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. 8382 _____ Ивлева О.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель.

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 Keras
- Настроить параметры обучения
- Обучить и оценить модель
- Изменить модель и провести сравнение. Объяснить результаты

Требуется:

1. Изучить влияние кол-ва нейронов на слое на результат обучения модели.
2. Изучить влияние кол-ва слоев на результат обучения модели
3. Построить графики ошибки и точности в ходе обучения
4. Провести сравнение полученных сетей, объяснить результат

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
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 60 входных параметров (X) и 1 выходной (Y).

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

Выходные параметры представлены строками (“R” и “M”), которые необходимо перевести в целочисленные значения 0 и 1 соответственно. Для этого применяется LabelEncoder из scikit-learn.

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

Далее была создана простая модель с двумя слоями: первый имеет 60 нейронов с функцией активации Relu (для обработки 60 входных параметров), второй – 1 нейрон с функцией активации Sigmoid (от 0 до 1 – определяет класс объекта на выходе).

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

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

# Обучение сети
H = model.fit(X, encoded_Y, epochs=100, 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.

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

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

# Обучение сети
H = model.fit(X, encoded_Y, epochs=100, batch_size=10, validation_split=0.1)
```

Результаты тестирования модели №1 представлены на рис. 1.

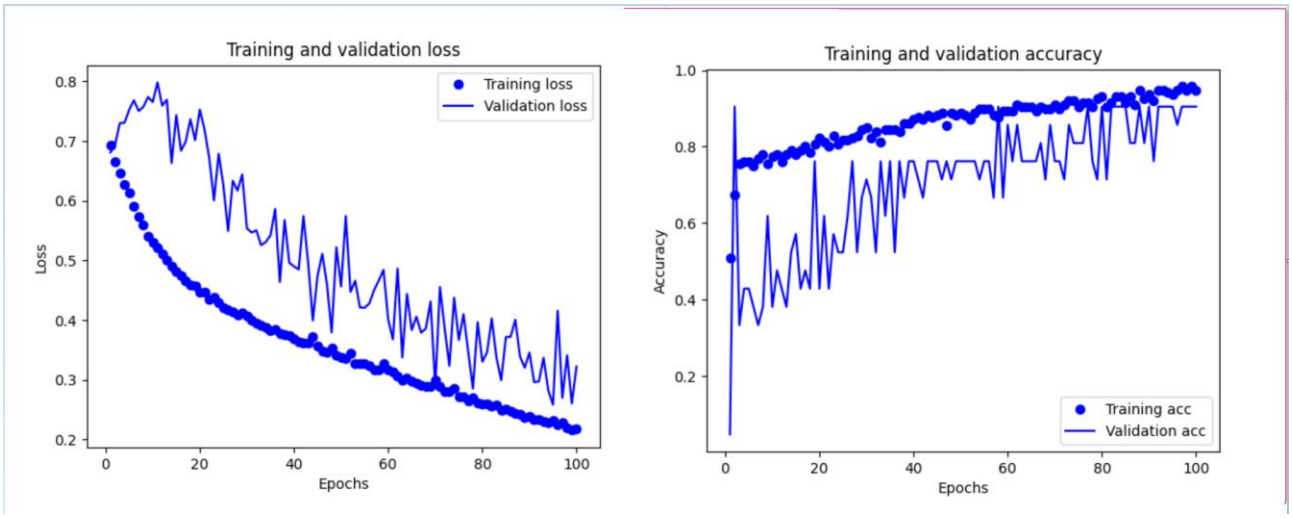

Рисунок 1 – Показатели ИНС №1

Из графиков видно, что ИНС №1 не достигает высокого показателя точности, особенно на проверочных данных, показатели после 45 эпохи практически не различаются. Высокие показатели ошибок, что говорит о том, что данная ИНС не подходит для решения задачи.

Значения при каждом запуске сильно отличаются друг от друга.

В представленном наборе данных присутствует некоторая избыточность, т.к. с разных углов описывается один и тот же сигнал.

Модель №2.

В ИНС №2 было уменьшено кол-во нейронов во входном слое в два раза.

```
model = Sequential()
model.add(Dense(60, input_dim=30, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

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

# Обучение сети
H = model.fit(X, encoded_Y, epochs=100, batch_size=10, validation_split=0.1)
```

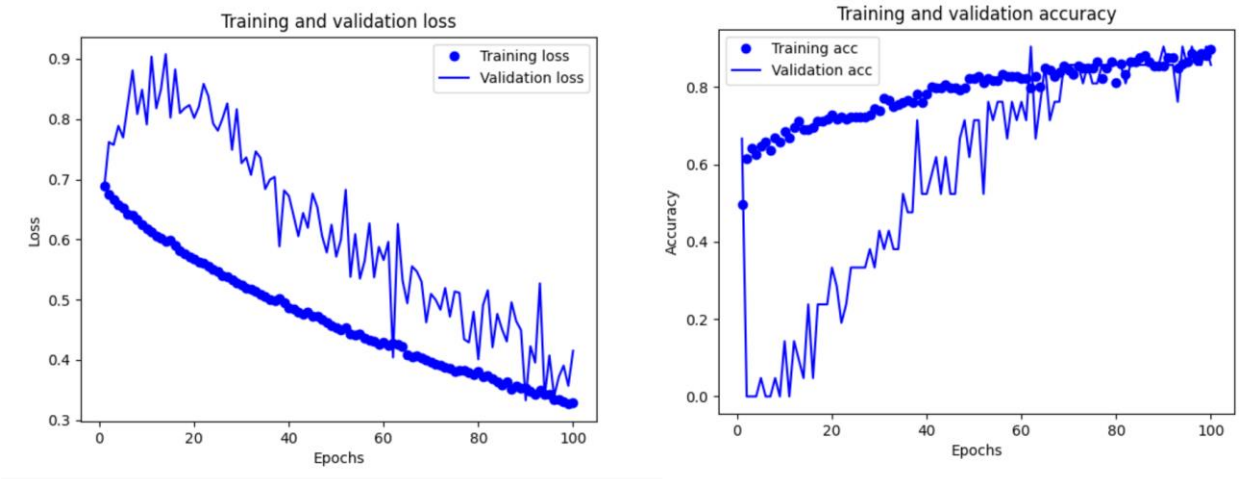

Рисунок 2 – Показатели ИНС №2

Из графиков видно, что ИНС №2 достигает достаточно высокого показателя точности на проверочных данных только после 65 эпохи. Высокие показатели ошибок, что говорит о том, что данная ИНС не подходит для решения задачи.

Значения при каждом запуске сильно отличаются друг от друга.

Модель №3.

Нейронная сеть с несколькими слоями позволяет находить закономерности не только во входных данных, но и в их комбинации. Также, дополнительные слои позволяют ввести нелинейность в сеть, что позволяет получать более высокую точность.

Был добавлен еще один скрытый слон нейронов с функцией активации Relu.

```
model = Sequential()
model.add(Dense(60, input_dim=30, activation='relu'))
model.add(Dense(15, activation="relu"))
model.add(Dense(1, activation='sigmoid'))

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

# Обучение сети
```

```
H = model.fit(X, encoded_Y, epochs=100, batch_size=10, validation_split=0.1)
```

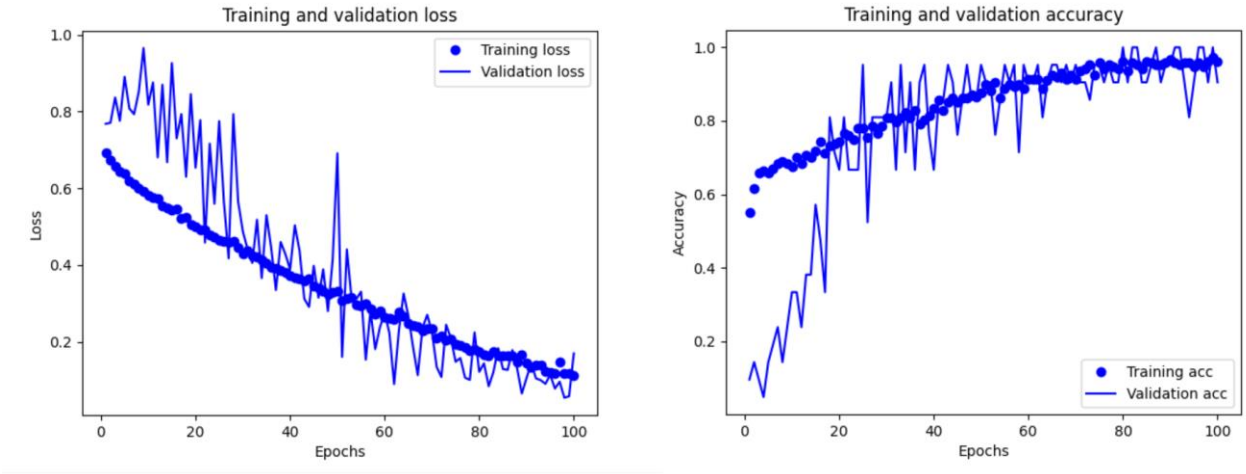

Рисунок 3 – Показатели ИНС №3

Из рисунка видно, что добавление скрытого слоя с 15 нейронами положительно повлияло на точность и уменьшило ошибки.

В начале обучения ИНС №3 уменьшает показатели ошибок. В то же время точность за 45 эпох достигает вполне хороших результатов.

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

Модель №4.

Был добавлен еще один скрытый слон нейронов с функцией активации Relu.

```
model = Sequential()
model.add(Dense(60, input_dim=30, activation='relu'))
model.add(Dense(15, activation="relu"))
model.add(Dense(30, activation="relu"))
model.add(Dense(1, activation='sigmoid'))

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

# Обучение сети
H = model.fit(X, encoded_Y, epochs=100, batch_size=10, validation_split=0.1)
```

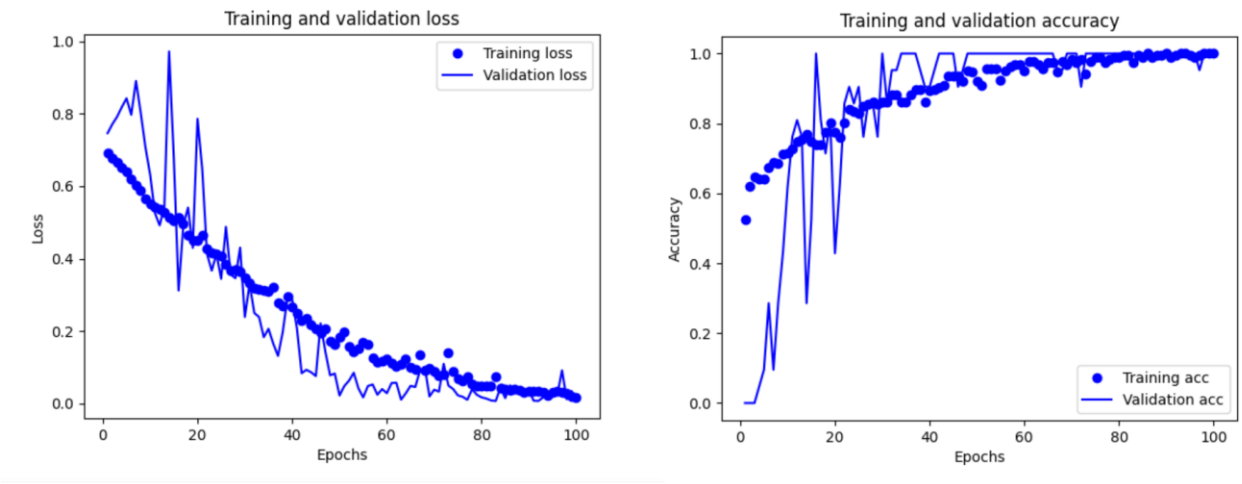

Рисунок 4 – Показатели ИНС №4

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 45 эпох достигает ~0.9 на обучаемых данных и на проверяемых, что является достаточно хорошим результатом.

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("sonar.csv", header=None)
dataset = dataframe.values
X = dataset[:, 0:30].astype(float)
Y = dataset[:, 60]
# Выходные параметры представлены строками (“R” и “M”),
# которые необходимо перевести в целочисленные значения 0 и 1 соответственно.
# Для этого применяется LabelEncoder из scikit-learn.
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)

model = Sequential()
model.add(Dense(60, input_dim=30, activation='relu'))
model.add(Dense(15, activation="relu"))
model.add(Dense(30, activation="relu"))
model.add(Dense(1, activation='sigmoid'))

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

# Обучение сети
H = model.fit(X, encoded_Y, epochs=100, 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()
```