Федеральное государственное бюджетное образовательное учреждение высшего образования

«Орловский государственный университет имени

И.С. Тургенева»

Кафедра «Мехатроника, механика и робототехника»

**Лабораторная работа №**

**Автоматизированная диагностика триботронных систем с использованием методов машинного обучения**

Выполнил:

Проверил:

Орёл, 2022г.

**Цель работы:**

**Выполнение работы:**

***Краткое изложение теоретической части:***

***Описание эксперимента:***

***Описание применяемого метода диагностики:***

Описание используемого инструментария, параметров для настройки нейронной сети, листинг кода с задаваемыми настройками.

*Пример листинга:*

WORK\_DIR = '/content' # Рабочая директория  
  
CLASSES = ['norm', 'dis\_1', 'dis\_2', 'dis\_both'] # Названия классов  
SENSORS = ['temp', 'pos\_x', 'pos\_y', 'accel\_x', 'accel\_y', 'accel\_z'] # Названия датчиков  
STRIDES = 10 # Шаг для генерации примеров  
N = 100 # Размер примеров  
N\_CL = len(CLASSES) # Количество классов  
  
# Средние значения измерений по отдельным датчикам  
MEAN = torch.tensor([1.55861268, 1.05951048, 0.67192477,   
 0.19552138, 0.06060946, -8.72093948],

dtype=torch.float32)  
  
# Стандартное отклонение измерений по отдельным датчикам  
STD = torch.tensor([0.03424773, 0.00415722, 0.00476956,  
 0.20554204, 0.97876029, 0.50078193], dtype=torch.float32)  
  
METRIC\_NAME = 'accuracy' # Название метрики  
METRIC = accuracy\_score # Функция расчета метрики  
CRITERION = torch.nn.CrossEntropyLoss() # Функция расчета потерь  
MODEL\_NAME = 'MLP' # Название модели  
BATCH\_SIZE = 128 # Размер пакетов для обучения  
EPOCHS = 100 # Максимальное количество эпох  
EARLY\_STOP\_PATIENCE = 10 # Максимальное количество эпох без улучшения  
DECAY = 1e-5 # Коэффициент регуляризации  
DEVICE = 'cuda' if torch.cuda.is\_available() else 'cpu' # Выбор ресурсов для расчета  
VERBOSE = True # Флаг для подробного вывода процесса обучения на экран  
  
HYPS = [] # Списко гиперпараметров  
  
hyps = {"learning\_rate": [1e-3],  
 "scheduler": ["None", "step"],  
 "layers": [[128, 128]],  
 "dropout": [0, 0.2]}  
  
for i in product(\*[hyps[j] for j in hyps]):  
 HYPS.append({a:b for a, b in zip(hyps, i)})

***Результаты обучения:***

Привести полученные результаты точности и ошибок, добавить графики обучения и матрицы ошибок, привести лучшие параметры нейронной сети

**Вывод:** сделать вывод о пригодности полученной модели для дальнейшей диагностики, вывод о том, какие параметры наиболее важны для получения более высокой точности. \*Предложить пути улучшения.