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

ОТЧЕТ

по лабораторной работе №2

"Бинарная классификация отраженных сигналов радара"

по дисциплине «Искусственные нейронные сети»

Студентка гр. 8383 _____ Ишанина Л.Н.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Реализовать классификацию между камнями (R) и металлическими цилиндрами (M) на основе данных об отражении сигналов радара от поверхностей.

60 входных значений показывают силу отражаемого сигнала под определенным углом. Входные данные нормализованы и находятся в промежутке от 0 до 1.

Задание.

- Ознакомиться с задачей бинарной классификации
- Загрузить данные
- Создать модель ИНС в tf.Keras
- Настроить параметры обучения
- Обучить и оценить модель
- Изменить модель и провести сравнение. Объяснить результаты

Изучить влияние кол-ва нейронов на слое на результат обучения модели.

Изучить влияние кол-ва слоев на результат обучения модели

Построить графики ошибки и точности в ходе обучения

Провести сравнение полученных сетей, объяснить результат

Выполнение работы.

Был скачен набор данных (файл sonar.all-data). Скачанный файл был переименован в “sonar.csv” и помещен в директорию проекта лабораторной работы. В листинге представлен пример данных из файла “sonar.csv”:

```
0.0200,0.0371,0.0428,0.0207,0.0954,0.0986,0.1539,0.1601,0.3109,0.21
11,0.1609,0.1582,0.2238,0.0645,0.0660,0.2273,0.3100,0.2999,0.5078,0
.4797,0.5783,0.5071,0.4328,0.5550,0.6711,0.6415,0.7104,0.8080,0.679
1,0.3857,0.1307,0.2604,0.5121,0.7547,0.8537,0.8507,0.6692,0.6097,0.
4943,0.2744,0.0510,0.2834,0.2825,0.4256,0.2641,0.1386,0.1051,0.1343
,0.0383,0.0324,0.0232,0.0027,0.0065,0.0159,0.0072,0.0167,0.0180,0.0
084,0.0090,0.0032,R
0.0453,0.0523,0.0843,0.0689,0.1183,0.2583,0.2156,0.3481,0.3337,0.28
72,0.4918,0.6552,0.6919,0.7797,0.7464,0.9444,1.0000,0.8874,0.8024,0
```

```
.7818,0.5212,0.4052,0.3957,0.3914,0.3250,0.3200,0.3271,0.2767,0.442
3,0.2028,0.3788,0.2947,0.1984,0.2341,0.1306,0.4182,0.3835,0.1057,0.
1840,0.1970,0.1674,0.0583,0.1401,0.1628,0.0621,0.0203,0.0530,0.0742
,0.0409,0.0061,0.0125,0.0084,0.0089,0.0048,0.0094,0.0191,0.0140,0.0
049,0.0052,0.0044,R
```

Затем были импортированы необходимые для работы классы и функции. Листинг подключения модулей представлен ниже:

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

Далее были загружены данные из файла и разделены атрибуты (столбцы) на 60 входных параметров (X) и 1 выходной (Y). Листинг представлен ниже:

```
dataframe = pandas.read_csv("sonar.csv", header=None)
dataset = dataframe.values
X = dataset[:,0:60].astype(float)
Y = dataset[:,60]
```

В первой строке кода происходит чтение из файла “sonar.csv”, а также указан параметр `header=None` который означает, что заголовки в файле отсутствуют. Во второй строке происходит преобразование `dataframe` в удобный для работы формат (массив). В третьей строке происходит отделение входных данных и приведение их к типу float. В четвертой строке происходит отделение выходных данных, то есть последнего столбца каждой строки.

Выходные параметры представлены строками (“R” и “M”), которые были переведены в целочисленные значения 0 и 1 соответственно. Для этого был применен LabelEncoder из scikit-learn. Листинг приведен ниже:

```
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
```

Затем была создана простая модель. Листинг представлен ниже:

```
model = Sequential()
model.add(Dense(60, input_dim=60, kernel_initializer='normal',
activation='relu'))
```

```
model.add(Dense(1, kernel_initializer='normal',
activation='sigmoid'))
```

Далее были инициализированы параметры обучения и проведено сам процесс обучения сети. Листинг приведен ниже:

```
model.compile(optimizer='adam',loss='binary crossentropy',
metrics=['accuracy'])
H = model.fit(X, encoded_Y, epochs=100, batch_size=10,
validation_split=0.1)
```

В процессе обучения отображаются четыре величины: потери сети на обучающих данных и точность сети на обучающих данных, а также потери и точность на данных, не участвовавших в обучении.

Для построения графика ошибок и точности в ходе обучения была подключена библиотека matplotlib. Листинг подключения представлен ниже:

```
import matplotlib.pyplot as plt
```

Затем были реализованы сами графики. Листинг приведен ниже:

```
loss = H.history['loss']
val_loss = H.history['val_loss']
acc = H.history['acc']
val_acc = H.history['val_acc']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'm*', 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()

plt.clf()
plt.plot(epochs, acc, 'm*', 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()
```

Анализ различных искусственных нейронных сетей.

Изначально, была простая модель, состоящая из последовательности двух слоев.

```
model.add(Dense(60, input_dim=60, kernel_initializer='normal',
activation='relu'))
model.add(Dense(1, kernel_initializer='normal',
activation='sigmoid'))
```

Лучший результат тестирования представлен на рис.1-2.

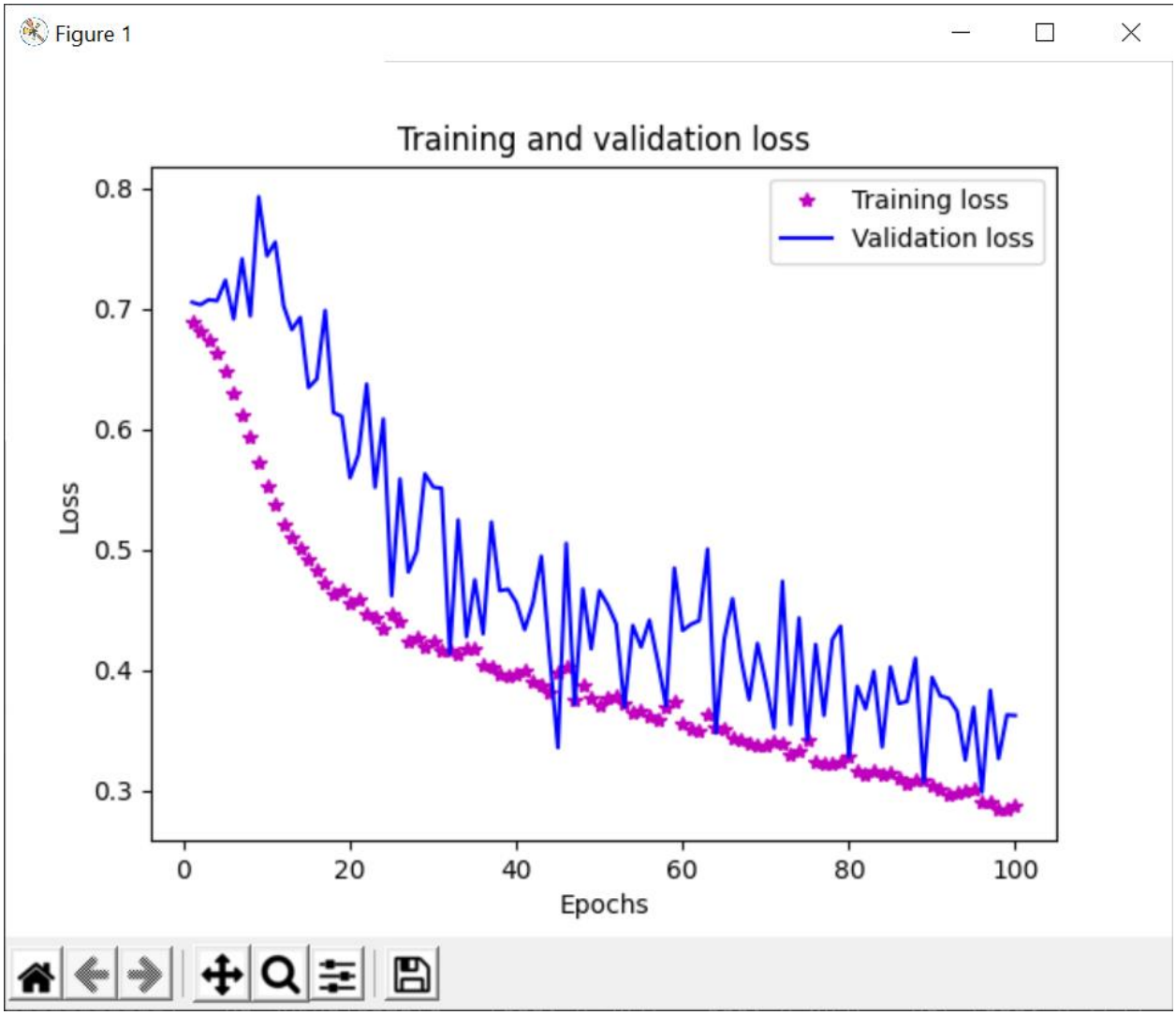

Рисунок 1 – графики потери сети на обучающих данных и данных, не участвовавших в обучении.

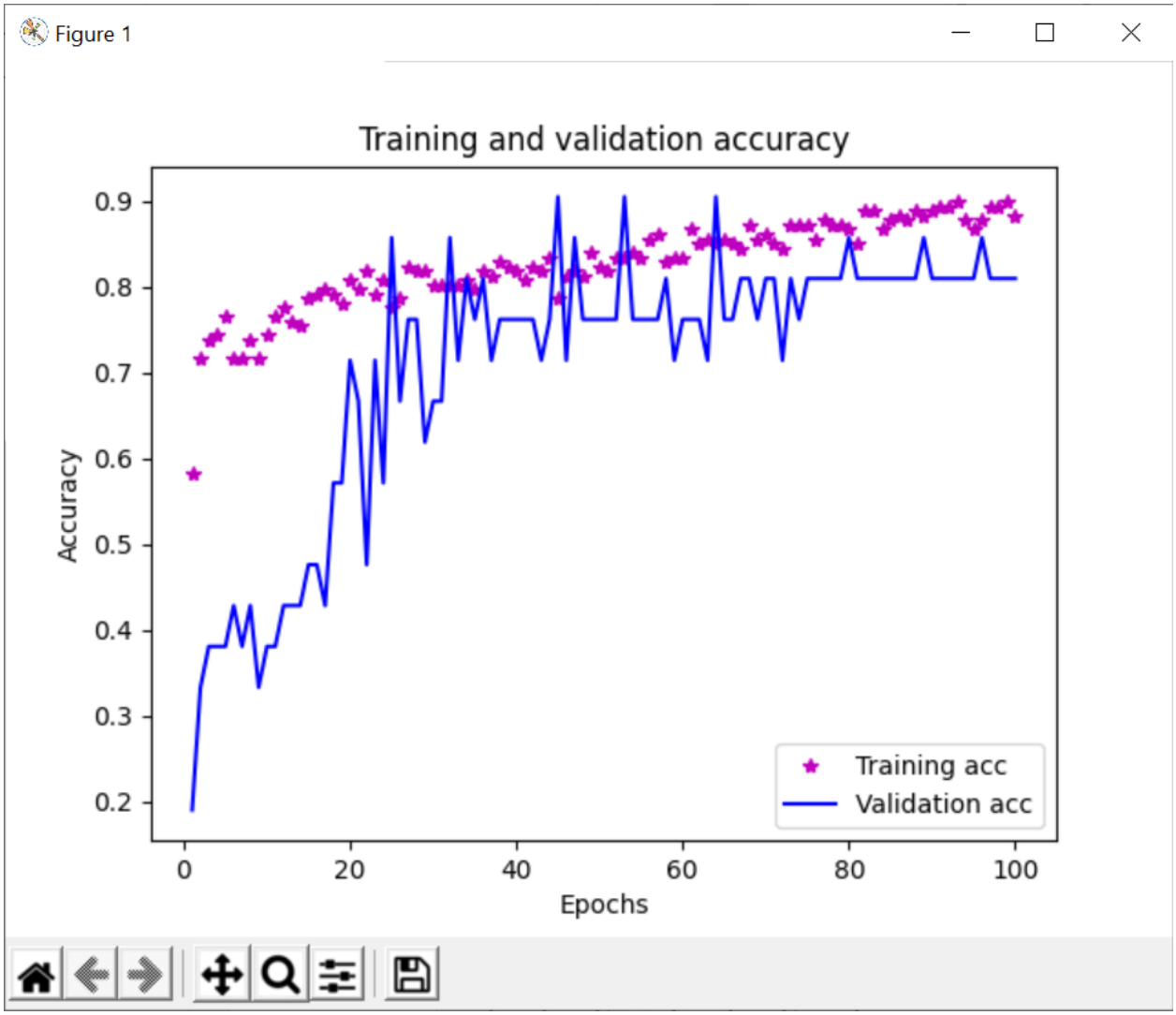

Рисунок 2 – графики точности сети на обучающих данных и данных, не участвовавших в обучении.

Исходя из полученных результатов, видно, что и на обучающих данных, и на данных, не участвовавших в обучении, точность невысокая (~ 0.80 – 0.85) и потери довольно высоки. Следовательно, обучение не очень эффективное.
ИНС достигла следующих показателей на 75 эпохе:
loss: 0.3325 - acc: 0.8717 - val_loss: 0.4432 - val_acc: 0.7619

Далее уменьшить размер входного слоя в два раза. Листинг приведен ниже: слоев.

```
model.add(Dense(60, input_dim=30, kernel_initializer='normal',
activation='relu'))
```

```
model.add(Dense(1, kernel_initializer='normal',
activation='sigmoid'))
```

Лучший результат тестирования представлен на рис.3-4.

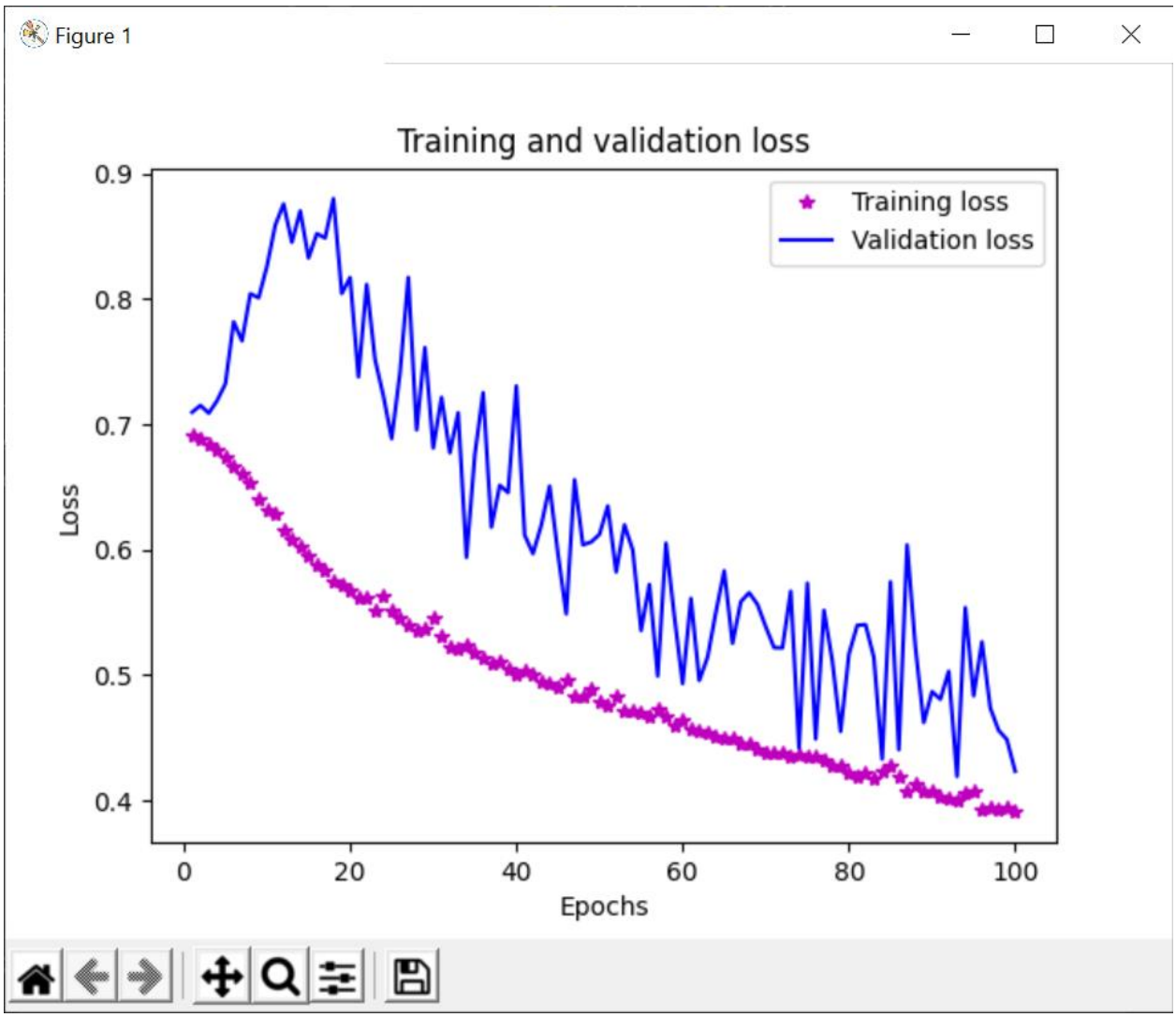

Рисунок 3 – графики потери сети на обучающих данных и данных, не участвовавших в обучении.

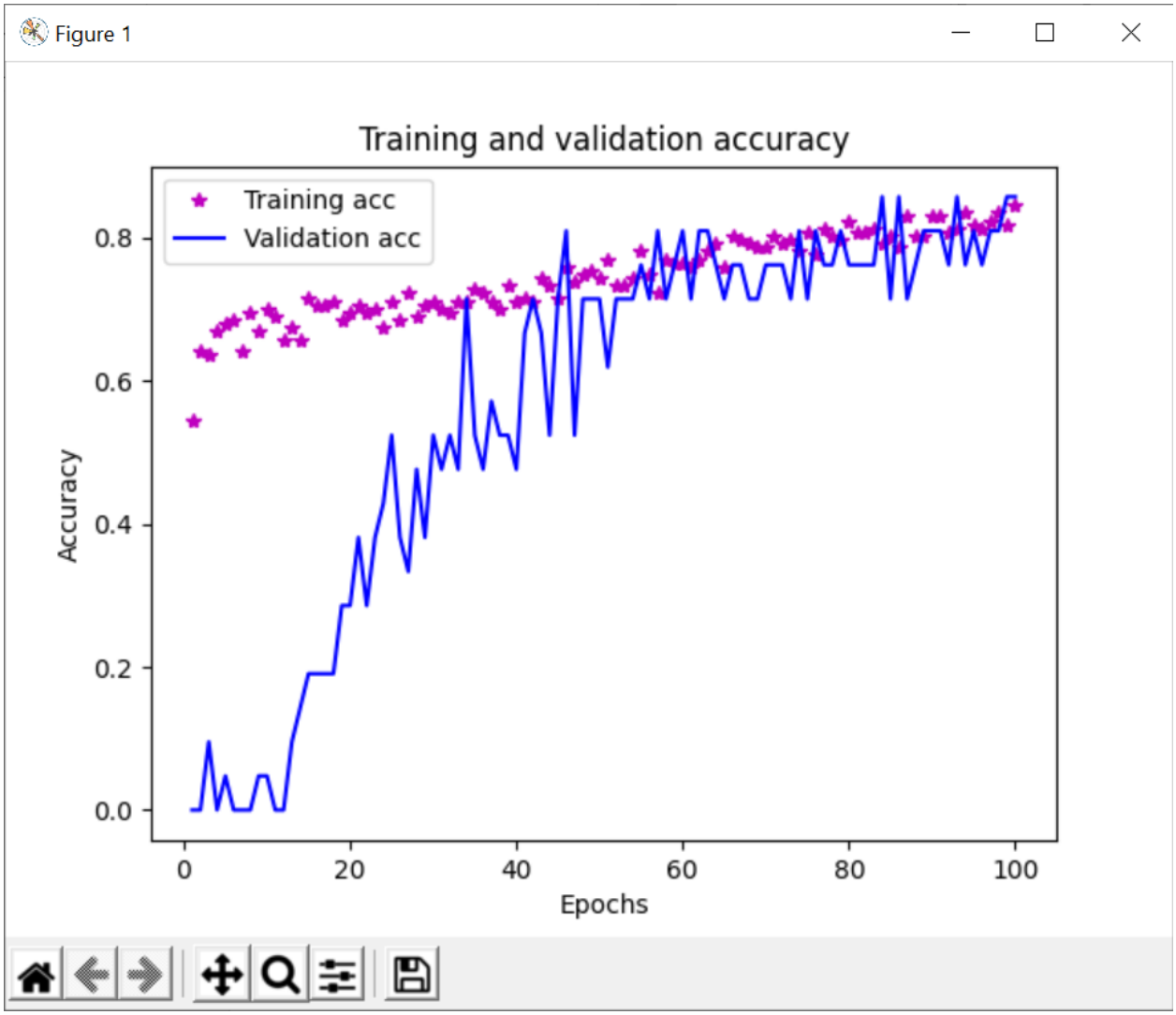

Рисунок 4 – графики точности сети на обучающих данных и данных, не участвовавших в обучении.

ИНС достигла следующих показателей на 75 эпохе:

loss: 0.4360 - acc: 0.7807 - val_loss: 0.4420 - val_acc: 0.8095

Можно сделать вывод, что показатели немного ухудшились, но в целом изменений особо не произошло.

Далее был добавлен промежуточный (скрытый) слой Dense в архитектуру сети с 15 нейронами. Листинг приведен ниже:

```
model.add(Dense(4, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

Лучший результат тестирования представлен на рис.5-6.

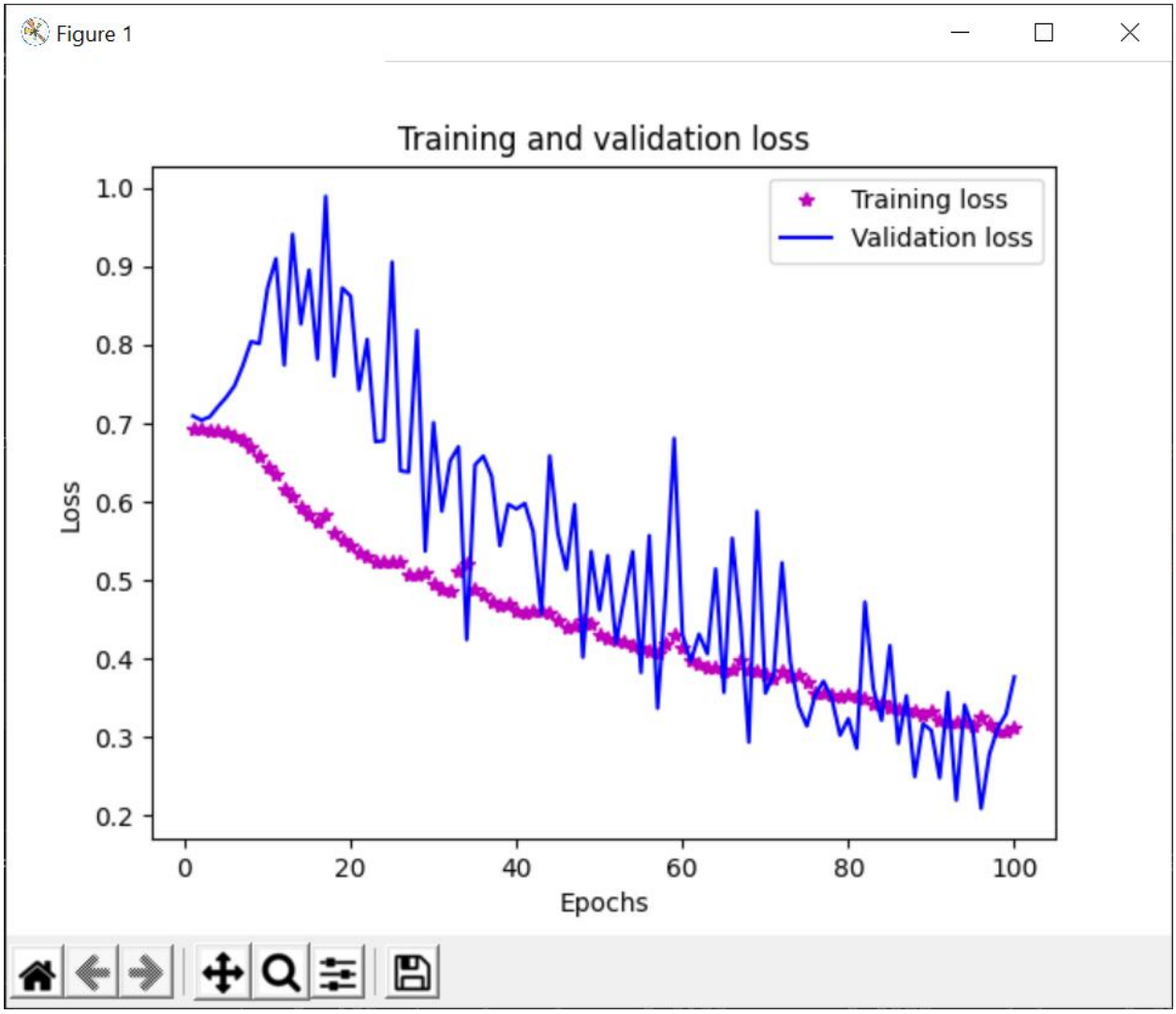

Рисунок 5 – графики потери сети на обучающих данных и данных, не участвовавших в обучении.

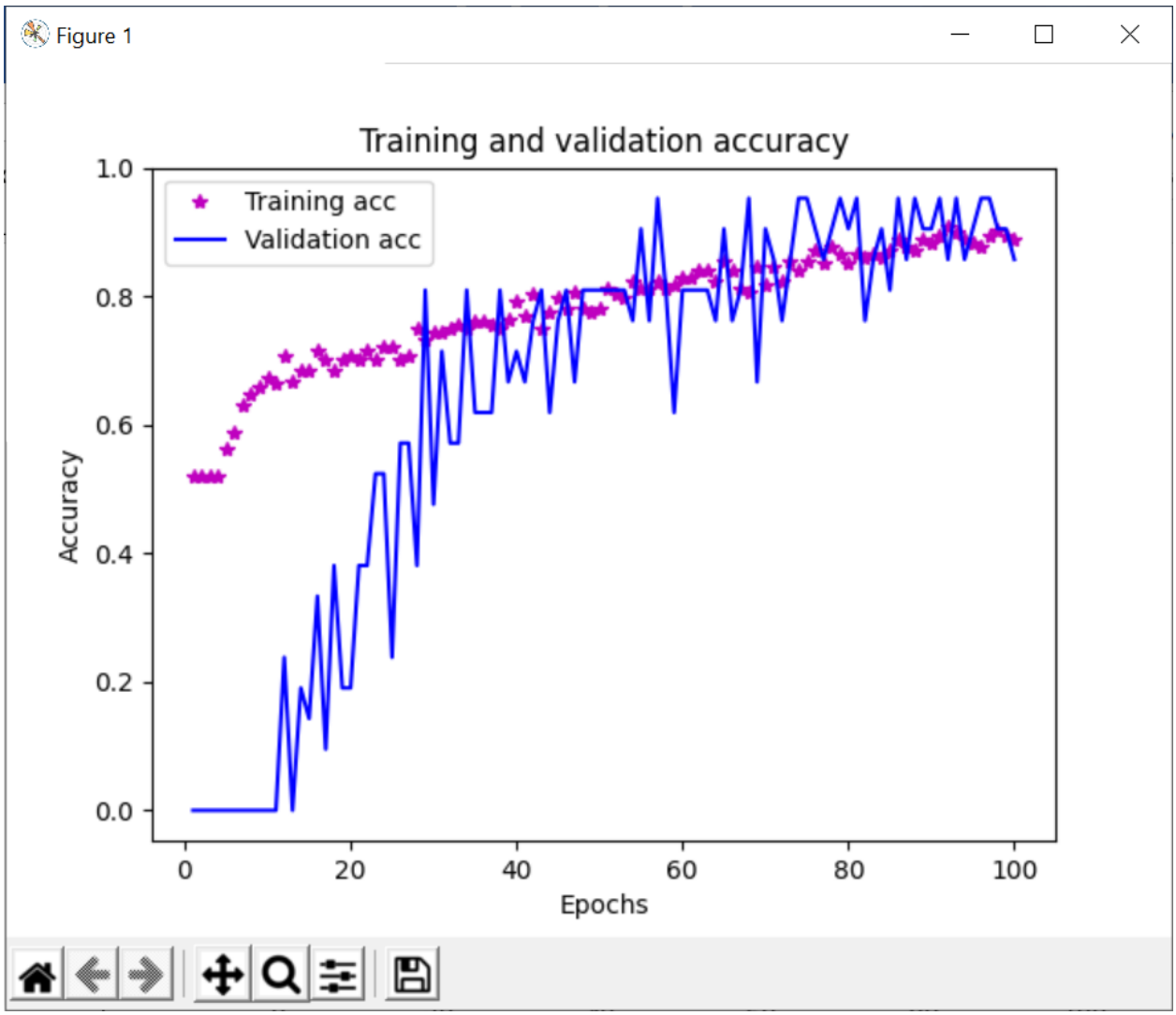

Рисунок 6 – графики точности сети на обучающих данных и данных, не участвовавших в обучении.

ИНС достигла следующих показателей на 75 эпохе:

loss: 0.3695 - acc: 0.8556 - val_loss: 0.3145 - val_acc: 0.9524

Сравнив, с предыдущими значениями, можно сделать вывод что все показатели немного улучшились, особенно точность. Таким образом, дополнительные слои позволяют ввести нелинейность в сеть, что позволяет получать более высокую точность.

Выводы.

В ходе выполнения лабораторной работы были изучены различные архитектуры ИНС, а также построены графики ошибок и точности в ходе обучения и выбрана наилучшая модель.