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

ОТЧЕТ

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

Студентка гр. 8383 _____ Кормщикова А.О.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы.

Реализовать классификацию сортов растения ирис (Iris Setosa - 0, Iris Versicolour - 1, Iris Virginica - 2) по четырем признакам: размерам пестиков и тычинок его цветков.

Задачи.

- Ознакомиться с задачей классификации
- Загрузить данные
- Создать модель ИНС в Keras
- Настроить параметры обучения
- Обучить и оценить модель

Ход работы

Были импортированы необходимые для работы классы и функции.

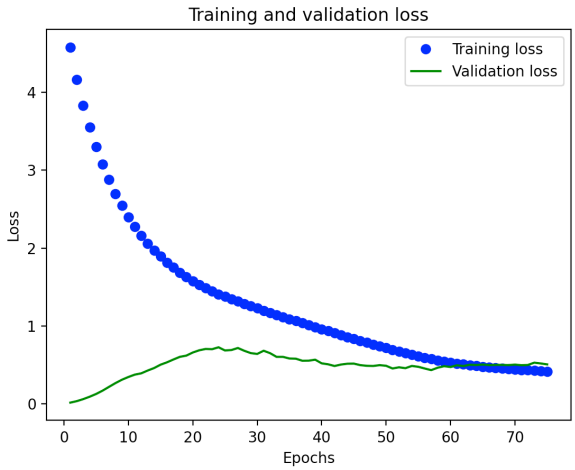
```
import matplotlib.pyplot as plt
import pandas
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
```

С помощью pandas был загружен набор данных, который были разделен на входные(X) и выходные данные(Y), также данные с помощью функции to_categorical() были переведены из вектора в матрицу.

Была построена следующая модель ИНС_1 с двумя слоями:

```
  model.add(Dense(4, activation='relu'))
  model.add(Dense(3, activation='softmax'))
  model.compile(optimizer='Adam',
loss='categorical_crossentrop, metrics=['accuracy'])
  history = model.fit(X, dummy_y, epochs=75, batch_size=10,
 validation_split=0.1)
```

Были построены графики, результаты запуска модели представлены на рис. 1



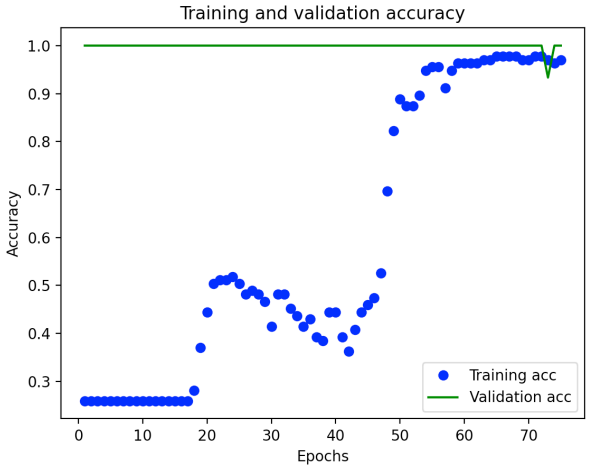


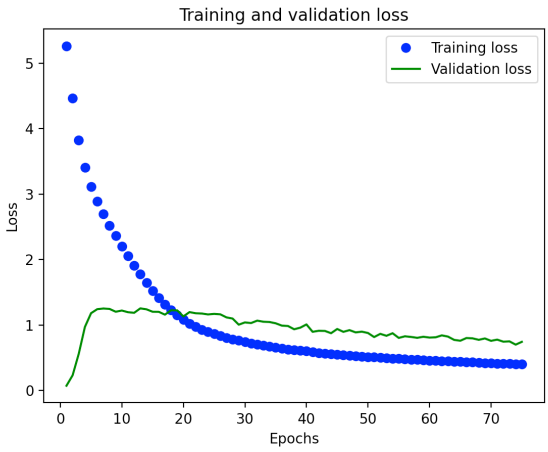
Рисунок 1 - Графики ошибок и точности ИНС_1

На этапе обучения потери снижаются с каждой эпохой, точность, в конечном итоге, выросла. От запуска к запуску данные значительно изменяются.

В архитектуру ИНС были внесены изменения:

Было увеличено количество нейронов во входном слое до 8. Результаты ИНС_2 представлены на рис.2

```
  model = Sequential()
  model.add(Dense(8, activation='relu'))
  model.add(Dense(3, activation='softmax'))
  model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
  history = model.fit(X, dummy_y, epochs=75, batch_size=10,
 validation_split=0.1)
```



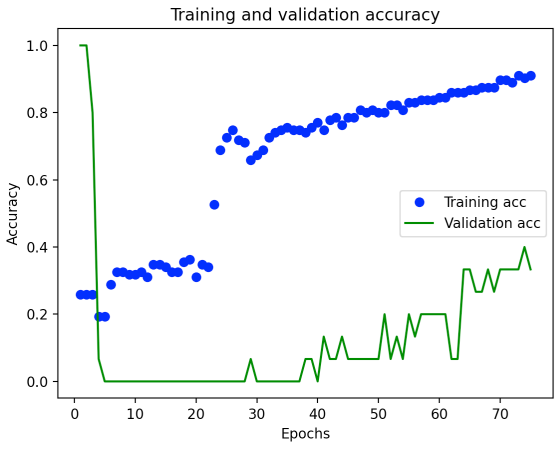


Рисунок 2 - Графики ошибок и точности ИНС_2

Ошибки на тренировочных данных уменьшались быстрее, на проверочных увеличились. Точность на проверочных данных значительно уменьшилась. Стоит отметить, что с каждым запуском данные на графике отличаются.

Был добавлен еще один слой. Количество нейронов на первом и втором слое - 4 Результаты ИНС_3 представлены на рис.3

```
  model.add(Dense(4, activation='relu'))
  model.add(Dense(4, activation='relu'))
  model.add(Dense(3, activation='softmax'))
  model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
  history = model.fit(X, dummy_y, epochs=75, batch_size=10,
validation_split=0.1)
```

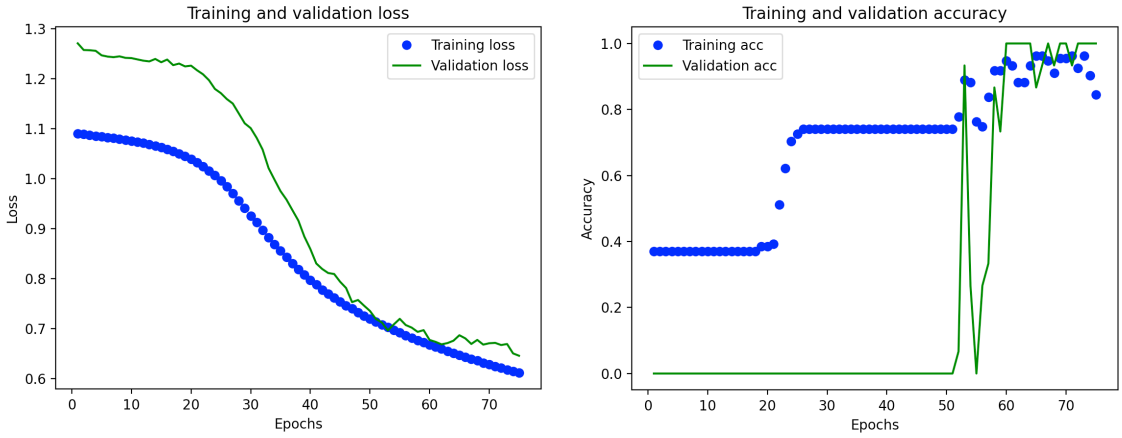


Рисунок 3 - Графики ошибок и точности ИНС_3

Данные с каждым запуском нестабильные. Было увеличено количество нейронов на первом и втором слоях до 16

```
  model.add(Dense(16, activation='relu'))
  model.add(Dense(16, activation='relu'))
  model.add(Dense(3, activation='softmax'))
  model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
  history = model.fit(X, dummy_y, epochs=75, batch_size=10,
validation_split=0.1)
```

Результаты ИНС_4 представлены на рис.4

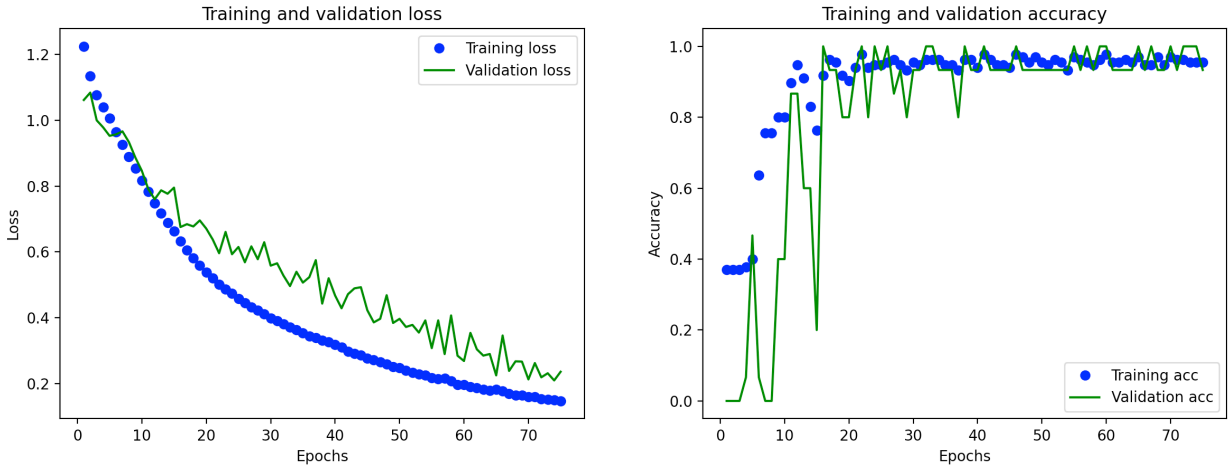


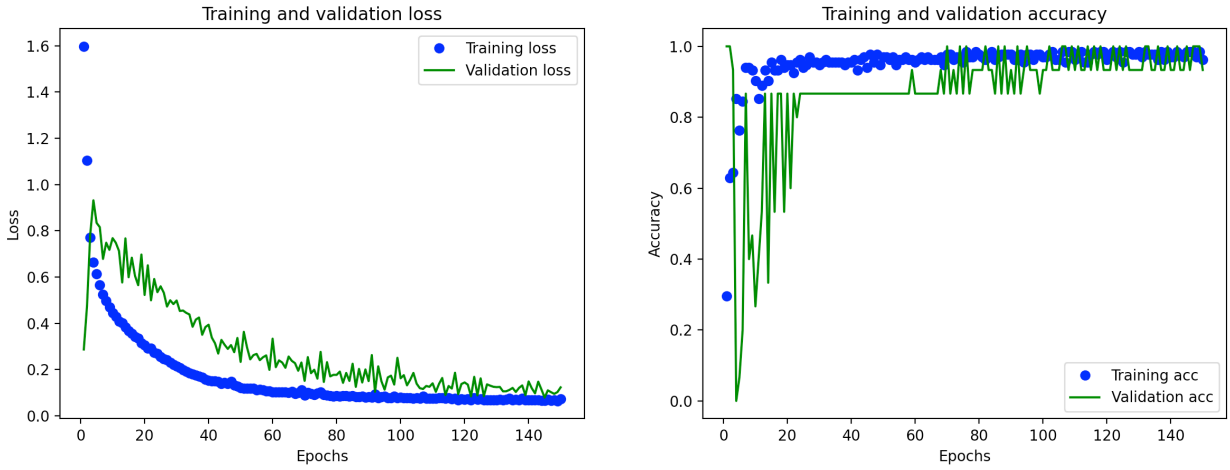
Рисунок 4 - Графики ошибок и точности ИНС_4

Данные, полученные на разных запусках, не так сильно расходятся. Наблюдается увеличение точности и уменьшение ошибок. Эта модель дает самые стабильные результаты с высокой точностью и небольшой ошибкой.

Были рассмотрены различные параметры функции fit. За основу взята четвертая модель.

```
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

Было увеличено количество эпох с 75 до 150 - рис. 5. Также модель была



запущена на 1000 эпох - рис. 6.

Рисунок 5 - Графики ошибок и точности ИНС_4 при 150 эпохах

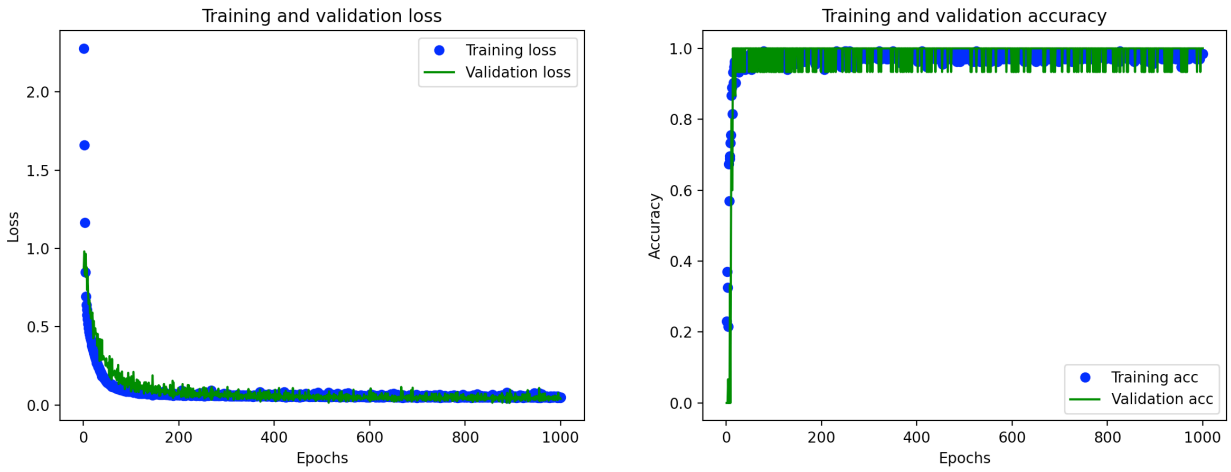


Рисунок 6- Графики ошибок и точности ИНС_4 при 1000 эпохах

Ошибки интенсивно снижаются до 70 эпохи на тренировочных данных, дальше скорость снижения достаточно мала, на проверочных данных это происходит немного позже. Точность также увеличивается достаточно быстро, высокие результаты на проверочных данных достигаются после 100 эпохи.

При 1000 эпох переобучения не произошло, но точность и ошибки не сильно поменялись.

Было оставлено 150 эпох для следующих проверок.

Был изменен параметр validation_split с 0.1 до 0.3. Результаты представлены на рис. 7

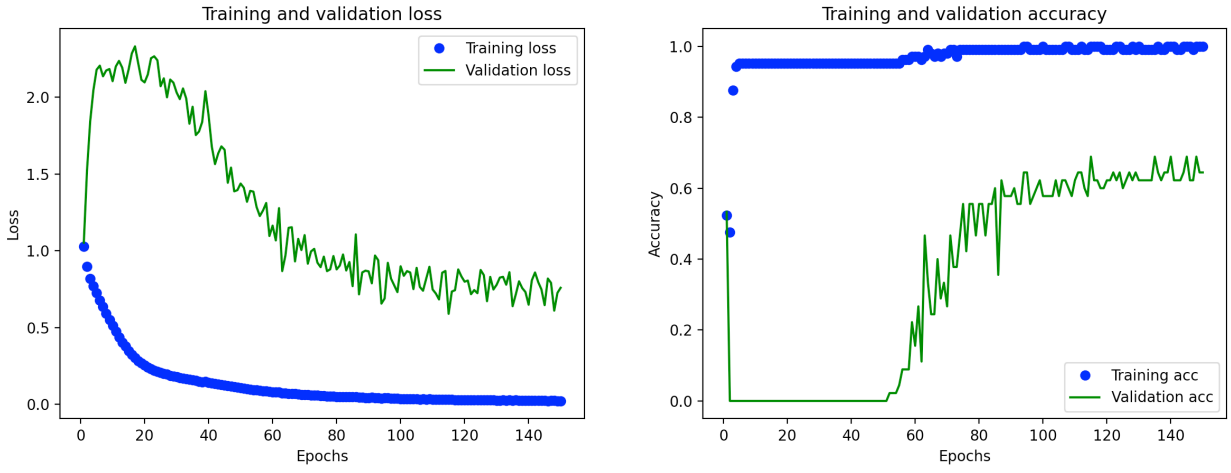


Рисунок 7- Графики ошибок и точности ИНС_4 при validation_split 0.3

Достижение высокой точности и низких ошибок на тренировочных данных не изменилось, однако на проверочных данных значения сильно ухудшились. Из этого можно сделать вывод, что более подходящее значение 0.1

Был изменен параметр batch_size с 10 до 20

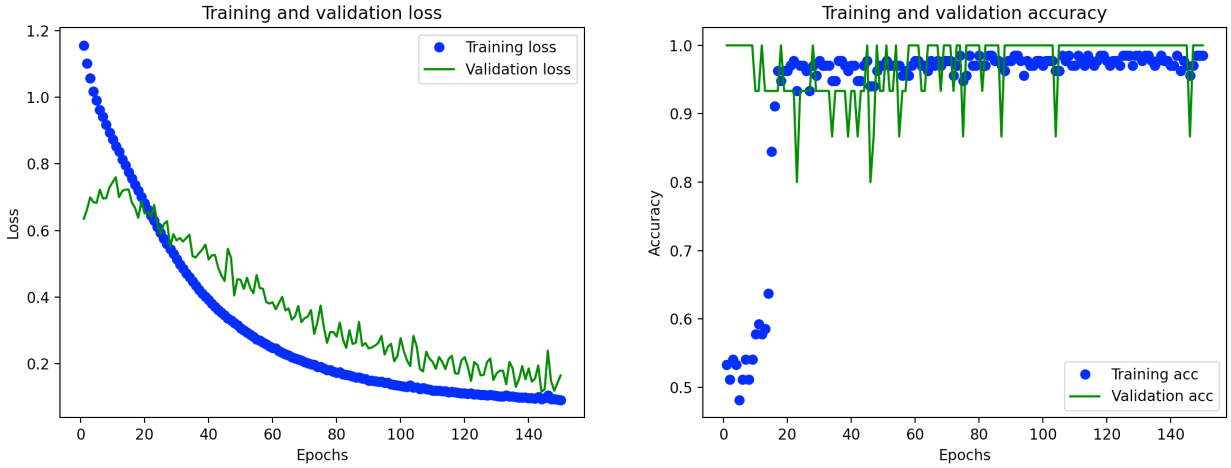


Рисунок 7- Графики ошибок и точности ИНС_4 при batch_size 20

Значение потерь уменьшается не так стремительно, изредка происходят скачки в точности.

Лучше всего показывает себя модель 4 с batch_size 10, epochs 150 и validation_split 0.1

```
  model.add(Dense(16, activation='relu'))
  model.add(Dense(16, activation='relu'))
  model.add(Dense(3, activation='softmax'))
  model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
  history = model.fit(X, dummy_y, epochs=150, batch_size=10,
validation_split=0.1)
```

Выводы.

Во время выполнения лабораторной работы была реализована классификация сортов растения ирис по четырем признакам. Были исследованы различные конфигурации ИНС, построены графики ошибок и точности в ходе обучения, а также была выбрана наилучшая модель.

ПРИЛОЖЕНИЕ А

```
import matplotlib.pyplot as plt  #импорт модуля для графиков
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

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

encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_y = to_categorical(encoded_Y)
# модель
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(X, dummy_y, epochs=150, batch_size=20,
validation_split=0.1)

history_dict = history.history
# график ошибки
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'g', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
# график точности
plt.clf()
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
plt.plot(epochs, acc_values, 'bo', label='Training acc')
plt.plot(epochs, val_acc_values, 'g', label='Validation acc')
```

```
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```