

## Содержание

Введение.....	4
1. Техническое задание .....	5
1.1 Назначение разработки и область применения.....	5
1.2 Технические требования .....	5
2. Анализ технического задания .....	6
2.1 Выбор операционной системы.....	6
2.2 Выбор языка программирования .....	7
2.3 Выбор среды разработки .....	8
2.4 Обзор существующих методов классификации.....	11
3. Разработка структуры системы.....	14
3.1 Разработка общей структуры системы .....	14
3.2 Алгоритм работы одномерной свёрточной нейронной сети .....	15
4. Разработка программных средств.....	19
4.1 Разработка программного интерфейса системы .....	19
4.2 Программная реализация модулей системы .....	22
5. Тестирование системы .....	34
5.1 Описание набора данных .....	34
5.2 Описание методики тестирования.....	35
5.3 Результаты вычислительного эксперимента.....	35
Заключение .....	40
Список литературы.....	41
Приложение А .....	45
Приложение Б .....	57

Иzm.	Лист	№ докум.	Подпись	Дата	BKR-НГТУ-09.03.01-(16-B-2)-007-2020 (П3)		
Разраб.	Гусев Н.А.			03.02.20	Программная система диагностики состояния механизма по вибрационному сигналу Пояснительная записка	Лист.	Лист
Проевр.	Гай В.Е.			03.02.20		4	67
Н. контр.				03.02.20			
Утврд.	Жевнерчук Д.В.						
НГТУ кафедра ВСТ							

## **Введение**

Анализ вибраций промышленного оборудования проводится в течение многих десятилетий, однако приобрел известность благодаря внедрению и широкому использованию персонального компьютера. Вибрационный анализ относится к процессу измерения уровней и частот вибраций промышленного оборудования и использования этой информации для определения состояния такого механизма и его компонентов [1].

Как отмечается в [2], когда работает промышленная машина (например, вентилятор или насос), она порождает вибрации. Эта вибрация может быть измерена с помощью устройства, называемого акселерометр. Акселерометр генерирует сигнал напряжения, пропорциональный величине вибрации, а также частоте вибрации или тому, сколько раз в секунду или минуту происходит вибрация. Этот сигнал напряжения от акселерометра подается в коллектор данных, который записывает этот сигнал либо в виде временной области (зависимость амплитуды от времени), либо в форме быстрого преобразования Фурье (зависимость амплитуды от частоты), либо в виде обеих форм. Затем этот сигнал может быть проанализирован обученным специалистом по вибрации или посредством эвристического алгоритма компьютерной программы. Затем проанализированные данные используются для определения исправности механизма и выявления любых надвигающихся проблем в нем, таких как перекос, дисбаланс, проблема подшипников или смазки, ослабление и многое другое [2-5].

Поэтому важно иметь эффективную систему контроля состояния механизма и систему диагностики неисправностей, чтобы можно было своевременно обнаружить и правильно диагностировать возникающие неисправности исследуемого механизма, для того чтобы предотвратить его дальнейшее ухудшение и повреждение. Например, раннее обнаружение зарождающегося дефекта в механизме высокоскоростного поезда, ветряной турбины или автомобиле может привести к своевременному техническому обслуживанию для предотвращения возможных катастрофических последствий, вызванных неожиданным отказом таких критических механических компонентов.

Вследствие вышеизложенного очевидна проблема своевременного обнаружения неисправностей механизма по вибрационному сигналу, что обуславливает актуальность разработки интеллектуальной программной системы, способной решить поставленную задачу.

					BKR-НГТУ-09.03.01-(16-B-2)-007-2020 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		5

## **1. Техническое задание**

### **1.1 Назначение разработки и область применения**

Программная система предназначена для обнаружения дефекта механизма по виброакустическому сигналу. Полученный с акселерометра сигнал механизма загружается в систему классификации, после чего разработанный алгоритм определяет класс анализируемого сигнала, таким образом, принимая конечное решение об исправности механизма.

Областями применения в рамках вибрационной диагностики разрабатываемой системы являются:

- Идентификация оснований изменения вибрационного состояния.
- Прогноз и контроль технического состояния механизма.

### **1.2 Технические требования**

Опишем требования, предъявляемые разрабатываемой системой к ЭВМ:

- В качестве операционной системы – Microsoft Windows, включая XP и выше.
- Требования к аппаратному обеспечению определяются операционной системой.
  - Устройства ввода: мышь, клавиатура.
  - Устройства вывода: монитор.

Проанализируем ключевой функционал разрабатываемой программной системы:

- Обеспечить пользовательскую загрузку данных виброакустического сигнала.
- Производить анализ полученного на вход системы сигнала.
- Предоставить статус механизма, виброакустический сигнал которого был загружен в систему.

Изм.	Лист	№ докум.	Подпись	Дата	БКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ)	Лист
						6

## **2. Анализ технического задания**

### **2.1 Выбор операционной системы**

Важным этапом для данной работы является выбор операционной системы, ввиду различных особенностей, влияющих на разработку программной системы. Рассмотрим наиболее универсальные и распространенные: Windows, Linux.

1) Операционная система Windows была впервые выпущена в 1985 году Microsoft. Теперь компания может похвастаться самой большой базой пользователей по всему миру. За прошедшие годы с серией обновлений каждая система ОС имеет уникальный графический интерфейс, который позволяет пользователям просматривать файлы и работать в своей системе. Большинство современных программ работают именно на операционной системе Windows, которая предназначена для работы на оборудовании x86, включая Intel и AMD. Последней версией Windows является Windows 10, выпущенная в июле 2015 года в качестве преемника Windows 8.1.

2) Операционная система Linux – это ОС с открытым исходным кодом, созданная в начале 1990-х годов Линусом Торвальдсом. Основанный на UNIX, Linux позволяет пользователям изменять существующий код и создавать его различные версии или дистрибутивы, которые они могут повторно использовать и даже продавать. Linux стал лучшим выбором для настройки серверов, поэтому большинство веб-страниц в Интернете обслуживаются системами, поддерживаемыми одним из его дистрибутивов. ОС также широко используется в настольных ПК, мобильных устройствах, консолях, устройствах для чтения электронных книг и многом другом.

Как Linux, так и Windows могут считаться одними из лучших операционных систем, и обе системы укрепили свое место в компьютерном мире.

В настоящем пункте рассмотрим основные характеристики и проведем содержательное сравнение, чтобы обосновать сделанный выбор, который помог при разработке системы.

1) Ни одна операционная система не является такой удобной для пользователя, как Microsoft Windows. Пользовательский интерфейс Microsoft Windows намного понятнее и проще в настройке по сравнению с открытыми операционными системами.

2) Доля Microsoft Windows на рынке составляет около 90%, в то время как Linux по-прежнему находится на уровне около 1%. Из-за своей большой пользовательской базы, Microsoft Windows имеет широкую экосистему и поддерживает большое количество программного обеспечения, которое предоставляет своему пользователю. Операционные системы с открытым исходным кодом также имеют огромный набор поддерживаемого программного обеспечения, но Microsoft Windows – лидер рынка, так как большинство программ построено с учетом требований Windows.

					<i>BKR-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ)</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

3) Поскольку ОС Windows имеет огромную базу пользователей по всему миру, для данной системы существует достойная онлайн-поддержка, в то время как для ОС Linux пользователям предоставляется поддержка через форумы и специализированные веб-сайты энтузиастов. Кроме того, в сети Интернет доступны многочисленные видео и книги о Microsoft Windows для людей всех уровней квалификации.

4) Microsoft Windows поддерживает широкий спектр оборудования. Большинство производителей устройств имеют поддержку в Microsoft Windows из-за его большей пользовательской базы. С другой стороны, операционные системы с открытым исходным кодом имеют сравнительно меньшую пользовательскую базу, и, следовательно, только некоторые производители поддерживают свои устройства в таких операционных системах с открытым исходным кодом, как Linux.

В результате проведенного сравнения была выбрана операционная система Microsoft Windows. Главным фактором для этого стало наибольшее число поддерживаемых продуктов для разработки. Еще одной важной причиной является то, что Windows ОС отвечает заявленным техническим требованиям и необходимым функциональным инструментарием для реализации программной системы.

## 2.2 Выбор языка программирования

Различные опросы, данные и источники показывают, что наиболее важным фактором при выборе языка является область применения разрабатываемой системы. Вследствие этого, выбор языка программирования является одной из важнейших частей данной работы.

Так как разрабатываемая в рамках настоящей работы интеллектуальная программная система включает в себя в качестве подзадач идентификацию сигнала, прогноз и контроль состояния механизма, что является атрибутами такого направления как машинное обучение (в частности глубокое обучение при непосредственном использовании нейронных сетей), достаточно трудно упомянуть только один язык программирования.

Рассмотрим основные языки: Python, Java и R, которые предоставляют наиболее функциональный инструментарий для разработки:

1) Python является широко используемым языком в сфере машинного и глубокого обучения, основными его особенностями являются выпуск TensorFlow, а также множество других библиотек, таких как Numpy, Keras и Scikit-learn. Например: Numpy – это библиотека, которая помогает решать работу с множеством сложнейших вычислений, а библиотека TensorFlow предназначена для решений задач связанных с обучением и тренировкой нейронной сети. Другая причина его популярности заключается в том, что синтаксис данного языка прост и может быть легко изучен, что делает алгоритмы легко реализуемыми.

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

2) Java это второй наиболее предпочтительный язык, используемый разработчиками в сфере машинного обучения. Это простой в использовании язык, который обеспечивает практичный процесс отладки, огромные сервисы пакетов, которые упрощают работу в проектах, графическое представление данных и интуитивное взаимодействие с пользователем. Также, Java считается безопасным языком благодаря использованию байт-кода. Неудивительно, что как новейшие, так и более старые алгоритмы машинного обучения написаны на Java. Это функциональный язык программирования, который позволит будущим системам машинного обучения работать со скоростью и точностью.

3) R – это динамический мультипарадигмальный язык, используемый для статистических вычислений, анализа и визуализации в машинном обучении. Он поддерживает объектно-ориентированные, императивные, функциональные, процедурные и рефлексивные парадигмы программирования. Причиной, почему R приобрел популярность среди ученых и разработчиков является его способность к вычислению статистики, визуализации данных, а также данный язык применим для задач машинного обучения, таких как: классификация данных и формирование дерева решений. R часто сравнивают с языком Python, но это два абсолютно разных языка, которые используются для разных целей.

Среди, различных языков программирования для разработки программной системы был выбран язык программирования Python. Решающими факторами в выборе данного языка являются: лаконичность и читабельность написанного кода, простота использования и разнообразие инструментов для разработки. В совокупности перечисленных оснований, выводом является, что выбранный язык дает возможность сосредоточиться на поиске решений и достижений целей при реализации данной программной системы.

### 2.3 Выбор среды разработки

Следующим шагом является выбор интегрированной среды разработки для реализации, редактирования, отладки, тестирования исходного кода и создания исполняемых файлов разрабатываемой программы. Правильный выбор среды для разработки, так же важен, как и методы решений поставленных в работе задач, так как корректно сделанный выбор увеличивает производительность разработки. Проанализируем несколько лидирующих сред разработки, которые совместимы с выбранным языком Python: PyCharm, Spyder и Google Colaboratory.

1) PyCharm – редактор кода PyCharm предоставляет обширную поддержку языка Python. Данная среда разработки имеет функции обнаружения ошибок, автозавершения кода и автоматического исправления кода, функцию интеллектуального поиска, которая может переходить к любому классу, файлу, символу или окну инструмента. PyCharm включает в

					ВКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		9

себя обширную коллекцию инструментов, среди них: встроенный отладчик, профилировщик Python и встроенный терминал. Также, редактор интегрируется с основными системами контроля версий, включая Git, SVN.

2) Spyder – предоставляет набор функций, таких как редактирование, анализ, отладка и профилирование. Это комплексный инструмент разработки, способный к исследованию данных, интерактивному выполнению, глубокой проверке и доступный интерфейс визуализации. Его возможности могут быть расширены с помощью различных плагинов и встроенного интерфейса. Функциональность редактора включает в себя многоязычный редактор с браузером классов, инструментами анализа кода, автоматическим завершением кода, переходом к определению, автокорректировкой, исправлением опечаток исходного кода и разделением окон по горизонтали и вертикали. Также, Spyder имеет дополнительный фактор при выборе для использования – это поддержка сообщества пользователей, которую можно получить вместе со всей полной конструктивной и многоязычной документацией.

3) Google Colaboratory – данный редактор имеет минималистичный дизайн и хорошо структурированная панель инструментов, делая рабочий процесс по разработке продуктивным. Google Colaboratory предоставляет такие функции, как: интерактивные виджеты, из которых код может выводить данные, например, изображения, моделировать графы, построение обученных моделей нейронной сети, а также позволяют манипулировать и визуализировать данные в режиме реального времени, как продемонстрировано на рисунках 2.1 – 2.3. Среда интерпретируется на виртуальных машинах, которые используют ресурсы того компьютера, на котором виртуальные узлы были запущены. Такая особенность данной среды разработки дает возможность использовать наиболее производительные машины при проектировании, обучении и тренировки нейронных сетей.

```

https://colab.research.google.com/drive/17zvpNL0M2Hhhd0C28XUu7y4FFry2paS
Часто посещаемые Диалоги Google Передача Твиттер Владелец - Почта М... Список, создать — Аудио... Instagram XiongMeijing/CWRU... SaveFrom.net
Копия Untitled0_Diploma.ipynb
Файл Изменить Вид Вставка Среда Выполнения Инструменты Справка Последнее изменение: 4 июня
+ Код + Текст
```python
# Train
from sklearn import *
epochs = 70
model, metrics = fit(epochs, model, loss_func, opt, train_dl, valid_dl, train_metric=True)

# Train
from train import *
model, metrics = fit(epochs, model, loss_func, opt, train_dl, valid_dl, train_metric=True)
```


Epoch	Train Loss	Val Loss	Train Acc	Val Acc
0	5.71656	5.54417	0.49499	0.50493
1	0.04903	2.05114	0.98679	0.98704
2	0.04903	0.84683	0.98540	0.98897
3	0.04903	0.62133	0.96614	0.90098
4	0.04903	0.55338	0.79810	0.91577
5	0.04903	0.44499	0.85631	0.92199
6	0.04903	0.39456	0.92652	0.96965
7	0.04903	0.12414	0.96339	0.96939
8	0.04903	0.18787	0.98174	0.95720
9	0.04903	0.04903	0.98103	0.89103
10	0.04903	0.04903	0.98654	0.92690
11	0.04903	0.04903	0.98654	0.92552
12	0.04903	0.19200	0.98646	0.92552
13	0.04903	0.19200	0.98646	0.92552
14	0.04903	0.19200	0.98646	0.92552
15	0.04903	0.19200	0.98646	0.92552
16	0.04903	0.19200	0.98646	0.92552
17	0.04903	0.19200	0.98646	0.92552
18	0.04903	0.19200	0.98646	0.92552
19	0.04903	0.19200	0.98646	0.92552
20	0.04903	0.19200	0.98646	0.92552
21	0.04903	0.19200	0.98646	0.92552


```

Рисунок 2.1 – Обзор исходного кода в Google Colaboratory

| Изм. | Лист | № докум. | Подпись | Дата | Лист                                     |    |
|------|------|----------|---------|------|--|----|
|      |      |          |         |      | BKP-НГТУ-09.03.01-(16-B-2)-007-2020 (ПЗ) |    |
|      |      |          |         |      |  | 10 |

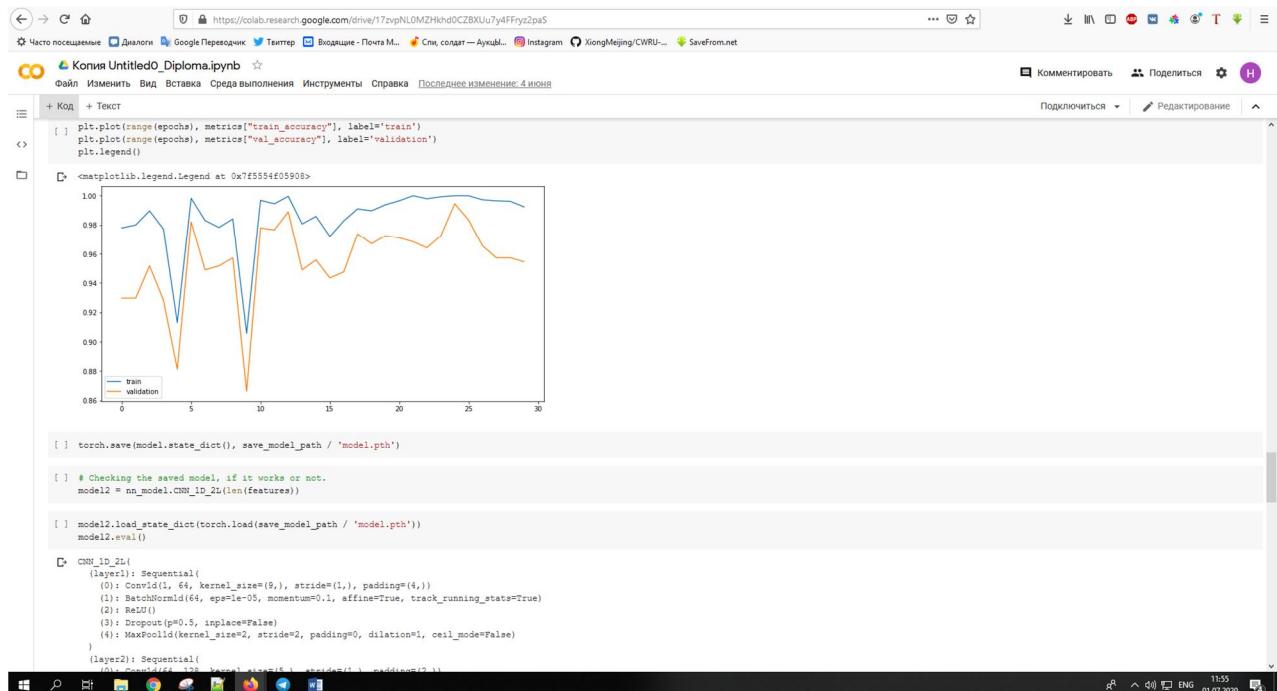


Рисунок 2.2 – Визуализация данных в Google Colaboratory

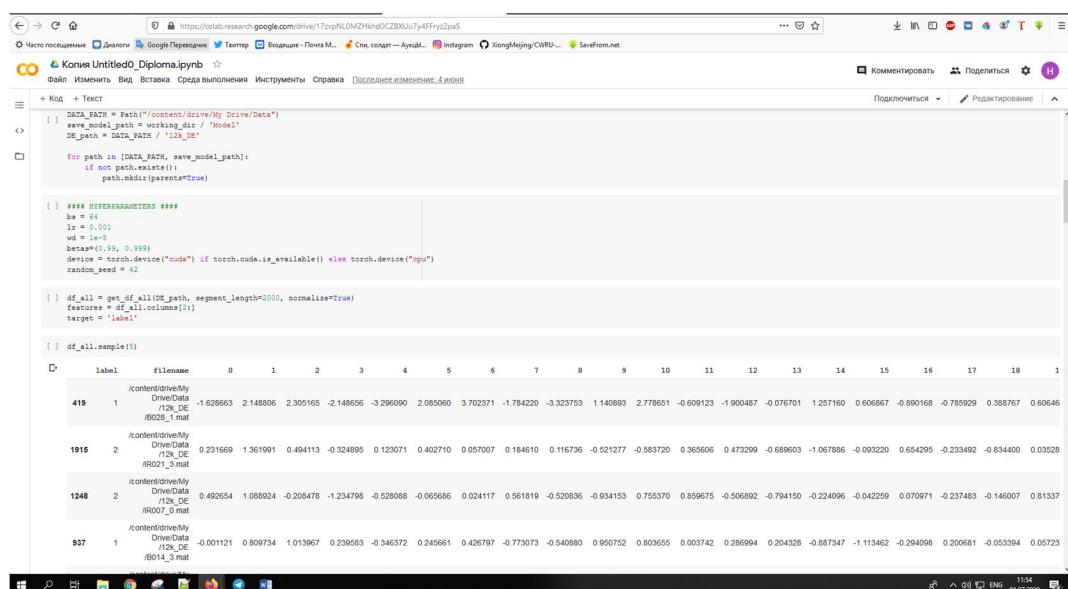


Рисунок 2.3 – Обзор данных в Google Colaboratory

Для реализации, тестирования данной программной системы были выбраны среды разработки: Spyder и Google Colaboratory в силу нескольких значительных причин. Вследствие предоставления редактором Spyder широкого выбора функциональных инструментов для разработки, сочетающихся с мощным пользовательским интерфейсом, данный редактор был задействован в реализации графической оболочки для программной системы. Благодаря обеспечению производительными вычислительными узлами и доступом к обширной документации Google Colaboratory соответствовал части, где проходила разработка и тестирование нейронной сети.

|      |      |          |         |      |      |
|------|------|----------|---------|------|------|
|      |      |          |         |      | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |      |

## **2.4 Обзор существующих методов классификации**

В последнее время наблюдается огромный подъем в изучении архитектуры нейронных сетей. Глубокое обучение, в частности, стало одной из наиболее применяемых методик обучения для решения различных проблем, начиная от распознавания дефекта подшипника по его сигналу и заканчивая транспортными средствами самостоятельного вождения [6-8]. Популярность таких систем заключается в том, что решения, которые основаны на глубоком обучении для многочисленных проблемных областей, имеют способность самостоятельно выполнять изучение зависимостей и математических функций для обучения, вместо того чтобы программист тратил усилия на разработку таких функций, создавая их вручную [9].

Согласно [10-16], тенденцию перехода от классических «неглубоких» алгоритмов машинного обучения к глубокому обучению можно объяснить следующими причинами:

1) **Данные.** Наблюдается всплеск появления крупномасштабных наборов данных во многих областях, таких как: ImageNet – для распознавания изображений, CRWU – для распознавания вибрационного сигнала [10, 12, 14]. А также, доступность огромного количества данных и применение к ним механизмов маркировки дает возможность воспользоваться ими [11, 13]. Согласно [15], с увеличением объема данных производительность глубокого обучения может значительно превзойти большинство классических алгоритмов машинного обучения. Для многих применений, включая диагностику неисправностей подшипников, такие большие наборы данных доступны, и их приобретение не будет дорогостоящим и длительным.

2) **Эволюция алгоритмов глубокого обучения.** Как сообщается в [11], изобретается и совершенствуется больше методов с точки зрения управления процессом обучения более глубоких моделей для достижения более высокой скорости, лучшей сходимости и улучшенного обобщения. Например, такие алгоритмы, как ReLU, помогают ускорить скорость сходимости. Методы отбора и объединения, помогают предотвратить переоснащение данных. Численные методы оптимизации, такие как градиентный спуск, оптимизация помогают использовать большое количество данных и обучать более глубокие модели [10, 14].

3) **Эволюция аппаратного обеспечения.** Обучение глубоких сетей, как говорится в [11-14], требует больших вычислительных ресурсов, но работа на высокопроизводительном графическом процессоре может значительно ускорить этот процесс обучения. В частности, GPU предлагает возможность параллельных вычислений и вычислительную совместимость с глубокими нейронными сетями, что делает их незаменимыми для обучения алгоритмов на основе глубокого обучения.

Все вышеперечисленные факторы способствуют новой эре применения алгоритмов в глубоком обучение для различных приложений, связанных с обработкой данных. Несколько основных видов нейронных сетей, лежащих в основе различных и текущей системы глубокого обучения, описаны ниже.

| Изм. | Лист | № докум. | Подпись | Дата | Лист<br>BKR-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | 12 |
|------|------|----------|---------|------|--|----|
|      |      |          |         |      |  |    |

1) Глубокие нейронные сети (DNN). DNN – это тип искусственной нейронной сети с несколькими скрытыми слоями [16]. Количество этих слоев может быть значительно большим, откуда и происходит термин «глубокий». Эта глубина позволяет изучать представления и отличительные признаки для классификации [14, 16], что позволяет использовать DNN-сети в широком спектре областей. Схематическое представление данной модели нейронной сети представлено на рисунке 2.4

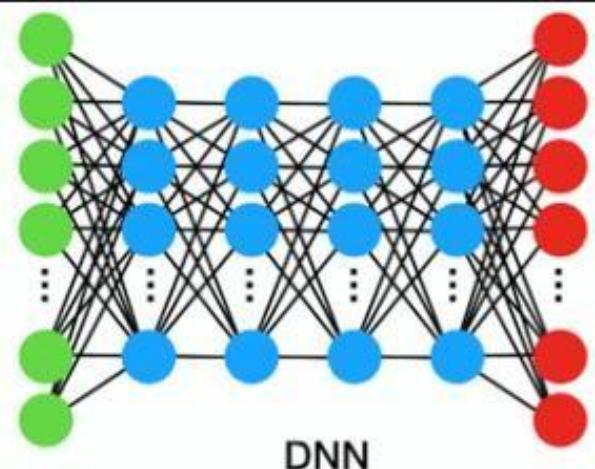


Рисунок 2.4 – Схематическое представление DNN нейронной сети

2) Сверточные нейронные сети (CNN). Как показал подробный анализ [17-28], CNN нейронная сеть является расширением DNN нейронной сети, которое позволяет работать с данными, поступающими в виде многомерных массивов [20]. В случае сигналов, которые могут быть представлены в виде одномерного массива, входной сигнал свернут с определенными фильтрами, после чего такие фильтры объединяются для уменьшения их размерности [24]. Это делается для того, чтобы получить локальную статистику входных данных, и при повторной итерации такие фильтры создают иерархию математических функций для обучаемой модели [23-27]. Схематическое представление модели CNN нейронной сети представлено на рисунке 2.5.

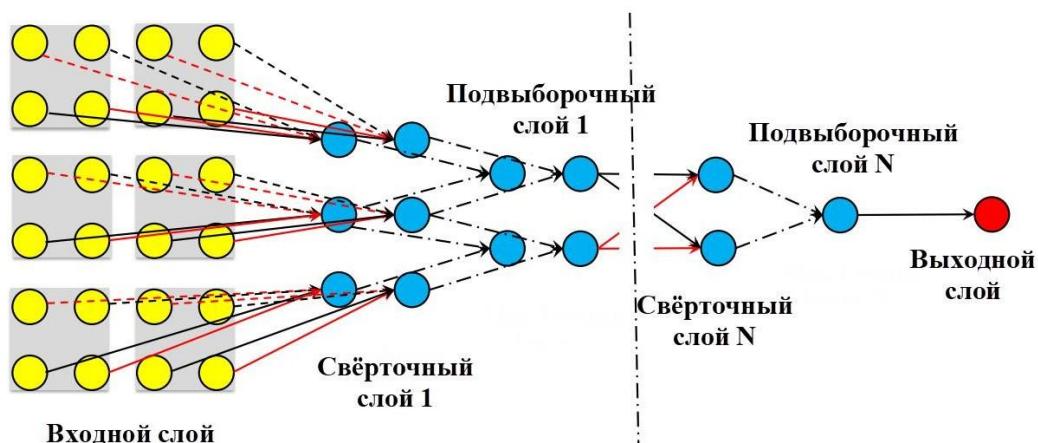


Рисунок 2.5 – Схематическое представление CNN нейронной сети

3) Рекурентные нейронные сети (RNN). Согласно [29, 30], данный тип модели нейронной сети поддерживает вектор состояния, который содержит информацию о последовательной истории всех прошлых элементов за счёт использования долгосрочной кратковременной памяти (LSTM). Однако, как отмечается в [31], градиенты, вычисленные во время обучения RNN, имеют тенденцию исчезать очень быстро в течение многих временных шагов, что приводит к плохому улавливанию долгосрочной зависимости. Таким образом, для обеспечения работы с длинными зависимостями без потери или переполнения как раз используются нейронные сети с кратковременной памятью (LSTM), которые имеют специальные скрытые блоки (шлюзы) для запоминания или забывания входных данных. Схематическое представление RNN нейронной сети представлено на рисунке 2.6.

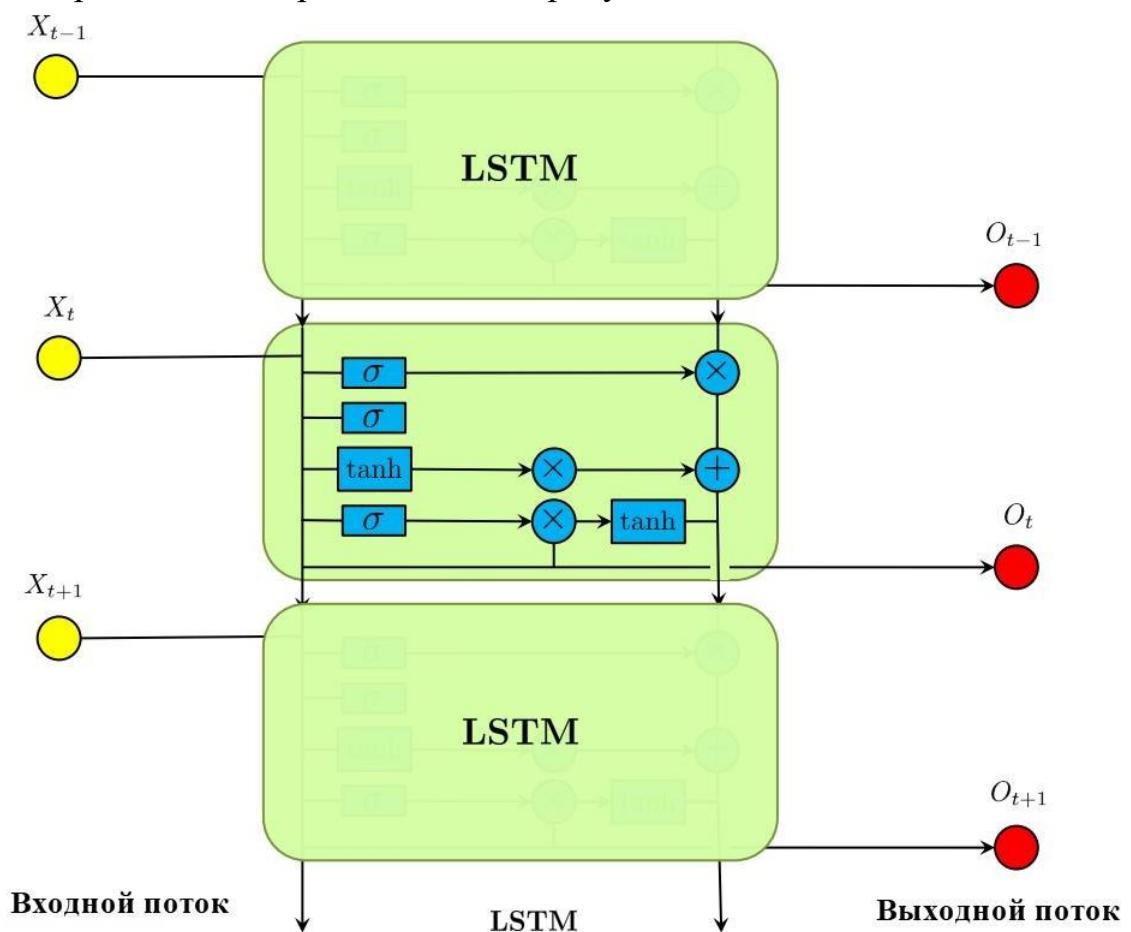


Рисунок 2.6 – Схематическое представление RNN нейронной сети

4) Авто-кодировщики. Как отмечается в [32-37], авто-кодировщики – это особый тип неконтролируемых нейронных сетей, использующие методы сжатия входных данных путём кодирования и методы восстановления выходных данных путём декодирования с целью шумоподавления в случае работы с зашумлёнными данными [38]. Иначе говоря, поступающий на вход сигнал кодируется с целью шумоподавления, обрабатывается в свёрточных слоях и затем выходная информация восстанавливается до исходной формы.

Следует отметить, что данная структура может быть использована не только для шумоподавления, но и для снижения размерности входных данных, как сообщается в [39-46]. Схематическое представление данного типа нейронной сети представлено на рисунке 2.7.

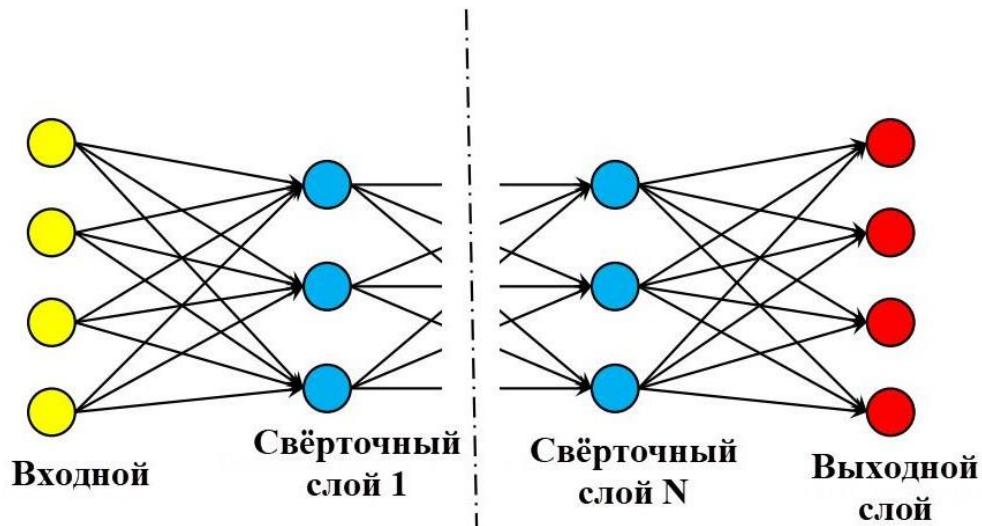


Рисунок 2.7 – Схематическое представление автокодировщика

Отметим, что в настоящей работе используется архитектура CNN. Для интеллектуальной диагностики состояния механизмов в качестве входных данных выступает записанный виброакустический сигнал, снятый с датчиков (акселерометров). Выбор был сделан в пользу модели свёрточной нейронной сети CNN, так как обученная сеть, как сообщается в [21, 23, 25], максимально приспособлена для решения задачи выявления признаков и классификации состояний многомерных объектов. Кроме того, данный тип нейронной сети с высокой точностью выявляет периодические закономерности в ходе анализа акустических сигналов (виброакустических в частности) [30], что позволяет в перспективах внедрить в разрабатываемую интеллектуальную программную систему модуль, осуществляющий предиктивный анализ виброакустических сигналов с целью обеспечить возможность предугадывать неисправности ещё на ранней стадии. Кроме того, следует отметить, что CNN нейронные сети, как показывают результаты экспериментов, представленные в [22-29], показывают, что данные сети способны обучаться на необработанных данных, что обеспечивает высокую стабильность и точность классификации.

### 3. Разработка структуры системы

#### 3.1 Разработка общей структуры системы

Как правило, в современных системах поддержки принятия решений выделяются два самостоятельных и технически не связанных между собой этапа – выявление признаков и классификация с использованием данных признаков. В настоящей работе предлагается метод, который, во-первых, не требует какой-либо конкретной формы преобразования данных, а во-вторых, позволяет объединить выявление признаков и классификацию состояний рабочего механизма в один обучающий корпус с применением одномерной свёрточной нейронной сети [47, 48]. Для обнаружения неисправностей для данного метода подойдут даже необработанные данные, что позволяет в случае реального применения данной нейронной сети при решении задачи диагностики механизма подавать на вход заранее обученной сети «сырой» вибраакустический сигнал. Предлагаемая нейронная сеть обучается на тестовых данных с использованием алгоритма обратного распространения. Единственной предварительной обработкой данных в предлагаемом методе является формирование выборки. На рисунке 3.1 представлена схема работы предлагаемой нейронной сети в тандеме с электромеханической системой, состояние которой контролируется нейронной сетью.

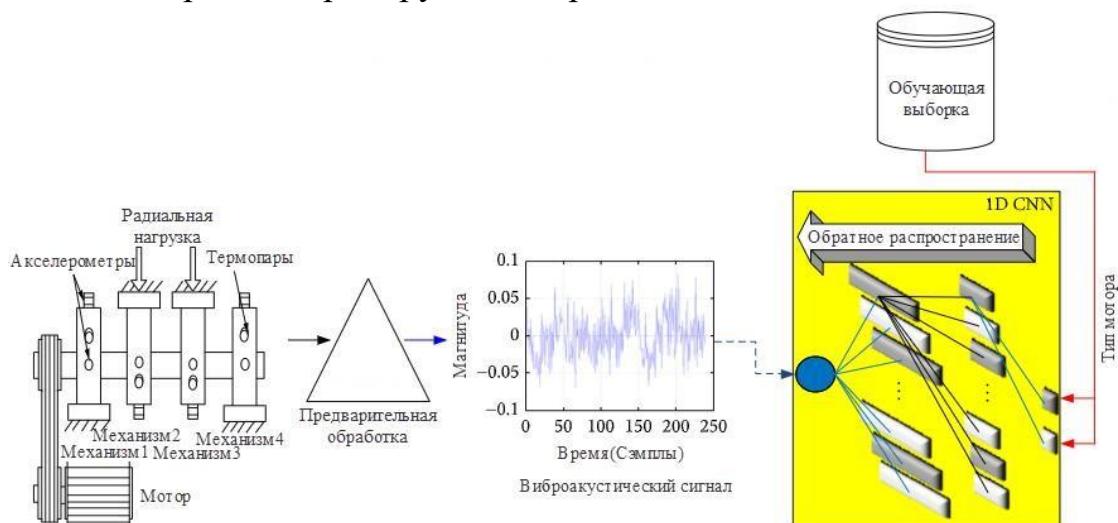


Рисунок 3.1 – Схема работы предлагаемой нейронной сети с электромеханической системой, состояние которой контролируется нейронной сетью.

Как видно из рисунка, во время работы мотора датчиками снимаются текущие показания работы, которые фиксируются и затем посылаются на блок предварительной обработки с целью нормирования данных. Таким образом имеется оцифрованный вибраакустический сигнал, с которым взаимодействует нейронная сеть, заранее обученная с использованием обучающей выборки, которая одновременно выступает в роли эталонной базы, из которой берутся сведения о типе мотора, для которого

осуществляется мониторинг состояния, для корректной оценки состояния механизмов.

Непосредственно сам процесс обучения нейронной сети концептуально выстроен аналогичным образом с тем лишь отличием, что эталонные виброакустические сигналы содержатся в эталонной базе, которую заранее подготавливают для корректного обучения нейронной сети, чтобы в итоге была выстроена максимально оптимальная структура сети с правильно подобранными параметрами для каждого слоя. Математическая часть данной процедуры представлена в следующем разделе.

### 3.2 Алгоритм работы одномерной свёрточной нейронной сети

Как говорилось ранее, в настоящей работе предлагается метод, который позволяет объединить выявление признаков и классификацию состояний рабочего механизма в единую процедуру с помощью адаптивной одномерной свёрточной нейронной сети. Любой снятый с датчиков виброакустический сигнал, поступающий на вход сети, может быть обработан в соответствии с топологией адаптивной свёрточной нейронной сети. Следует отметить, что в предлагаемой модели нейронной сети нейроны в свёрточных слоях способны выполнять как операции свёртки, так и операции формирования подвыборки, что позволяет упростить структуру и исключить из классической модели свёрточной нейронной сети подвыборочные слои. Данная идея отображена на рисунке 3.2.

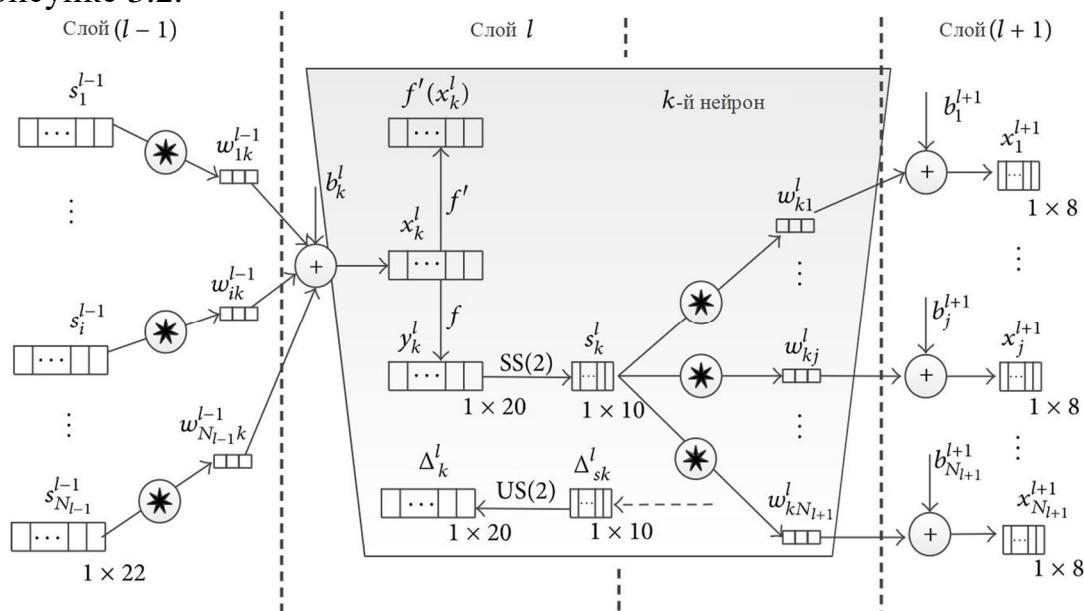


Рисунок 3.2 – Структура свёрточных слоёв предлагаемой модели одномерной свёрточной нейронной сети

Концептуально предлагаемая модель нейронной сети включает в себя явные входной и выходной слои, между которыми расположен ряд скрытых свёрточных слоёв.

Главным отличием между предлагаемой моделью одномерной свёрточной нейронной сети и классической моделью двумерной свёрточной сети является использование одномерных векторов вместо двумерных массивов, как для атрибутивного набора, так и для синапсов. Таким образом, одномерная свёртка вместе с разворачиванием вектора заменяют двумерную свёртку и последующее транспонирование массива. Кроме того, важным преимуществом предлагаемой модели является тот факт, что параметры, определяющие размер синапса и размер подвыборки, являются скалярными величинами. С другой стороны, обучение нейронной сети для обоих случаев осуществляется в соответствии с алгоритмом обратного распространения, который заключается в наличии обратной связи для коррекции параметров модели в ходе обучения.

Теперь опишем непосредственно сам процесс функционирования нейронной сети.

Прямое распространение одномерного вектора данных от сверточного слоя  $l-1$  к входу нейрона в  $l$ -ом слое определяется следующим выражением:

$$x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} conv1D(\omega_{ik}^{l-1}, s_i^{l-1}), \quad (1)$$

где  $x_k^l$  –  $k$ -й вход  $l$ -го слоя свёрточной нейронной сети,  $b_k^l$  – скалярное смещение  $k$ -го нейрона,  $s_i^{l-1}$  – выход  $i$ -го нейрона в  $(l-1)$ -ом слое,  $\omega_{ik}^{l-1}$  – значение весового коэффициента синапса, соединяющего  $i$ -й нейрон  $(l-1)$ -го слоя и  $k$ -й нейрон  $l$ -го слоя.

Согласно следующей формуле, в качестве промежуточного выхода  $k$ -го нейрона  $l$ -го слоя  $y_k^l$  выступает функция от входа  $x_k^l$ , а выходом  $k$ -го нейрона  $l$ -го слоя  $s_k^l$  является подвыборка из значений  $y_k^l$ :

$$\begin{cases} y_k^l = f(x_k^l) \\ s_k^l = y_k^l \downarrow ss \end{cases} \quad (2)$$

Единственным жёстким требованием к конфигурации аддитивной свёрточной нейронной сети, фрагмент которой представлен на рисунке 3.2, является задание коэффициента сэмплирования  $ss$  для выходного слоя равным количеству выделяемых признаков, по которым оценивается состояние механизма. В остальном конструкция рассматриваемой нейронной сети позволяет обрабатывать наборы данных различной длины с использованием любого количества свёрточных слоёв с различными коэффициентами сэмплирования.

Рассматривая алгоритм обратного распространения, ошибка начинает распространяться с выходного слоя и выражается через среднеквадратичное отклонение (СКО) выходного вектора от эталонного в соответствии с выражением:

|      |      |          |         |      |   |      |
|------|------|----------|---------|------|---|------|
|      |      |          |         |      | <i>BKR-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ)</i> | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |   | 18   |

$$E_p = CKO[t_i^p, (y_1^L, \dots, y_{N_L}^L)] = \sum_{i=1}^{N_L} (y_i^L - t_i^p)^2, \quad (3)$$

где  $N_L$  – мощность эталонной базы (количество классов в обучающей выборке),  $p$  – номер эталона,  $t^p$  и  $(y^L, K, y^L)$  – вектор  $p$ -го эталона и выходной вектор соответственно.

Алгоритм обратного распространения направлен на минимизацию вклада в эту ошибку внутренних параметров сети, а именно собственного веса  $\omega_{ik}^{l-1}$  и смещения  $b_k^l$   $k$ -го нейрона. Для этого итеративно используется метод градиентного спуска в отношении частных производных СКО по данным параметрам:

$$\begin{cases} \frac{\partial E}{\partial \omega_{ik}^{l-1}} = \Delta_k^l y_i^{l-1} \\ \frac{\partial E}{\partial b_k^l} = \Delta_k^l \end{cases} \quad (4)$$

Как мы видим из (4), пересчёт смещения  $k$ -го нейрона и весов всех сопряжённых с ним нейронов предыдущего слоя осуществляется с использованием дельта-функции  $\Delta_k^l$   $l$ -го слоя.

Обычное обратное распространение выполняется от входного слоя до последнего свёрточного слоя, в соответствии с выражением:

$$\frac{\partial E}{\partial s_k^l} = \Delta s_k^l = \sum_{i=1}^{N_{l+1}} \left( \frac{\partial E}{\partial x_i^{l+1}} * \frac{\partial x_i^{l+1}}{\partial s_k^l} \right) = \sum_{i=1}^{N_{l+1}} (\Delta_i^{l+1} * \omega_{ki}^l) \quad (5)$$

Как только первый раз выполняется обратное распространение от слоя  $(l+1)$  к  $l$ -му слою, мы можем далее отправить результат данной операции на вход дельта-функции  $\Delta_k^l$ . Результат перезаписывается в соответствии с выражением:

$$\Delta_k^l = \frac{\partial E}{\partial y_k^l} * \frac{\partial y_k^l}{\partial x_k^l} = \frac{\partial E}{\partial u s_k^l} * \frac{\partial u s_k^l}{\partial y_k^l} = f'(x_k^l) = u * p(\Delta_s^l) * \beta * f'(x_k^l), \quad (6)$$

где  $\beta = (ss)^{-1}$ , так как каждый элемент  $s_k^l$  получен путём усреднения количества элементов промежуточного выхода.

Обратное распространение дельта-ошибки определяется, как:

$$\Delta s_k^l = \sum_{i=1}^{N_{l+1}} conv1Dz(\Delta_i^{l+1}, rev(\omega_{ki}^l)), \quad (7)$$

где оператор  $rev()$  используется для переворачивания массива, а оператор  $conv1Dz()$  – для выполнения полной свёртки в одно измерение с добавлением  $K-1$  нулей.

Следующей формулой выражаются чувствительности веса и смещения:

|      |      |          |         |      |   |      |
|------|------|----------|---------|------|---|------|
|      |      |          |         |      | <i>BKR-HГТУ-09.03.01-(16-B-2)-007-2020 (ПЗ)</i> | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |   | 19   |

$$\begin{cases} \frac{\partial E}{\partial \omega_{ki}^l} = conv1D(s_k^l, \Delta_i^{l+1}) \\ \frac{\partial E}{\partial b_k^l} = \sum_n \Delta_k^l(n) \end{cases} \quad (8)$$

В результате алгоритм обратного распространения, представленный в [48], используется итеративно с коэффициентом обучения  $\varepsilon$  для коррекции веса и смещения, согласно следующей формуле:

$$\begin{cases} \omega_{ik}^{l-1}(t+1) = \omega_{ik}^{l-1}(t) - \varepsilon \frac{\partial E}{\partial \omega_{ki}^l} \\ b_k^l(t+1) = b_k^l(t) - \varepsilon \frac{\partial E}{\partial b_k^l} \end{cases} \quad (9)$$

Анализируя выше описанный математический аппарат, выраженный в виде формул (1 – 9), нетрудно сделать вывод, что процесс формирования адаптивной свёрточной нейронной сети заключается в итеративной корректировке параметров нейронов всех слоёв в ходе подгонки выходных данных под эталонную базу с целью получения корректного классификатора, который в дальнейшем будет использоваться для диагностирования текущего состояния механизма по записанному виброакустическому сигналу, снятыму с акселерометра.

В заключение отметим, что на рисунке 3.3 представлена структурная модель разрабатываемой в настоящей работе нейронной сети.

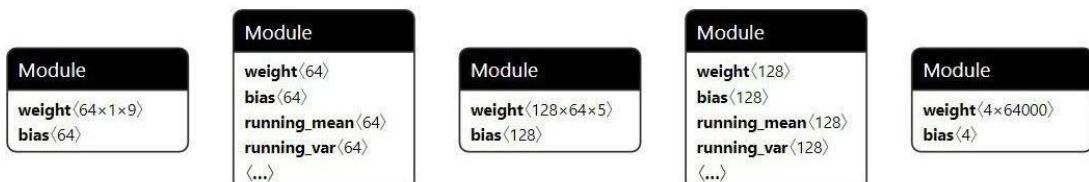


Рисунок 3.3 – Структурная модель разрабатываемой одномерной свёрточной нейронной сети

## **4. Разработка программных средств**

### **4.1 Разработка программного интерфейса системы**

Разработанная в рамках выполнения выпускной квалификационной работы система включает в себя две подсистемы: подсистему обучения нейронной сети и подсистему оценки состояния механизма посредством обученной нейронной сети. Дадим подробное описание каждой из подсистем.

#### **4.1.1 Подсистема обучения нейронной сети**

При разработке подсистемы, отвечающей за обучение нейронной сети, было разработано 3 модуля, каждый из которых является самостоятельным .ру-модулем:

- 1) nn\_model.py – модуль, содержащий прототип реализуемой одномерной двуслойной нейронной сети
- 2) train\_helper.py – модуль, содержащий модель методики обучения нейронной сети
- 3) helper.py – вспомогательный модуль, содержащий вспомогательные функции, отвечающие за считывание данных из файла, а также за предварительную обработку считанных данных. Кроме того, данный модуль подгружает необходимые библиотеки, отвечающие за математические вычисления.

В качестве исходных данных для подсистемы обучения нейронной сети являются:

- 1) прототип модели нейронной сети
- 2) обучающая выборка

В качестве выходных данных служит файл с расширением .pth, содержащий в себе реальную модель обученной нейронной сети, который используется подсистемой оценки состояния механизма.

#### **4.1.2 Подсистема оценки состояния механизма посредством обученной нейронной сети**

Данная подсистема так же включает в себя три модуля, являющиеся самостоятельными .ру-модулями:

1) model.py – модуль, отвечающий за загрузку в программу реальной модели обученной нейронной сети, которую мы получаем на выходе первой подсистемы, а также за считывание и обработку считываемых данных из файла, содержащего сэмплированную форму вибраакустического сигнала, на базе которого принимается решение о состоянии механизма.

2) controller.py – модуль управления, отвечающий за отображение файлов, подвергаемых анализу нейронной сетью, корректировку списка файлов (добавление и удаление), построение графиков вибраакустического сигнала, сохранение данных, а также запуска процедуры оценки состояния механизма, которому соответствует анализируемый файл.

3) view.py – модуль, отвечающий за внешний вид и поведенческие функции элементов графического интерфейса программы, которые

|      |      |          |         |      |  |      |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      | BKR-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |  | 21   |

используются пользователем для управления программой. Следует отметить, что методы, описанные в модуле controller.py, по сути, являются обработчиками событий для элементов графического интерфейса (кнопки, таблица, график), размещаемых на главной форме приложения.

#### 4.1.3 Описание пользовательского графического интерфейса подсистемы оценки состояния механизма посредством обученной нейронной сети

Внешний вид программы представлен на рисунке 4.1.

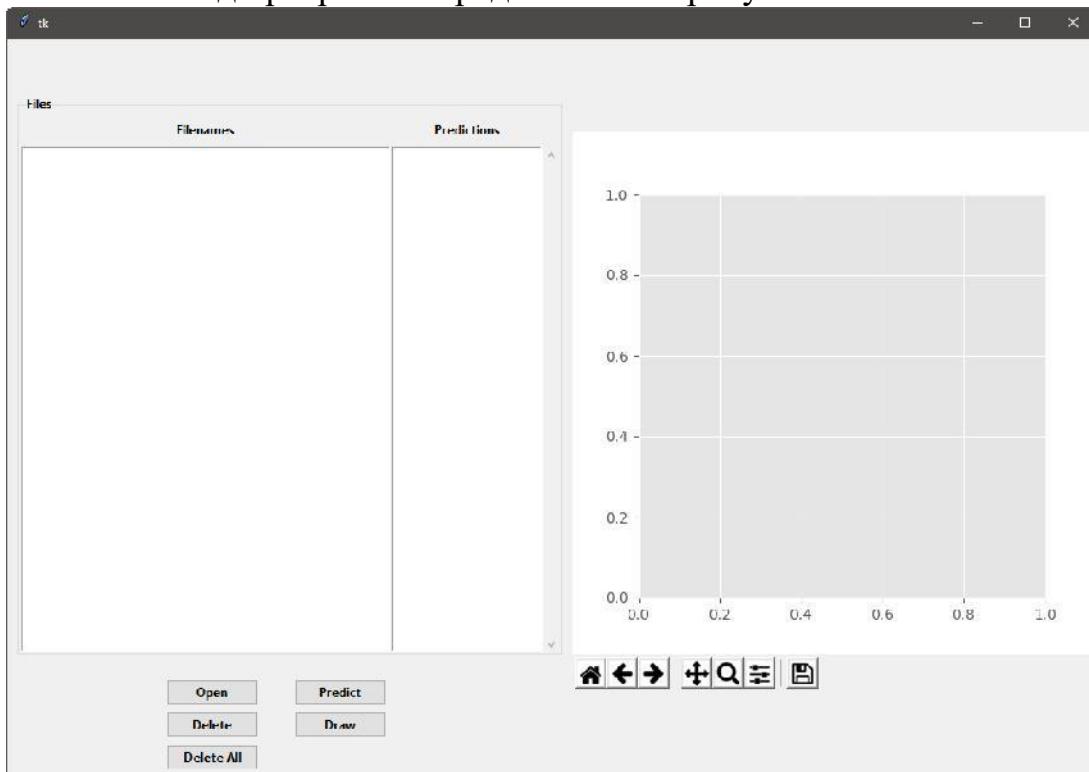


Рисунок 4.1 – Внешний вид программы

Как видно из рисунка, на главной форме приложения расположены следующие элементы управления:

- одномерная таблица Filenames, предназначенная для хранения и отображения загруженных в память программы имён файлов, содержащих данные для анализа нейронной сетью;
- одномерная таблица Predictions, предназначенная для хранения и отображения результатов анализа сигналов, соответствующих загруженным в файлам;
- кнопка Open – предназначена для открытия файла, содержащего данные вибраакустического сигнала для обработки нейронной сетью и добавления его имени в таблицу Filenames;
- кнопка Delete – предназначена для удаления имени файла из таблицу Filenames;
- кнопка Delete All – предназначена для очистки таблицы Filenames;

- кнопка Predict – предназначена для запуска процедуры обработки выбранного файла используемой нейронной сетью;
- кнопка Draw – предназначена для визуального отображения формы виброакустического сигнала в виде графика, который строится на базе данных, считываемых из файла;
- кнопка возврата к изначальному графику (домик) – используется для выставления графика по умолчанию;
- кнопка возврата к предыдущему состоянию графика (стрелка влево);
- кнопка перехода к следующему состоянию графика (стрелка вправо);
- кнопка смещения графика при увеличении (четырёхнаправленный крест);
- кнопка увеличения графика (лупа);
- кнопка вызова настроек полотна, на котором отображается график (горизонтальные ползунки);
- кнопка сохранения текущего графика в файл в формате \*.PNG (дискета);
- окно построения графика, на котором отображается считанный из файла график сигнала с учётом выставления дополнительных настроек отображения (масштабирование, смещения и т.д.).

На рисунке 4.2 представлен внешний вид дополнительной формы, которая открывается при нажатии на кнопку настроек графика.

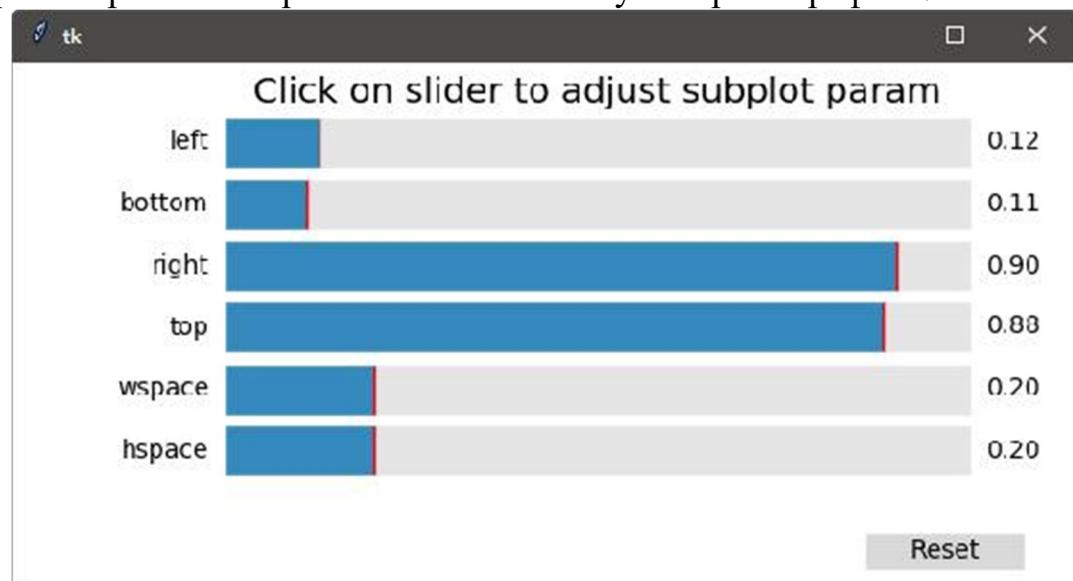


Рисунок 4.2 – Форма управления настройками графика

Согласно рисунку, здесь представлены ползунки, отвечающие за отступы графика от краёв полотна графика в долевом соотношении (от 0 до 1), а также кнопка Reset используемая для выставления оптимальных значений, отмеченных на ползунках красными пороговыми чертами.

## 4.2 Программная реализация модулей системы

В данном разделе дадим детальное описание каждого модуля с позиции внутренней реализации, а именно опишем используемые в каждом модуле функции и их назначение. Для удобства восприятия отобразим все модули подсистем и их функции в виде таблиц 4.1 и 4.2. Информация о входных и выходных параметрах для каждой функции содержится в комментариях исходного кода для каждой подсистемы (приложения А и Б).

Таблица 4.1 – Модули подсистемы обучения нейронной сети и их функции

| Модуль          | Назначение модуля  | Функция   | Назначение функции   |
|-----------------|--|---|--|
| nn_model.py     | Хранение прототипа одномерной свёрточной двуслойной нейронной сети         | <code>__init__(self, n_in)</code>                           | Инициализация начального состояния необученной нейронной сети  |
|                 |  | <code>forward(self, x)</code>                               | Переход нейронной сети к следующему состоянию в процессе обучения после перераспределения синапсов и их связей |
| train_helper.py | Хранение методов обучения нейронной сети в соответствии с заданной моделью | <code>get_dataloader(train_ds, valid_ds, bs)</code>         | Загрузка обучающей и контрольной выборок данных  |
|                 |  | <code>loss_batch(model, loss_func, xb, yb, opt=None)</code> | Подсчёт потерь (ошибок классификации) для получения качественных характеристик работы обученной нейронной сети |

|           |   |  |  |
|-----------|---|--|--|
|           |   | fit(epochs, model,<br>loss_func, opt, train_dl,<br>valid_dl,<br>one_cycle=None,<br>train_metric=False) | Обучение<br>нейронной сети за<br>конечное число<br>шагов при<br>заданной<br>точности и вывод<br>сведений о<br>качественных<br>характеристиках<br>обученной<br>нейронной сети |
|           |   | validate(model, dl,<br>loss_func)  | Оценка<br>качественных<br>характеристик<br>обученной<br>нейронной сети<br>при заданной<br>точности   |
| helper.py | Считывание<br>данных из<br>файла,<br>предварительн<br>ая обработка<br>считанных<br>данных | matfile_to_dic(folder_path)  | Считывание<br>данных из файла<br>в активный<br>словарь<br>подсистемы<br>обучения<br>нейронной сети   |
|           |   | remove_dic_items(dic)  | Очистка<br>активного<br>словаря<br>подсистемы  |
|           |   | matfile_to_df(folder_path)   | Предварительная<br>обработка<br>считанных<br>данных,<br>хранящихся в<br>активном словаре<br>подсистемы   |
|           |   | divide_signal(df,<br>segment_length)   | Разбиение<br>считанного<br>сигнала на<br>сегменты для<br>ускорения<br>предварительной<br>обработки   |

| Изм. | Лист | № докум. | Подпись | Дата | БКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | Лист |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      |  | 25   |

|  |  |  |  |
|--|--|--|--|
|  |  | normalize_signal(df)                                       | Нормализация считанного сигнала                      |
|  |  | get_df_all(data_path, segment_length=512, normalize=False) | Подготовка данных для начала обучения нейронной сети |

Таблица 4.2 – Модули подсистемы оценки состояния механизма посредством обученной нейронной сети

| Модуль        | Назначение модуля                           | Функция                             | Назначение функции   |
|---------------|---|-------------------------------------|--|
| model.py      | Управление моделью обученной нейронной сети | __init__(self)                      | Инициализация модели обученной нейронной сети  |
|               |   | predict(self, file_index)           | Оценка состояния механизма, которому соответствует файл с данными вибраакустического сигнала |
|               |   | update_prediction(self, file_index) | Обновление результата оценки состояния механизма   |
|               |   | read_files(self, filepaths)         | Считывание файлов для анализа состояния  |
| controller.py | Управление программой                       | __init__(self, page)                | Инициализация главного окна программы  |
|               |   | select_files(self)                  | Выбор файла для анализа  |
|               |   | delete_list_all(self)               | Очистка списка файлов для анализа  |
|               |   | delete_list_selected(self)          | Удаление выбранного элемента из списка файлов для анализа                                    |

|         |                                 |                                    |  |
|---------|---------------------------------|------------------------------------|--|
|         |                                 | get_list_selection(self)           | Отображение списка файлов для анализа                                      |
|         |                                 | plot_something(self)               | Отображение графика  |
|         |                                 | get_predictions(self)              | Возвращение результата анализа состояния механизма на базе файла из списка |
| view.py | Графический интерфейс программы | __init__(self, parent, controller) | Инициализация положений и размеров элементов управления                    |

Теперь продемонстрируем работу пользовательской программы оценки состояния механизма на базе обученной сети.

На рисунке 4.3 представлено стартовое окно программы.

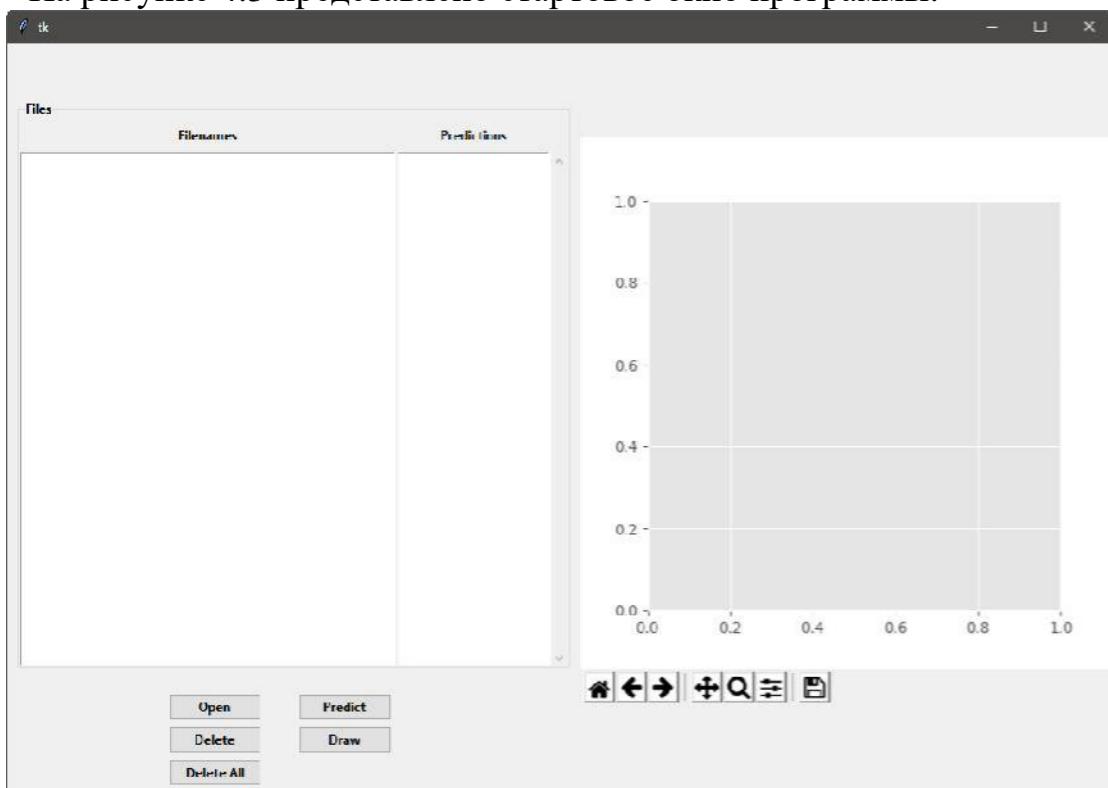


Рисунок 4.3 – Стартовое окно программы

Для добавления файла в список нажимаем кнопку Open. В результате откроется диалоговое окно выбора файла, как показано на рисунке 4.4.

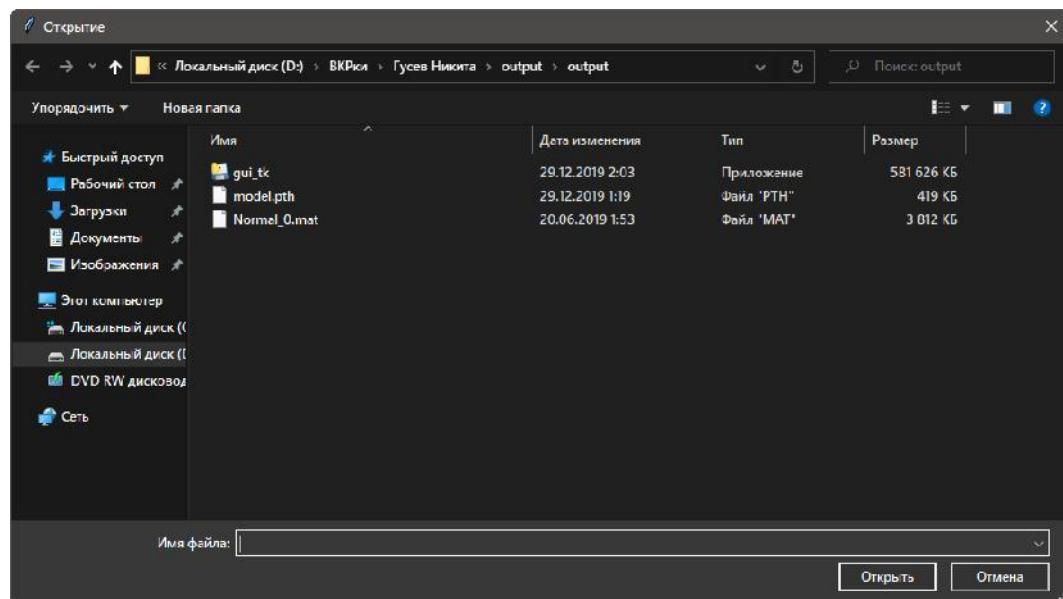


Рисунок 4.4 – Диалоговое окно открытия файла

Выбираем файл с расширением .mat (в нашем случае Normal\_0.mat). В результате его имя добавится в список Filenames, как показано на рисунке 4.5.

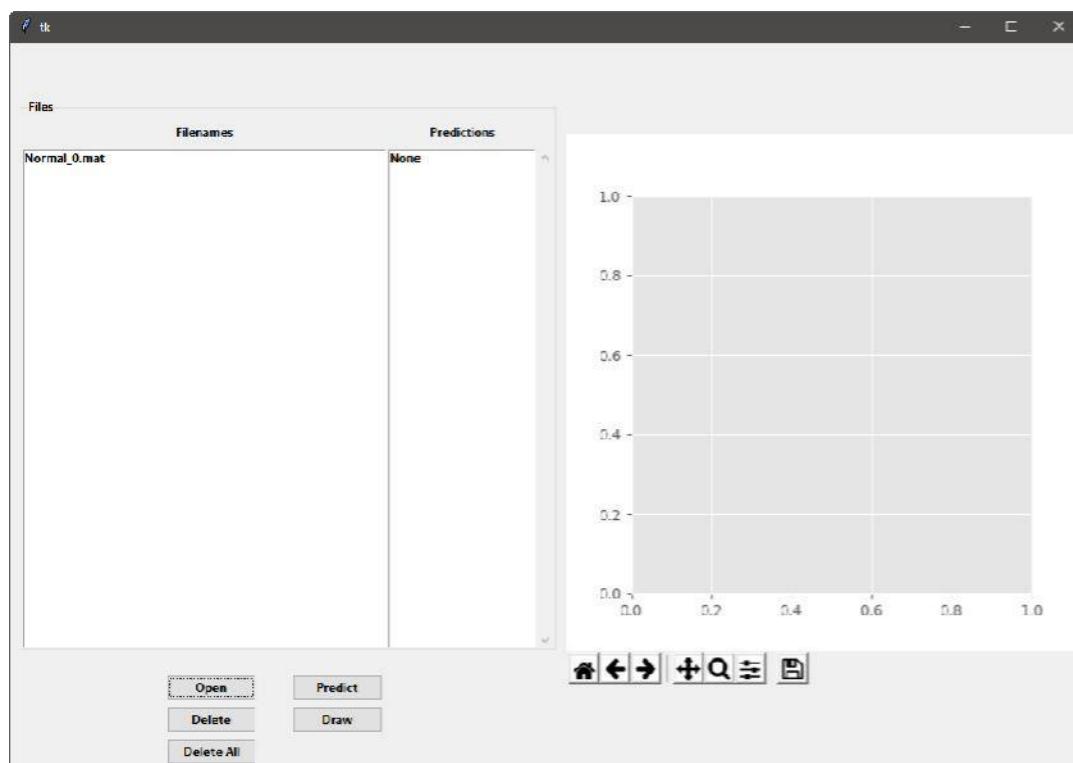


Рисунок 4.5 – Программа после добавления файла в список

Как видно из рисунка, в списке Predictions выставлено значение по умолчанию None. Это вызвано тем, что для механизма, состоянию которого соответствует выбранный файл, ещё не проводилась оценка состояния посредством обученной нейронной сети.

|      |      |          |         |      |  |      |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      | BKR-НГТУ-09.03.01-(16-B-2)-007-2020 (ПЗ) | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |  | 28   |

Чтобы удалить данный файл из списка, нужно навести на него курсор мыши и нажать левую кнопку мыши, чтобы программа знала, что был осуществлён выбор данного файла, как показано на рисунке 4.6.

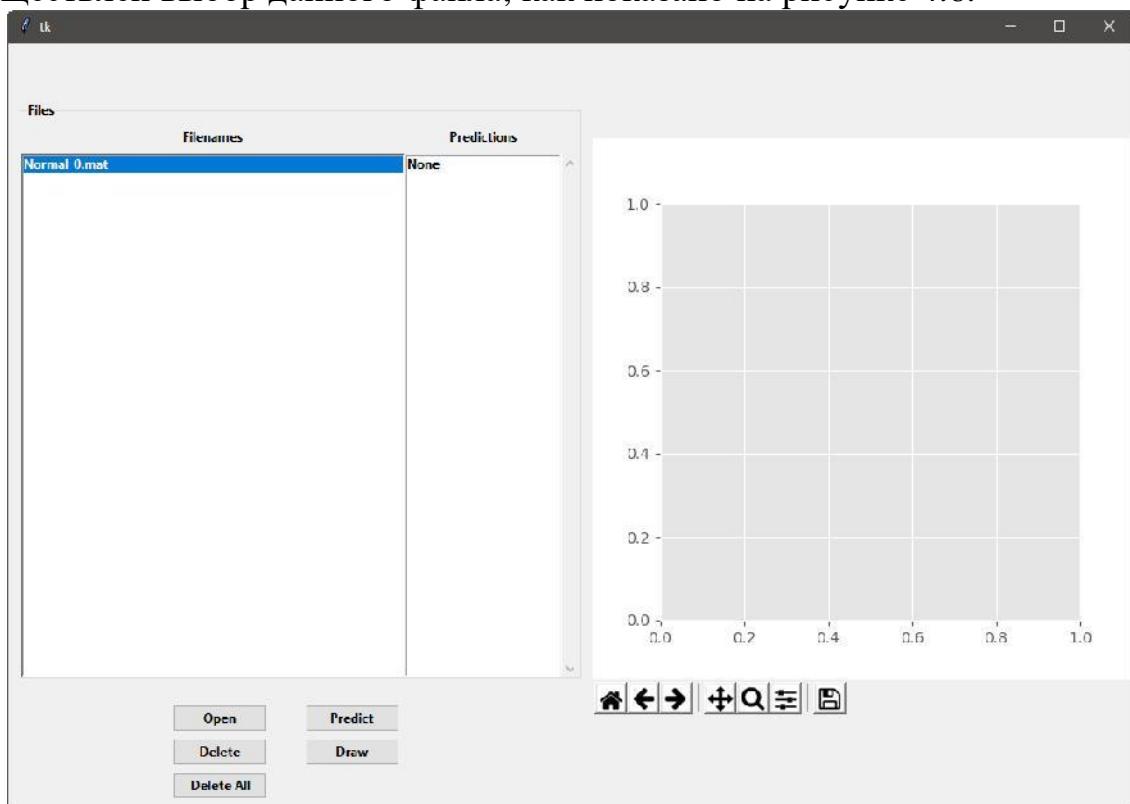


Рисунок 4.6 – Выбор файла из списка

После этого пользователь может либо удалить его, нажав кнопку Delete, исход чего показан на рисунке 4.7 (сразу отметим, что при нажатии кнопки Delete All будет тот же результат с той лишь разницей, что не нужно выбирать файлы – произойдёт полная очистка списка), либо выполнить отображение графика, нажав кнопку Draw, как показано на рисунке 4.8, либо выполнить оценку состояния механизма нажатием кнопки Predict, в результате чего значение None изменится на N (соответствует механизму, который в норме) или F (соответствует механизму, состояние которого не соответствует норме), как показано на рисунке 4.9

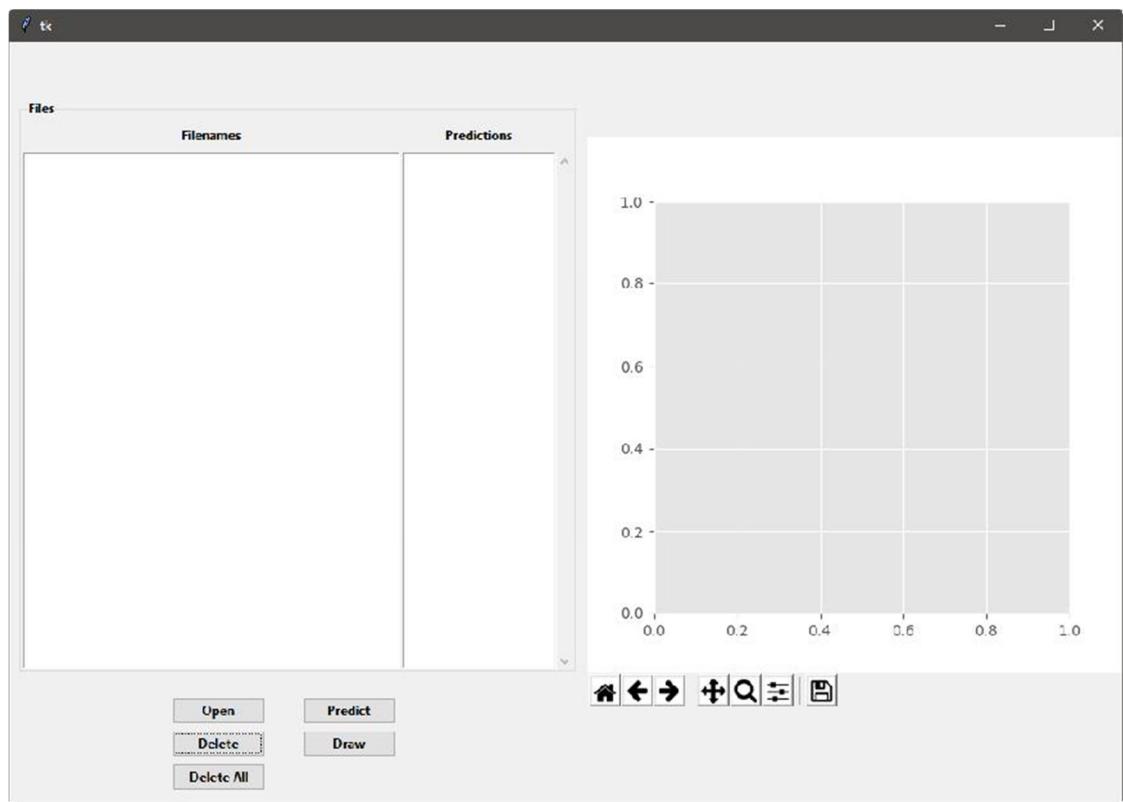


Рисунок 4.7 – Результат нажатия кнопки Delete

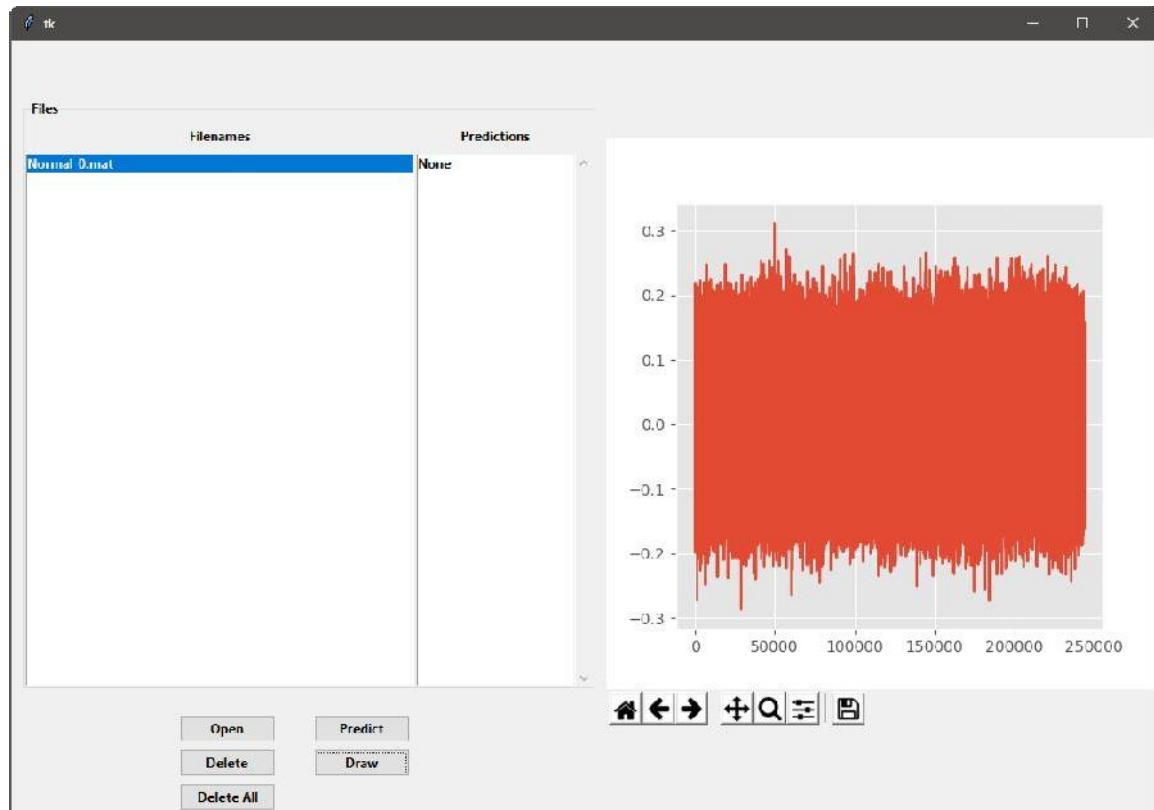


Рисунок 4.8 – Результат нажатия кнопки Draw

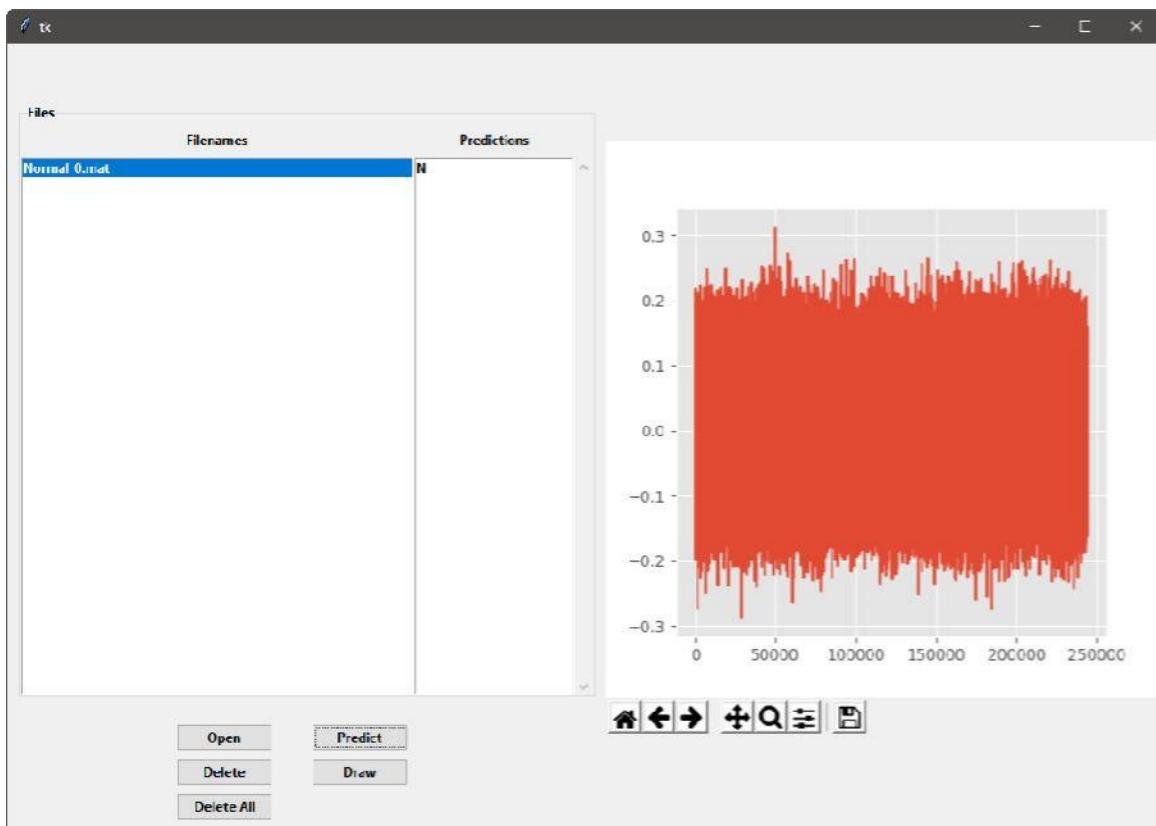


Рисунок 4.9 – Результат нажатия кнопки Predict

Для более детального рассмотрения формы виброакустического сигнала можно произвести масштабирование графика нажатие кнопки с изображением лупы, после чего с помощью мыши необходимо выделить область, которую хотим увеличить в масштабе. Результат данного действия представлен на рисунке 4.10. Если необходимо сместиться по графику, то нажимаем кнопку с изображением четырёхнаправленного креста, затем щёлкаем левой кнопкой мыши по точке на графике, зажимаем её и двигаем мышь от отпускания кнопки. Результат данной операции показан на рисунке 4.11, где хорошо видно смещение графика относительно его предыдущего состояния. Кроме того отметим, что при нажатии кнопки с изображением стрелки влево, пользователь вернётся к состоянию с рисунка 4.10. А если после этого нажмёт кнопку с изображением стрелки вправо, то вернётся назад к состоянию графика с рисунка 4.11. Нажатие кнопки с изображением дома вернёт график в исходное состояние, продемонстрированное на рисунке 4.8.

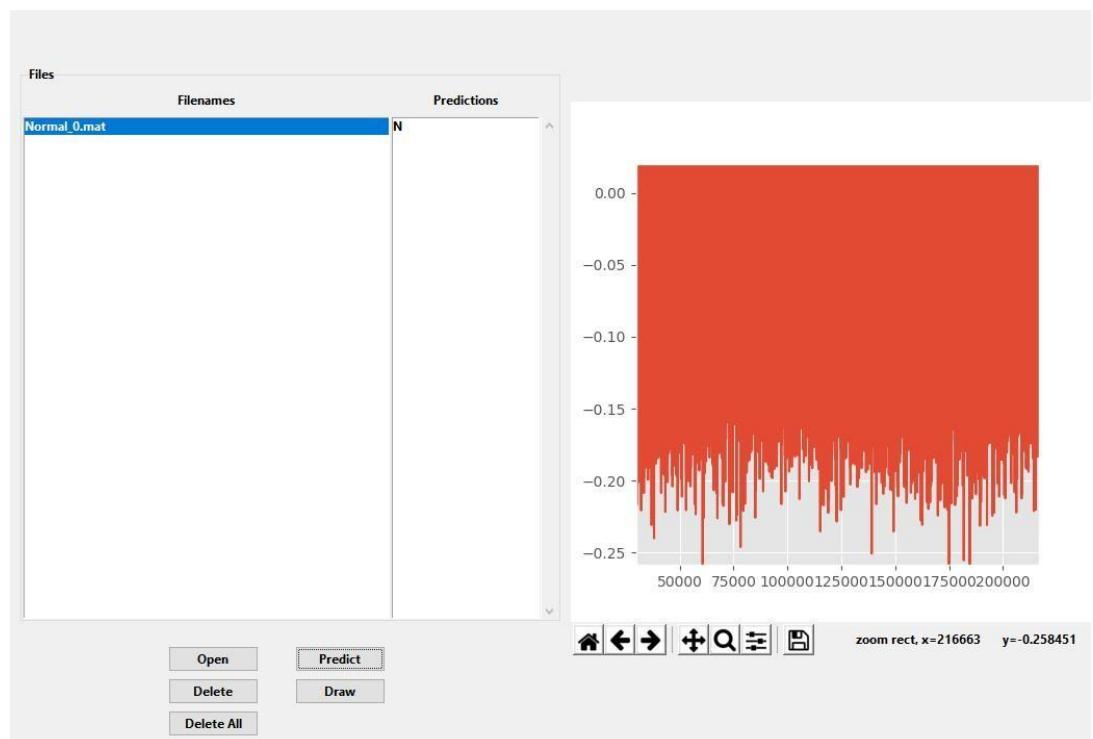


Рисунок 4.10 – Масштабирование графика

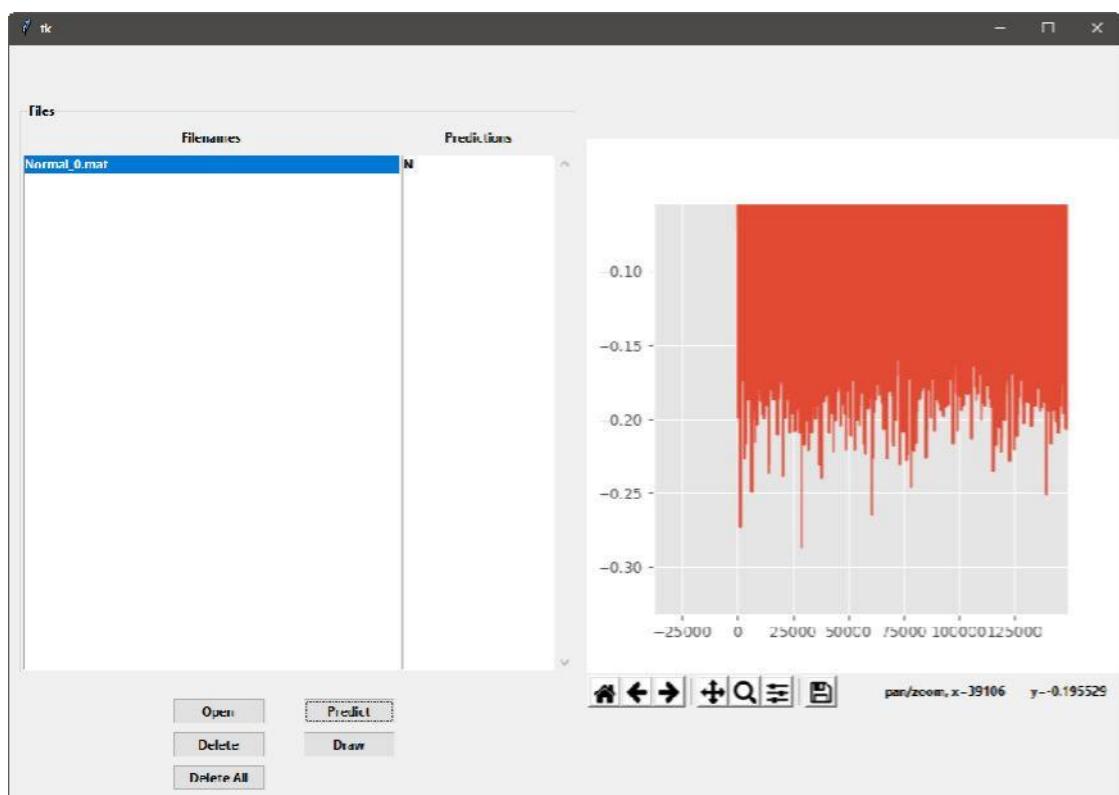


Рисунок 4.11 – Смещение графика

Если необходимо сохранить текущий график в том виде, в котором он представлен на текущий момент времени, то необходимо нажать кнопку с

изображением дискеты. В результате вызовется диалоговое окно сохранения файла, как показано на рисунке 4.12.

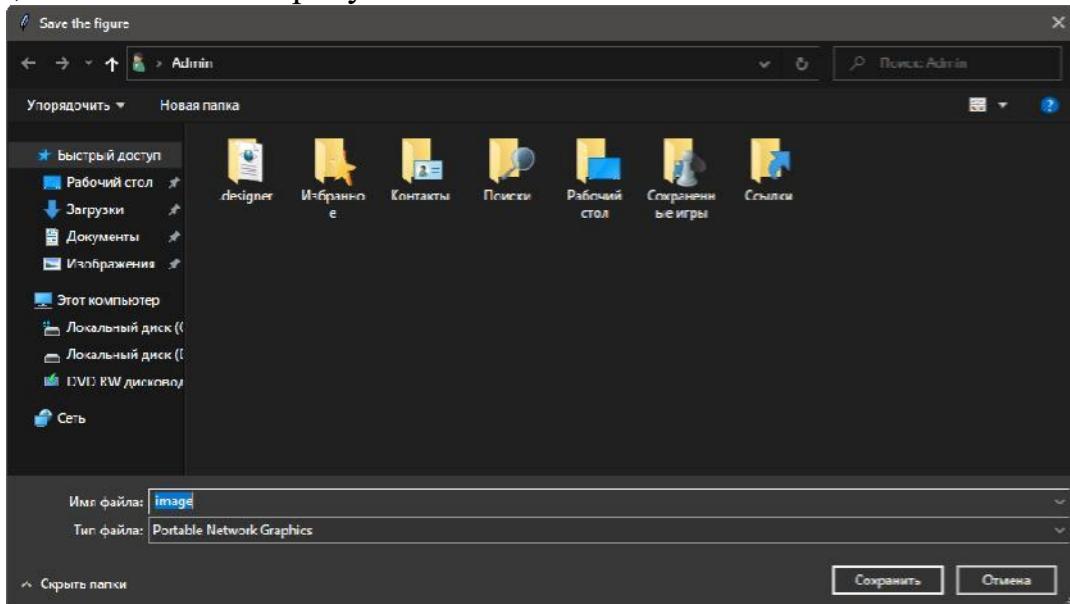


Рисунок 4.12 – Диалоговое окно сохранения рисунка в формате PNG

Как говорилось ранее, нажатие на кнопку с изображением горизонтальных ползунков открывает дополнительную форму с ползунками для настройки графика относительно полотна (рисунок 4.13).

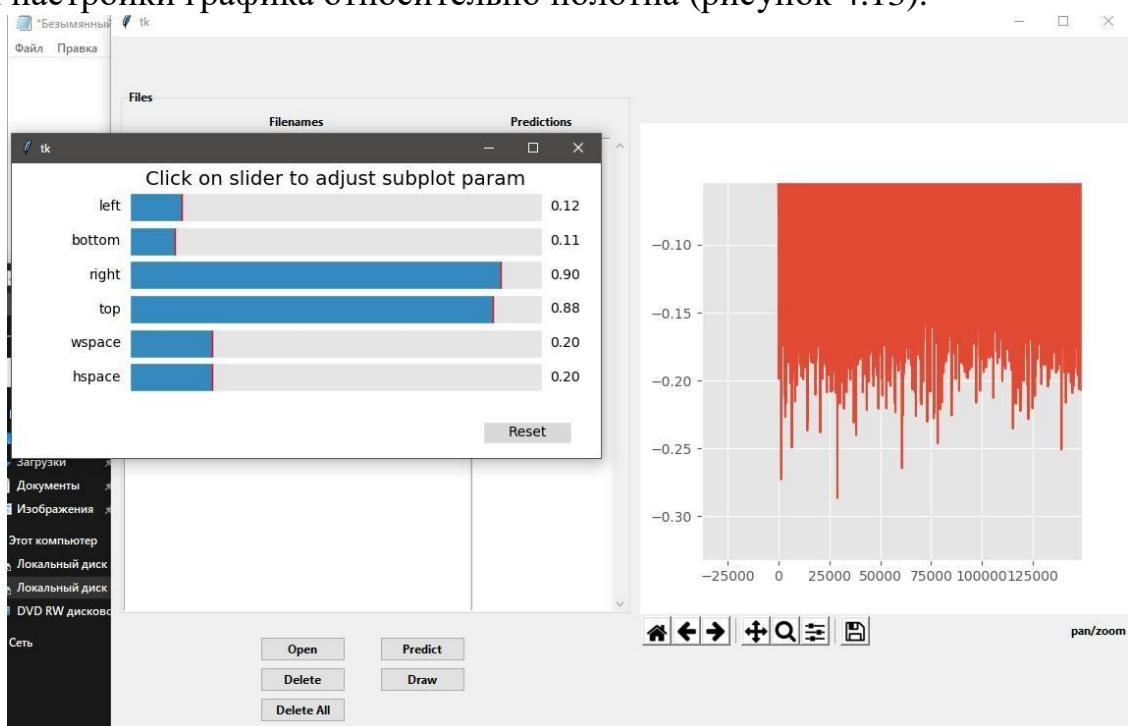


Рисунок 4.13 – Настройка параметров графика

На рисунках 4.14 и 4.15 показано, как меняется график вследствие изменения положений ползунков. А нажатие на кнопку Reset, как уже было

сказано выше, возвращает настройки, а, следовательно, и график к оптимальным значениям, отмеченным красными линиями.

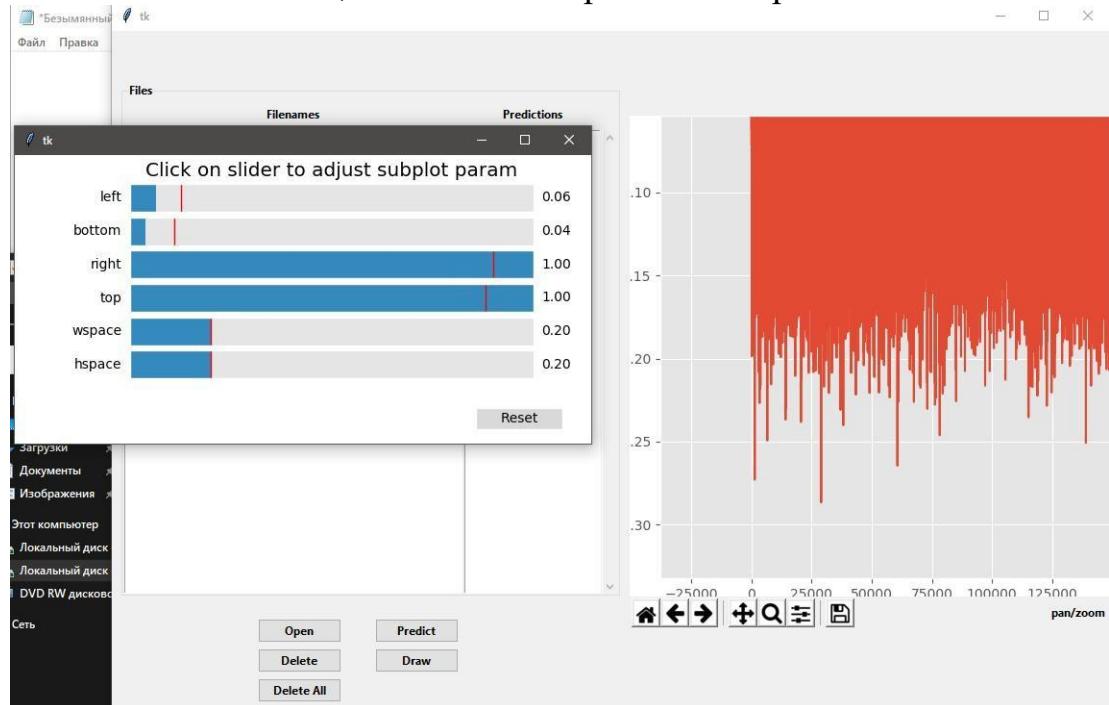


Рисунок 4.14 – Настройка параметров графика 2

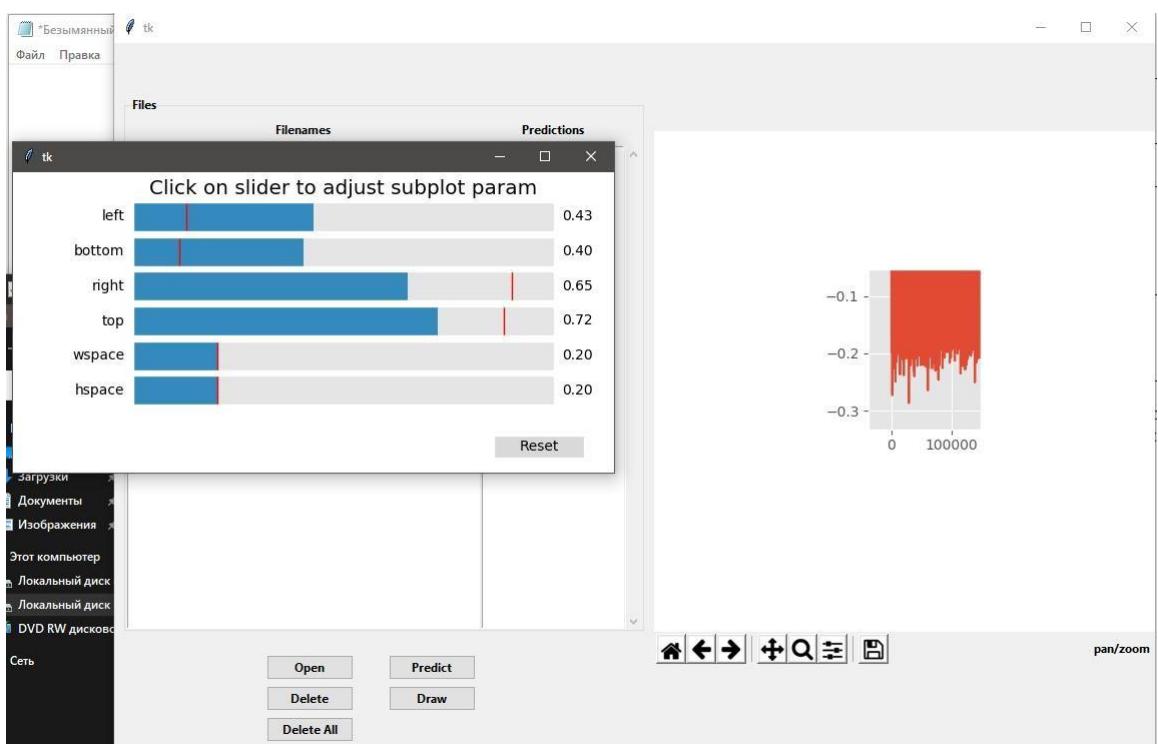


Рисунок 4.15 – Настройка параметров графика 3

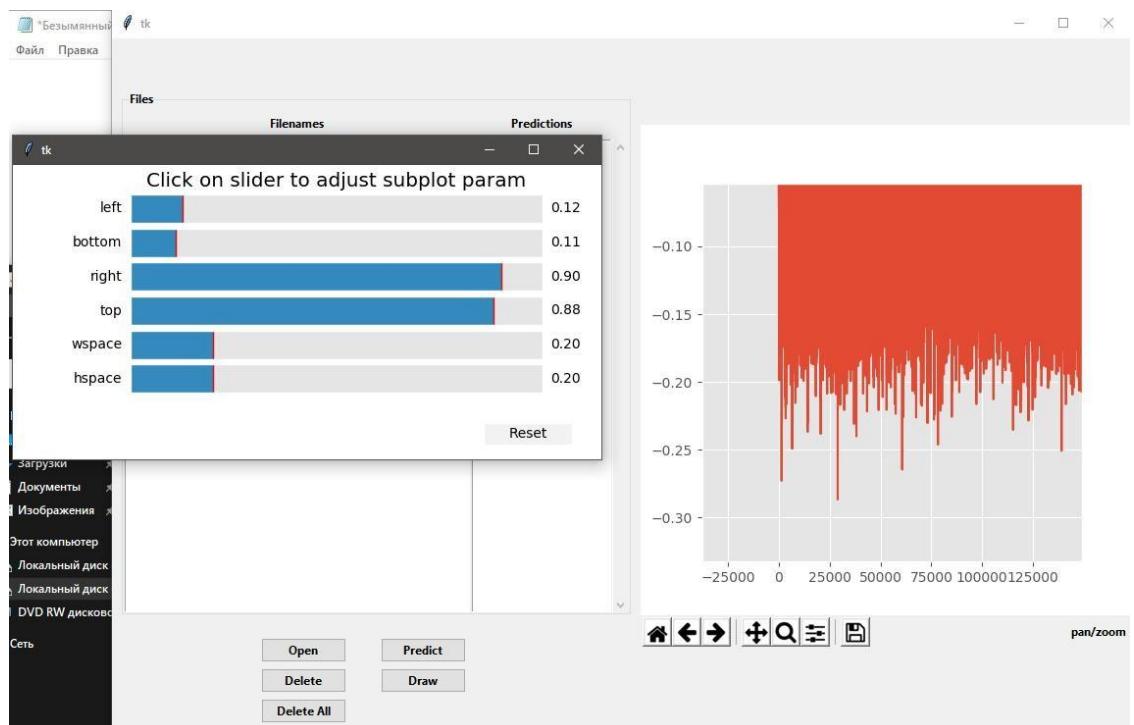


Рисунок 4.16 – Результат нажатия кнопки Reset

## **5. Тестирование системы**

### **5.1 Описание набора данных**

Эталонные данные являются базисной основой для любого метода глубокого обучения. Для разработки предлагаемого алгоритма, равно как и для обнаружения дефектов различных механизмов, необходим правильно подобранный корректный набор данных.

Поскольку процесс разрушения механизмов, как правило, занимает несколько лет, проведение экспериментов и сбор данных осуществляется либо с использованием механизмов с искусственно введенными неисправностями, либо с помощью метода ускоренного тестирования ресурса.

Несмотря на то, что сбор данных все еще занимает много времени, к счастью, несколько организаций предприняли усилия и предоставили несущие наборы данных о сбоях для работы над исследованием вибрационного сигнала. Эти наборы данных также служат стандартами для оценки и сравнения различных алгоритмов.

Кроме того, в ходе настоящей работы для проведения дополнительных испытаний был разработан контрольно-испытательный стенд, включающий в себя электродвигатель мощностью 2 л.с., акселерометр, который представлен на рис. 5.1.

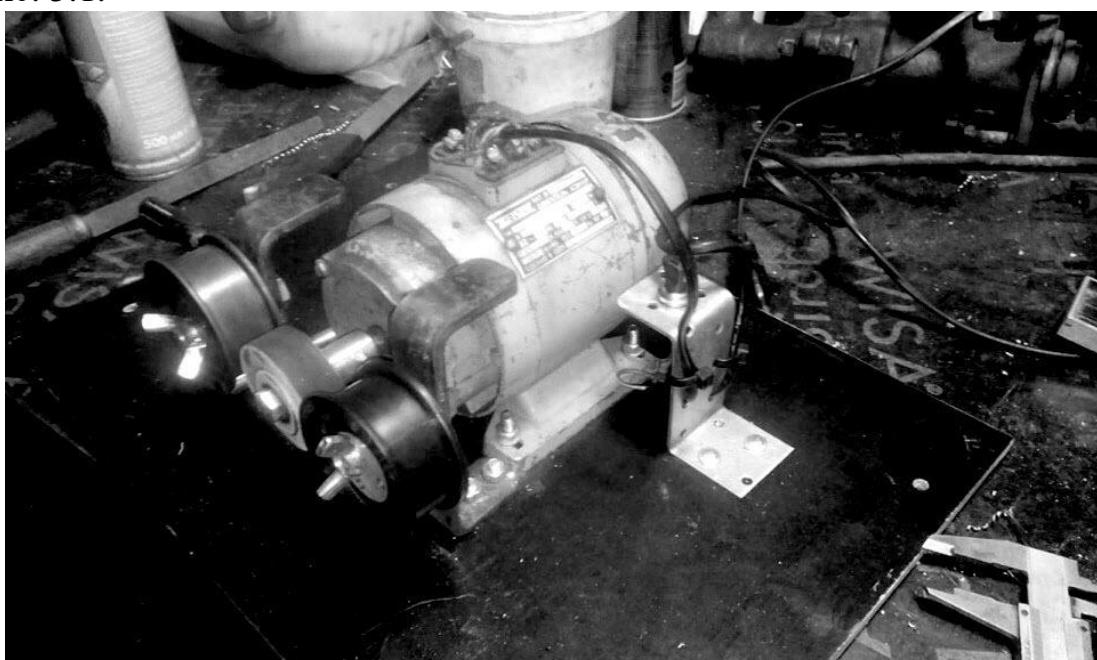


Рисунок 5.1 – Контрольно-испытательный стенд

Измерения осуществляются в точках, которые расположены рядом с подшипниками двигателя и удаленно от них.

Рассмотрим предоставленный набор исходных данных, в котором содержатся данные о подшипниках исправной и неисправной конструкции. В текущих материалах содержатся искусственные дефекты диаметров, которые производятся в различных местах подшипников: внутренние дорожки

|      |      |          |         |      |   |
|------|------|----------|---------|------|---|
|      |      |          |         |      | Лист<br><i>BKR-НГТУ-09.03.01-(16-B-2)-007-2020 (ПЗ)</i> |
| Изм. | Лист | № докум. | Подпись | Дата |   |

качения, наружные дорожки качения и шаровые. Данные вибро-акустических сигналов записывались для нагрузок двигателя от 0 до 2 л.с. (соответствует циклическим скоростям двигателя от 1797 до 1720 об / мин) с подключением акселерометров со стороны привода и со стороны вентилятора. Снятые с датчиков данные при частоте дискретизации 12 кГц сохранялись в виде файлов Matlab. Таким образом, каждый файл Matlab содержит от ~ 120k до ~ 240k точек выборки.

## 5.2 Описание методики тестирования

В настоящей работе при тестировании разработанной нейронной сети основной упор делается на оценку качественных показателей работы сети, а именно в ходе тестирования оценивается точность работы (классификации) в долевом соотношении и уровень потерь (оценка ошибок классификации первого рода) в форме СКО выходного вектора от эталона.

Точность работы нейронной сети определяется как отношение числа верно определённых состояний к мощности всей выборки, согласно ниже представленной формуле:

$$Acc = \frac{N - E}{N} * 100\% , (10)$$

где  $N$  – мощность выборки, а  $E$  – количество ошибок классификации.

Также напомним, что количество потерь оценивается в соответствии с (3) как расстояние в метрическом пространстве между выходным вектором, полученным по анализируемому сигналу, и эталонным вектором, в пользу которого было принято решение, вычисляемое по Евклидовой метрике.

## 5.3 Результаты вычислительного эксперимента

Для детального анализа сведем результаты обучения и проверки качественных параметров разработанной нейронной сети в таблицу, в которой для каждого измерения вычисляются долевое значение точности работы нейронной сети при обучении и при проверке работы, а также процентное количество потерь от общей выборки при тех же формах работы нейронной сети.

Суммарно было произведено 100 измерений, результаты которых сведены в таблицу 5.1.

Таблица 5.1 – Результаты эксперимента

| Измерение | Потери при обучении | Потери при проверке | Точность при обучении | Точность при проверке |
|-----------|---------------------|---------------------|-----------------------|-----------------------|
| 0         | 5.71656             | 5.54417             | 0.49499               | 0.50483               |
| 1         | 1.84585             | 2.22714             | 0.68347               | 0.63034               |
| 2         | 0.56988             | 0.86683             | 0.85640               | 0.78897               |
| 3         | 0.32133             | 0.66614             | 0.90093               | 0.82897               |
| 4         | 0.55338             | 0.73910             | 0.91577               | 0.88276               |

| Измерение | Потери при обучении | Потери при проверке | Точность при обучении | Точность при проверке |
|-----------|---------------------|---------------------|-----------------------|-----------------------|
| 5         | 0.44499             | 0.85631             | 0.92199               | 0.85931               |
| 6         | 0.29456             | 0.69552             | 0.91060               | 0.85655               |
| 7         | 0.12414             | 0.40639             | 0.96099               | 0.91724               |
| 8         | 0.18787             | 0.58174             | 0.95720               | 0.89103               |
| 9         | 0.04904             | 0.31000             | 0.98654               | 0.92690               |
| 10        | 0.04202             | 0.30946             | 0.98619               | 0.92552               |
| 11        | 0.01920             | 0.19087             | 0.99551               | 0.95310               |
| 12        | 0.09091             | 0.44027             | 0.97031               | 0.91034               |
| 13        | 0.05751             | 0.33841             | 0.98240               | 0.93793               |
| 14        | 0.01399             | 0.19014             | 0.99586               | 0.95310               |
| 15        | 0.04182             | 0.33136             | 0.98274               | 0.93103               |
| 16        | 0.03836             | 0.32382             | 0.98516               | 0.93103               |
| 17        | 0.04583             | 0.29518             | 0.98619               | 0.94069               |
| 18        | 0.01374             | 0.21046             | 0.99551               | 0.94345               |
| 19        | 0.04524             | 0.32437             | 0.98826               | 0.93241               |
| 20        | 0.04834             | 0.34305             | 0.98723               | 0.92414               |
| 21        | 0.03526             | 0.32857             | 0.98895               | 0.93103               |
| 22        | 0.05408             | 0.40158             | 0.98171               | 0.92552               |
| 23        | 0.00976             | 0.22820             | 0.99689               | 0.94759               |
| 24        | 0.01975             | 0.30644             | 0.99241               | 0.93241               |
| 25        | 0.00903             | 0.15623             | 0.99689               | 0.96000               |
| 26        | 0.04787             | 0.31329             | 0.98516               | 0.93931               |
| 27        | 0.07165             | 0.37022             | 0.97791               | 0.92276               |
| 28        | 0.08311             | 0.41068             | 0.97963               | 0.92414               |
| 29        | 0.02909             | 0.22646             | 0.99137               | 0.96000               |
| 30        | 0.03682             | 0.31647             | 0.98516               | 0.94621               |
| 31        | 0.01979             | 0.30570             | 0.99310               | 0.93241               |
| 32        | 0.00181             | 0.09851             | 0.99965               | 0.97379               |
| 33        | 0.01672             | 0.22619             | 0.99413               | 0.94759               |
| 34        | 0.03590             | 0.16946             | 0.99137               | 0.97103               |
| 35        | 0.05713             | 0.29706             | 0.98723               | 0.95172               |
| 36        | 0.04367             | 0.24826             | 0.99137               | 0.95586               |
| 37        | 0.02791             | 0.21974             | 0.99344               | 0.95310               |
| 38        | 0.01460             | 0.18682             | 0.99586               | 0.96138               |
| 39        | 0.02882             | 0.26528             | 0.99068               | 0.94621               |
| 40        | 0.00957             | 0.18566             | 0.99689               | 0.95862               |
| 41        | 0.04958             | 0.31001             | 0.98757               | 0.94069               |
| 42        | 0.03221             | 0.24372             | 0.99206               | 0.95172               |
| 43        | 0.00855             | 0.16449             | 0.99724               | 0.96414               |
| 44        | 0.01861             | 0.23707             | 0.99344               | 0.95172               |

| Измерение | Потери при обучении | Потери при проверке | Точность при обучении | Точность при проверке |
|-----------|---------------------|---------------------|-----------------------|-----------------------|
| 45        | 0.00563             | 0.17899             | 0.99793               | 0.96138               |
| 46        | 0.26816             | 0.85337             | 0.91854               | 0.85517               |
| 47        | 0.00448             | 0.16608             | 0.99793               | 0.96138               |
| 48        | 0.03231             | 0.23489             | 0.99103               | 0.96000               |
| 49        | 0.13648             | 0.43286             | 0.96790               | 0.92690               |
| 50        | 0.03111             | 0.17226             | 0.99103               | 0.96414               |
| 51        | 0.09066             | 0.54343             | 0.97308               | 0.91034               |
| 52        | 0.07532             | 0.39614             | 0.97549               | 0.93103               |
| 53        | 0.08610             | 0.40154             | 0.98516               | 0.94207               |
| 54        | 0.03756             | 0.29358             | 0.98654               | 0.94207               |
| 55        | 0.08691             | 0.45345             | 0.97135               | 0.92000               |
| 56        | 0.01113             | 0.19603             | 0.99586               | 0.96138               |
| 57        | 0.17891             | 0.54858             | 0.95927               | 0.90897               |
| 58        | 0.00902             | 0.15268             | 0.99724               | 0.97241               |
| 59        | 0.10507             | 0.48165             | 0.97549               | 0.92690               |
| 60        | 0.41455             | 1.07360             | 0.93200               | 0.86483               |
| 61        | 0.00163             | 0.04623             | 0.99931               | 0.98483               |
| 62        | 0.00742             | 0.14093             | 0.99758               | 0.96690               |
| 63        | 0.67702             | 1.26993             | 0.91923               | 0.86897               |
| 64        | 0.07825             | 0.24494             | 0.97791               | 0.95034               |
| 65        | 0.09353             | 0.55347             | 0.97825               | 0.92828               |
| 66        | 0.24339             | 0.76777             | 0.96065               | 0.92828               |
| 67        | 0.24243             | 0.88076             | 0.95858               | 0.90345               |
| 68        | 0.00002             | 0.04626             | 1.00000               | 0.99310               |
| 69        | 0.36695             | 1.33624             | 0.93614               | 0.87310               |
| 70        | 0.11307             | 0.65193             | 0.97791               | 0.92966               |
| 71        | 0.12938             | 0.57708             | 0.97998               | 0.92966               |
| 72        | 0.03989             | 0.32134             | 0.98964               | 0.95172               |
| 73        | 0.11520             | 0.68035             | 0.97722               | 0.92828               |
| 74        | 0.78238             | 1.55256             | 0.91301               | 0.88138               |
| 75        | 0.00452             | 0.11253             | 0.99827               | 0.98207               |
| 76        | 0.08034             | 0.44734             | 0.98309               | 0.94897               |
| 77        | 0.13069             | 0.48065             | 0.97825               | 0.95172               |
| 78        | 0.09219             | 0.42594             | 0.98412               | 0.95724               |
| 79        | 0.80148             | 1.74862             | 0.90576               | 0.86621               |
| 80        | 0.00992             | 0.17134             | 0.99689               | 0.97793               |
| 81        | 0.02524             | 0.25879             | 0.99448               | 0.97655               |
| 82        | 0.00185             | 0.09400             | 0.99965               | 0.98897               |
| 83        | 0.08145             | 0.45166             | 0.98067               | 0.94897               |
| 84        | 0.08229             | 0.42859             | 0.98585               | 0.95586               |

| Измерение | Потери при обучении | Потери при проверке | Точность при обучении | Точность при проверке |
|-----------|---------------------|---------------------|-----------------------|-----------------------|
| 85        | 0.19932             | 0.69337             | 0.97204               | 0.94345               |
| 86        | 0.19849             | 0.65429             | 0.98274               | 0.94759               |
| 87        | 0.11621             | 0.36588             | 0.99103               | 0.97379               |
| 88        | 0.12844             | 0.41497             | 0.98964               | 0.96690               |
| 89        | 0.07164             | 0.30259             | 0.99379               | 0.97241               |
| 90        | 0.02001             | 0.22377             | 0.99655               | 0.97103               |
| 91        | 0.00117             | 0.20849             | 1.00000               | 0.96828               |
| 92        | 0.00490             | 0.29586             | 0.99793               | 0.96414               |
| 93        | 0.00188             | 0.23267             | 0.99931               | 0.97241               |
| 94        | 0.00000             | 0.03840             | 1.00000               | 0.99448               |
| 95        | 0.00047             | 0.10282             | 1.00000               | 0.98345               |
| 96        | 0.00380             | 0.18238             | 0.99724               | 0.96552               |
| 97        | 0.00898             | 0.26972             | 0.99655               | 0.95724               |
| 98        | 0.00811             | 0.24056             | 0.99620               | 0.95724               |
| 99        | 0.02737             | 0.33180             | 0.99241               | 0.95448               |

Для обеспечения большей наглядности на рисунке 5.2 представлены графики потерь при обучении и при проверке для первых 70 измерений, а на рисунке 5.3 представлены графики точности при обучении и при проверке для последних 30 измерений.

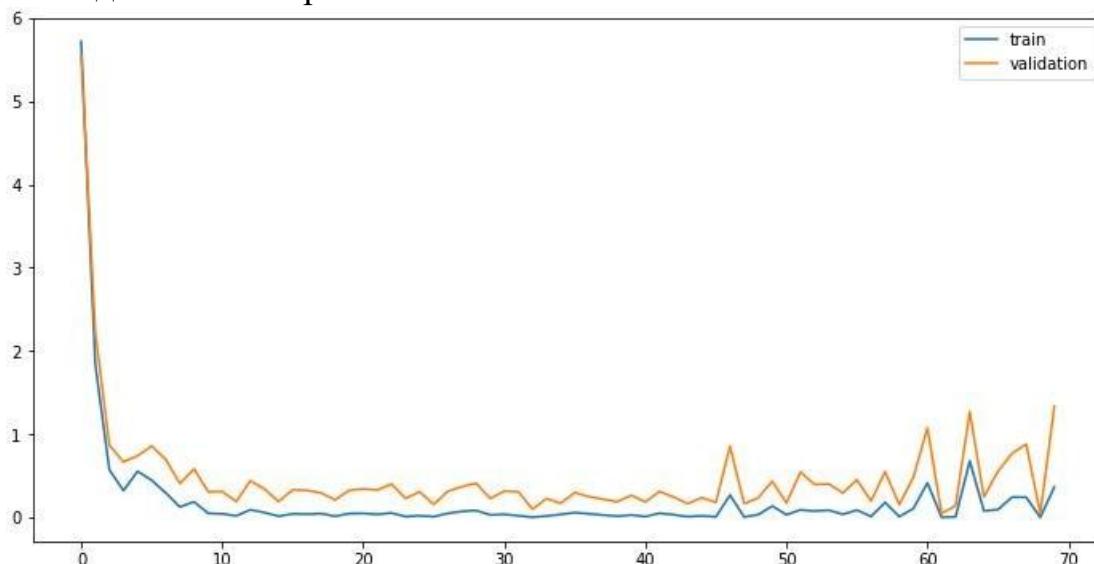


Рисунок 5.2 – Графики потерь при обучении и при проверке работы нейронной сети

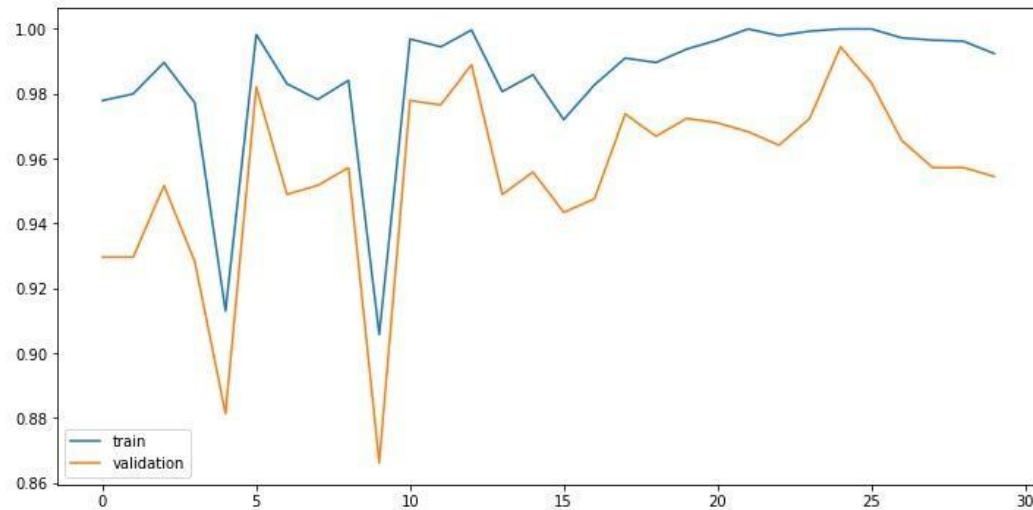


Рисунок 5.3 – Графики точности при обучении и при проверке работы нейронной сети

Как видно из полученных графиков, качественные показатели точности сходятся к отметке 0.95 (95%) в среднем при схождении уровня СКО к отметке 0.33, соответствующей примерно 5% потерь от выборки, что является достаточно высоким показателем.

По результатам проведённого эксперимента для одного измерения непосредственно время обучения нейронной сети в среднем составило 2 секунды, а суммарное время обработки данных для формирования модели свёрточной сети, которая впоследствии будет использоваться подсистемой оценки состояния механизма посредством обученной нейронной сети (вместе с предварительной обработкой сигналов, их нормировкой и т.д.) составило в среднем 3,32 секунды.

Общее время, затраченное на проведение эксперимента, составило 56 минут и 28 секунд.

## **Заключение**

В результате выполнения выпускной квалификационной работы была спроектирована и программно реализована система для обнаружения дефекта механизма по вибрационному сигналу. Для выполнения работы был выбран язык программирования Python, за основу разработанного алгоритма был взят метод из теории активного восприятия.

Вследствие этого, данная программная система предоставляет высокоэффективную и многофункциональную разработку алгоритма для определения неисправностей различных механизмов по вибрационному сигналу. В свою очередь, тестированием разработанной системы была доказана эффективная и функциональная работоспособность, возможность применения данной системы для сформулированной задачи в выпускной работе и корректный выбор действенных средств разработки.

Также, разработанная программная система имеет высокую практическую ценность, так как решает довольно актуальную и общую задачу по классификации вибрационного сигнала. Данная система, равным образом, применима с различным набором данных и легко интегрируется для определения неисправностей у множества механизмов.

Дальнейшая работа на основание разработанного прототипа включает в себя несколько путей развития. Первым шагом является интегрирование системы в узкую область анализа, то есть разработать интерфейс для определения дефектов подшипников качения с применением мультиклассификации, а именно обнаружения неисправностей внутри и снаружи детали. Вторым шагом предстоит произвести разбор подходов к построению нейронной сети, произведя анализ, выбрать самый подходящий для заданной области. Третий шаг – это изменение исходного кода, целью которого является оптимизация реализуемого алгоритма, тем самым улучшая производительность системы в целом.

|      |      |          |         |      |  |      |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      | ВКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | Лист |
| Иzm. | Лист | № докум. | Подпись | Дата |  | 42   |

## **Список литературы**

1. Колобов А.Б. Вибродиагностика: теория и практика: учебное пособие / А.Б. Колобов. - М.: Инфра-Инженерия, 2019. - 252 с.
2. Генкин М.Д., Соколова А.Г. Виброакустическая диагностика машин и механизмов / М.Д. Генкин, А.Г. Соколова. - М.: Машиностроение, 1987. - 288 с.
3. H. A. Toliyat, S. Nandi, S. Choi, and H. Meshgin-Kelk, Electric Machines: Modeling, Condition Monitoring, and Fault Diagnosis, CRC Press, 2012.
4. R. B. Randall, Vibration-Based Condition Monitoring: Industrial, Aerospace and Automotive Applications, John Wiley & Sons, Chichester, UK, 2011.
5. H. Henao, G.-A. Capolino, M. Fernandez-Cabanas et al., "Trends in fault diagnosis for electrical machines: a review of diagnostic techniques," IEEE Industrial Electronics Magazine, vol. 8, no. 2, pp. 31–42, 2014.
6. P. D. McFadden and J. D. Smith, "Model for the vibration produced by a single point defect in a rolling element bearing," Journal of Sound and Vibration, vol. 96, no. 1, pp. 69–82, 1984.
7. R. R. Schoen, T. G. Habetler, F. Kamran, and R. G. Bartheld, "Motor bearing damage detection using stator current monitoring," IEEE Transactions on Industry Applications, vol. 31, no. 6, pp. 1274–1279, 1995.
8. F. Filippetti, A. Bellini, and G.-A. Capolino, "Condition monitoring and diagnosis of rotor faults in induction machines: State of art and future perspectives," in Proceedings of the 1st Workshop on Electrical Machines Design, Control and Diagnosis, WEMDCD 2013, pp. 196–209, Paris, France, March 2013.
9. R. Di Stefano, S. Meo, and M. Scarano, "Induction motor faults diagnostic via artificial neural network (ANN)," in Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE' 94), pp. 220–225, IEEE, Santiago, Chile, May 1994.
10. L. Eren, A. Karahoca, and M. J. Devaney, "Neural network based motor bearing fault detection," in Proceedings of the 21st IEEE Instrumentation and Measurement Technology Conference, pp. 1657–1660, May 2004.
11. C. T. Kowalski and T. Orlowska-Kowalska, "Neural networks application for induction motor faults diagnosis," Mathematics and Computers in Simulation, vol. 63, no. 3–5, pp. 435–448, 2003.
12. W.-Y. Chen, J.-X. Xu, and S. K. Panda, "Application of artificial intelligence techniques to the study of machine signatures," in Proceedings of the 20th International Conference on Electrical Machines (ICEM '12), pp. 2390–2396, Marseille, France, September 2012.
13. K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," Biolog. Cybernetics, vol. 36, pp. 193–202, 1980.

|      |      |          |         |      |  |      |
|------|------|----------|---------|------|--|------|
| Изм. | Лист | № докум. | Подпись | Дата | BKR-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | Лист |
|      |      |          |         |      |  | 43   |

14. J. Pan, Y. Zi, J. Chen, Z. Zhou and B. Wang, “LiftingNet: A novel deep learning network with layerwise feature learning from noisy mechanical data for fault classification,” *IEEE Trans. Ind. Electron.*, vol. 65, no. 6, pp. 4973–4982, June 2018.
15. Y. Chen, G. Peng, C. Xie, W. Zhang, C. Li, and S. Liu, “ACDIN: Bridging the gap between artificial and real bearing damages for bearing fault diagnosis,” *Neurocomput.*, vol. 294, pp. 61–71, 2018.
16. B. Li, M.-Y. Chow, Y. Tipsuwan, and J. C. Hung, “Neural-network-based motor rolling bearing fault diagnosis,” *IEEE Transactions on Industrial Electronics*, vol. 47, no. 5, pp. 1060–1069, 2000.
17. S. Singh, C. Q. Howard, and C. Hansen, “Convolutional neural network based fault detection for rotating machinery,” *J. Sound Vib.*, vol. 377, pp. 331–345, 2016.
18. X. Guo, L. Chen, and C. Shen, “Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis,” *Meas.*, vol. 93, pp. 490–502, 2016.
19. R. Liu, G. Meng, B. Yang, C. Sun and X. Chen, “Dislocated time series convolutional neural architecture: An intelligent fault diagnosis approach for electric machine,” *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1310–1320, June 2017.
20. C. Lu, Z. Wang, and Bo Zhou, “Intelligent fault diagnosis of rolling bearing using hierarchical convolutional network based health state classification,” *Adv. Eng. Informat.*, vol. 32, pp. 139–157, 2017.
21. M. Xia, T. Li, L. Xu, L. Liu and C. W. de Silva, “Fault diagnosis for rotating machinery using multiple sensors and convolutional neural networks,” *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 1, pp. 101–110, Feb. 2018.
22. L. Wen, X. Li, L. Gao and Y. Zhang, “A new convolutional neural network-based data-driven fault diagnosis method,” *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5990–5998, July 2018.
23. W. Zhang, F. Zhang, W. Chen, Y. Jiang and D. Song, “Fault state recognition of rolling bearing based fully convolutional network,” *Comput. in Sci. & Eng.*, vol. PP, no. PP, pp. PP–PP, 2018.
24. Z. Zhuang and Q. Wei, “Intelligent fault diagnosis of rolling bearing using one-dimensional multi-scale deep convolutional neural network based health state classification,” in Proc. 2018 IEEE 15th Int. Conf. Netw., Sens. & Control (ICNSC), Zhuhai, China, 2018, pp. 1–6.
25. W. Zhang, C. Li, G. Peng, Y. Chen, Z. Zhang, “A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load,” *Mech. Syst. Signal Process.*, vol. 100, pp. 439–458, 2018.
26. S. Li, G. Liu, X. Tang, J. Lu, and J. Hu, “An ensemble deep convolutional neural network model with improved D-S evidence fusion for bearing fault diagnosis,” *Sensors*, vol. 17, art. no. 1729, Jul. 2017.

|      |      |          |         |      |  |      |
|------|------|----------|---------|------|--|------|
| Изм. | Лист | № докум. | Подпись | Дата | БКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | Лист |
|      |      |          |         |      |  | 44   |

27. S. Guo, T. Yang, W. Gao, C. Zhang, and Y. Zhang, “An intelligent fault diagnosis method for bearings with variable rotating speed based on Pythagorean spatial pyramid pooling CNN,” Sensors, vol. 18, art. no. 3857, Nov. 2018.
28. W. Qian, S. Li, J. Wang, Z. An, and X. Jiang, “An intelligent fault diagnosis framework for raw vibration signals: adaptive overlapping convolutional neural network,” Meas. Sci. Technol., vol. 29, no. 6, art. no 095009, Aug. 2018.
29. W. Abed, S. Sharma, R. Sutton, and A. Motwani, “A robust bearing fault detection and diagnosis technique for brushless DC motors under non-stationary operating conditions,” J. Control Autom. Electr. Syst., vol. 26, no. 3, pp. 241–254, Jun. 2015.
30. L. Guo, N. Li, F. Jia, Y. Lei, and J. Lin, “A recurrent neural network based health indicator for remaining useful life prediction of bearings,” Neurocomput., vol. 240, pp. 98–109, 2017.
31. H. Pan, X. He, S. Tang, F. Meng, “An improved bearing fault diagnosis method using one-dimensional CNN and LSTM,” J. Mech. Eng., vol. 64, no. 7–8, pp. 443–452, 2018.
32. J. Schmidhuber, “Deep learning in neural networks: An overview,” Neural Netw., vol. 61, pp. 85–117, Jan. 2015.
33. D. H. Ballard, “Modular learning in neural networks,” in Proc. AAAI’87 National Conf. AI, Seattle, WA, 1987, pp. 279–284.
34. H. Shao, H. Jiang, H. Zhao, and F. Wang, “A novel deep autoencoder feature learning method for rotating machinery fault diagnosis,” Mech. Syst. Signal Process., vol. 95, pp. 187–204, 2017.
35. F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu, “Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data,” Mech. Syst. Signal Process., vol. 100, pp. 743–765, 2016.
36. W. Mao, J. He, Y. Li, and Y. Yan, “Bearing fault diagnosis with autoencoder extreme learning machine: A comparative study,” Proc. Inst. Mech. Eng. C, J. Mech. Eng. Sci., vol. 231, no. 8, pp. 1560–1578, 2016.
37. C. Lu, Z.-Y. Wang, W.-L. Qin, J. Ma, “Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification,” Signal Process., vol. 130, pp. 377–388, 2017.
38. X. Guo, C. Shen, and L. Chen, “Deep fault recognizer: An integrated model to denoise and extract features for fault diagnosis in rotating machinery,” Appl. Sci., 2017, vol. 7, no. 41, pp. 1–17.
39. F. Wang, B. Dun, G. Deng, H. Li and Q. Han, “A deep neural network based on kernel function and auto-encoder for bearing fault diagnosis,” in Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC), Houston, TX, USA, 2018, pp. 1–6.
40. H. Shao, H. Jiang, Y. Lin, and X. Li, “A novel method for intelligent fault diagnosis of rolling bearings using ensemble deep autoencoders,” Knowl.-Based Syst., vol. 119, pp. 200–220, 2018.

|      |      |          |         |      |  |            |
|------|------|----------|---------|------|--|------------|
| Изм. | Лист | № докум. | Подпись | Дата | BKP-НГТУ-09.03.01-(16-Б-2)-007-2020 (ПЗ) | Лист<br>45 |
|------|------|----------|---------|------|--|------------|

41. H. Shao, H. Jiang, X. Li, S. Wu, “Intelligent fault diagnosis of rolling bearing using deep wavelet auto-encoder with extreme learning machine,” *Knowl.-Based Syst.*, vol. 140, pp. 1–14, 2018.
42. J. Sun, C. Yan and J. Wen, “Intelligent bearing fault diagnosis method combining compressed data acquisition and deep learning,” *IEEE Trans. Instrum. Meas.*, vol. 67, no. 1, pp. 185–195, Jan. 2018.
43. F. Jia, Y. Lei, L. Guo, J. Lin, and S. Xing, “A neural network constructed by deep learning technique and its application to intelligent fault diagnosis of machines,” *Neurocomput.*, vol. 272, pp. 619–628, 2018.
44. W. Mao, S. Tian, X. Liang, and J. He, “Online bearing fault diagnosis using support vector machine and stacked auto-encoder,” in Proc. IEEE Int. Conf. Prognost. Health Manag. (ICPHM), Seattle, WA, 2018, pp. 1–7.
45. F. Xu, W. P. Tse, and Y.-L. Tse, “Roller bearing fault diagnosis using stacked denoising autoencoder in deep learning and Gath-Geva clustering algorithm without principal component analysis and data label,” *Appl. Soft Comput. J.*, vol. PP, pp. PP–PP, 2018.
46. C. Li, W. Zhang, G. Peng, and S. Liu, “Bearing fault diagnosis using fully-connected winner-take-all autoencoder,” *IEEE Access*, vol. 6, pp. 6103–6115, 2018.
47. S. Kiranyaz, T. Ince, and M. Gabbouj, “Real-time patient-specific ECG classification by 1-D convolutional neural networks,” *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 3, pp. 664–675, 2016.
48. T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, “Real-time motor fault detection by 1-D convolutional neural networks,” *IEEE Transactions on Industrial Electronics*, vol. 63, no. 11, pp. 7067–7075, 2016.

| Изм. | Лист | № докум. | Подпись | Дата | Лист |
|------|------|----------|---------|------|------|
|      |      |          |         |      | 46   |

## Приложение А (обязательное)

Исходный код программы, реализующей подсистему обучения нейронной сети

### A.1 Текст модуля nn\_model.py

```
import torch
from torch import nn
from torch.nn import functional as F
# 2 Layers CNN
class CNN_1D_2L(nn.Module):
    def __init__(self, n_in):
        super().__init__()
        self.n_in = n_in
        self.layer1 = nn.Sequential(
            nn.Conv1d(1, 64, (9,), stride=1, padding=4),
            nn.BatchNorm1d(64),
            nn.ReLU(),
            nn.Dropout(p=0.5),
            nn.MaxPool1d(2,stride=2)
        )

        self.layer2 = nn.Sequential(
            nn.Conv1d(64, 128, (5,), stride=1, padding=2),
            nn.BatchNorm1d(128),
            nn.ReLU(),
            nn.Dropout(p=0.5),
            nn.AvgPool1d(2,stride=2)
        )

        self.linear1 = nn.Linear(self.n_in*128 //4, 4)

    def forward(self, x):
        x = x.view(-1, 1, self.n_in)
        x = self.layer1(x)
        x = self.layer2(x)
        x = x.view(-1, self.n_in*128//4)
        return self.linear1(x)
```

| Изм. | Лист | № докум. | Подпись | Дата | БКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | Лист |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      |  | 47   |

```

# 3 Layers CNN
class CNN_1D_3L(nn.Module):
    def __init__(self, n_in):
        super().__init__()
        self.n_in = n_in
        self.layer1 = nn.Sequential(
            nn.Conv1d(1, 64, (9,), stride=1, padding=4),
            nn.BatchNorm1d(64),
            nn.ReLU(),
            nn.Dropout(p=0.5),
            nn.MaxPool1d(2,stride=2)
        )

        self.layer2 = nn.Sequential(
            nn.Conv1d(64, 64, (5,), stride=1, padding=2),
            nn.BatchNorm1d(64),
            nn.ReLU(),
            nn.Dropout(p=0.5),
            nn.MaxPool1d(2,stride=2)
        )

        self.layer3 = nn.Sequential(
            nn.Conv1d(64, 128, (5,), stride=1, padding=2),
            nn.BatchNorm1d(128),
            nn.ReLU(),
            nn.Dropout(p=0.5),
            nn.MaxPool1d(2,stride=2)
        )

        self.linear1 = nn.Linear(self.n_in*128 //8, 4)

    def forward(self, x):
        x = x.view(-1, 1, self.n_in)
        x = self.layer1(x)
        x = self.layer2(x)
        x = self.layer3(x)
        x = x.view(-1, self.n_in*128//8)
        return self.linear1(x)

```

| Изм. | Лист | № докум. | Подпись | Дата | БКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | Лист |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      |  | 48   |

## A.2 Текст модуля train\_helper.py

```
import torch
from torch import nn
from torch.nn import functional as F
from torch import Tensor
from torch.utils.data import TensorDataset, DataLoader
from torch import optim
from torch.nn.modules.loss import CrossEntropyLoss

from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd
from one_cycle import OneCycle, update_lr, update_mom
```

# Functions for training

```
def get_dataloader(train_ds, valid_ds, bs):
```

```
    """
```

Get dataloaders of the training and validation set.

Parameter:

train\_ds: Dataset

Training set

valid\_ds: Dataset

Validation set

bs: Int

Batch size

Return:

(train\_dl, valid\_dl): Tuple of DataLoader

Dataloaders of training and validation set.

```
"""
```

```
return (
```

DataLoader(train\_ds, batch\_size=bs, shuffle=True),

DataLoader(valid\_ds, batch\_size=bs \* 2),

```
)
```

```
def loss_batch(model, loss_func, xb, yb, opt=None):
```

```
    """
```

Parameter:

model: Module

Your neural network model

loss\_func: Loss

|      |      |          |         |      |
|------|------|----------|---------|------|
| Изм. | Лист | № докум. | Подпись | Дата |
|------|------|----------|---------|------|

```

        Loss function, e.g. CrossEntropyLoss()
xb: Tensor
    One batch of input x
yb: Tensor
    One batch of true label y
opt: Optimizer
    Optimizer, e.g. SGD()

Return:
loss.item(): Python number
    Loss of the current batch
len(xb): Int
    Number of examples of the current batch
pred: ndarray
    Predictions (class with highest probability) of the minibatch
    input xb
"""
with torch.no_grad():
    out = model(xb)
    loss = loss_func(out, yb)
    pred = torch.argmax(out, dim=1).cpu().numpy()

    if opt is not None:
        loss.backward()
        opt.step()
        opt.zero_grad()

return loss.item(), len(xb), pred

def fit(epochs, model, loss_func, opt, train_dl, valid_dl, one_cycle=None,
train_metric=False):
"""
Train the NN model and return the model at the final step.
Lists of the training and validation losses at each epochs are also
returned.

Parameter:
epochs: int
    Number of epochs to run.
model: Module
    Your neural network model
loss_func: Loss
    Loss function, e.g. CrossEntropyLoss()

```

|      |      |          |         |      |  |      |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      | БКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |  | 50   |

```

opt: Optimizer
    Optimizer, e.g. SGD()
train_dl: DataLoader
    Dataloader of the training set.
valid_dl: DataLoader
    Dataloader of the validation set.
one_cycle: OneCycle
    See one_cycle.py. Object to calculate and update the learning
    rates and momentums at the end of each training iteration (not
    epoch) based on the one cycle policy.
train_metric: Bool
    Default is False. If False, the train loss and accuracy will be
    set to 0.
    If True, the loss and accuracy of the train set will also be
    computed.

Return:
model: Module
    Trained model.
metrics: DataFrame
    DataFrame which contains the train and validation loss at each
    epoch.
"""
print(
    'EPOCH', '\t',
    'Train Loss', '\t',
    'Val Loss', '\t',
    'Train Acc', '\t',
    'Val Acc', '\t')
# Initialize dic to store metrics for each epoch.
metrics_dic = {}
metrics_dic['train_loss'] = []
metrics_dic['train_accuracy'] = []
metrics_dic['val_loss'] = []
metrics_dic['val_accuracy'] = []
device = torch.device("cuda") if torch.cuda.is_available() else
torch.device("cpu")

for epoch in range(epochs):
    # Train
    model.train()
    train_loss = 0.0
    train_accuracy = 0.0

```

|      |      |          |         |      |
|------|------|----------|---------|------|
| Изм. | Лист | № докум. | Подпись | Дата |
|------|------|----------|---------|------|

```

num_examples = 0
for xb, yb in train_dl:
    xb, yb = xb.to(device), yb.to(device)
    loss, batch_size, pred = loss_batch(model, loss_func, xb, yb, opt)
    if train_metric == False:
        train_loss += loss
        num_examples += batch_size

    if one_cycle:
        lr, mom = one_cycle.calc()
        update_lr(opt, lr)
        update_mom(opt, mom)

# Validate
model.eval()
with torch.no_grad():
    val_loss, val_accuracy, _ = validate(model, valid_dl, loss_func)
if train_metric:
    train_loss, train_accuracy, _ = validate(model, train_dl, loss_func)
else:
    train_loss = train_loss / num_examples

metrics_dic['val_loss'].append(val_loss)
metrics_dic['val_accuracy'].append(val_accuracy)
metrics_dic['train_loss'].append(train_loss)
metrics_dic['train_accuracy'].append(train_accuracy)

print(
    f'{epoch} \t',
    f'{train_loss:.05f}', '\t',
    f'{val_loss:.05f}', '\t',
    f'{train_accuracy:.05f}', '\t'
    f'{val_accuracy:.05f}', '\t')

metrics = pd.DataFrame.from_dict(metrics_dic)

return model, metrics

def validate(model, dl, loss_func):
    total_loss = 0.0
    total_size = 0
    predictions = []
    y_true = []

```

| Изм. | Лист | № докум. | Подпись | Дата | Лист  |
|------|------|----------|---------|------|---|
|      |      |          |         |      | БКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) 52 |

```

device = torch.device("cuda") if torch.cuda.is_available() else
torch.device("cpu")
for xb, yb in dl:
    xb, yb = xb.to(device), yb.to(device)
    loss, batch_size, pred = loss_batch(model, loss_func, xb, yb)
    total_loss += loss*batch_size
    total_size += batch_size
    predictions.append(pred)
    y_true.append(yb.cpu().numpy())
mean_loss = total_loss / total_size
predictions = np.concatenate(predictions, axis=0)
y_true = np.concatenate(y_true, axis=0)
accuracy = np.mean((predictions == y_true))
return mean_loss, accuracy, (y_true, predictions)

```

|      |      |          |         |      |   |      |
|------|------|----------|---------|------|---|------|
|      |      |          |         |      | <i>BKP-НГТУ-09.03.01-(16-Б-2)-007-2020 (ПЗ)</i> | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |   | 53   |

### A.3 Текст модуля helper.py

```
# Helper functions to read and preprocess data files from Matlab format
# Data science libraries
import scipy
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Others
from pathlib import Path
from tqdm.auto import tqdm
import requests

def matfile_to_dic(folder_path):
    """
```

Read all the matlab files of the CWRU Bearing Dataset and return a dictionary. The key of each item is the filename and the value is the data of one matlab file, which also has key value pairs.

Parameter:

folder\_path:

Path (Path object) of the folder which contains the matlab files.

Return:

output\_dic:

Dictionary which contains data of all files in the folder\_path.

""

output\_dic = {}

for \_, filepath in enumerate(folder\_path.glob('\*.\*mat')):

# strip the folder path and get the filename only.

key\_name = str(filepath).split('\\')[-1]

output\_dic[key\_name] = scipy.io.loadmat(filepath)

return output\_dic

```
def remove_dic_items(dic):
    """
```

Remove redundant data in the dictionary returned by matfile\_to\_dic inplace.

""

# For each file in the dictionary, delete the redundant key-value pairs

for \_, values in dic.items():

del values['\_\_header\_\_']

del values['\_\_version\_\_']

del values['\_\_globals\_\_']

|      |      |          |         |      |
|------|------|----------|---------|------|
| Изм. | Лист | № докум. | Подпись | Дата |
|------|------|----------|---------|------|

```

def rename_keys(dic):
    """
    Rename some keys so that they can be loaded into a
    DataFrame with consistent column names
    """
    # For each file in the dictionary
    for _,v1 in dic.items():
        # For each key-value pair, rename the following keys
        for k2,_ in list(v1.items()):
            if 'DE_time' in k2:
                v1['DE_time'] = v1.pop(k2)
            elif 'BA_time' in k2:
                v1['BA_time'] = v1.pop(k2)
            elif 'FE_time' in k2:
                v1['FE_time'] = v1.pop(k2)
            elif 'RPM' in k2:
                v1['RPM'] = v1.pop(k2)

```

```

def label(filename):
    """
    Function to create label for each signal based on the filename. Apply this
    to the "filename" column of the DataFrame.
    Usage:
    df['label'] = df['filename'].apply(label)
    """

```

```

if 'B' in filename:
    return 'B'
elif 'IR' in filename:
    return 'IR'
elif 'OR' in filename:
    return 'OR'
elif 'Normal' in filename:
    return 'N'

```

```

def matfile_to_df(folder_path):
    """
    Read all the matlab files in the folder, preprocess, and return a DataFrame
    Parameter:

```

| Изм. | Лист | № докум. | Подпись | Дата | Лист  |
|------|------|----------|---------|------|---|
|      |      |          |         |      | БКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) 55 |

```

folder_path:
    Path (Path object) of the folder which contains the matlab files.

Return:
    DataFrame with preprocessed data
"""

dic = matfile_to_dic(folder_path)
remove_dic_items(dic)
rename_keys(dic)
df = pd.DataFrame.from_dict(dic).T
df = df.reset_index().rename(mapper={'index':'filename'},axis=1)
df['label'] = df['filename'].apply(label)
return df.drop(['BA_time','FE_time', 'RPM', 'ans'], axis=1, errors='ignore')

```

def divide\_signal(df, segment\_length):

""

This function divide the signal into segments, each with a specific number of points as defined by segment\_length. Each segment will be added as an example (a row) in the returned DataFrame. Thus it increases the number of training examples. The remaining points which are less than segment\_length are discarded.

Parameter:

df:

DataFrame returned by matfile\_to\_df()

segment\_length:

Number of points per segment.

Return:

DataFrame with segmented signals and their corresponding filename and label

"""

dic = {}

idx = 0

for i in range(df.shape[0]):

n\_sample\_points = len(df.iloc[i,1])

n\_segments = n\_sample\_points // segment\_length

for segment in range(n\_segments):

dic[idx] = {

'signal': df.iloc[i,1][segment\_length \* segment:segment\_length \*

(segment+1)],

'label': df.iloc[i,2],

'filename' : df.iloc[i,0]

}

| Изм. | Лист | № докум. | Подпись | Дата | Лист                                     |
|------|------|----------|---------|------|--|
|      |      |          |         |      | BKP-НГТУ-09.03.01-(16-B-2)-007-2020 (ПЗ) |

```

    idx += 1
    df_tmp = pd.DataFrame.from_dict(dic, orient='index')
    df_output = pd.concat(
        [df_tmp[['label', 'filename']],
         pd.DataFrame(np.hstack(df_tmp["signal"].values).T)
        ],
        axis=1 )
    return df_output

```

```
def normalize_signal(df):
```

```
""
```

Normalize the signals in the DataFrame returned by matfile\_to\_df() by subtracting

the mean and dividing by the standard deviation.

```
""
```

```
mean = df['DE_time'].apply(np.mean)
```

```
std = df['DE_time'].apply(np.std)
```

```
df['DE_time'] = (df['DE_time'] - mean) / std
```

```
def get_df_all(data_path, segment_length=512, normalize=False):
```

```
""
```

Load, preprocess and return a DataFrame which contains all signals data and labels and is ready to be used for model training.

Parameter:

normal\_path:

Path of the folder which contains matlab files of normal bearings

DE\_path:

Path of the folder which contains matlab files of DE faulty bearings

segment\_length:

Number of points per segment. See divide\_signal() function

normalize:

Boolean to perform normalization to the signal data

Return:

df\_all:

DataFrame which is ready to be used for model training.

```
""
```

```
df = matfile_to_df(data_path)
```

```
if normalize:
```

```
    normalize_signal(df)
```

| Изм. | Лист | № докум. | Подпись | Дата | BKP-НГТУ-09.03.01-(16-Б-2)-007-2020 (ПЗ) | Лист |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      |  | 57   |

```

df_processed = divide_signal(df, segment_length)

map_label = {'N':0, 'B':1, 'IR':2, 'OR':3}
df_processed['label'] = df_processed['label'].map(map_label)
return df_processed

def download(url:str, dest_dir:Path, save_name:str, suffix=None) -> Path:
    assert isinstance(dest_dir, Path), "dest_dir must be a Path object"
    if not dest_dir.exists():
        dest_dir.mkdir()
    if save_name == None: filename = url.split('/')[-1]
    else: filename = save_name+suffix
    file_path = dest_dir / filename
    if not file_path.exists():
        print(f"Downloading {file_path}")
        with open(f'{file_path}', 'wb') as f:
            response = requests.get(url, stream=True)
            total = int(response.headers.get('content-length'))
            with tqdm(total=total, unit='B', unit_scale=True, desc=filename) as pbar:
                for data in response.iter_content(chunk_size=1024*1024):
                    f.write(data)
                    pbar.update(1024*1024)
    else:
        return file_path
    return file_path

```

| Изм. | Лист | № докум. | Подпись | Дата | Лист |
|------|------|----------|---------|------|------|
|      |      |          |         |      | 58   |

## Приложение Б (обязательное)

Исходный код программы, реализующей подсистему оценки состояния  
механизма посредством обученной нейронной сети

### Б.1 Текст модуля model.py

```
import torch
from torch import nn
from torch.nn import functional as F
from torch import Tensor
from pathlib import Path
save_model_path = Path("./Model")
import scipy.io
import nn_model

import numpy as np
import pandas as pd

class _Model:
    """
    Base class for Model which stores read signal files and stores states.
    Inherit this class, extend the init method to load your serialized ml model
    and implement the predict method.
    """
    def __init__(self):
        self.data = {}
        self.data['filenames'] = []
        self.data['filepaths'] = []
        self.data['signals'] = []
        self.data['prediction'] = []

        self._pred2class_map = {
            0:'N',
            1:'B',
            2:'IR',
            3:'OR',
        }

    def update_prediction(self, file_index):
        """
        Get prediction of the file_index coresponding signal and update the data
        prediction state of this signal.
        """
        prediction = self.predict(file_index)
```

| Изм. | Лист | № докум. | Подпись | Дата | Лист                                     | 59 |
|------|------|----------|---------|------|--|----|
|      |      |          |         |      | ВКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) |    |

```

self.data['prediction'][file_index] = self._pred2class_map[prediction]

def predict(self, file_index):
    """
    Implement this method in subclass to return the prediction by your ml
    model as Python object, e.g. int, str.
    """
    print("Not implemented")

def read_files(self, filepaths):
    """
    Read the .mat signal files and update the states of data

    Parameter:
        filepaths: List/tuple of .mat signal files' paths
    """
    for filepath in filepaths:
        file_name = str(filepath).split('/')[-1]
        self.data['filenames'].append(file_name)
        self.data['filepaths'].append(Path(filepath))
        self.data['signals'].append(mat_to_ndarray(Path(filepath)))
        self.data['prediction'].append('None')
    return self.data['filenames']

```

```

class CNN_1D(_Model):
    def __init__(self):
        super().__init__()
        # Instantiate the CNN_1D_2L with input size=500 and load the trained
        # model parameters
        self.pred_model = nn_model.CNN_1D_2L(500)
        self.pred_model.load_state_dict(
            torch.load(save_model_path / 'model.pth', map_location='cpu')
        )
        self.pred_model.eval()

    def predict(self, file_index):
        """
        Split the signal data into segments of 500 points (same preprocessing
        during training). Then predict the class of all segments and return the
        most frequent class (mode) as output.
        """
        x = self.data['signals'][file_index]

```

| Изм. | Лист | № докум. | Подпись | Дата | BKP-НГТУ-09.03.01-(16-Б-2)-007-2020 (ПЗ) | Лист |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      |  | 60   |

```

x = preprocess_signal(x, 500)
x = torch.tensor(x, dtype=torch.float32)
out = self.pred_model(x)
pred = torch.argmax(out, dim=1)
mode, _ = torch.mode(pred, dim=0)
return mode.item()

def rename_matfile_keys(dic):
    """
    Rename some keys so that they can be loaded into a DataFrame with
    consistent column names
    """
    # For each key-value pair, rename the following keys
    for k2, _ in list(dic.items()):
        if 'DE_time' in k2:
            dic['DE_time'] = dic.pop(k2)
        elif 'BA_time' in k2:
            dic['BA_time'] = dic.pop(k2)
        elif 'FE_time' in k2:
            dic['FE_time'] = dic.pop(k2)
        elif 'RPM' in k2:
            dic['RPM'] = dic.pop(k2)

def mat_to_ndarray(matfile_path):
    """
    Read the .mat signal file and return the drive end signal data (DE_time).
    """
    matfile_dic = scipy.io.loadmat(matfile_path)
    rename_matfile_keys(matfile_dic)
    return matfile_dic['DE_time']

def preprocess_signal(array, segment_length):
    """
    Preprocess the raw signal data by normalizing and divide the signal into
    smaller segments with size of segment_length. Segment length must be the
    same as the value used in training.
    """

    Parameter:
        array: array
        segment_length: int
    """
    array = normalize_signal_array(array)

```

|      |      |          |         |      |  |      |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      | БКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |  | 61   |

```

    return divide_signal_array(array, segment_length)

def normalize_signal_array(array):
    """ Normalize the signal array"""
    mean = np.mean(array)
    std = np.std(array)
    return (array - mean) / std

def divide_signal_array(array, segment_length):
    """

```

This function divide the signal into segments, each with a specific number of points as defined by segment\_length. Each segment will be added as an example (a row) in the returned DataFrame. Thus it increases the number of training examples. The remaining points which are less than segment\_length are discarded.

Parameter:

array:

Numpy array which contains the signal sample points

segment\_length:

Number of points per segment.

Return:

DataFrame with segmented signals and their corresponding filename

""

dic = {}

idx = 0

n\_sample\_points = len(array)

n\_segments = n\_sample\_points // segment\_length

for segment in range(n\_segments):

dic[idx] = {

'signal': array[segment\_length \* segment:segment\_length \* (segment+1)],

}

idx += 1

df\_tmp = pd.DataFrame.from\_dict(dic, orient='index')

return np.hstack(df\_tmp["signal"].values).T

|      |      |          |         |      |  |      |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      | БКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |  | 62   |

## Б.2 Текст модуля controller.py

```
import gui_core.model as model
from tkinter import *

Model = model.CNN_1D

class StartPageController:
    def __init__(self, page):
        self.page = page
        self.model = Model()
        self.page.button_open.config(command=self.select_files)
        self.page.button_del_all.config(command=self.delete_list_all)
        self.page.button_del_selected.config(command=self.delete_list_selected)
        self.page.button_pred.config(command=self.get_predictions)
        self.page.draw_button.config(command=self.plot_something)

    def select_files(self):
        paths = filedialog.askopenfilenames()
        filenames = self.model.read_files(paths)
        self.page.labelframe.lb_filenames.delete(0,END)
        self.page.labelframe.lb_predictions.delete(0,END)
        for i, (filename, pred) in enumerate(zip(self.model.data['filenames'],
                                                self.model.data['prediction'])):
            self.page.labelframe.lb_filenames.insert(i, filename)
            self.page.labelframe.lb_predictions.insert(i, pred)

    def delete_list_all(self):
        for k,v in self.model.data.items():
            self.model.data[k] = []
        self.page.labelframe.lb_filenames.delete(0, END)
        self.page.labelframe.lb_predictions.delete(0, END)

    def delete_list_selected(self):
        # Each time listbox.delete method is called, the index of the listbox is
        # reset. Hence the index of the selected item needs to be corrected by
        # subtracting the count of deleted items.
        del_indices = self.get_list_selection()
        count = 0
        for del_idx in del_indices:
            for k,v in self.model.data.items():
                v.pop(del_idx - count)

            self.page.labelframe.lb_filenames.delete(del_idx - count)
```

| Изм. | Лист | № докум. | Подпись | Дата | БКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | Лист |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      |  | 63   |

```

        self.page.labelframe.lb_predictions.delete(del_idx - count)
        count += 1

    def get_list_selection(self):
        return self.page.labelframe.lb_filenames.curselection()

    def plot_something(self):
        try:
            idx = self.get_list_selection()
            idx = idx[0]
            arr = self.model.data['signals'][idx]
            self.page.plotframe.a.clear()
            self.page.plotframe.a.plot(arr)
            self.page.plotframe.canvas.draw()
        except Exception as e:
            print(e)

    def get_predictions(self):
        pred_indices = self.get_list_selection()
        for pred_index in pred_indices:
            self.model.update_prediction(pred_index)
            self.page.labelframe.lb_predictions.insert(pred_index,
self.model.data['prediction'][pred_index])
            self.page.labelframe.lb_predictions.delete(pred_index + 1)

```

|      |      |          |         |      |            |
|------|------|----------|---------|------|------------|
|      |      |          |         |      | Лист<br>64 |
| Изм. | Лист | № докум. | Подпись | Дата |            |

### Б.3 Текст модуля view.py

```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
import tkinter.ttk as ttk
LARGE_FONT= ("Verdana", 12)

import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg,
NavigationToolbar2Tk
from matplotlib.figure import Figure
from matplotlib import style
style.use('ggplot')

class StartPage(tk.Frame):

    def __init__(self, parent, controller):
        super().__init__(parent)
        label = tk.Label(self, text="", font=LARGE_FONT)
        label.grid(row=1, column=1, pady=10,padx=10)

        # Frame to contain the list views
        self.labelframe = ListViewFrame(self, text="Files")
        self.labelframe.grid(row=2, column=1, pady=10,padx=10, sticky=NSEW)

        # Frame to contain the buttons
        self.button_frame = ttk.Frame(self)
        self.button_frame.grid(row=3, column=1, pady=10,padx=10)

        self.button_open =      ttk.Button(self.button_frame, text="Open")
        self.button_del_selected = ttk.Button(self.button_frame, text="Delete")
        self.button_del_all =   ttk.Button(self.button_frame, text="Delete All")
        self.button_pred =     ttk.Button(self.button_frame, text="Predict")
        self.draw_button =     ttk.Button(self.button_frame, text="Draw")

        self.button_open.grid    (row=1, column=1, pady=3,padx=5, rowspan=1)
        self.button_del_selected.grid(row=2, column=1, pady=3, padx=5, rowspan=1)
        self.button_del_all.grid  (row=3, column=1, pady=3, padx=5, rowspan=1)
        self.button_pred.grid    (row=1, column=3, pady=3, padx=30, rowspan=1)
        self.draw_button.grid    (row=2, column=3, pady=3, padx=30, rowspan=1)
```

|      |      |          |         |      |
|------|------|----------|---------|------|
| Изм. | Лист | № докум. | Подпись | Дата |
|------|------|----------|---------|------|

```

# Frame to contain the PlotFrame
self.plotframe = PlotFrame(self)
self.plotframe.grid(row=1, column=2, rowspan=3)

class ListViewFrame(ttk.LabelFrame):

    def __init__(self, parent, **kwargs):
        super().__init__(parent, **kwargs)
        label_fn = tk.Label(self, text="Filenames")
        label_fn.grid(row=1, column=1, pady=5, padx=5)
        label_pred = tk.Label(self, text="Predictions")
        label_pred.grid(row=1, column=2, pady=5, padx=5)

        scrollbar = ttk.Scrollbar(self)
        self.lb_filenames = tk.Listbox(
            self,
            height=30,
            width=50,
            selectmode=EXTENDED,
            yscrollcommand=scrollbar.set,
        )
        self.lb_filenames.grid(row=2, column=1)
        self.lb_predictions = tk.Listbox(
            self,
            height=30,
            width=20,
            selectmode=EXTENDED,
            yscrollcommand=scrollbar.set,
        )
        self.lb_predictions.grid(row=2, column=2)
        self.lb_filenames.bind("<MouseWheel>", self.on_mouse_wheel)
        self.lb_predictions.bind("<MouseWheel>", self.on_mouse_wheel)
        scrollbar.grid(row=2, column=3, rowspan=1, sticky=N+S+W)
        scrollbar.config(command=self.yview)

    def yview(self, *args):
        self.lb_filenames.yview(*args)
        self.lb_predictions.yview(*args)

    def on_mouse_wheel(self, event):
        """
        """

```

Source: <https://stackoverflow.com/questions/17355902/python-tkinter-binding-mousewheel-to-scrollbar>

|      |      |          |         |      |
|------|------|----------|---------|------|
| Изм. | Лист | № докум. | Подпись | Дата |
|------|------|----------|---------|------|

```

self.lb_filenames.yview("scroll", int(-1*(event.delta/60)), "units")
self.lb_predictions.yview("scroll", int(-1*(event.delta/60)), "units")
# this prevents default bindings from firing, which
# would end up scrolling the widget twice
return "break"

class PlotFrame(ttk.Frame):

    def __init__(self, parent):
        super().__init__(parent)
        self.f = Figure(figsize=(5,5), dpi=100)
        self.a = self.f.add_subplot(111)

        self.canvas = FigureCanvasTkAgg(self.f, self)
        self.canvas.draw()
        self.canvas.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH,
                                         expand=True)

        toolbar = NavigationToolbar2Tk(self.canvas, self)
        toolbar.update()
        self.canvas._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

```

| Изм. | Лист | № докум. | Подпись | Дата | БКР-НГТУ-09.03.01-(16-В-2)-007-2020 (ПЗ) | Лист |
|------|------|----------|---------|------|--|------|
|      |      |          |         |      |  | 67   |