**Практики**

На практиках производилось знакомство с базовым функционалом языка Python3. Основные темы:

- Знакомство со средой Google Colab;

- Типы данных и операторы;

- Переменные;

- Функции и условия;

- Ввод данных с клавиатуры;

- Списки;

- Циклы;

- Кортежи;

- Множества и словари;

- Библиотека NumPy;

- Библиотека MatplotLib;

- Библиотека Pandas.

Материалы с рассматриваемыми примерами можно найти здесь:

<https://github.com/stebakov-i-n/OGU_Stebakov_it/blob/main/python_get_started.ipynb>

**Лабораторные работы**

**Теоретическая часть**

Одним из направлений интеллектуализации машин является применение методов машинного обучения для автоматизированной диагностики. Модуль автоматизированной диагностики может быть включен в состав триботронной системы и отслеживать различные виды повреждений: трещины ротора, изменение дисбаланса ротора, износ подшипников и т.д.

Методы машинного обучения включают в себя обучение с учителем, обучение без учителя и обучение с подкреплением. Обучение с учителем – способ машинного обучения, в котором алгоритм получает необходимую входные и соответствующие выходные данные. В частности, классификация и регрессия являются задачами обучения с учителем.

Широкое распространение для обучения с учителем получили искусственные нейронные сети. Распространенной реализацией искусственных нейронных сетей является перцептрон [1]. Значение нейрона рассчитываются следующим образом. На вход нейрона подаются выходные значения всех нейронов предыдущего слоя, включая базовый элемент, который всегда равен единице. При передаче значение умножается на соответствующий ему вес. Далее все входные сигналы суммируются и подаются на активационную функцию, которая выдает выходное значение нейрона. Иллюстрация этого процесса представлена на рисунке 1.

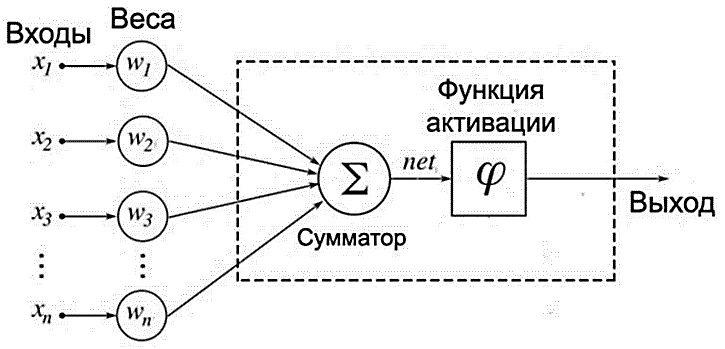


Рисунок 1 – Математическая модель искусственного нейрона

Функции используются для добавления нелинейности. Наиболее распространенные функции активации представлены на рисунке 2.

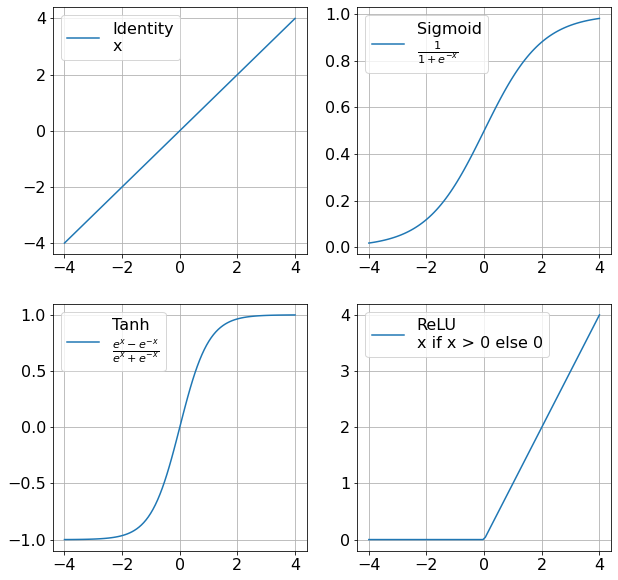


Рисунок 2 – Функции активации

Для решения более сложных задач используется архитектура многослойного перцептрона, представленная на рисунке 3.

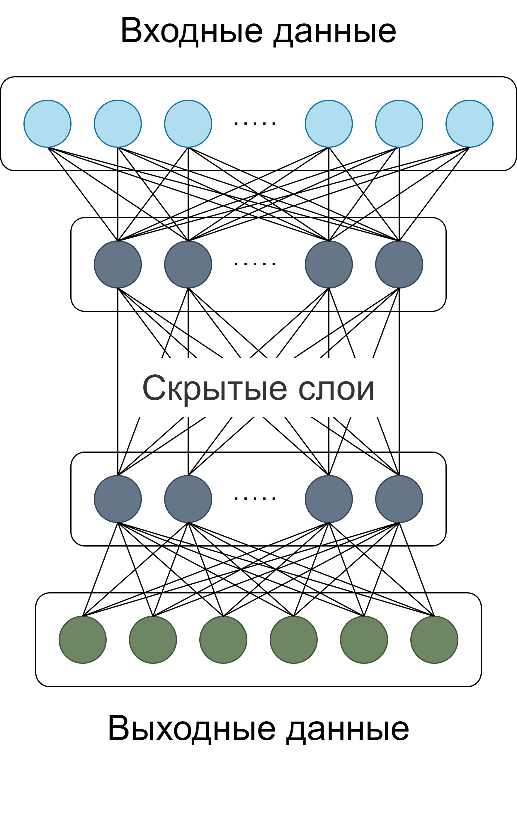


Рисунок 3 – Архитектура многослойного перцептрона

В данной архитектуре нейроны объединяются в слои так, что каждый нейрон текущего слоя соединен с нейронами следующего слоя весовыми коэффициентами. Входящие данные являются входным слоем, последний слой – выходным, а оставшиеся слои называют скрытыми. Для решения задачи классификации в выходном слое применяется функция активации SoftMax:

 (1)

где  – значения нейронов выходного слоя;

*K* – количество нейронов выходного слоя.

С ее помощью выходные нейроны принимают значения от 0 до 1 выражая «уверенность» нейросети в том, что входные данные относятся к соответствующему классу. Каждый выходной нейрон соответствует определенному классу. Предсказанный класс определяется по наибольшему значению нейрона в выходном слое.

Для получения весовых коэффициентов нейронной сети применяется минимизация функции потерь. Для минимизации применяются методы, основанные на методе градиентного спуска. Для классификации применяется функция потерь CrossEntropyLoss:

 (2)

где  – 1 если *i* соответствует действительному классу, иначе 0;

 –значение нейрона;

*n – размер выхода.*

В данной работе используется лабораторная роторная установка с двумя дисками на роторе для исследования дисбаланса. В дисках предусмотрены резьбовые отверстия для вкручивания винтов. При помощи винтов устанавливается дисбаланс ротора. В состав измерительной системы входит: мини-компьютер NVidia Jetson Nano в составе, аналогово-цифровой преобразователь (АЦП) MCP3204 в составе модуля EasyADC, а также датчиков: двух датчиков перемещений IMA12-06BE1ZC0S и трехосевого MEMS-акселерометра MPU6050. Данные от АЦП на мини-компьютер поступают в виде значений напряжения, соответствовавших значению перемещения. Данные от акселерометра на мини-компьютер поступали в виде значений ускорений (м/с2). Акселерометр установлен на одну из опор роторной системы и был закреплен на опоре с помощью пчелиного воска (рис. 3). В опорах установлен подшипник качения 304 серии. На рисунке 4 представлена схема экспериментальной установки.

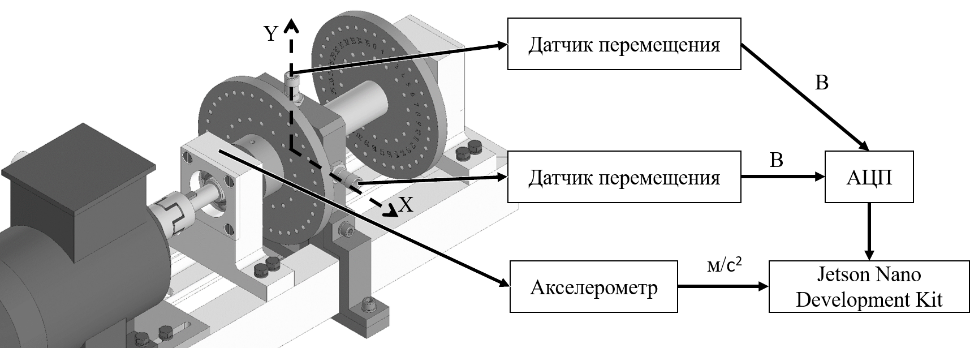


Рисунок 4 – Экспериментальная установка

Исследуется влияние дисбаланса ротора на показания датчиков. Всего исследуется 4 состояния: норма, дисбаланс на первом диске, дисбаланс на втором диске и дисбаланс на двух дисках. Дисбаланс составляет 750 грамм/мм для каждого из дисков. Частота вращения ротора составляет 1200 об/мин. Ротор разгоняется до заданной частоты вращения, затем в течение 1 минуты с частотой 50 измерений в секунду производилась запись с 2 датчиков положения ротора, датчика температуры и акселерометра. План эксперимента включает по 6 параллельных опытов, все опыты выполняются в случайном порядке для снижения возможного эффекта дрейфа. Для установки дисбаланса в диски на роторе предварительно вкручиваются винты. При дисбалансе на двух дисках винты вкручиваются на одном уровне. На рисунке 5 представлен вид установки с установленным на двух дисках дисбалансом.

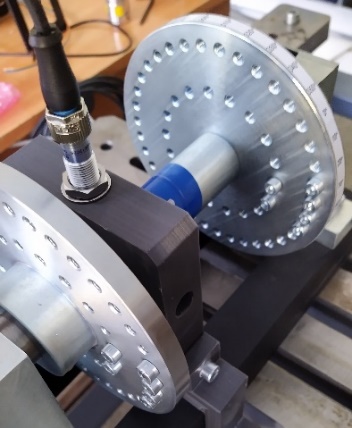


Рисунок 5 – Роторная установка с установленным дисбалансом

В результате эксперимента получены данные, включающие в себя по 3000 измерений всех датчиков для каждого 6 опытов. Данные распределены следующим образом: на обучение нейросети приходится по 3 опыта для каждого класса, для валидации: по 2 опыта, а для тестирования по 1 опыту. Далее данные разбиты на фрагменты по 100 измерений с шагом равным 10. Для предварительной обработки данных используется нормализация. На рисунке 6 представлены графики полученных фрагментов, отсортированные по классам.

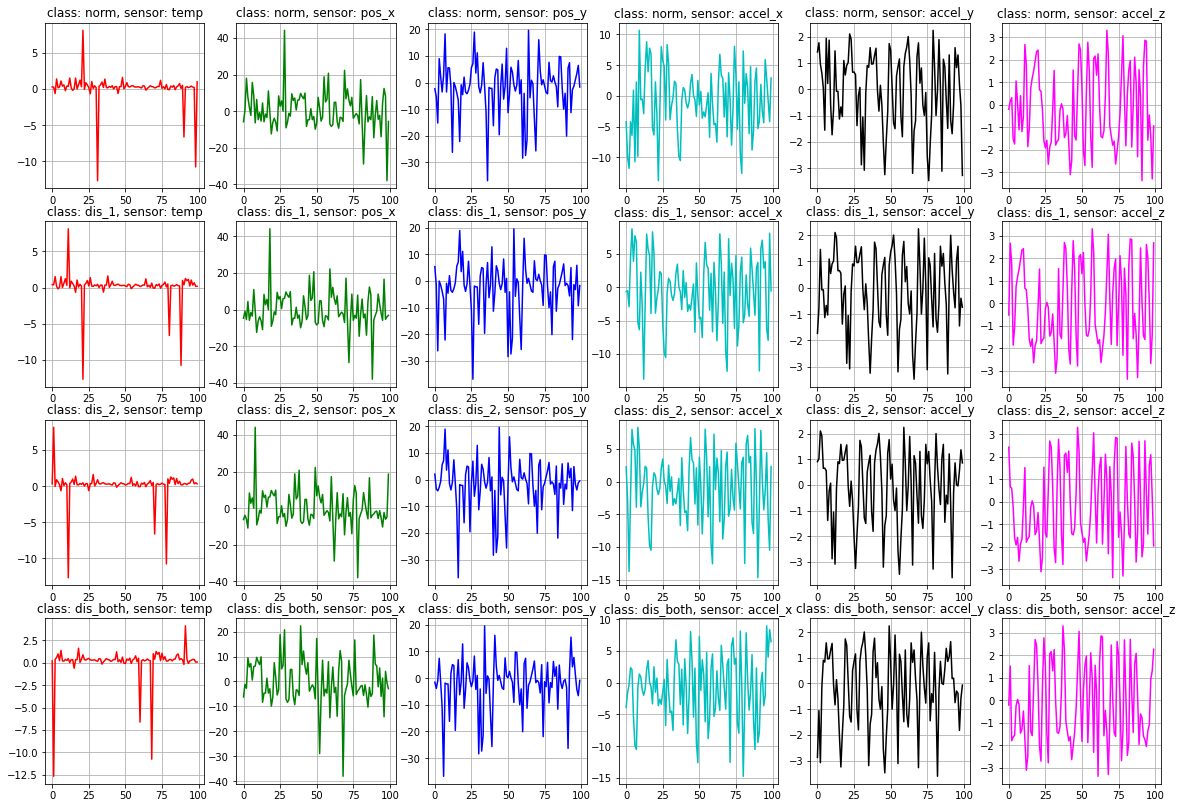


Рисунок 6 – Сэмплы данных разных классов неисправностей

Таким образом, размер выборок составил: обучение – 3480 примеров, валидация – 2320 примеров, тестирование – 1160 примеров. В результате получены матрицы размеров NxMxC, где N – количество примеров, M – количество измерений, С – количество каналов. Также получены матрицы с разметкой классов.

**Практическая часть**

Данные, полученные в ходе эксперимента, сохранялись с расширением “\*.npy”. Данные файлы можно открыть с помощью библиотеки NumPy [3] языка программирования Python 3 [2]. Для обучения предиктивного модуля необходимо использовать облачный сервис Google Colab [4]. Данный сервис позволяет производить выполнение кода языка программирования Python 3 на удаленном сервере.

Для выполнения работы предлагается выполнения ряда лабораторных работ включающих рассмотрение следующих вопросов:

- Базовый функционал библиотеки Pytorch;

- Загрузка и визуализация данных;

- Формирование наборов данных для обучения;

- Реализация функций для обработки предсказаний многослойного перцептрона;

- Реализация модели многослойного перцептрона;

- Визуализация процесса обучения;

- Подготовка программы для обучения многослойного перцептрона;

- Проведение вычислительного эксперимента.

Для использования методов машинного обучения используется библиотека PyTorch [5]. Для создания нейронной используется модуль torch.nn. Для подбора оптимальной модели предлагается изменять следующие параметры: шаг обучения, количество и размер скрытых слоев коэффициент регуляризации, решатель для процесса оптимизации.

Для оценки модели используется функция accuracy, которая рассчитывается как отношение верно предсказанных состояний к общему числу примеров. Необходимо реализовать данную функцию в программном коде.

Для подведения итогов необходимо сделать выводы о том, какие параметры нейронной сети являются оптимальными, какая точность ожидается на новых данных.

**Содержание лабораторных работ**

Предварительный код для выполнения лабораторных работ можно найти здесь: <https://github.com/stebakov-i-n/OGU_Stebakov_Pytorch.git>

**Лабораторная работа №1 "Базовый функционал библиотеки Pytorch и реализация градиентного спуска для линейной регрессии"**

1. Рассмотреть пример с квадратичной функцией

2. Добавить датасет в рабочую директорию. Добавьте разбиение на выборки (train\_test\_split работает и с тензорами) для тренировки, валидации и теста.

3. Реализуйте функцию для линейной регрессии вида: . Для реализации используйте матричное умножение.

4. Добавьте в процесс обучения выбор случайных сэмплов для реализации SGD.

5. Реализуйте поиск шага обучения по валидации. При каждом новом цикле обучения заново инициализируйте веса модели.

6. Обучите модель с лучшими параметрами. Выполните оценку и визуализацию работы модели.

**Лабораторная работа №2 "Загрузка и визуализация данных при помощи библиотек NumPy и Matplotlib"**

1. Дописать функцию для визуализации сигналов с выбором датчиков и длины сигнала. Добавить подпись датчика.

2. Дописать расчет среднего значения и стандартного отклонения для всех датчиков.

3. Модифицировать визуализацию боксплотов. Добавить название класса. Сделать боксплоты разного цвета для каждого класса. Для этого сделать словари аналогичный словарю с датчиками. Названия цветов: <https://matplotlib.org/stable/gallery/color/named_colors.html>

**Лабораторная работа №3 "Использование функционала Pytorch для формирования наборов данных для обучения"**

1. Сделать список с индексами используемых датчиков на основе USED\_SENSORS.

2. Ознакомиться с агрегацией всех данных в списке data\_all.

3. Ознакомиться с формированием массивов inputs и targets, которые содержат фрагменты измерений размером n.

4. Добавить в датасет загрузку данных и формирование массивов inputs и targets. Сделать обработку параметров датасета. Используйте USED\_SENSORS\_INDEX для выбора указанных датчиков.

5. Сделать экземпляры датасетов для всех выборок данных. Вывести кол-во элементов в каждом датасете.

6. Сделать экземпляры DataLoader для всех выборок данных. Для обучающих данных shuffle=True, drop\_last=True, для остальных - False. Вывести кол-во элементов в каждом DataLoader.

**Лабораторная работа №4 "Реализация функций для обработки предсказаний многослойного перцептрона"**

1. Написать функцию SoftMax.

2. Написать функцию CELoss с усреднением по примерам.

3. Написать код для расчета точности.

**Лабораторная работа №5 "Реализация модели многослойного перцептрона"**

1. Модифицировать функцию формирования слоев добавив следующие параметры: размер входа и выхода, параметр p операции Dropout.

2. Модифицировать класс создания нейронной добавив следующие параметры: кол-во слоев, кол-во нейронов в промежуточных слоях, размер входа, кол-во классов, параметр p операции Dropout.

**Лабораторная работа №6 "Визуализация процесса обучения"**

1. Написать функцию для визуализации истории обучения. На двух отдельных рисунках должны быть графики для функции потерь и точности. Разным цветом обозначить train и val. Вывест легенду в формате ['Train', 'Val']. Заголовок не использовать. Подпись оси х Epochs. Подпись оси у подписать Loss или Accuracy. Сделать логарифмическую шкалу для графиков с функцией потерь.

2. Написать функицию, которая будет выводить модифицированныую матрицу ошибок. Необходимо по предсказаниям определить предсказанные классы применив функцию SoftMax и argmax (см. Л.Р. № 5). Добавить заголовок, подписи классов на шкале.

**Лабораторная работа №7 "Подготовка программы для обучения многослойного перцептрона"**

1. Вставить код для функций SoftMax, CELoss, accuracy в файл fn\_utils.py

2. Вставить код для функций plot\_cm, plot\_history в файл vis\_utils.py

3. Вставить код для классов DefectsDataset, MLP в основной файл

4. Проверить работоспособность кода

**Лабораторная работа №8 "Проведение вычислительного эксперимента"**

1. Провести серию расчетов с разными параметрами.

2. Провести анализ результатов.

**Список источников**

1. Goodfellow, I., Bengio, Y. & Courville, A. Deep Learning. (MIT Press, 2016).

2. Python. Официальный сайт. URL: https://pytorch.org/

3. NumPy. Официальный сайт. URL: https://numpy.org/

4. PyTorch. Официальный сайт. URL: https://www.python.org/

5. Google colab. Официальный сайт. URL: https://colab.research.google.com/