****

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

ИНСТИТУТ ИНФОРМАТИКИ И КИБЕРНЕТИКИ

Кафедра программных систем

Дисциплина

Нейронные сети

ОТЧЕТ

по лабораторной работе №4

«Использование библиотеки PyTorch»

Студент: Соколова А.Д.

Группа: 6301-020302D

Проверил:

профессор Тюгашев А.А.

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара  
2025

СОДЕРЖАНИЕ

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc197905845)

[1 Исходный текст программы 4](#_Toc197905846)

[2 Протокол исполнения 6](#_Toc197905847)

[ЗАКЛЮЧЕНИЕ 7](#_Toc197905848)

# ПОСТАНОВКА ЗАДАЧИ

Цель работы заключается в создании и обучении нейронной сети с использованием библиотеки TensorFlow/Keras для решения задачи многоклассовой классификации данных из набора Iris.

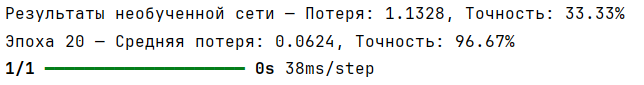
В рамках выполнения работы предполагается выполнение следующих задач:

1. реализация простой полносвязной нейронной сети;
2. обучение модели на датасете Iris;
3. оценка эффективности работы модели на тестовых данных, включая измерение точности и потерь;
4. визуализация процесса обучения и анализа потерь модели по батчам и по эпохам.
5. Исходный текст программы

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.datasets import load\_iris  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler, LabelBinarizer  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense, Input  
from tensorflow.keras.optimizers import Adam  
from tensorflow.keras.callbacks import Callback  
  
*# 1. загрузка и подготовка данных*iris = load\_iris()  
X = iris.data  
y = iris.target  
class\_names = iris.target\_names  
  
scaler = StandardScaler()  
X = scaler.fit\_transform(X)  
lb = LabelBinarizer()  
y\_encoded = lb.fit\_transform(y)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_encoded, test\_size=0.2, random\_state=42)  
  
*# 2. необученная модель*model\_untrained = Sequential([  
 Input(shape=(4,)),  
 Dense(10, activation='relu'),  
 Dense(3, activation='softmax')  
])  
model\_untrained.compile(optimizer=Adam(0.01), loss='categorical\_crossentropy', metrics=['accuracy'])  
loss\_untrained, acc\_untrained = model\_untrained.evaluate(X\_test, y\_test, verbose=0)  
print(f"Результаты необученной сети — Потеря: {loss\_untrained:.4f}, Точность: {acc\_untrained \* 100:.2f}%")  
  
*# 3. основная модель*model = Sequential([  
 Input(shape=(4,)),  
 Dense(10, activation='relu'),  
 Dense(3, activation='softmax')  
])  
model.compile(optimizer=Adam(0.01), loss='categorical\_crossentropy', metrics=['accuracy'])  
  
*# 4.логи потерь по батчам*class BatchLossLogger(Callback):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.batch\_losses = []  
  
 def on\_train\_batch\_end(self, batch, logs=None):  
 self.batch\_losses.append(logs.get('loss'))  
  
batch\_logger = BatchLossLogger()  
  
*# 5. обучение*history = model.fit(  
 X\_train, y\_train,  
 validation\_data=(X\_test, y\_test),  
 epochs=20,  
 batch\_size=8,  
 verbose=0,  
 callbacks=[batch\_logger]  
)  
  
*# 6. оценка модели*loss, accuracy = model.evaluate(X\_test, y\_test, verbose=0)  
print(f"Эпоха 20 — Средняя потеря: {loss:.4f}, Точность: {accuracy \* 100:.2f}%")  
  
*# 7. график потерь на обучении по батчам*plt.figure(figsize=(8, 4))  
plt.plot(batch\_logger.batch\_losses)  
plt.title('График изменения потерь на обучении по батчам')  
plt.xlabel('Номер батча')  
plt.ylabel('Потеря')  
plt.grid(True)  
plt.tight\_layout()  
plt.show()  
  
*# 8. график изменения функции потерь на тесте по эпохам*plt.figure(figsize=(8, 4))  
plt.plot(history.history['val\_loss'], label='Потери на тесте')  
plt.title('Потери на тестовых данных по эпохам')  
plt.xlabel('Эпоха')  
plt.ylabel('Потеря')  
plt.grid(True)  
plt.tight\_layout()  
plt.show()

1. Протокол исполнения

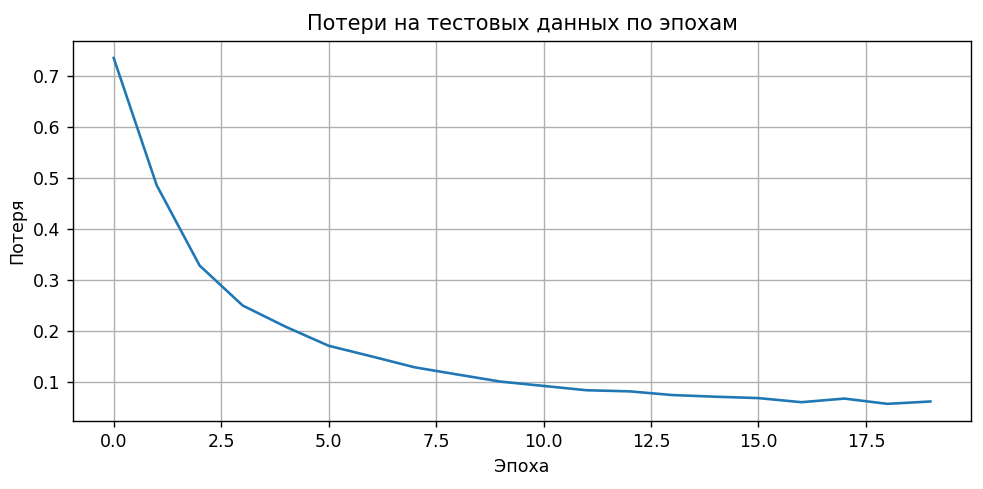
На рисунке 1 представлены результаты теста при необученной сети (средняя потеря 1.1328, точность классификации 33.33%) и после обучения нейронной сети (средняя потеря 0.0624, точность классификации 96.67%).

  
Рисунок 1 – Результаты тестов

На рисунке 2 показан график изменения потерь на обучении по мере увеличения числа батчей.

  
Рисунок 2 – График потерь на обучении

На рисунке 3 изображён график изменения функции потерь на тестовых данных в зависимости от количества эпох

  
Рисунок 3 – График потерь на тестировании

# ЗАКЛЮЧЕНИЕ

В рамках лабораторной работы была разработана нейронная сеть для классификации цветков из датасета Iris с использованием библиотеки Keras. В качестве входных данных использовались стандартизованные признаки длины и ширины чашелистиков и лепестков трёх видов цветков: setosa, versicolor и virginica.

Для оценки начального состояния модели была протестирована необученная сеть, которая показала точность 33.33%, что соответствует случайному угадыванию. После обучения точность модели на тестовых данных достигла 96.67%, а средняя потеря составила 0.0624, что свидетельствует о высокой способности нейросети к обобщению на данной задаче.

В процессе работы были построены графики изменения потерь на обучающей выборке по батчам и изменение функции потерь на тестовой выборке по эпохам. Также выполнена визуализация результатов классификации, наглядно демонстрирующая соответствие предсказанных и реальных классов.

Полученные результаты подтверждают эффективность простых полносвязных нейронных сетей в задачах многоклассовой классификации при наличии хорошо структурированных данных.