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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Искусственные нейронные сети»

Тема: Многоклассовая классификация цветов

Студент гр.8382 _____ Терехов А.Е.

Преподаватель _____ Жангриров 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

Требования

1. Изучить различные архитектуры ИНС (Разное кол-во слоев, разное кол-во нейронов на слоях).
2. Изучить обучение при различных параметрах обучения (параметры ф-ций fit).
3. Построить графики ошибок и точности в ходе обучения.
4. Выбрать наилучшую модель.

Основные теоретические положения

Задача многоклассовой классификации является одним из основных видов задач, для решения которых применяются нейронные сети. В листинге 1 представлен пример данных.

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
```

Листинг 1: Пример данных

Набор данных доступен для скачивания по ссылке (файл iris.data). Скачанный файл необходимо переименовать в “iris.csv” и поместить в директорию своего проекта.

Импортируем необходимые для работы классы и функции. Кроме Keras понадобится Pandas для загрузки данных и scikit-learn для подготовки данных и оценки модели (листинг 2).

```
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
```

Листинг 2: Подключение модулей

Набор данных загружается напрямую с помощью pandas. Затем необходимо разделить атрибуты (столбцы) на входные данные(X) и выходные данные(Y) (листинг 3).

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

Листинг 3: Загрузка данных

При решении задач многоклассовой классификации хорошей практикой является преобразование выходных атрибутов из вектора в матрицу к виду

представленных в листинге 4.

```
Iris-setosa, Iris-versicolor, Iris-virginica
1,      0,      0
0,      1,      0
0,      0,      1
```

Листинг 4: Представление данных

Для этого необходимо использовать функцию `to_categorical()`(Листинг 5).

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

Листинг 5: Переход от текстовых меток к категориальному вектору

Теперь можно задать базовую архитектуру сети (листинг 6).

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

Листинг 6: Создание модели

Основным строительным блоком нейронных сетей является слой (или уровень), модуль обработки данных, который можно рассматривать как фильтр для данных. Он принимает некоторые данные и выводит их в более полезной форме. В частности, слои извлекают представления из подаваемых в них данных, которые, как мы надеемся, будут иметь больше смысла для решаемой задачи. Фактически методика глубокого обучения заключается в объединении простых слоев, реализующих некоторую форму поэтапной очистки данных. Модель глубокого обучения можно сравнить с ситом, состоящим из последовательности фильтров все более тонкой очистки данных — слоев.

В данном случае наша сеть состоит из последовательности двух слоев Dense, которые являются тесно связанными (их еще называют полносвязными) нейронными слоями. Второй (и последний) слой — это 3-переменный слой потерь (softmax layer), возвращающий массив с 3 оценками вероятностей (в сумме дающих 1). Каждая оценка определяет вероятность принадлежности текущего изображения к одному из 3 классов цветов.

Чтобы подготовить сеть к обучению, нужно настроить еще три параметра для этапа компиляции:

1. функцию потерь, которая определяет, как сеть должна оценивать качество своей работы на обучающих данных и, соответственно, как корректировать ее в правильном направлении;
2. оптимизатор — механизм, с помощью которого сеть будет обновлять себя, опираясь на наблюдаемые данные и функцию потерь;
3. метрики для мониторинга на этапах обучения и тестирования — здесь нас будет интересовать только точность (доля правильно классифицированных изображений).

```
model.compile(optimizer='adam',loss='categorical_crossentropy', metrics=['
    accuracy'])
```

Листинг 7: Инициализация параметров обучения

Теперь можно начинать обучение сети (листинг 8), для чего в случае использования библиотеки Keras достаточно вызвать метод fit сети — он пытается адаптировать (fit) модель под обучающие данные.

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

Листинг 8: Инициализация параметров обучения

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.

Ход работы

Для работы была написана функция, конфигурирующая модель искусственной нейронной сети (листинг 9).

```
def model_configurator(X: pd.array, y: pd.array, hidden_layers: list,
    epochs: int, batch_size: int,
                         validation_split: float):
     model = Sequential()
     model.add(Dense(4, activation='relu'))
     for layer in hidden_layers:
         model.add(Dense(layer.neuron_count, activation=layer.activation))
     model.add(Dense(3, activation='softmax'))
     model.compile(optimizer='adam', loss='categorical_crossentropy',
    metrics=['accuracy'])
     return model.fit(X, y, epochs=epochs, batch_size=batch_size,
    validation_split=validation_split)
```

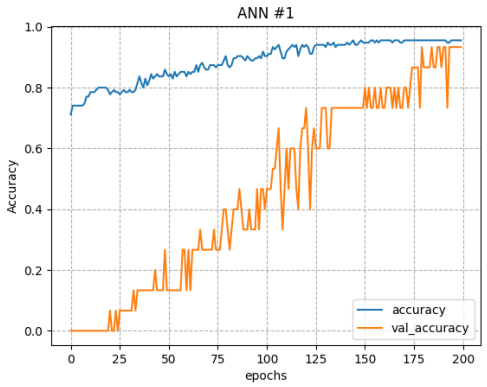
Листинг 9: Конфигуратор модели

Для вывода графиков точности и ошибок написана функция, представленная в листинге 10.

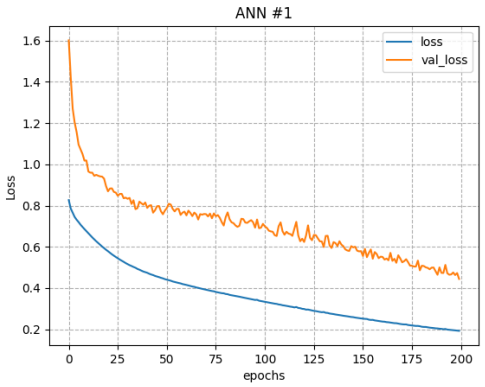
```
def plot(data: pd.DataFrame, label: str, title: str):
     axis = sns.lineplot(data=data)
     axis.set(ylabel=label, xlabel='epochs', title=title)
     plt.show()
```

Листинг 10: Вывод графиков

Для начала запустим программу с параметрами из методических указаний, только изменим количество эпох до 200. Графики потерь и точности первой сети представлены на рисунках 1a и 1b.



(a) Точность

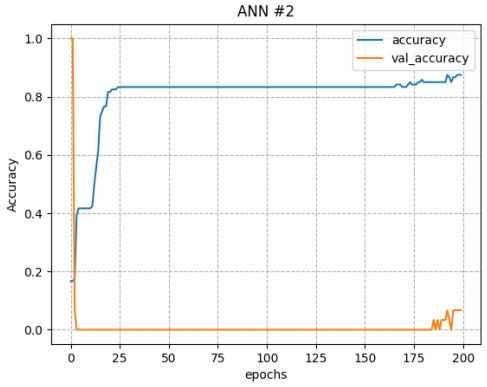


(b) Потери

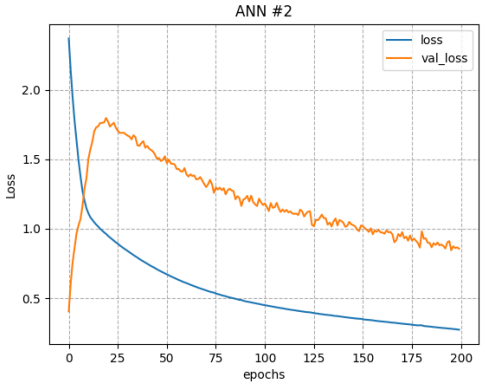
Рис. 1: Графики для первой ИНС

Из графиков видно, что такая сеть за 200 эпох смогла неплохо уменьшить показатели ошибок, но по графику точности нельзя утверждать, что сеть смогла достичь стабильных результатов, тем не менее на 200 эпохе точность на валидационных данных составила примерно 90%.

Теперь попробуем изменить параметр `validation_split` до 20% и 60%. Результаты обучения представлены на рисунках 2 и 3.

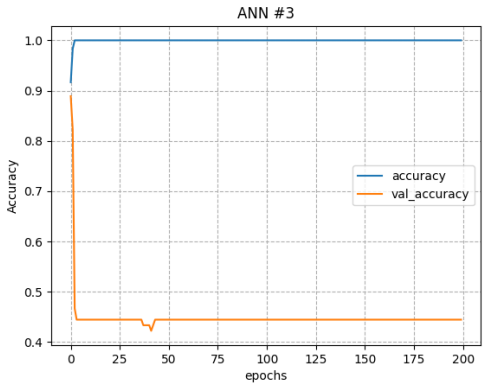


(a) Точность

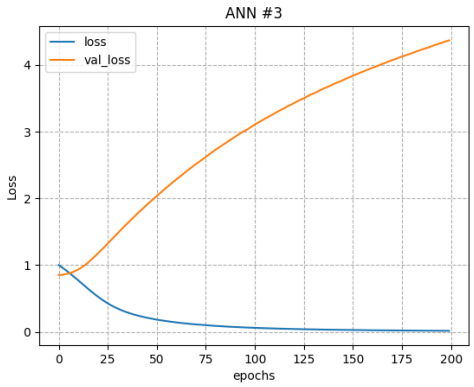


(b) Потери

Рис. 2: Графики для ИНС при validation_split=0.2



(a) Точность

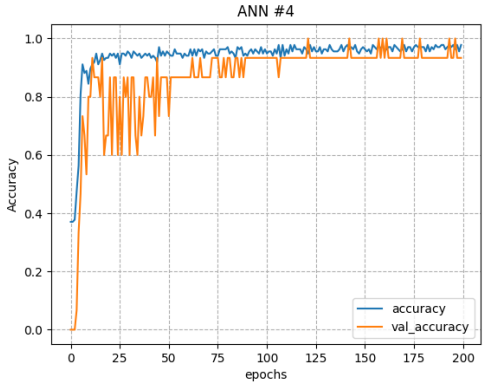


(b) Потери

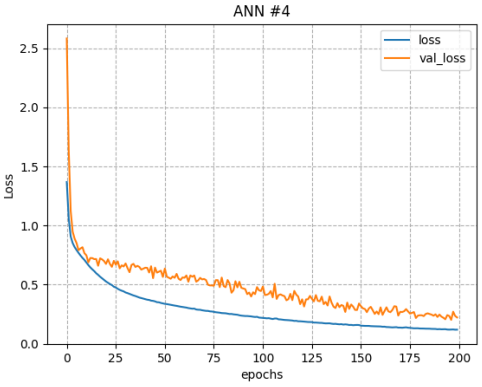
Рис. 3: Графики для ИНС при validation_split=0.6

По графикам можно сделать вывод, что увеличение доли валидационных данных не приводит ни к чему хорошему, графики расходятся и сеть не может обучиться за 200 эпох. Это можно объяснить тем, что сети не хватает тренировочных данных.

При изменении параметра `batch_size` до 3 и 30 были получены графики, представленные на рисунках 4-5.

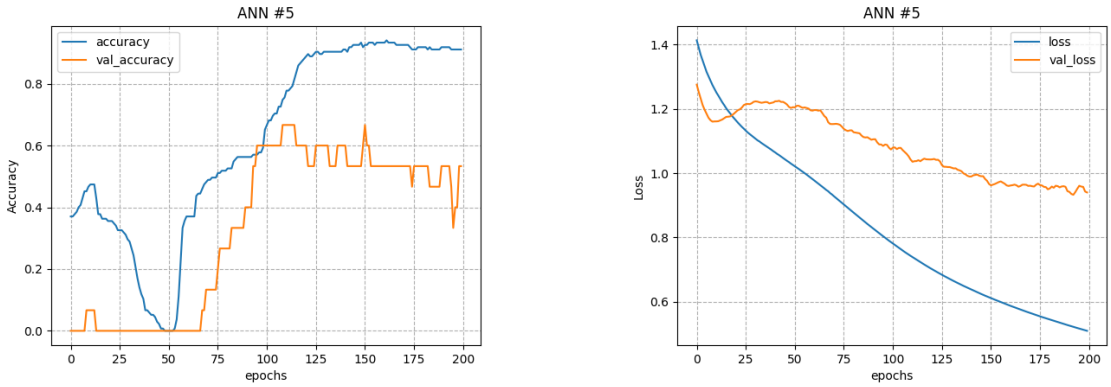


(a) Точность



(b) Потери

Рис. 4: Графики для ИНС при batch_size=3

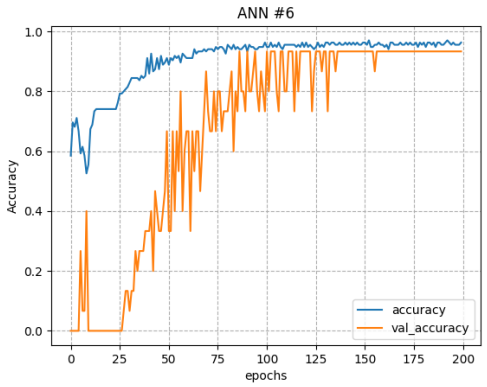


(a) Точность (b) Потери

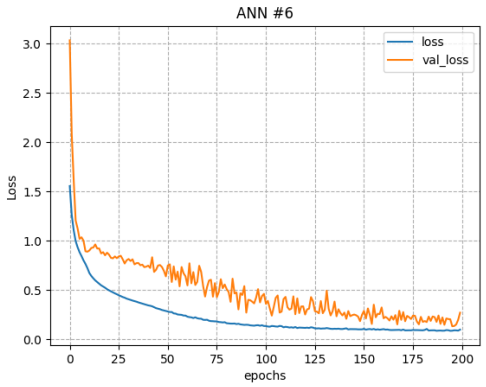
Рис. 5: Графики для ИНС при batch_size=30

Параметр `batch_size` отвечает за количество обработанных строк до обновления весов. То есть чем меньше значения, тем чаще будут обновляться веса, но растет вероятность застрять в локальном минимуме. При увеличении этого значения будет проводиться меньше вычислений, а значит сеть будет быстрее работать, но точность может упасть, а ошибки возрасти.

При добавлении одного скрытого слоя с функцией активации relu и количеством нейронов 8 были получены графики предстваленные на рисунках 6a и 6b. Обучение происходило при параметрах: `validation_split=0.1` и `batch_size=3`.



(a) Точность



(b) Потери

Рис. 6: Графики для ИНС с одним скрытым слоем ($n = 8$)

Если сравнивать данную конфигурацию с исходной (без скрытых слоев) при таких же параметрах обучения (рисунок 4), то можно сделать вывод, что добавление скрытых слоев не сильно повлияло на обучение модели. Даже можно сказать, что при отсутствии скрытых слоев модель обучилась быстрее. Уже к 100 эпохам были получены более-менее стабильные результаты, чего нельзя сказать о модели со скрытым слоем.

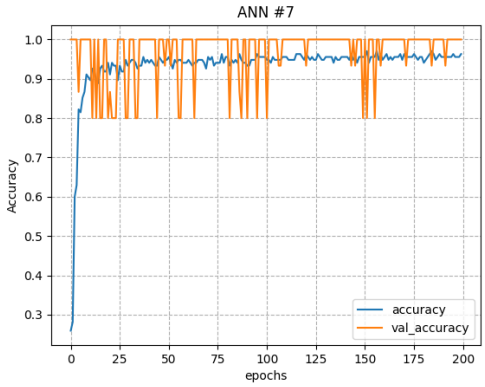
Следующие три исследованных ИНС имеют слои, представленные в таблице 1.

Таблица 1: Скрытые слои сетей

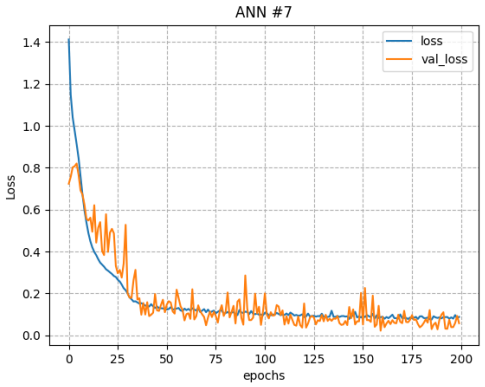
№ сети	Функция активации	Количество нейронов
7	relu	8
	relu	8
8	relu	16
9	relu	16
	relu	16

При их обучении были получены графики, представленные на рисун-

ках 7-9.

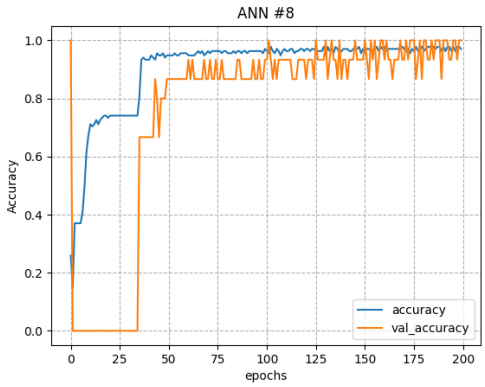


(a) Точность

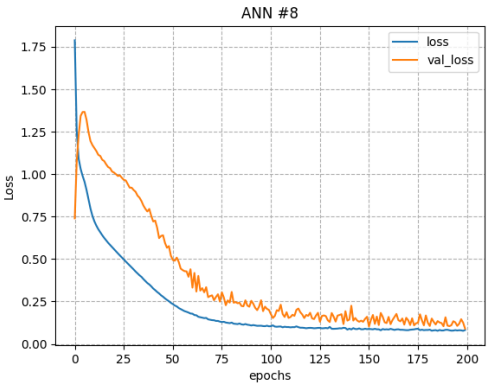


(b) Потери

Рис. 7: Графики для ИНС с двумя скрытыми слоями ($n_1 = n_2 = 8$)

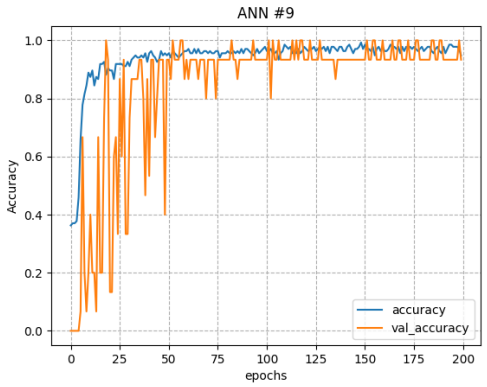


(a) Точность

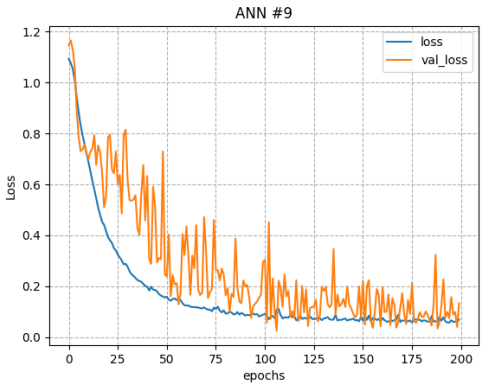


(b) Потери

Рис. 8: Графики для ИНС с одним скрытым слоем ($n = 16$)



(a) Точность



(b) Потери

Рис. 9: Графики для ИНС с двумя скрытыми слоями ($n_1 = n_2 = 16$)

Из полученных сетей можно выделить седьмую, с двумя слоями по 8 нейронов, она смогла быстро минимизировать потери, примерно за 30 эпох до уровня 0.17-0.18. Также неплохие результаты показала следующая сеть с одним слоем в 16 нейронов, но по сравнению с предыдущей она обучалась значительно дольше. Остальные сети не продемонстрировали ничего выдающегося, лишь вели себя достаточно нестабильно.

Вывод

В ходе данной лабораторной работы была написана искусственная нейронная сеть для классификации сортов ириса. Были исследованы конфигурации сетей представленные в таблице 2.

Таблица 2: Скрытые слои и параметры обучения всех сетей

№ сети	Ф-ция акт-ции	Кол-во нейронов	batch_size	val_split
1	-	-	10	0.1
2	-	-	10	0.2
3	-	-	10	0.6
4	-	-	3	0.1
5	-	-	30	0.1
7	relu	8	3	0.1
	relu	8		
8	relu	16	3	0.1
9	relu	16	3	0.1
	relu	16		

Из всех сетей можно выделить 4, 6, 7 и 8, но следует учитывать, что многое зависит от начального выбора весов, которые устанавливаются случайным образом.