации, создали ИНС и настроили оптимальные параметры ее обучения, при которых ИНС выдает стабильные и достаточно хорошие результаты.

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

# 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(64, activation="relu"))
model.add(Dense(64, activation="relu"))
model.add(Dense(3, activation="softmax"))

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

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

# Построить графики ошибок и точности в ходе обучения
#Получение ошибки и точности в процессе обучения
loss = H.history['loss']
val_loss = H.history['val_loss']
acc = H.history['accuracy']
val_acc = H.history['val_accuracy']
epochs = range(1, len(loss) + 1)
#Построение графика ошибки
```

```
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
#Построение графика точности
plt.clf()
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```