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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Искусственные нейронные сети»

Тема: Бинарная классификация отраженных сигналов радара

Студент гр. 8382 _____ Мирончик П.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Задание

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.

Постановка задачи:

- 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

Ход работы

Данные считываются из файла `sonar.csv` в виде 60 столбцов параметров X и одного столбца Y:

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

Данные столбца Y представлены в виде строк R и M, их нужно перевести в целочисленные значения 0 и 1:

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

Создается и компилируется начальный вариант модели:

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

После этого производится обучение модели и отображаются результаты обучения:

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

`plot_single_history` - функция, использовавшаяся в первой лабораторной работе. Она отображает функции точности и потерь для данных для обучения и проверочных данных:

```
def plot_single_history(history, color="blue"):
    keys = ["loss", "accuracy", "val_loss", "val_accuracy"]
    titles = ["Loss", "Accuracy", "Val loss", "Val accuracy"]
    xlabels = ["epoch", "epoch", "epoch", "epoch"]
    ylabels = ["loss", "accuracy", "loss", "accuracy"]
    ylims = [3, 1.1, 3, 1.1]
    for i in range(len(keys)):
        plot.subplot(2, 2, i + 1)
        plot.title(titles[i])
        plot.xlabel(xlabels[i])
        plot.ylabel(ylabels[i])
        plot.gca().set_ylim([0, ylims[i]])
        plot.grid()
        values = history[keys[i]]
        plot.plot(range(len(values)), values, color=color)
```

Проверим результаты начальной модели (`ИНС1`):

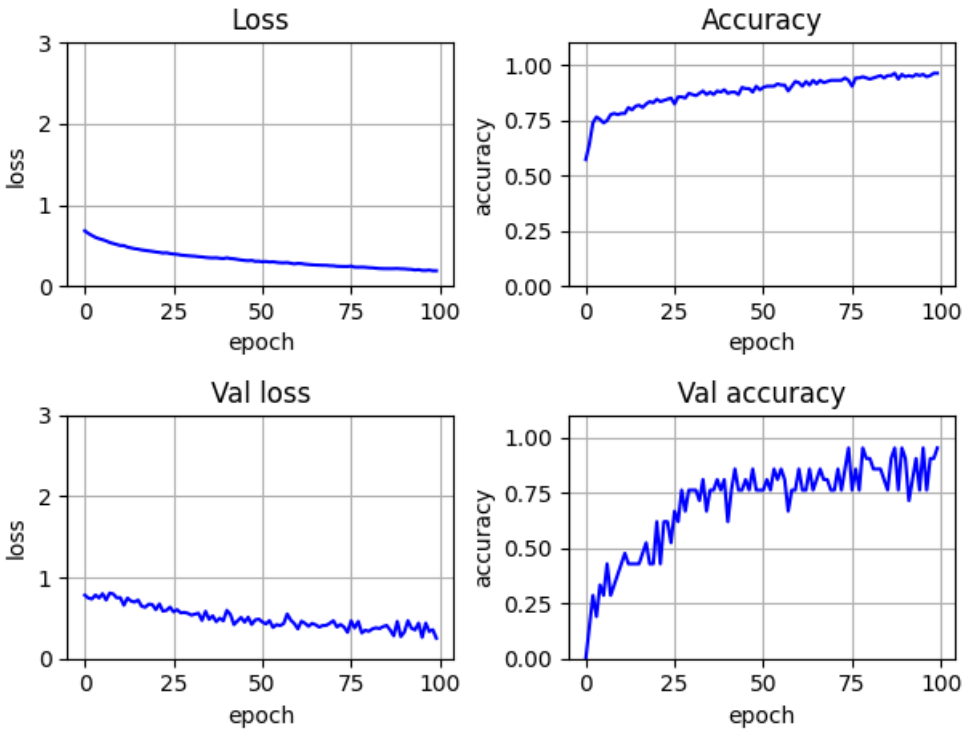

Рис.1,ИНС1

Заметно, что ИНС не достигает высокой точности в пределах 100 эпох как на данных для обучения, так и на данных для проверки.

Проверим версию избыточности входных данных. Для этого уменьшим количество нейронов на входном слое и возьмем первые 30 параметров (ИНС2):

```
X = dataset[:, 0:30].astype(float)
…
model = Sequential()
model.add(Dense(60, input_dim=30, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

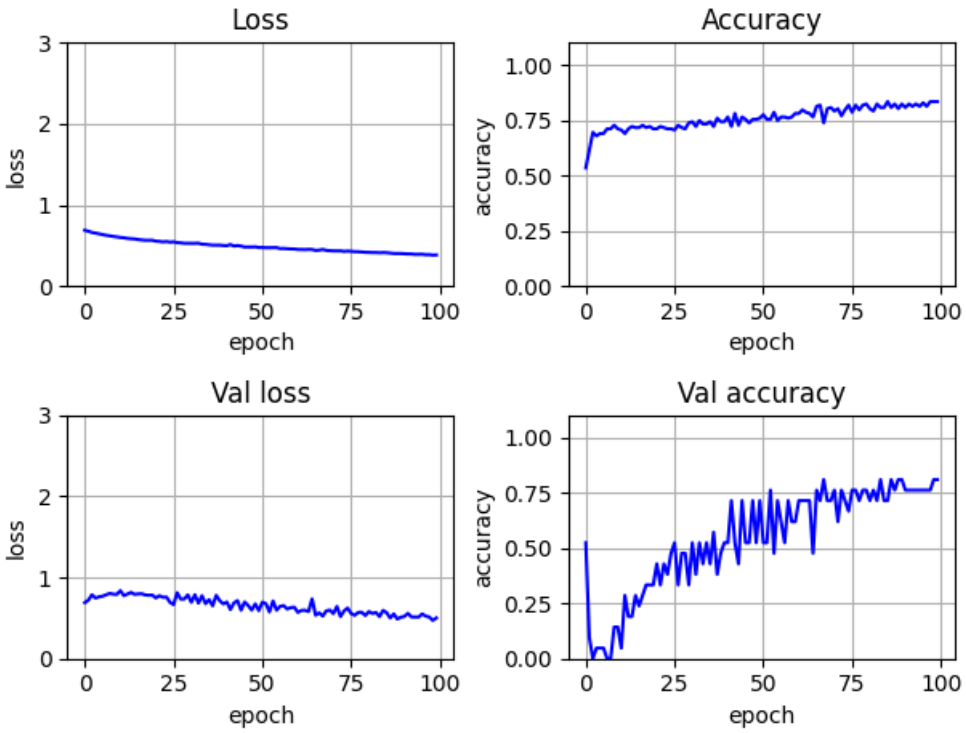

Рис.2, ИНС2

Заметно, что ИНС обучается более стабильно - меньше колебания точности и потерь на проверочных данных. Однако эта ИНС показывает более плохие результаты точности и потерь в сравнении с начальной ИНС.

Попробуем добавить промежуточный слой с 15 нейронами для поиска более сложных зависимостей в данных в ИНС1 (обозначим получившуюся сеть `ИНС3`):

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

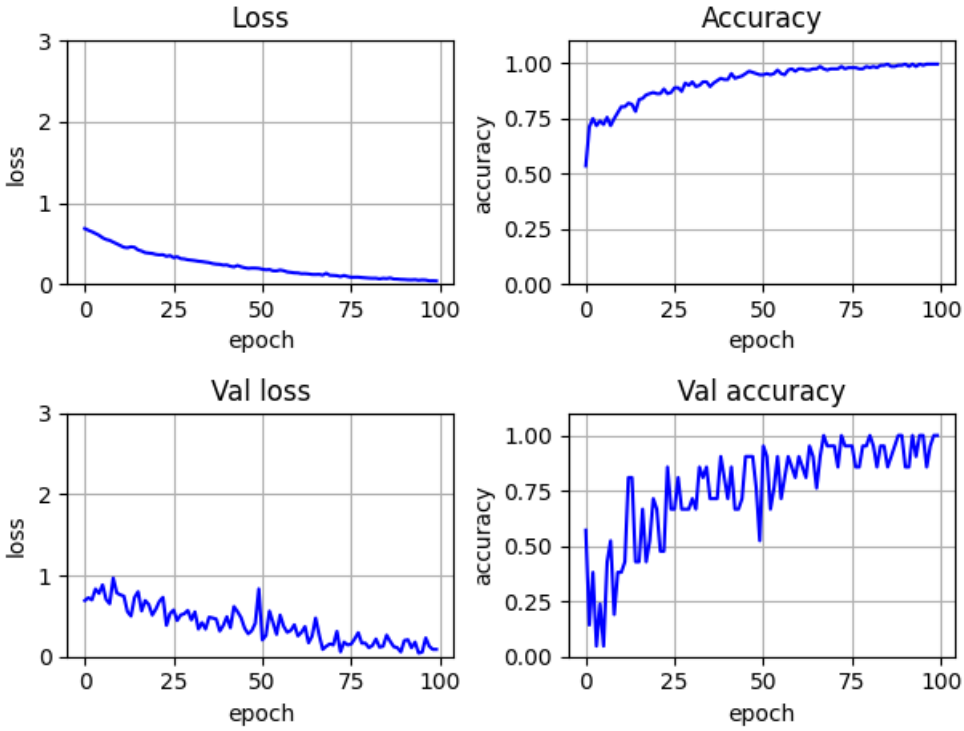

Рис.3, ИНС3

Функции точности и потерь показали значительно лучшие результаты в сравнении с предыдущими ИНС.

Попробуем также добавить скрытый слой к `ИНС2` (назовем сеть `ИНС4`):

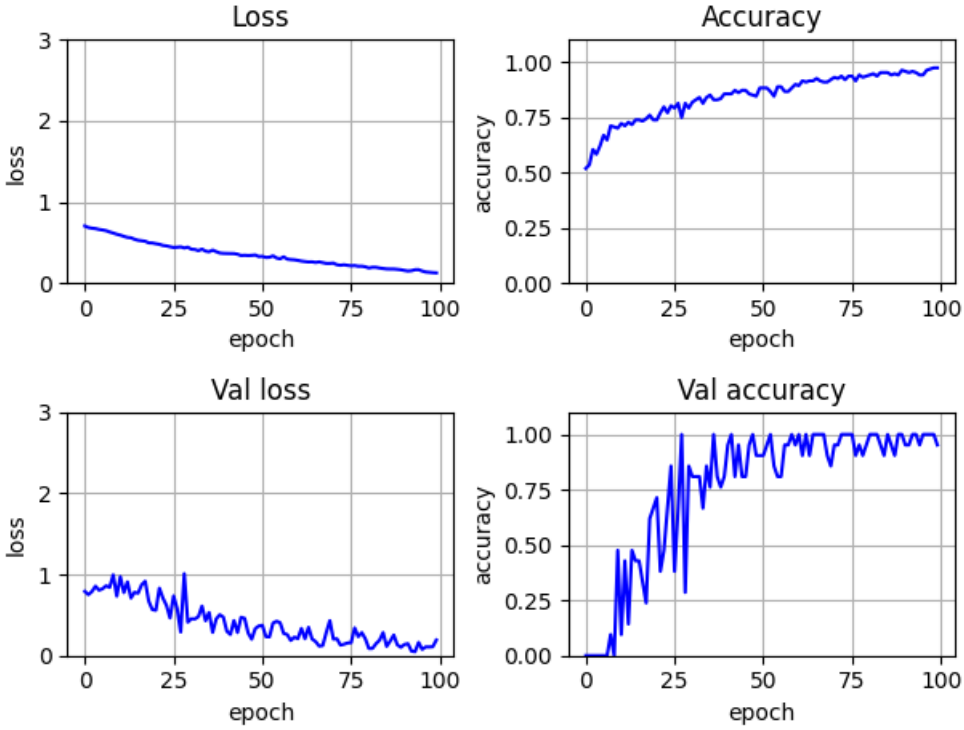

Рис.4, ИНС4

Видно, что `ИНС4` показывает лучшие результаты среди всех сетей как по итоговым результатам точности и потерь, так и по скорости обучения.

Вывод

В ходе выполнения лабораторной работы была построена модель классификации между камнями и металлическими цилиндрами на основе данных об отражении сигналов радара от поверхностей. Экспериментально было изучено влияние количества исходных данных и скрытых слоев на точность результатов и скорость обучения. По результатам исследований выбрана ИНС, показавшая лучшие результаты.

ПРИЛОЖЕНИЕ А

Исходный код программы. main.py

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


def plot_single_history(history, color="blue"):
    keys = ["loss", "accuracy", "val_loss", "val_accuracy"]
    titles = ["Loss", "Accuracy", "Val loss", "Val accuracy"]
    xlabels = ["epoch", "epoch", "epoch", "epoch"]
    ylabels = ["loss", "accuracy", "loss", "accuracy"]
    ylims = [3, 1.1, 3, 1.1]
    for i in range(len(keys)):
        plot.subplot(2, 2, i + 1)
        plot.title(titles[i])
        plot.xlabel(xlabels[i])
        plot.ylabel(ylabels[i])
        plot.gca().set_ylim([0, ylims[i]])
        plot.grid()
        values = history[keys[i]]
        plot.plot(range(len(values)), values, color=color)


dataframe = pandas.read_csv("sonar.csv", header=None)
dataset = dataframe.values
X = dataset[:, 0:30].astype(float)
Y = dataset[:, 60]

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(1, activation='sigmoid'))

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

plot_single_history(H.history)
plot.show()
```