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»

Студент гр. 8383 _____ Киреев К.А.

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

Требования

- Изучить различные архитектуры ИНС (Разное кол-во слоев, разное кол-во нейронов на слоях);
- Изучить обучение при различных параметрах обучения (параметры функции fit);
- Построить графики ошибок и точности в ходе обучения;
- Выбрать наилучшую модель;

Выполнение работы

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

Листинг 1 – 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
```

Был скачан файл iris.data и переименован в iris.csv. Набор данных состоит из данных о 150 экземплярах ириса, по 50 экземпляров из трёх видов — *Iris setosa*, *Iris virginica* и *Iris versicolor*. Для каждого экземпляра измерялись четыре характеристики: длина наружной доли околоцветника; ширина наружной доли околоцветника; длина внутренней доли околоцветника; ширина внутренней доли околоцветника.

Набор данных загружается напрямую с помощью библиотеки pandas. Затем были разделены атрибуты (столбцы) на входные данные(X) и выходные данные(Y). Далее выходные данные были приведены к нормальному виду.

Листинг 2

```
df = pd.read_csv('iris.csv', header=None)
dataset = df.values
X = dataset[:,0:4].astype(float)
Y = dataset[:,4]

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

Далее была создана нейронная сеть. В данном случае сеть состоит из последовательности двух слоев Dense, которые являются тесно связанными нейронными слоями. Второй 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. Далее были выбраны параметры для этапа компиляции и параметры для обучения сети. Количество эпох выставлено в значение 150.

Листинг 3

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

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

H = model.fit(X, dummy_y, epochs=150, batch_size=10, validation_split=0.1, verbose=2)
```

Далее были построены графики ошибок и точности в ходе обучения.

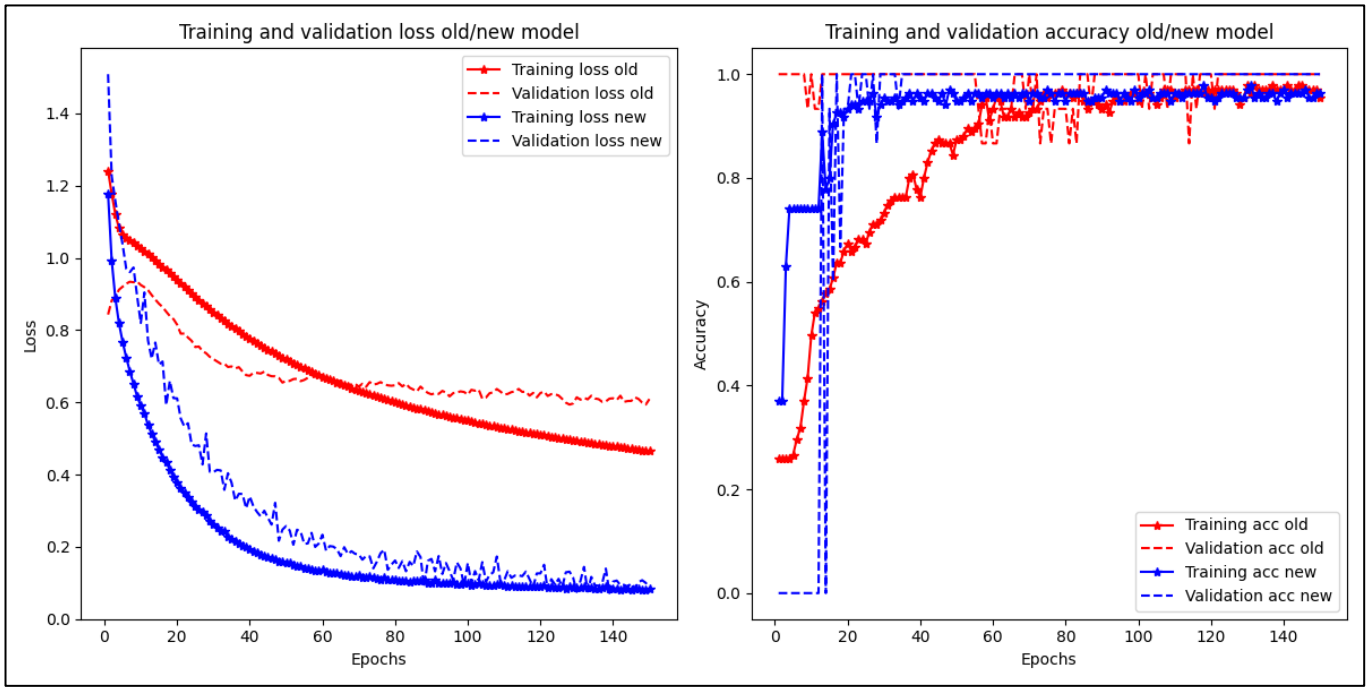
Листинг 4

```
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)

fig = plt.figure(figsize=(12,6))
gs = gridspec.GridSpec(1, 2, width_ratios=[3, 3])
plt.subplot(gs[0])
plt.plot(epochs, loss, 'r--', label='Training loss')
plt.plot(epochs, val_loss, 'g--', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(gs[1])
plt.plot(epochs, acc, 'r--', label='Training acc')
plt.plot(epochs, val_acc, 'g--', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

Изучение различных архитектур ИНС

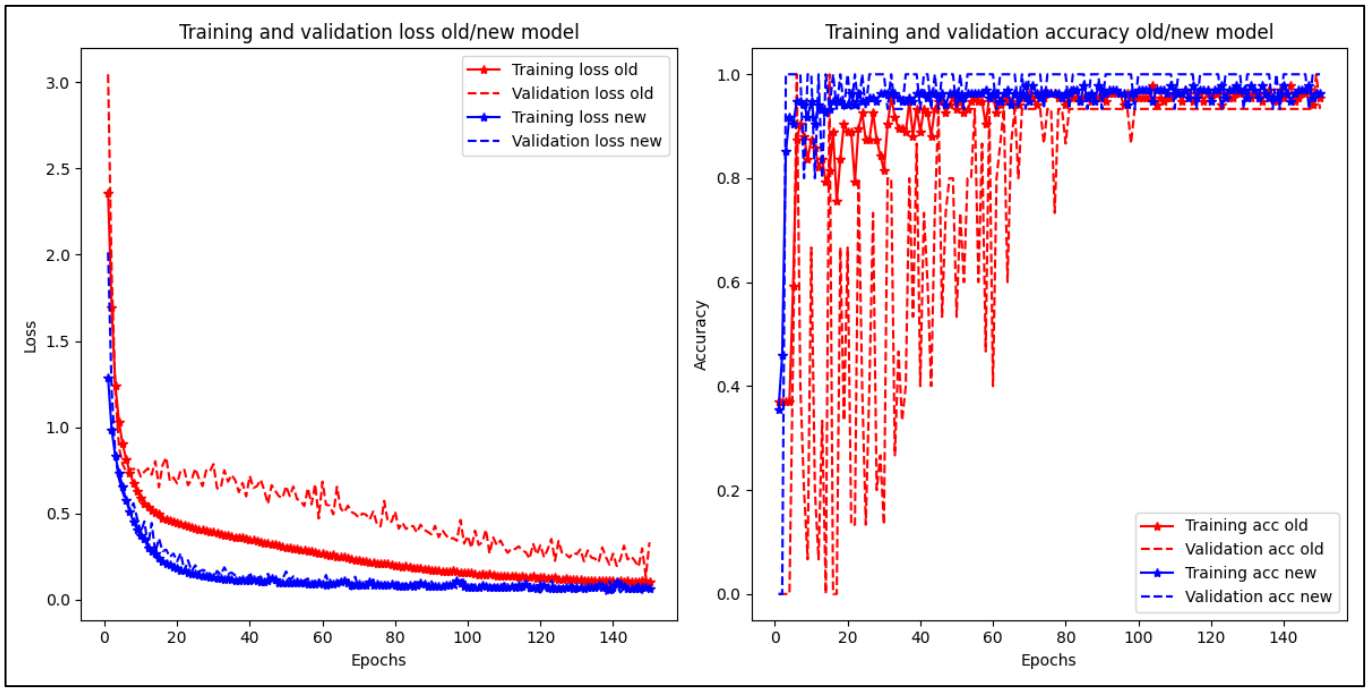
- **Изменение слоев в ИНС**

Далее на графиках синим изображены данные измененной нейронной сети, а красным старой. Графики слева – ошибки, справа – точность. Был добавлен новый скрытый слой с 24 нейронами.



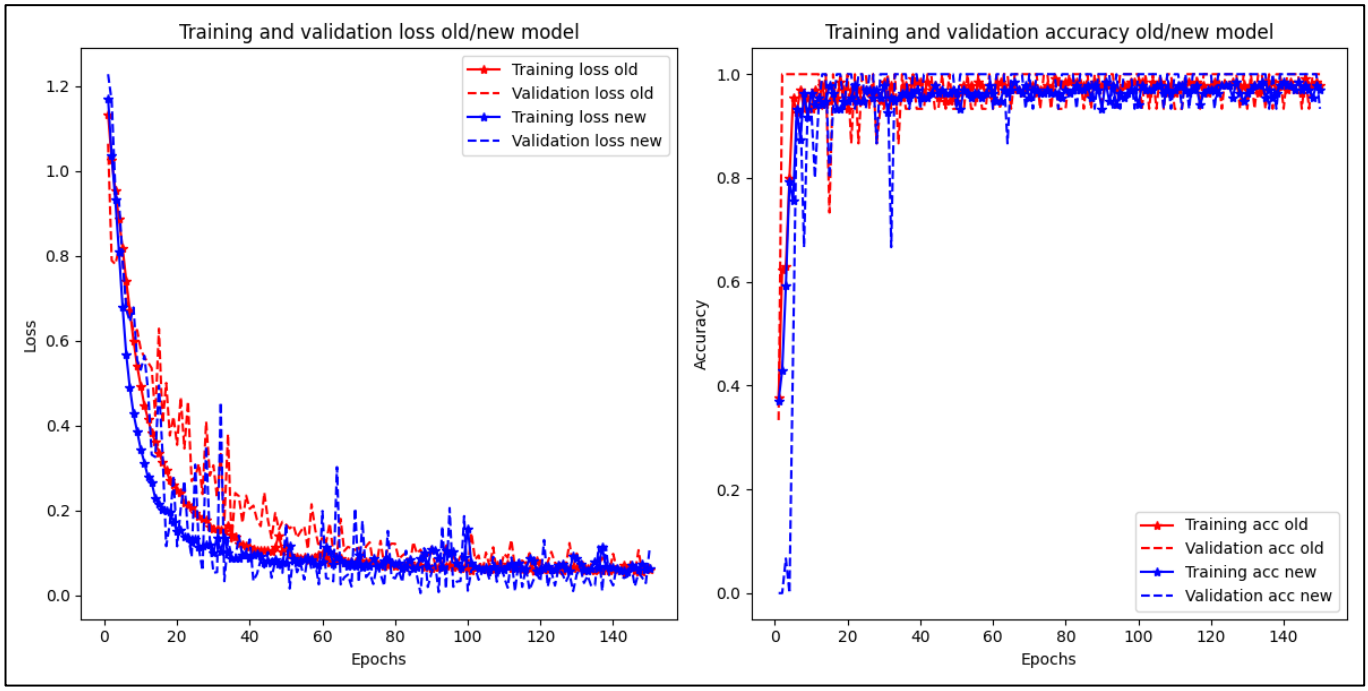
Видно, что в новой сети быстрее уменьшается показатель ошибок, а точность уже около 20 эпохи показывает стабильные результаты.

Был добавлен четвёртый слой с 24 нейронами.



Результаты еще немного улучшились.

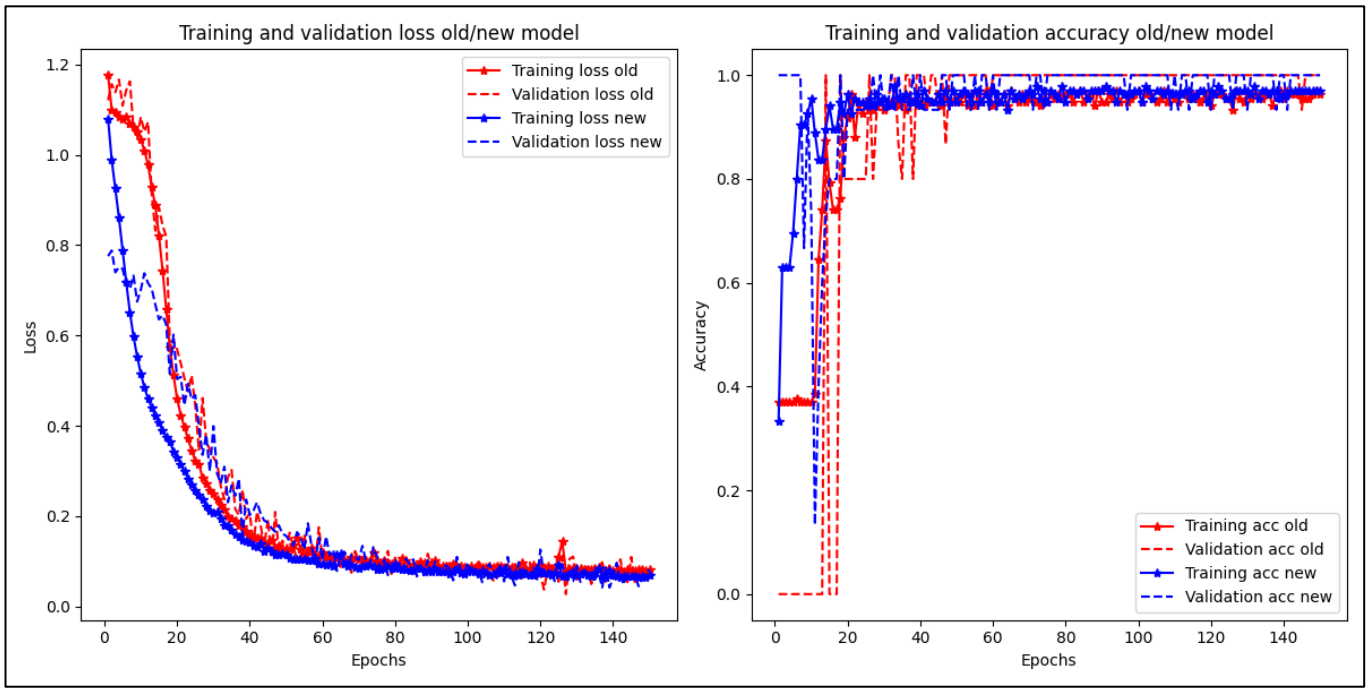
Был добавлен пятый слои с 24 нейронами.



Значительных отличий не наблюдается, поэтому добавлять новые слои не имеет смысла.

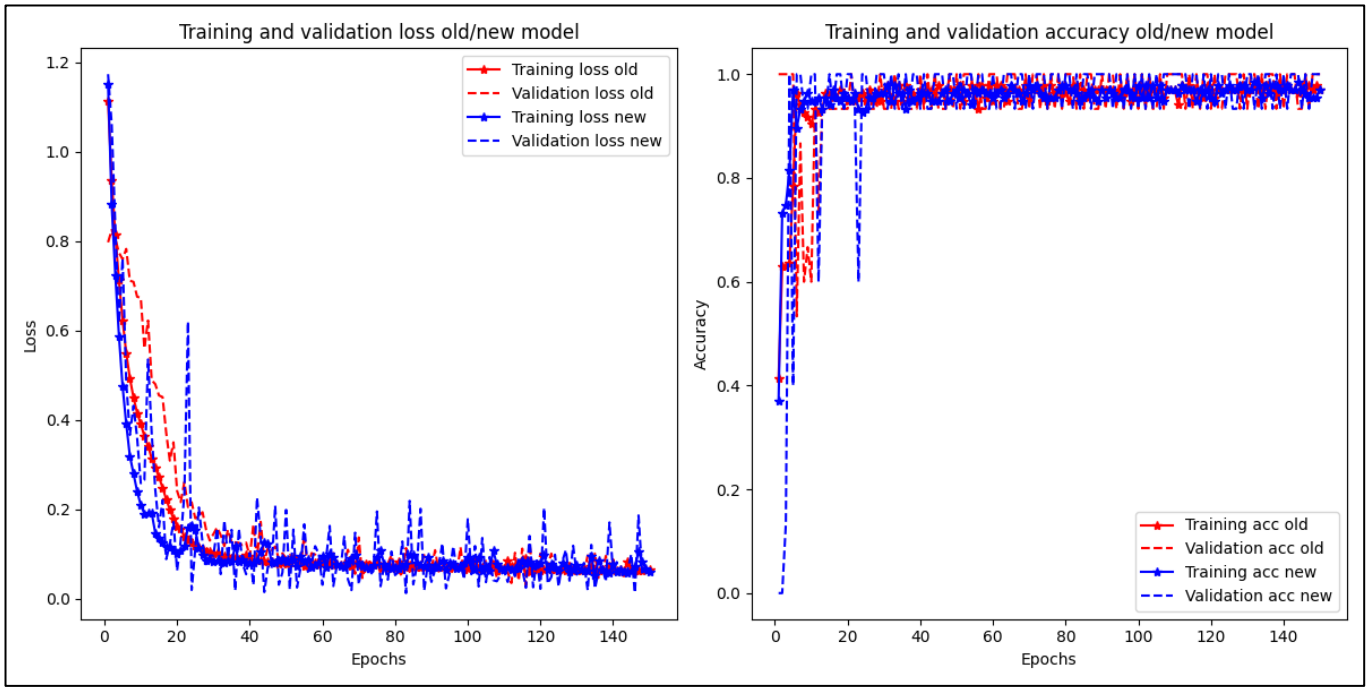
- **Изменение количества нейронов в слоях ИНС**

Было изменено количество нейронов во входном слое с 4 до 8.



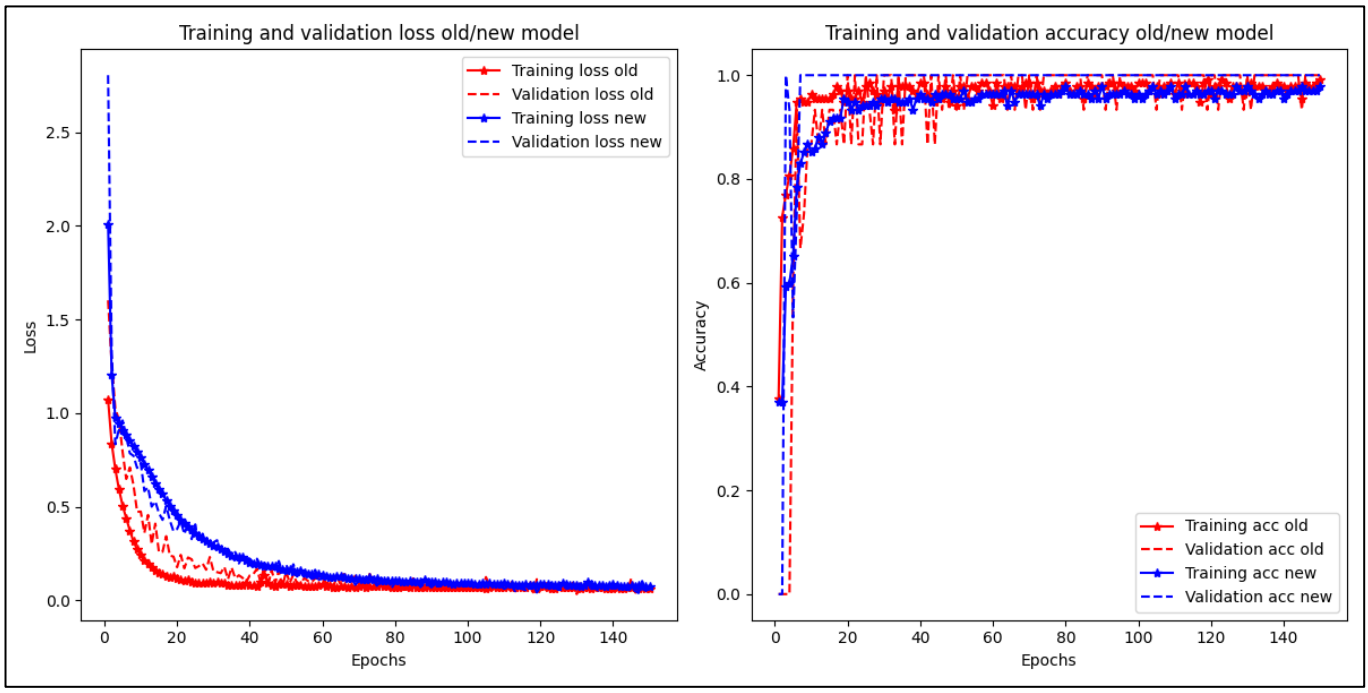
Видно, что в новой сети быстрее уменьшается показатель ошибок, а точность раньше показывает стабильные результаты.

Было изменено количество нейронов на втором и третьем слое до 96 и 72 соответственно.



Значительных отличий не наблюдается, поэтому добавлять нейроны нет необходимости.

Было изменено количество нейронов на втором и третьем слое на 12 и 10 соответственно.



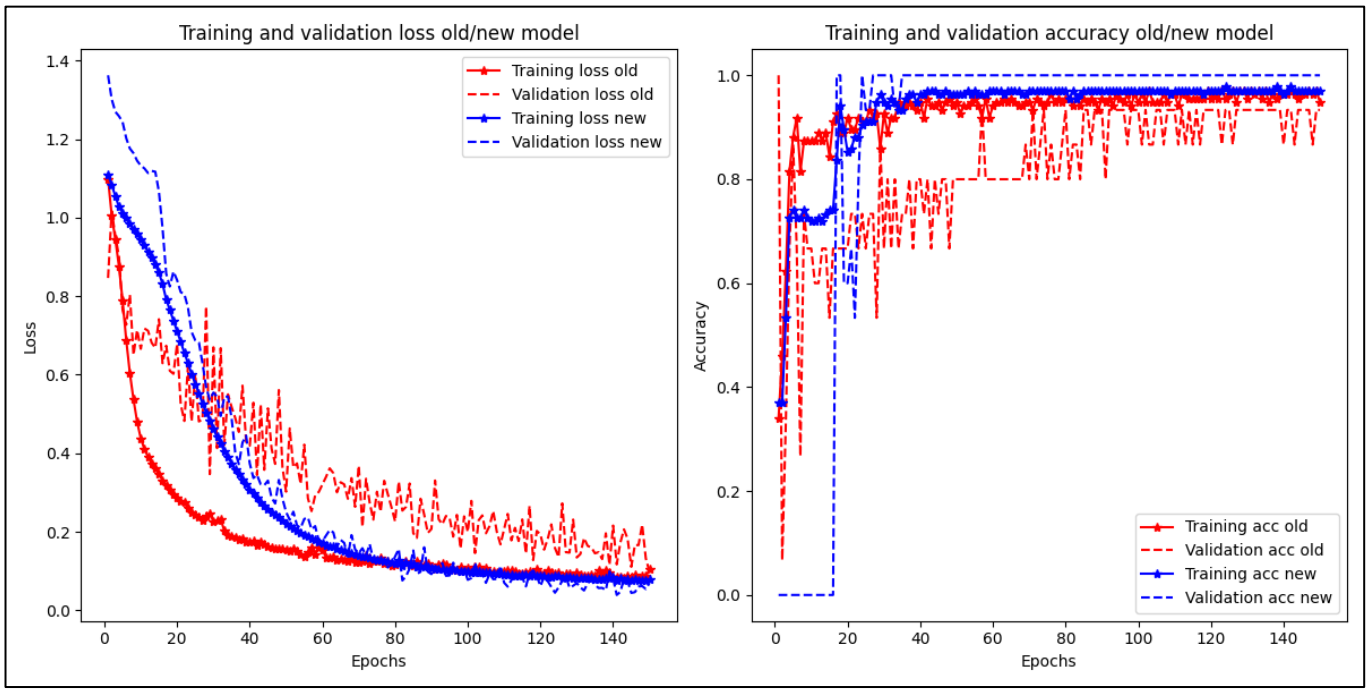
Результаты как на графике ошибок, так и на графике точности ухудшились, поэтому изменять количество нейронов нет смысла.

Изучение обучения при различных параметрах обучения

- **Изменение значения переменной batch_size**

Переменная указывает на общее число тренировочных объектов, представленных в одном батче. При увеличении параметра уменьшается количество итераций в одной эпохе, но зато уменьшается точность обучения.

Была изменена переменная batch_size на 25.

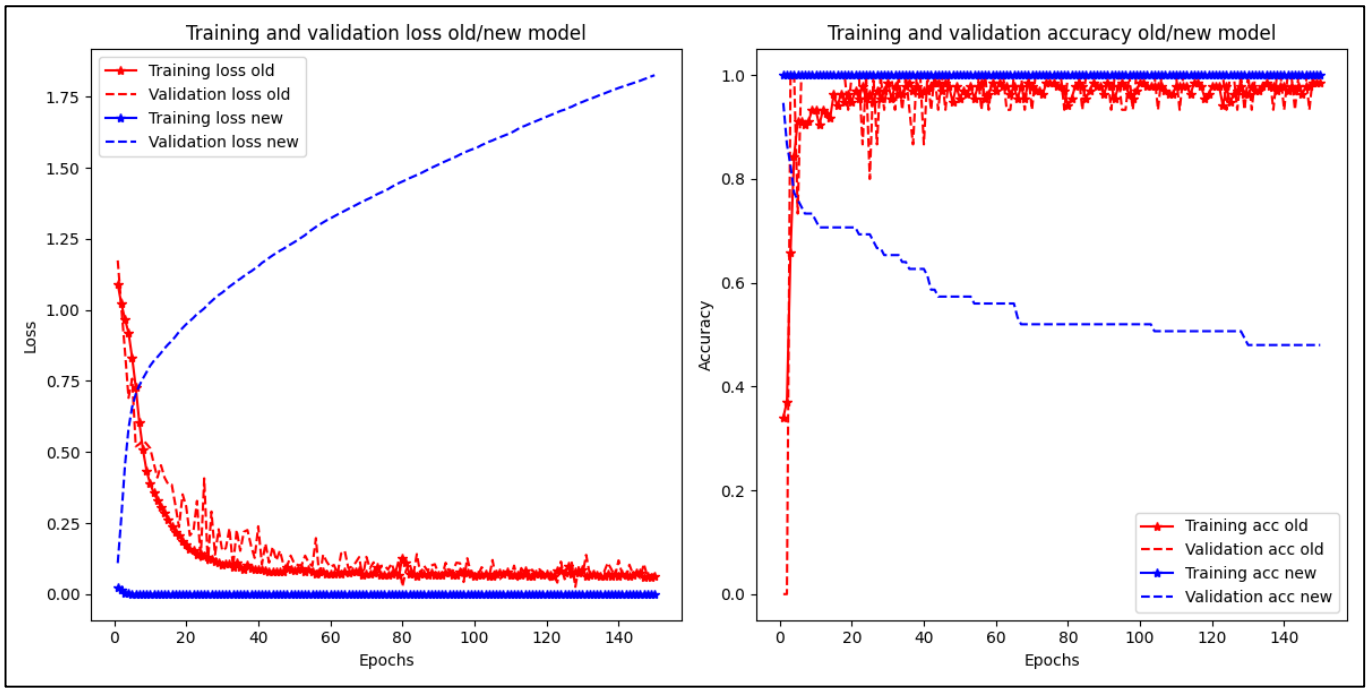


Как и ожидалось результаты ухудшились, поэтому оставляем 10.

- **Изменение значения переменной validation_split**

Переменная указывает на количество данных для обучения.

Была изменена переменная validation_split на 0.5.



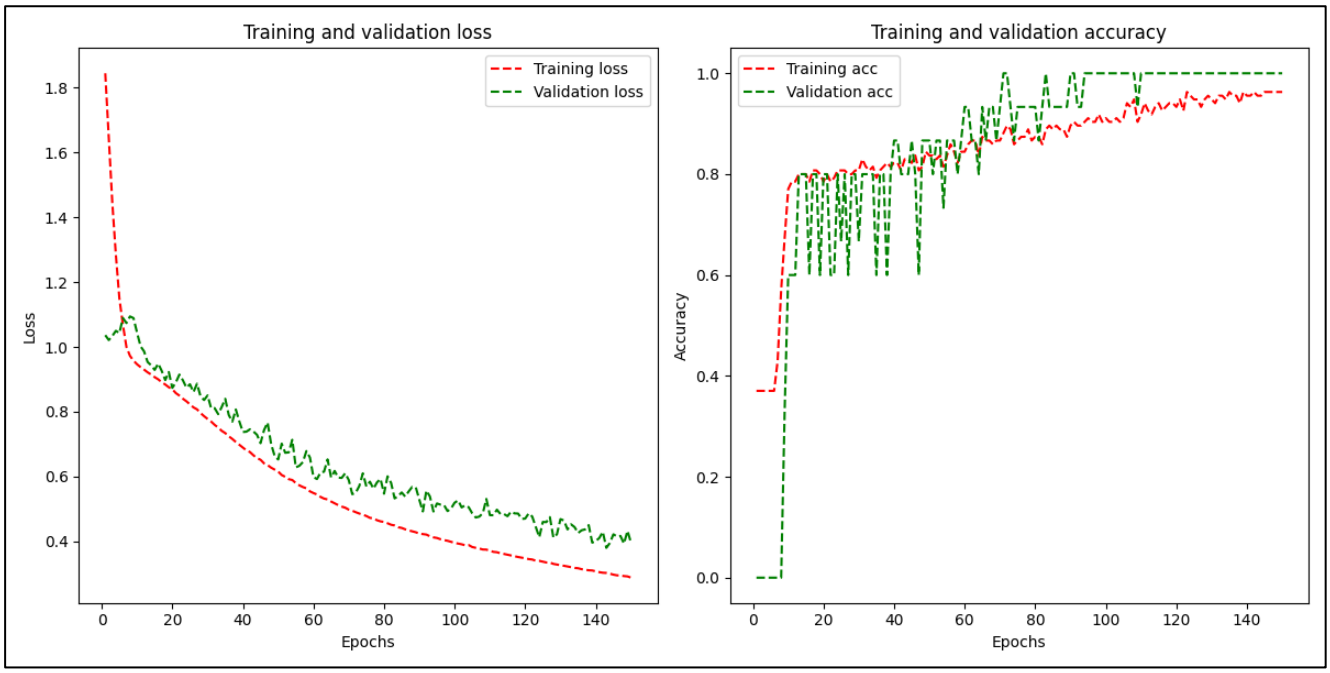
Видно, что сети не хватает данных для корректного обучения. Поэтому оставляем 0.1.

Можно сделать вывод, что лучшая конфигурация сети:

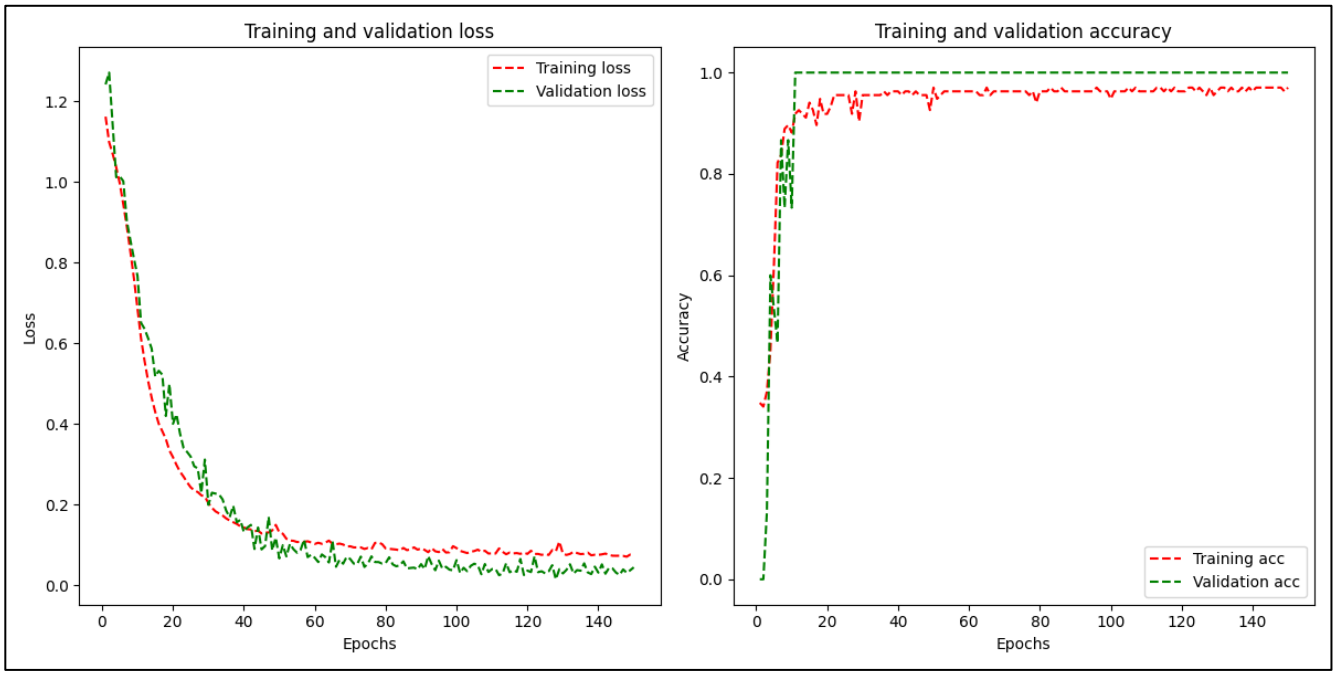
```
model = Sequential()
model.add(Dense(8, input_shape=(4,), activation='relu'))
model.add(Dense(24, activation='relu'))
model.add(Dense(24, activation='relu'))
model.add(Dense(3, activation='softmax'))

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

H = model.fit(X, dummy_y, epochs=150, batch_size=10, validation_split=0.1, verbose=2)
```

Далее представлены графики ошибок и точности для первоначальной и конечной модели.

Начальная модель:



Конечная модель:



Выводы

В ходе выполнения лабораторной работы были получены базовые теоретические и практические навыки по работе с ИНС и библиотекой Keras.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД. ФАЙЛ TASK.PY

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '1'
import pandas as pd
import numpy as np
import tensorflow.keras
from tensorflow.keras import layers
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
from matplotlib import gridspec

df = pd.read_csv('iris.csv', header=None)
dataset = df.values
X = dataset[:,0:4].astype(float)
Y = dataset[:,4]

encoder = LabelEncoder() # sklearn
encoder.fit(Y) # sklearn
encoded_Y = encoder.transform(Y) # sklearn
dummy_y = to_categorical(encoded_Y) # keras

model = Sequential()
model.add(Dense(8, input_shape=(4,), activation='relu'))
model.add(Dense(24, activation='relu'))
model.add(Dense(24, activation='relu'))
model.add(Dense(3, activation='softmax'))

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

H = model.fit(X, dummy_y, epochs=150, batch_size=10,
validation_split=0.1, verbose=2)


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)
```

```
fig = plt.figure(figsize=(12,6))
gs = gridspec.GridSpec(1, 2, width_ratios=[3, 3])
plt.subplot(gs[0])
plt.plot(epochs, loss, 'r--', label='Training loss')
plt.plot(epochs, val_loss, 'g--', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(gs[1])
plt.plot(epochs, acc, 'r--', label='Training acc')
plt.plot(epochs, val_acc, 'g--', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```