Рассматриваются вопросы эффективного применения методов и средств мониторинга и идентификации геофизических процессов с помощью технологий машинного обучения, а также особенности интегрирования систем искусственного интеллекта в информационно-измерительные системы. Особое внимание уделено методам мониторинга и идентификации геофизических процессов на базе технологий глубокого обучения.

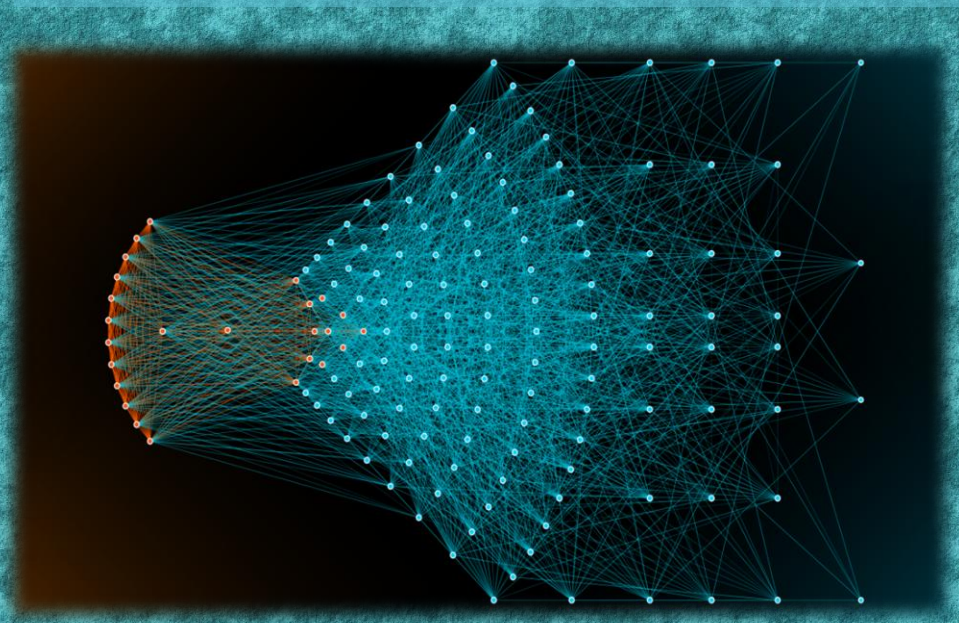
Анализируются принципы использования систем искусственного интеллекта как основных компонентов различных информационно-измерительных систем в таких областях, как разработка измерительных приборов; прогнозирование временных рядов, возникающих при изучении различных геофизических процессов, а также в медицине и системах контроля силовых кабельных линий.

Рекомендуется в качестве методического пособия научным и инженерно-техническим работникам, занимающимся применением технологий машинного обучения для мониторинга, идентификации и прогнозирования в различных областях научных исследований.

Верзунов Сергей Николаевич
кандидат технических наук,
доцент кафедры «Информационно-вычислительные технологии» КРСУ,
в.н.с. лаборатории **информационно-измерительных систем** Института машиноведения и автоматики НАН Кыргызской Республики

МОНИТОРИНГ И ИДЕНТИФИКАЦИЯ ГЕОЭКОЛОГИЧЕСКИХ ПРОЦЕССОВ НА БАЗЕ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

С.Н. ВЕРЗУНОВ



Kindle Direct Publishing, Seattle, Washington, United States

MONITORING AND IDENTIFICATION OF GEOECOLOGICAL PROCESSES ON THE BASIS OF INTELLIGENT DATA ANALYSIS

SERGEI N. VERZUNOV

ISBN: 9798444069325
2022

ИЗДАТЕЛЬСКИЙ ЦЕНТР «ИЛИМ» ПРИ ПРЕЗИДИУМЕ
НАЦИОНАЛЬНОЙ АКАДЕМИИ НАУК
КЫРГЫЗСКОЙ РЕСПУБЛИКИ

С.Н. ВЕРЗУНОВ

МОНИТОРИНГ И ИДЕНТИФИКАЦИЯ ГЕОЭКОЛОГИЧЕСКИХ ПРОЦЕССОВ НА БАЗЕ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

Бишкек 2022
ИЛИМ

УДК 004.65
ББК 16.333
В 31

Рекомендовано к печати
Ученым советом Института машиноведения и
автоматики НАН КР
Рецензенты:
д-р техн. наук, проф. Н.М. Лыченко,
д-р техн. наук, доц. А.Б. Бакасова

Верзунов С.Н.
В31 МОНИТОРИНГ И ИДЕНТИФИКАЦИЯ ГЕОЭКОЛОГИЧЕСКИХ ПРОЦЕССОВ НА БАЗЕ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ — Б.: Илим, 2022. — 168с.

ISBN 978-9967-12-943-6

Рассматриваются вопросы эффективного применения методов и средств мониторинга и идентификации геофизических процессов с помощью технологий машинного обучения, а также особенности интегрирования систем искусственного интеллекта в информационно-измерительные системы. Особое внимание уделено методам мониторинга и идентификации геофизических процессов на базе технологий глубокого обучения.

Анализируются принципы использования систем искусственного интеллекта как основных компонентов различных информационно-измерительных систем в таких областях, как разработка измерительных приборов; прогнозирование временных рядов, возникающих при изучении различных геофизических процессов; а также в медицине и системах контроля силовых кабельных линий.

Рекомендуется в качестве методического пособия научным и инженерно-техническим работникам, занимающимся применением технологий машинного обучения для мониторинга, идентификации и прогнозирования в различных областях научных исследований.

ISBN 978-9967-12-943-6

УДК 004.65
ББК 16.333

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В последние несколько лет, как в профессиональной литературе, так и в средствах массовой информации, постоянно появляется информация о последних достижениях и улучшениях, которые системы искусственного интеллекта (СИИ) привнесли в широкий спектр областей – от технических и естественных наук до лингвистики и многих других областей. ИИ является одной из ключевых технологий, обеспечивающих четвертую промышленную революцию, наступление которой мы наблюдаем в настоящее время. Применение ИИ уже меняет наш мир и влияет на все аспекты жизни общества, экономику и технический прогресс в целом Область мониторинга и измерительных процессов — не исключение, и на них уже также повлияло применение СИИ.

В настоящее время применение технологий машинного обучения позволяет улучшить систему прогнозирования загрязнения воздуха и дальности видимости в аэропорту, определить тип и место возникновения неисправности, выполнить диагностику различных заболеваний легких, повысить точность работы прибора для измерения параметров индуктивного компонента магнитометра. Ученые ожидают, что в будущем СИИ будут соответствовать или даже превосходить человеческий интеллект. Итак, что же это за методы?

На рис. В.1 показаны некоторые из наиболее часто применяемых технологий ИИ, которые, как правило, вдохновлены тем, как работает биологический мозг или биологические системы в целом.

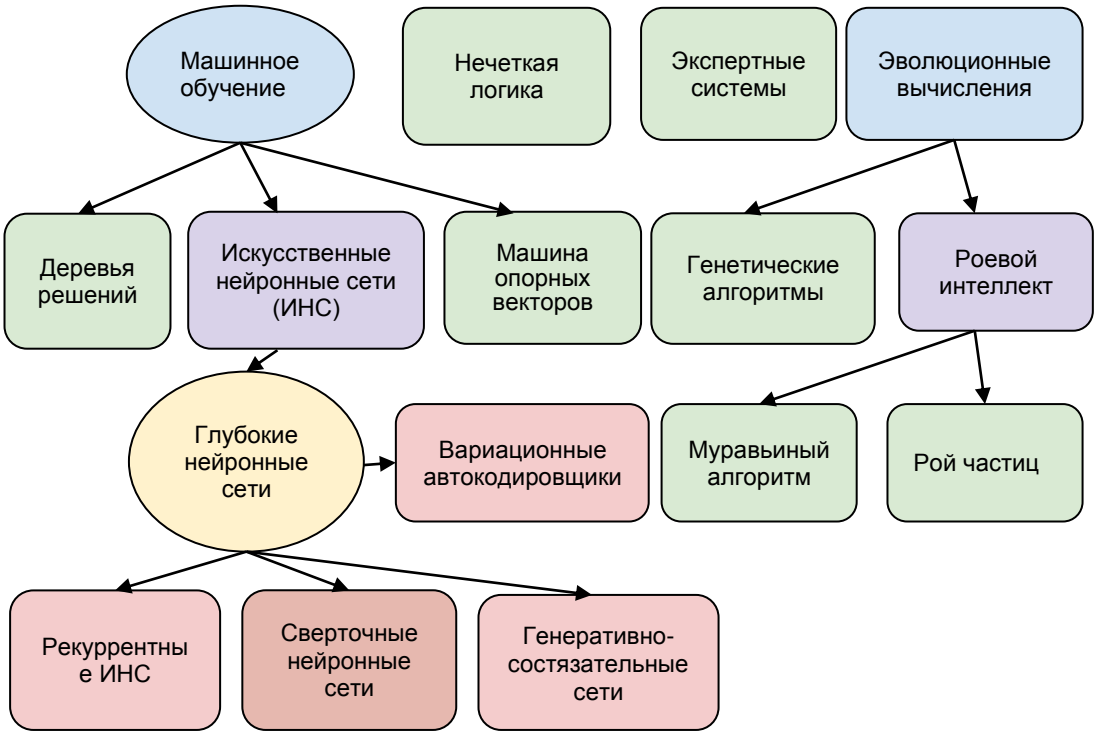


Рисунок В.1 – Некоторые из наиболее часто используемых технологий ИИ

Например: искусственные нейронные сети пытаются эмулировать нейронные сети в живом мозге, нечеткая логика пытается действовать подобно тому, как люди принимают решения без сложного математического моделирования с использованием неточной или расплывчатой информации.

Что сделало ИИ такой прорывной технологией, так это тот факт, что эти методы позволяют нам выполнять такие задачи, как классификация, кластеризация, прогнозирование и оптимизация без необходимости разработки аналитической модели предметной области. Это очень важно, поэтому давайте рассмотрим это подробнее.

Большинство этих методов машинного обучения принимают в качестве входных данных набор ранее собранных данных и пытаются создать модель, наилучшим образом соответствующую этим входным данным и желаемым выходным результатам. Затем эту модель можно использовать для того, чтобы получить выходные значения для новых входных данных. Это означает, что модель, которую создает метод ИИ, основана исключительно на сопоставлении входных и выходных данных (черный ящик), в отличие от аналитической модели, которую создает эксперт в предметной области (белый ящик). Например, предположим, что необходимо создать систему, которая определяет заболевание легких по рентгеновскому снимку грудной клетки пациента. Чтобы аналитически создать такую систему, инженеру-биоинформатику необходимо разработать математическую модель, основанную на форме поражений легких, характере, интенсивности, распределении, расположении, площади и многих других параметрах, некоторые из которых сложно или даже невозможно учесть аналитически. Разработка такой аналитической модели может быть очень сложной, особенно если модель должна быть обобщаемой, т.е. работать с учетом индивидуальных особенностей строения легких. Необходимо учесть слишком много параметров, чтобы придумать общую модель, которая будет работать с приемлемой точностью. Но, если мы используем технологии ИИ, например глубокую нейронную сеть, необходимо только обучить ее на достаточно большом наборе ранее сделанных рентгеновских снимков. Затем метод создает собственную модель того, как сопоставлять особенности на этих рентгеновских снимках с наличием того или иного заболевания. Это не только значительно снижает сложность, но и в некоторых случаях дает даже лучший результат, чем аналитическая модель. Конечно, точность этой модели на основе ИИ зависит от конкретной выбранной технологии и алгоритма обучения ИИ, а также от качества имеющегося набора данных. Имея в виду это определение ИИ, давайте теперь посмотрим, как мы можем использовать его в области мониторинга и идентификации геофизических процессов.

Актуальность применения СИИ в информационно-измерительных системах объясняется тем, что методы ИИ вполне подходят для

обнаружения, отслеживания, мониторинга, характеристики, идентификации, оценки, диагностики или прогнозирования. На самом деле огромный потенциал использования ИИ в информационно-измерительных системах был замечен уже в 1990-х годах, например, в [1], где очень хорошо описано применение нейронных сетей, нечеткой логики и экспертных систем в таких областях, как проектирование и калибровка датчиков, прогнозирование и интерполяция данных измерений, программные компоненты информационно-измерительных систем, косвенные измерения, обнаружение неисправностей и идентификация систем. ИИ особенно удобен, когда построение аналитической модели измерительной системы или имеющегося инструмента очень сложно, сильно нелинейно, очень динамично или невозможно из-за отсутствия знаний о системе, за исключением ограниченного числа параметров. В таких случаях или когда точность конечного измерения важнее понимания того, как именно работает система, СИИ может предложить привлекательное и практичное решение. Одним из примеров, который хорошо иллюстрирует вышеупомянутое понятие, является измерение на основе зрения (Vision-Based Measurement, VBM) [2], использующее ИИ в качестве основного компонента измерительной системы. Примером VBM-системы является описанная в работе [3] система, которая косвенно измеряет количество калорий и питательных веществ в пище по изображению еды. Такая система была бы просто невозможна без ИИ из-за очень сложной природы проблемы и отсутствия полного знания всех параметров. Еще одной областью применения ИИ является калибровка, например, в [4], где изображение пользователя используется для простой калибровки протеза руки.

Прогнозирование измерений – это другая полезная функция ИИ, которую можно применять, когда фактическое измерение либо дорого, либо невозможно. Например, в сетевой системе с большим количеством источников невозможно явно измерить сквозную задержку между всеми парами узлов в сети из-за $O(N^2)$ сложности проблемы. Было показано, что в таком случае ИИ предсказывает измерения более точно и на несколько порядков быстрее, чем без использования искусственного интеллекта [5].

На рис. В.1 такой раздел машинного обучения, как глубокое обучение (Deep Learning, DL), выделен особо, потому что в настоящее время именно этот раздел нашел наиболее широкое практическое применение. Хотя глубокое обучение существует с 1980-х годов, только в 2011-м и 2012 годах оно вызвало широкую огласку среди исследователей и практиков, когда системы распознавания визуальных объектов на основе глубокого обучения, такие как GoogLeNet и AlexNet, опередили своих конкурентов с огромным отрывом и с точки зрения повышения частоты ошибок [6]. Это произошло отчасти благодаря графическим процессорам, обеспечивающим значительное ускорение, необходимое для DL. Другая причина их популярности заключается в том, что алгоритмы DL автоматизируют

процесс выделения значимых представлений признаков в наборе данных, который обычно требует знания предметной области и значительных человеческих усилий. Вот почему в последние годы DL произвело революцию в области применения ИИ. Чтобы понять, как ИИ может достигать таких замечательных результатов, необходимо проанализировать методы применения машинного обучения в области мониторинга и идентификации процессов в различных направлениях науки и техники.

Далее материал в монографии организован следующим образом. В первой главе анализируются особенности применения индуктивных магнитометров, использующихся для мониторинга геоэкологических процессов, рассматриваются способы измерения параметров индуктивных компонентов для разработки способа измерения их параметров на основе методов машинного обучения. Кроме того, рассмотрена архитектура программных средств, реализующих предложенный метод.

Во второй главе рассматриваются методы прогнозирования временных рядов, порождаемых различными геоэкологическими процессами, разработана модель прогноза временных рядов на основе полиморфной мультивейвлетной сети и архитектуры глубоких нейронных сетей на GRU и LSTM нейронных слоев. Сравниваются прогностические возможности предложенных моделей на примере прогноза уровня загрязнённости воздуха.

В третьей главе раскрываются особенности диагностики заболеваний легких с помощью рентгеновских снимков. Обсуждаются архитектура глубокой нейронной сети для диагностики COVID-19 и программные средства для диагностики болезней легких.

В четвертой главе освещаются проблемы создания системы мониторинга подземных силовых кабельных линий с использованием технологий глубокого обучения. Глубоко раскрывается инженерная часть, посвященная разработке системы искусственного интеллекта для мониторинга неисправностей.

1. Определение параметров индуктивных компонентов

1.1 Особенности измерения с помощью индукционных магнитометров

На современном этапе развития геомагнитных измерений, характеризующемся широким внедрением информационных технологий, необходима работа геомагнитных обсерваторий и их взаимодействие с потребителями информации на качественно новом техническом уровне, включающем цифровые методы обработки и накопления данных, что позволило бы значительно повысить эффективность геофизических исследований.

Однако на геомагнитных обсерваториях Кыргызстана традиционно используются трёхкомпонентные аналоговые магнитовариационные станции на основе кварцевых магнитных вариометров Боброва с записью данных на фотобумагу. Они позволяют измерять вариации трёх компонент вектора напряженности магнитного поля Земли. Несмотря на высокую надёжность и простоту использования магнитовариационных станций, многие из них зачастую простаивают из-за отсутствия дорогостоящей фотобумаги. К тому же возникают дополнительные сложности при переводе магнитограмм в цифровую форму для удобного анализа и хранения. Это говорит о том, что уровень геомагнитных измерений еще недостаточно высок и магнитометрическое оборудование на отечественных обсерваториях нуждается в модернизации для максимального упрощения и автоматизации получения необходимой информации о состоянии магнитного поля Земли, ее обработке, накоплению и хранению. Разработка технических средств и методов измерений геомагнитного поля позволила бы геомагнитным обсерваториям Кыргызстана включиться в Международную программу INTERMAGNET с целью создания глобальной сети цифровых магнитных обсерваторий, внедрения современных стандартов измерения и регистрации магнитных вариаций.

В связи с этим в настоящее время достаточно актуально создание высокочувствительных цифровых трёхкомпонентных магнитометров, позволяющих с достаточной точностью регистрировать изменение не только модуля, но и компонент вектора геомагнитного поля, что позволит существенно улучшить аппаратную базу и эффективность геофизических исследований в Кыргызстане.

Существует множество типов магнитометров, различающихся принципом действия, способом регистрации и хранения измеренных данных. Индукционные магнитометры применяются для измерения земного и космических магнитных полей, а также различных технических полей. Принцип действия такого магнитометра основан на явлении электромагнитной индукции – возникновении э.д.с. в измерительной катушке при изменении проходящего сквозь её контур магнитного потока:

$$\Phi = \mu_0 \mu Sw(H_T n + H_n n),$$

где μ_0 – магнитная постоянная, μ – эффективная проницаемость сердечника, S – площадь контура, w – число витков, H_T – постоянное магнитное поле, δH_n – переменное магнитное поле, n – нормаль к контуру.

Электродвижущая сила, возникающая в контуре:

$$\varepsilon = -\mu_0 \mu Sw \frac{\partial}{\partial t} \delta H_n = -\mu_0 \mu Sw \frac{\partial}{\partial t} \delta Xk + \delta Yl + \delta Zm \ , \qquad (1.1)$$

где k, l , m – единичные орты по осям X, Y, Z [4].

Из (1.1) видно, что при произвольном расположении контура к вектору H э.д.с. будет линейной функцией производных всех составляющих этого вектора, а электродвижущая сила, возникающая в контуре, является функцией любого параметра – μ, r, w, δH_n. Так как все они, кроме δH_n, остаются постоянными, используемый метод измерения является пассивным, а индукционный магнитометр такого типа регистрирует не величину магнитного поля, а скорость его изменения, и, кроме того, для одновременной регистрации всех трёх компонент необходимо иметь три контура, расположенных в трёх взаимно перпендикулярных направлениях.

Произведение μSw – коэффициент преобразования, зависящий только от конструктивных характеристик преобразователя. Эффективная проницаемость μ сердечника зависит от отношения его диаметра к длине, или, другими словами, от коэффициента размагничивания D. Эффективная магнитная проницаемость μ с учетом коэффициента размагничивания D рассчитывается по формуле:

$$\mu = \frac{\mu_r}{1 + D(\mu_r - 1)},$$

где μ_r – начальная магнитная проницаемость материала сердечника.

Например, при D= 0,l и μ_r >1000 максимальное значение μ составит не более 10. С укорочением длины сердечника при сохранении его диаметра коэффициент преобразования уменьшается. Стабильность коэффициента преобразования зависит от стабильности свойств сердечника, поэтому необходимо учитывать изменение магнитной проницаемости от температуры, частоты и изменения магнитного поля, величины постоянного поля. Уровень собственного шума индукционного датчика определяется также тепловыми шумами в катушке. Спектральная плотность флюктуаций магнитного поля, эквивалентных тепловым шумам индукционного датчика, определяется как

$$N = \frac{\sqrt{4kTr_n}}{\omega w \mu S},$$

где k – постоянная Больцмана, T – температура в градусах Кельвина, r_n – сопротивление катушки датчика ω — циклическая частота [8].

Потенциальные возможности индукционного датчика с сердечником ограничиваются фактором размагничивания. Выбор конструктивных размеров пассивного индукционного датчика не является однозначным для достижения максимальной чувствительности. Недостатком индукционного датчика при геофизических исследованиях являются относительно большие геометрические размеры датчика, что снижает точность локализации источников магнитного поля. Потеря чувствительности индукционного датчика на низких частотах снижает возможности измерения геомагнитных полей, но для сигналов с частотой выше 100 Гц индукционный датчик может быть более чувствительным, чем современные сверхпроводящие магнитоизмерительные приборы [9].

Таким образом, известен индукционный датчик (рис. 1.1), состоящий из трёх катушек – продольной (X), поперечной (Y) и вертикальной (Z), закреплённых на платформе. При этом каждая из катушек X, Y и Z содержит несколько сотен тысяч витков и состоит из двух последовательно соединённых частей, намотанных в несколько слоёв. Снижение шума индукционного датчика достигается введением ферритового сердечника на ферритовом стержне начальной магнитной проницаемостью 2000.

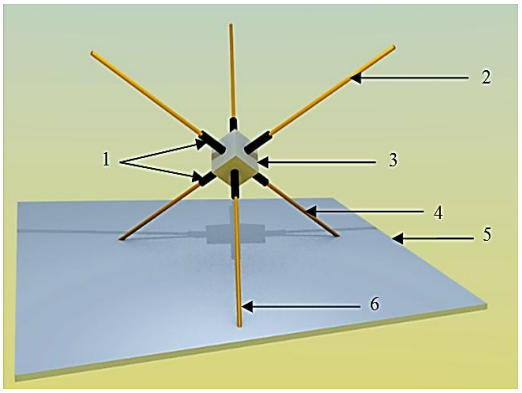


Рисунок 1.1 – Трёхкомпонентный индукционный датчик:1 – крепёжные стойки, 2 – поперечная катушка Y, 3 – соединительная коробка, 4 – продольная катушка X, 5 – платформа, 6 – вертикальная катушка Z

Однако на точность измерения магнитного поля, как, впрочем, на многие другие измерения, сильно влияет промышленная электрическая сеть, дающая помеху частотой 50 Гц, от которой практически невозможно полностью избавиться.

Для того чтобы уменьшить влияние электрической сети, предлагается использовать двойной Т-образный фильтр с буферным усилителем, препятствующим влиянию фильтра на индукционный датчик (рис.1.2).

Для измерения АЧХ двойного Т-образного фильтра используется плата сбора данных PCI-1710HG. На вход фильтра подается сигнал с аналогового выхода платы сбора данных, генерируемый с использованием возможностей среды MatLab в сочетании с пакетами Simulink, Real-Time WorkShop. Частота этого сигнала линейно возрастает на протяжении всего

времени снятия АЧХ. Отфильтрованный сигнал, снимаемый с выхода фильтра, оцифровывается АЦП платы сбора данных, и к полученным данным применяется быстрое преобразование Фурье с помощью Signal Processing Toolbox. Измеренная таким образом АЧХ двойного Т-образного фильтра показана на рис. 1.3.

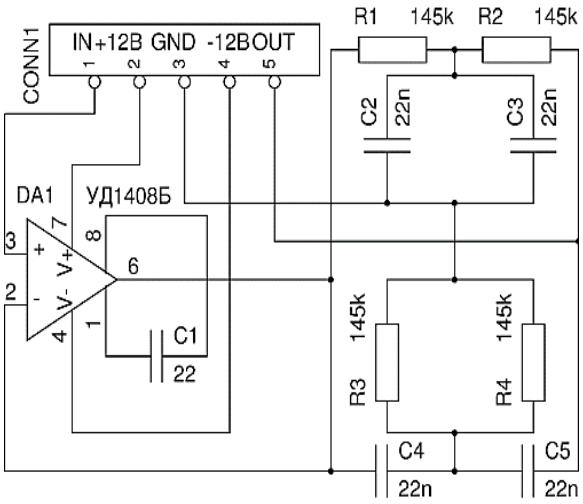


Рисунок 1.2 - Принципиальная схема двойного Т-образного фильтра с буферным усилителем

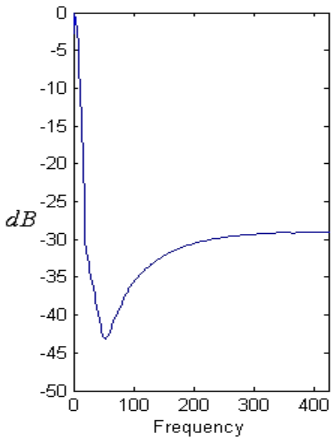


Рисунок 1.3 – АЧХ двойного Т-образного фильтра

Параметры фильтра рассчитываются по формуле:

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 145 \cdot 10^3 \text{Ом} \cdot 22 \cdot 10^{-9} \Phi} \approx 50 \text{Гц}. \qquad (1.2)$$

Величина ослабления помехи двойным Т-образным фильтром на частоте 50 Гц достигает 43 дБ по сравнению с полезным сигналом на частоте 1 Гц. И 15 дБ по сравнению с полезным сигналом на частоте 500 Гц. Следовательно, отношение сигнал/шум увеличивается по меньшей мере в 30 раз, что позволяет значительно повысить точность измерений.

Обычно в качестве коммутационных элементов в подобных схемах используются электронные ключи. Это позволяет сократить структурную избыточность за счет использования только одного буферного усилителя и фильтра (рис. 1.4).

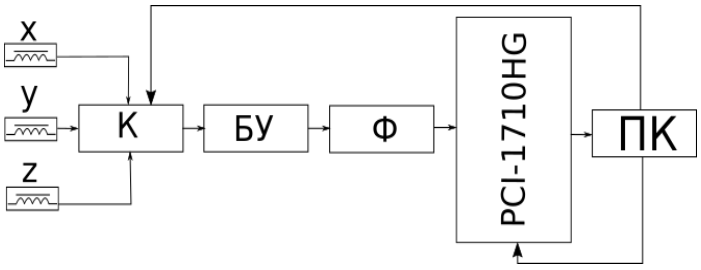


Рисунок 1.4 – Структурная схема индукционного вариометра с электронным ключом

Демультиплексирование отфильтрованного сигнала можно осуществить при обработке полученных данных в ПК, используя Simulink-модель, показанную на рис. 1.5.

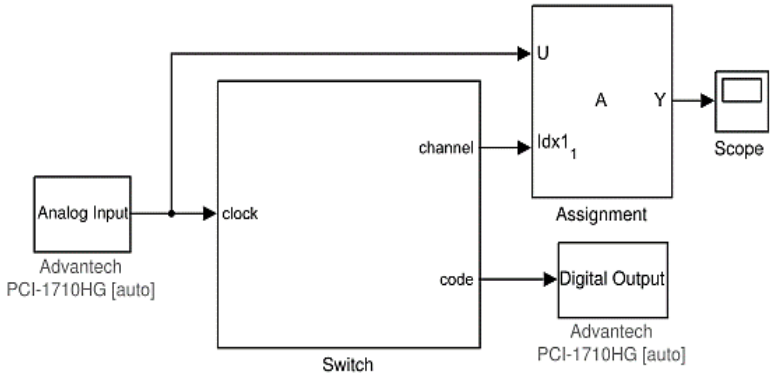


Рисунок 1.5 – Simulink-модель для коммутации датчиков

Для управления коммутатором можно использовать цифровые выходы платы сбора данных, сигнал управления формируется с помощью блока, структура которого показана на рис. 1.6.

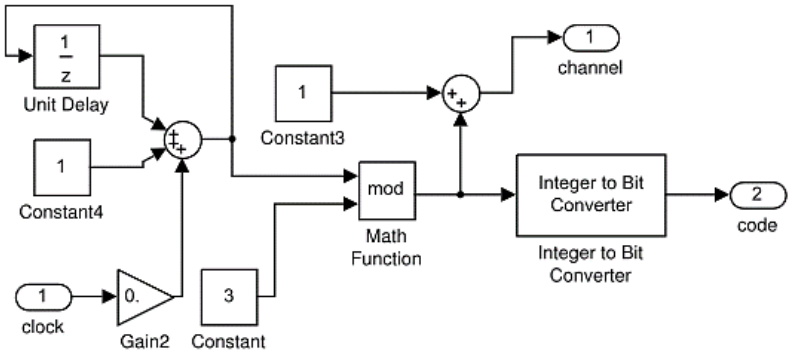


Рисунок 1.6 – Simulink-модель блока управления коммутатором

Однако в момент коммутации возникают переходные процессы (рис.1.7) в колебательном контуре, образованном большой индуктивностью датчика и емкостью полупроводникового электронного ключа, что не позволяет коммутировать датчики с частотой более 20 Гц.

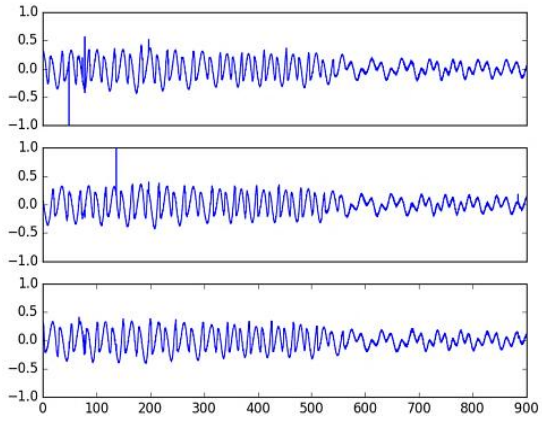


Рисунок 1.7 – Переходные процессы, возникающие при коммутации индукционного датчика

В связи с этим в конструкции магнитометра (рис. 1.8) предлагается использовать три одинаковых двойных Т-образных фильтра с буферными усилителями, поскольку большая ёмкость электронных ключей, обычно

используемых в качестве коммутационных элементов, не позволяет получить данные о геомагнитном поле с высокой точностью квантования и частотой дискретизации.

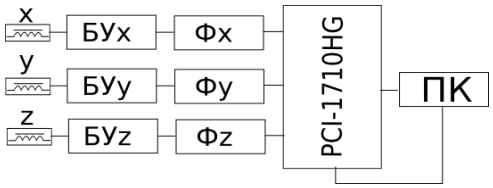


Рисунок 1.8 – Структурная схема индукционного вариометра

Выходы фильтров соединены с входными каналами платы собора данных PCI-1710HG, управляемой персональным компьютером через PCI шину, что позволяет регулировать усиление измеряемого сигнала, а также выбирать необходимые для записи данные и частоту их дискретизации.

На рис. 1.9 показана магнитограмма, полученная с использованием разработанного нами магнитометра. На рисунке видны искусственно введенные возмущения от электрической дуги.

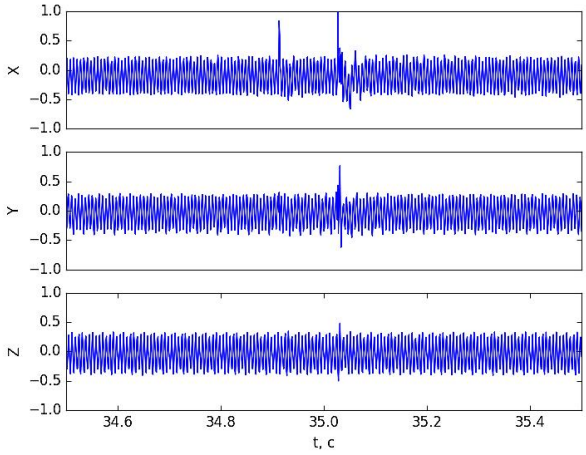


Рисунок 1.9 – Пример записи показаний магнитометра

1.2 Измерение параметров индуктивных компонентов

Как было показано выше, индуктивные компоненты магнитометров отличаются от типовых индуктивностей тем, что содержат, как правило, сотни тысяч витков, в связи с чем обладают достаточно большой индуктивностью, паразитной емкостью и активным сопротивлением, выходящим за пределы измерений серийных приборов. Кроме того, индуктивный компонент магнитометра с большим числом витков сам по себе является активным элементом, в котором генерируется значительная ЭДС индукция, оказывающая сильное влияние на работу серийных LCR-метров любого типа. Тем не менее задача определения параметров индуктивных элементов часто возникает при создании, отладке и ремонте магнитометров. В связи с этим возникает острая необходимость разработки устройства, способного работать в широких диапазонах

измерения *L*, *C*, *R*, несмотря на присутствие переменной ЭДС в измеряемом элементе, случайной по своей величине (см. рис. 1.9).

Проблема измерения параметров электротехнических компонентов очень часто возникает и при конструировании, наладке и ремонте различных измерительных систем. Например, во многих устройствах индуктивный элемент используется для получения резонансной цепи, настроенной на определенную частоту. Кроме того, это и электронные балласты питания, радиоприемные устройства, фильтры разделения частот и так далее. Иногда, помимо значения индуктивности *L*, крайне желательно знать ещё и паразитную ёмкость *C*, а также ее активное сопротивление потерь *R*.

Кроме того, измерение параметров электротехнических компонентов может иметь самостоятельное значение в ходе экспериментальных и научных работ. К примеру, для измерений неэлектрических величин, таких, как температура, влажность, давление, ускорение, часто применяют резистивные, емкостные и индуктивные приемные преобразователи, в этом случае параметры измеряют для оценки значений неэлектрических величин. Наконец, задача определения всех параметров индуктивности возникает при моделировании поведения различных измерительных систем и датчиков.

Для индуктивного компонента применяются различные схемы замещения с сосредоточенными параметрами. Правильность результатов расчета параметров индуктивности во многом зависит от того, насколько выбранная схема замещения реальной индуктивности соответствует принятым тем или иным допущениям, что определяется влиянием побочных параметров на уравнения, связывающие между собой параметры режима работы, например полное комплексное сопротивление на различных частотах, и параметры схемы замещения. Традиционно эквивалентной схемой замещения реального индуктивного элемента является схема, показанная на рисунке 1.10. Основной параметр – это индуктивность, а побочные параметры – сопротивление потерь и собственная емкость [11].

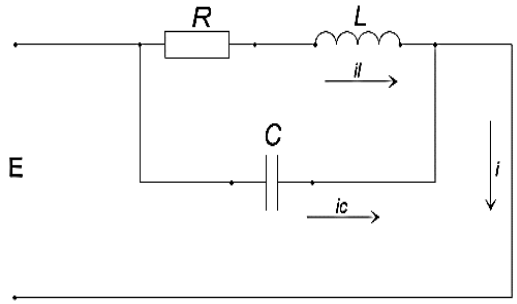


Рисунок 1.10 – Традиционная схема замещения индуктивного компонента

Оценка использования разных схем замещения реальной индуктивности приведена в работе [10]. Для схемы электрической цепи, показанной на рисунке 1.10, по второму закону Кирхгофа можно записать

следующие дифференциальные уравнения:

$$E = i_L \cdot R + \frac{d\Psi}{dt}, \tag{1.3}$$

$$u_C = i_L \cdot R + \frac{d\Psi}{dt}, \tag{1.4}$$

а по первому закону Кирхгофа:

$$i = i_L + i_C. \tag{1.5}$$

Ток i_C, протекающий через конденсатор при его зарядке:

$$i_C = C \cdot \frac{du_C}{dt} \tag{1.6}$$

Из (1.6) определяется напряжение на конденсаторе:

$$u_C = \frac{1}{C}\int i_C \cdot dt.$$

Затем формула напряжения на конденсаторе (1.6) подставляется в (1.4):

$$\frac{1}{C}\int i_C \cdot dt = i_L \cdot R + \frac{d\Psi}{dt}, \tag{1.7}$$

в результате уравнение (1.7) дифференцируется:

$$\frac{1}{C}\int i_C \cdot dt = i_L \cdot R + \frac{d\Psi}{dt} \tag{1.8}$$

$$\frac{i_C}{C} = R \cdot \frac{di_L}{dt} + \frac{d^2\Psi}{dt^2} \tag{1.9}$$

и из (1.9) определяется отличное от (1.6) выражение тока, протекающего через конденсатор

$$i_C = C \cdot (R \cdot \frac{di_L}{dt} + \frac{d^2\Psi}{dt^2}). \tag{1.10}$$

Из (1.3) выражается ток i_L, проходящий через индуктивность:

$$i_L = \frac{E - \frac{d\Psi}{dt}}{R}, \tag{1.11}$$

а после дифференцирования (1.11) – его производная

$$i_L = \frac{E - \frac{d\Psi}{dt}}{R}, \tag{1.12}$$

Для определения связи между током через индуктивность i (ток в общей ветви на рис. 1.10), ЭДС E и параметрами схемы замещения выражения (1.10), (1.11) и (1.12) подставляются в (1.5):

$$\begin{aligned} i = i_L + i_C &= \frac{E}{R} - \frac{1}{R} \cdot \frac{d\Psi}{dt} + C \cdot R \cdot \frac{di_L}{dt} + C \cdot \frac{d^2\Psi}{dt^2} = \\ &= \frac{E}{R} - \frac{1}{R} \cdot \frac{d\Psi}{dt} + C \cdot R \cdot (-\frac{1}{R} \cdot \frac{d^2\Psi}{dt^2}) + C \cdot \frac{d^2\Psi}{dt^2} = \\ &= \frac{E}{R} - \frac{1}{R} \cdot \frac{d\Psi}{dt} - C \cdot \frac{d^2\Psi}{dt^2} + C \cdot \frac{d^2\Psi}{dt^2} = \\ &= \frac{E}{R} - \frac{1}{R} \cdot \frac{d\Psi}{dt}. \end{aligned} \tag{1.13}$$

В результате преобразований в формуле (1.12) отсутствует собственная емкость индуктивного компонента, что свидетельствует о том, что ток в общей ветви схемы замещения не зависит от этого параметра.

Уравнение, идентичное (1.12), получается, если его составить по второму закону Кирхгофа для схемы замещения индуктивности без учета ее емкости (рис. 1.11):

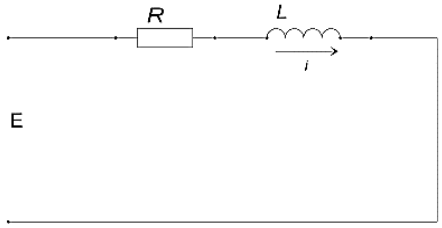


Рисунок 1.11 – Упрощенная схема замещения индуктивного компонента

$$E = i \cdot R + \frac{d\Psi}{dt} \tag{1.14}$$

Сопоставляя между собой выражения (1.13) и (1.14), можно заметить, что схема замещения (см. рис. 1.10) неадекватно отражает процессы, протекающие в реальном индуктивном элементе, подключённом к идеальному источнику ЭДС, так как физически емкость существует. Аналогично анализу применения схемы (см. рис. 1.10) проводится оценка адекватности использования схемы замещения индуктивности, показанной (рис. 1.12).

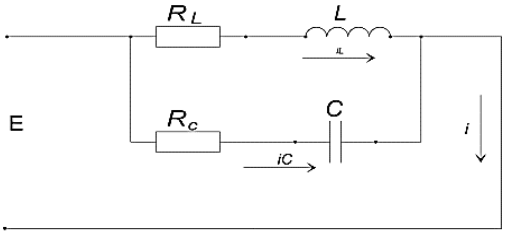


Рисунок 1.12 – Полная схема замещения индуктивного компонента

Система уравнений, составленная по законам Кирхгофа:

$$E = i_L \cdot R_L + \frac{d\Psi}{dt}, \tag{1.15}$$

$$\frac{1}{C} \cdot \int i_C \cdot dt + i_C \cdot R_C = i_L \cdot R_L + \frac{d\Psi}{dt}, \tag{1.16}$$

$$i = i_L + i_C. \tag{1.17}$$

Решением системы (1.16) относительно тока в общей ветви i является следующее уравнение:

$$i = \frac{E}{R_L} - \frac{1}{R_L} \cdot \frac{d\Psi}{dt} - C \cdot R_C \cdot \frac{di_C}{dt}. \tag{1.18}$$

В уравнении (1.12) параметры режима и их производные связаны со всеми параметрами схемы замещения индуктивного компонента, следовательно, схема замещения (см. рис. 1.12) адекватна реальной индуктивности и более точно учитывает потери энергии при подключении ее к идеальному источнику ЭДС.

Таким образом, при моделировании реального индуктивного компонента, подключенного к идеальному источнику ЭДС, схему замещения, указанную на рисунке 1.10, нельзя использовать, так как, несмотря на присутствие в схеме конденсатора, собственная емкость не влияет на выражение тока (1.13), протекающего по обмотке индуктивного компонента.

Напротив, полная схема замещения, как следует из (1.12), более полно отражает физические процессы при включении реальной индуктивности на идеальный источник ЭДС, значит, применение этой схемы замещения при моделировании электрических цепей более предпочтительно по сравнению с первой схемой (см. рис. 1.10).

Физический смысл сопротивления *Rc* заключается в учете потерь в изоляции проводов и каркасе индуктивного компонента, возникающих при прохождении по ней переменного тока. Они в свою очередь складываются из потерь в диэлектрике межвиткового конденсатора (межвитковые утечки и прочие потери, характерные для диэлектриков конденсаторов) и потерь,

обусловленных магнитными свойствами диэлектрика (эти потери аналогичны потерям в сердечнике). Следует заметить, что в общем случае для современных компонентов единого применения потери в диэлектрике чаще всего достаточно малы, но для индуктивных компонентов магнитометров ввиду большого количества витков и значительной межвитковой ёмкости могут достигать значительной величины. Потери в сердечнике, в случае его наличия, складываются из потерь на вихревые токи, потерь на перемагничивание ферромагнетика — на «гистерезис». Как известно, переменное магнитное поле индуцирует вихревые ЭДС в окружающих проводниках, например, в сердечнике, экране и в проводах соседних витков. Возникающие при этом вихревые токи (токи Фуко) становятся источником потерь из-за омического сопротивления проводников. Кроме того, на практике более применима схема замещения, показанная на рис. 1.12, в связи с чем возникает задача определения и других параметров, кроме L – C, R_L, еще и дополнительного параметра R_c [11].

1.3 Способы определения параметров индуктивных компонентов

Параметры L, C и R измеряют тремя основными способами: мостовым, резонансным и преобразованием во временной интервал или напряжение. При мостовом методе измеряемый элемент вводят в одно из плеч сбалансированного моста, что приводит к нарушению баланса. Затем с помощью образцовых элементов баланс восстанавливают. По изменению последних судят о параметре измеряемого элемента. Резонансный метод основан на известных зависимостях между параметрами контура L, C, R и его резонансной частотой и добротностью. Он пригоден для измерений в диапазоне частот от единиц килогерц до сотен мегагерц.

Параметры L, C и R элемента цепи можно определить как отношение напряжения на элементе к протекающему через него току. К таким измерениям часто прибегают на постоянном токе. На переменном токе приходится измерять действительную и мнимую составляющие комплексных амплитуд тока и напряжения, а затем вычислять искомые параметры. В современных приборах, реализующих этот метод, используют встроенные микропроцессорные средства, осуществляющие управление процессом измерений, коррекцию погрешностей и расчет результатов для разных схем замещений. Кроме того, параметры L, C, R можно преобразовать во временной интервал, воспользовавшись переходной характеристикой цепи, в которую входит измеряемый и образцовый элементы, например, образцовый резистор и измеряемый конденсатор.

Для точных измерений параметров индуктивных компонентов необходимы эталоны. Эталонирование параметров L, C и R основано на точном воспроизведении одного размера индуктивности, емкости или сопротивления с помощью эталонных элементов индуктивности,

конденсаторов и резисторов. Диапазон воспроизводимых размеров расширяют, используя несколько эталонных мер, периодически сравниваемых с эталоном мостовыми методами. Эталоны аттестованы на частоте 1 кГц. Эталон индуктивности состоит из четырех тороидальных катушек с индуктивностью 10 мГ, среднеквадратическим отклонением 10-6, не исключенным остатком систематической погрешности 5·10-6. Эталон емкости содержит плоский вакуумный конденсатор, размеры которого контролируют интерферометром. Конденсатор воспроизводит емкость 0,2 пФ. Содержащиеся в эталоне меры воспроизводят емкости 1 пФ – 1 мкФ с среднеквадратическим отклонением 2–10·10-6 и не исключенным остатком систематической погрешности 10–30·10-6. В эталоне сопротивления используют 10 резисторов с сопротивлением 1 Ом. Набор мер воспроизводит сопротивления 10-3– 109 Ом с среднеквадратическим отклонением 3·10-8 и не исключенным остатком систематической погрешности 3·10-7.

1.3.1 Мостовые методы. Измерительные приборы основаны на четырехплечих или более сложных мостовых схемах, обычно работающих на низких частотах или на постоянном токе. Упрощенная схема измерителя с четырехплечим мостом приведена на рисунке 1.13.

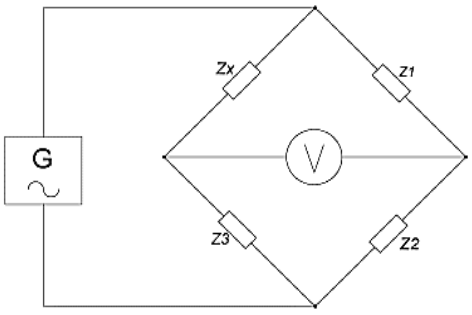


Рисунок 1.13 – Четырехплечий мост

К диагонали моста подведено синусоидальное напряжение от генератора *G*. Индикатор регистрирует напряжение, возникающее в другой диагонали моста. Искомое полное сопротивление вводят в одно из плеч моста. Затем мост балансируют, изменяя сопротивления остальных плеч.

Состояние баланса фиксируют по нулевому показанию индикатора. Условием баланса моста является соотношение:

$$\mathbf{Z}_x\mathbf{Z}_2 = \mathbf{Z}_1\mathbf{Z}_3, \tag{1.19}$$

из которого определяют измеряемое сопротивление.

Сопротивление плеч моста в показательной форме:

$$\mathbf{Z}_x = Z_x\, e^{j\varphi_x};\ \mathbf{Z}_1 = Z_1\, e^{j\varphi_1};\ \mathbf{Z}_2 = Z_2 e^{j\varphi_2};\ \mathbf{Z}_3 = Z_3 e^{j\varphi_3}.$$

Путем подстановки этих значений в (1.19) получается два условия баланса моста:

$$Z_xZ_2 = Z_1Z_3;\ \varphi_x + \varphi_2 = \varphi_1 + \varphi_x,$$

которые можно представить в форме:

$$\mathrm{Re}(\mathbf{Z}_x\mathbf{Z}_2) = \mathrm{Re}(\mathbf{Z}_1\mathbf{Z}_3), \tag{1.20}$$

$$\mathrm{Im}(\mathbf{Z}_x\mathbf{Z}_2) = \mathrm{Im}(\mathbf{Z}_1\mathbf{Z}_3). \tag{1.21}$$

Для балансирования моста необходимо изменять как модуль, так и фазу, по крайней мере одного из сопротивлений, т. е. иметь не менее двух регулируемых элементов. Для четырехплечих мостов регулировать отдельно модуль и фазу не удается. Мост балансируют методом последовательных приближений: поочередно регулируют каждый из элементов до получения минимального показания индикаторного прибора. Эти операции повторяют многократно, пока индикаторный прибор не зафиксирует нулевое показание. Минимально необходимое число операций при балансировке характеризует сходимость моста. Сходимость зависит от сопротивлений плеч моста, а также от чувствительности и типа применяемого индикатора. Если индикатор моста выполнен на основе магнитоэлектрического прибора с детектором, то по отклонению стрелки можно судить лишь об амплитуде напряжения в диагонали моста. Значительно удобнее индикатор с электролучевой трубкой. На одну пару отклоняющих пластин подают некоторое опорное напряжение, а на другую – напряжение в диагонали. При таком способе индикации можно судить об изменении как модуля, так и фазы напряжения, что позволяет ускорить процесс балансировки.

В качестве регулировочных элементов в мостах используют образцовые резисторы и конденсаторы. Изменяя R или C, можно добиться выполнения условия (1.20) при любом емкостном или индуктивном характере измеряемого полного сопротивления. Условие же (1.21) можно выполнить только при определенном характере измеряемого сопротивления.

Например, если сопротивления $\mathbf{Z}_2$ и $\mathbf{Z}_3$ моста активные, а $\mathbf{Z}_1$ емкостное, то неизвестное сопротивление $\mathbf{Z}_x$ должно быть также емкостным.

Существует много разновидностей мостовых схем для измерения сопротивления, емкости и индуктивности.

Например, для измерения емкости и сопротивления потерь конденсатора применяют схему, показанную на рисунке 1.14.

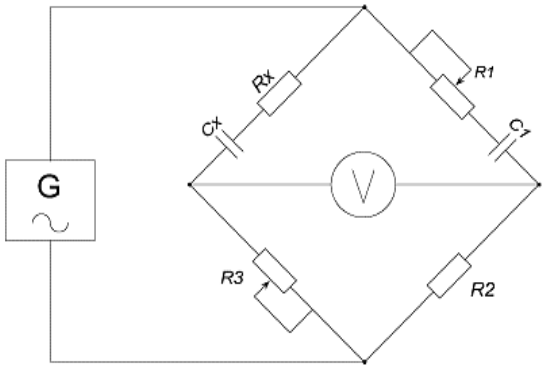


Рисунок 1.14 – Мост для измерения емкости

Здесь

$$\mathbf{Z}_x = R_x + \frac{1}{j\omega C_x}; \mathbf{Z}_1 = R_1 + \frac{1}{j\omega C_1}; \mathbf{Z}_2 = R_2; \mathbf{Z}_3 = R_3.$$

При условии баланса моста

$$C_x = C_1 R_2 / R_3, R_x = R_1 R_3 / R_2, \tag{1.22}$$

$$tg\delta = \omega C_x R_x = \omega R_1 R_1, \tag{1.23}$$

Мост балансируют переменными резисторами R_1 и R_3. Шкалу резистора R_3 градуируют в единицах измеряемой емкости, отсчет сопротивления производят по шкале резистора R_1. Иногда применяют более сложные мосты, например, шестиплечие, мосты с сильной индуктивной связью, т.н. трансформаторные мосты. Такие мосты уравновешивают, изменяя параметры образцовых элементов или число витков трансформаторов. На основе трансформаторных мостов строят цифровые мосты с автоматической балансировкой. Метод трансформаторного моста реализован в выпускаемых промышленностью приборах, позволяющих измерять L, C, R и tg δ с основной погрешностью не менее 0,1 % на частоте 1 кГц в широком диапазоне значений измеряемых параметров. Диапазон измеряемых индуктивностей составляет 10^{-7}–10^{3} Гн, емкостей 10^{-2}–10^{8} пФ, сопротивлений 10^{-3}–10^{7} Ом.

1.3.2 Генераторный (резонансный) метод. Резонансные методы можно реализовать, вводя измеряемый элемент в контур автогенератора или в колебательный контур, слабо связанный с генератором. Эти методы иногда называют генераторным и контурным.

Прибор (рис. 1.15) включает в себя два генератора *G1* и *G2* высокой частоты. Генератор *G1* настроен на фиксированную частоту. Контур генератора *G2* можно перестраивать образцовым конденсатором *C*. Перед началом измерений частоту генератора *G2* устанавливают равной частоте генератора *G1*.

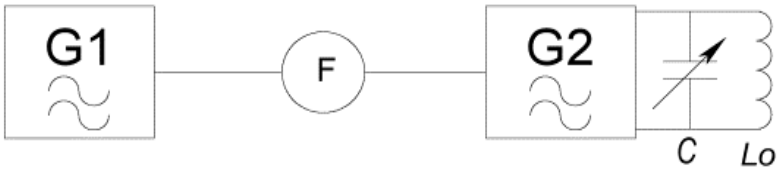


Рисунок 1.15 – Измерение емкости генераторным методам

Равенство частот генераторов контролируют индикатором равенства частот. При этом емкость $C_к$ контура генератора *G2* складывается из емкости C_1 образцового конденсатора и некоторой паразитной емкости $C_п$, включающей собственную ёмкость индуктивного компонента. Следовательно, $Cк=C_1+C_п$. Измеряемый конденсатор $C_х$ включают в контур генератора *G2* параллельно образцовому. Равенство частот нарушается, а затем его восстанавливают, уменьшая емкость образцового конденсатора до значения C_2. При этом $C_к = C_х + C_2 + C_п$. Сравнивая полученные соотношения, получаем $C_х=C_1-C_2$.

Погрешность измерения образцовой емкости $\Delta C_х=\Delta C_1-\Delta C_2$ зависит от неточности градуировки шкалы образцового конденсатора $\Delta C_{1ш}$, погрешности при считывании $\Delta C_{1с}$ и погрешности индикации равенства частот $\Delta C_{1р}$. На погрешность ΔC_2, помимо перечисленных причин, влияет относительная нестабильность частот генераторов за время между отсчетами C_1 и C_2, вызывающая погрешность $\Delta C_н$. Следовательно,

$$\Delta C_х = (\Delta C_{1ш} - \Delta C_{2ш}) + (\Delta C_{1с} - \Delta C_{2с}) + (\Delta C_{1р} - \Delta C_{2р}) + \Delta C_н .$$

Первые два слагаемых в правой части имеют систематический характер, остальные слагаемые имеют случайный характер, и практически их можно считать некоррелированными.

Случайная составляющая погрешности, обусловленная погрешностью при считывании, зависит от конструкции шкалы. Составляющая $\Delta C_{1р}$ определяется погрешностью Δf_1 фиксации равенства частот:

$$\frac{\Delta C_{1р}}{C_1} = -2\frac{\Delta f_1}{f_1},$$

где f_1 – частота генератора.

Максимальное значение Δf_1 при использовании индикатора равенства частот с электролучевой трубкой может составлять доли герца, а частота f выбирается достаточно высокой – до единиц мегагерц. В этих условиях относительная погрешность $\Delta C_{1р}/C_1$ весьма мала и составляет 10^{-6}–10^{-7}. Такого же порядка и погрешность $\Delta C_{2р}/C_2$. Погрешность $\Delta C_н$ обусловлена взаимной нестабильностью генераторов за время измерений.

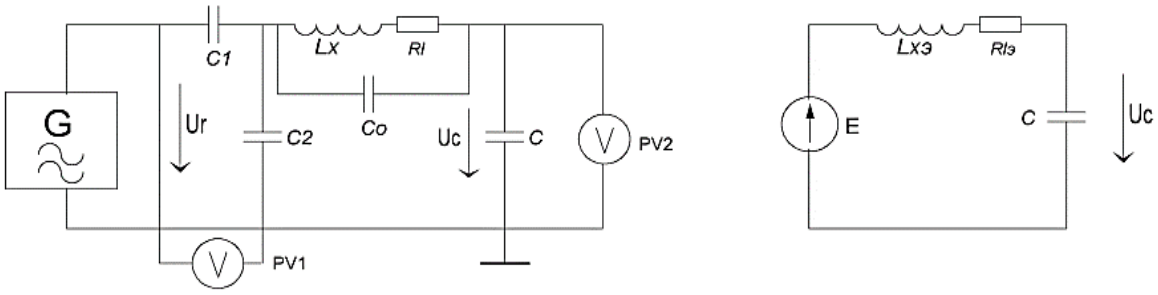
Обычно генераторы реализуют по идентичным схемам, и их взаимная нестабильность за время измерений, длящихся несколько секунд, не превышает 10^{-6}–10^{-7}. Следовательно, погрешность $\Delta C_н$ весьма мала. Погрешность измерения в основном определяется неточностью

градуировки шкалы конденсатора и погрешностью при считывании значений емкости.

Приборы рассмотренного типа характеризуются высокой разрешающей способностью, зависящей главным образом от конструкции образцового конденсатора, и могут применяться для измерения малых емкостей и индуктивностей на частотах до десятков мегагерц. Активную составляющую сопротивления, а следовательно, и потери прибор измерять не может.

1.3.3 Контурный метод. Этот метод положен в основу приборов, называемых измерителями добротности (куметрами). Измеритель добротности (рис. 1.16, а) содержит последовательный колебательный контур, состоящий из переменного образцового конденсатора С и образцового индуктивного компонента L и подключенного к генератору G.

Измеряемый компонент подключают вместо образцового, а конденсатор C_x подключают параллельно образцовому. Вводимое в контур напряжение фиксируют по показаниям вольтметра *PV1* уровня, а напряжение на образцовом конденсаторе измеряют вольтметром *PV2*. Большая часть измерений, проводимых куметром, основана на резонансных свойствах контура. Настройку в резонанс осуществляют, изменяя емкость образцового конденсатора.



а) б)

Рисунок 1.16 – Контуры для измерения добротности

Для работы электронного вольтметра, измеряющего напряжение на конденсаторе, необходимо резонансное напряжение в единицы вольт, следовательно, в контур с добротностью порядка 100 необходимо вводить напряжение около 0,01 В. Вольтметр уровня, контролирующий вводимое в контур напряжение, измеряет выходное напряжение генератора U_r, достаточно большое по амплитуде. Аттенюатор не только упрощает измерение вводимого в контур напряжения, но и уменьшает влияние выходного сопротивления генератора на контур. Так, в рассматриваемой схеме аттенюатор образован конденсаторами C_1 и C_2 ($C_1 \approx 0.01 C_2$), и независимо от выходного сопротивления генератора можно приближенно считать, что выходное сопротивление аттенюатора в основном определяется емкостью C_2. Иногда применяют аттенюаторы и других типов, например индуктивные.

Измерительный контур можно представить в виде, показанном на рис. 1.16, б, где E – эквивалентная ЭДС, вводимая в контур. Выходное сопротивление источника ЭДС очень мало по сравнению с сопротивлением контура, и поэтому его можно не учитывать.

Измерение добротности индуктивных компонентов. Индуктивность L_x вводят в контур вместо образцовой и, изменяя C, настраивают контур в резонанс. Заменив, например, индуктивный компонент эквивалентной схемой (рис. 1.16, б), если $r_c << r_{Lэ}$, то при резонансе ток контура $I_p \approx E/r_{Lэ}$, а напряжение на образцовом конденсаторе:

$$U_{C_p} = \frac{E}{r_L}\frac{1}{\omega C_p},$$

где C_p — резонансное значение образцовой емкости. Значение добротности индуктивного компонента:

$$Q_L = \frac{U_{Cp}}{E}\frac{1}{r_L \omega C_p} \qquad (1.24)$$

отсчитывают по шкале вольтметра, поскольку амплитуда вводимого в контур напряжения постоянна.

На результат измерения добротности влияют потери в элементах контура. Полное сопротивление контура:

$$r_k = r_{Lэ} + r_C, \qquad (1.25)$$

где r_C — сопротивление образцового конденсатора, учитывающее потери в нем и шунтирующее действие вольтметра.

Поделив (1.25) на резонансное сопротивление $1/\omega C_p$, получим

$$\frac{1}{Q} = \frac{1}{Q_L} + \frac{1}{Q_C}.$$

Образцовые конденсаторы имеют добротность порядка 10^4, поэтому приближенно можно считать $Q \approx Q_L$. Самая существенная погрешность измерения Q_L возникает из-за неточности измерения напряжений. Основная погрешность для выпускаемых промышленностью измерителей добротности составляет 3–15 %.

Добротность можно измерять, используя метод вариации емкости. Для этого измеряют значения C_1 и C_2 емкости C, взятые при напряжении на образцовом конденсаторе $0{,}707U_{Cp}$. Значения образцовой емкости C_1 и C_2 отсчитывают со шкалы. Погрешность измерения Q_L в основном обусловлена погрешностями измерения C_1 и C_2.

Измерение индуктивности и собственной емкости. Измеряемую индуктивность вводят в контур последовательно с образцовым конденсатором. После настройки в резонанс эквивалентную индуктивность рассчитывают по формуле $L_э = 1/C_p\omega^2_p$.

На фиксированных частотах генератора индуктивность $L_э$ однозначно зависит от C. Поэтому со шкалой образцового конденсатора совмещают шкалу индуктивностей, справедливую для ряда фиксированных частот,

указанных в паспортных данных прибора. Погрешность измерения индуктивности $\delta Lэ = -\delta C_p - 2\delta\omega_p$.

Погрешность установки частоты для промышленных образцов измерителей добротности составляет не менее 1 %. Неточность определения резонансной емкости зависит от погрешности градуировки шкалы, погрешности считывания и погрешности фиксации экстремального уровня напряжения на конденсаторе.

Измерение сопротивления. Методика измерения сопротивления основана на измерении изменения добротности контура при введении в него неизвестного сопротивления. Если характеристическое сопротивление контура $x_p >> r_x$, то измеряемое сопротивление вводят в контур последовательно с его элементами. Полное сопротивление контура при резонансе $r=r_C+r_L + r_x$. Разделив обе части равенства на x_p, получается:

$$\frac{1}{Q_2} = \frac{1}{Q_1} + \frac{r_x}{x_p},$$

где Q_1 и Q_2 – значения добротности образцового контура, измеренные до и после введения r_x. Отсюда

$$r_x = \frac{Q_1 - Q_2}{Q_1 Q_2} \cdot \frac{1}{2\pi f (C + C_{0)}}, \qquad (1.26)$$

где f – частота генератора.

Для промышленных образцов измерителей добротности, перекрывающих частотный диапазон 1 кГц...200 МГц, основная погрешность измерения сопротивления составляет 5–10%.

Преобразование измеряемого сопротивления в напряжение. К этому методу прибегают при построении простых измерителей активного сопротивления. В измерителе больших сопротивлений измеряемый резистор R_x образует вместе с образцовым резистором R_0 делитель, к которому подводят напряжение питания от генератора G переменного напряжения низкой частоты (рис. 1.17).

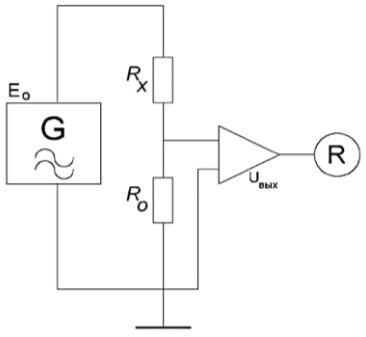


Рисунок 1.17 – Преобразование измеряемого сопротивления в напряжение

Напряжение на образцовом резисторе усиливают усилителем, результат измерений отсчитывают по прибору, в качестве которого обычно

используют магнитоэлектрический микроамперметр. Входное сопротивление усилителя $R_{вх} >> R_0$, тогда

$$R_x = [(KE_0 / U_{вых}) - 1]R_0, \quad (1.27)$$

где K – коэффициент передачи усилителя.

Схему применяют для измерений больших сопротивлений $R_x > R_0$. Выпускаемые промышленностью приборы измеряют сопротивления $2\text{-}10^8$ Ом с основной погрешностью 1,5 %.

Малые сопротивления в пределах 10^{-4}–10^2 Ом измеряются также с помощью схемы, показанной на рисунке 1.17, но сопротивления делителя меняются местами.

Измеряемое сопротивление $R_x = R_0 / [(KU_0 / U_{вых}) - (R_0 / R_{вх}) - 1]$.

При $R_{вх} >> R_0$ $R_x = R_0 / [(RU_0 / U_{вых}) - 1]$.

Преобразование полного сопротивления в напряжение с помощью операционного усилителя. Полное сопротивление **Z**x можно измерить по отношению падения напряжения $\mathbf{U}_x$ на нем к току **I**x: $\mathbf{U}_R = \mathbf{I}x R_0$. На практике удобнее поочередно измерять два однородных параметра – напряжения $\mathbf{U}_x$ и $\mathbf{U}_R = \mathbf{I}_x R_0$ на образцовом резисторе R_0, через который протекает ток $\mathbf{I}_x$.

Возможная структурная схема прибора показана на рисунке 1.18. Измеряемое сопротивление **Z***x* образует последовательную цепь с резистором R и образцовым резистором R_0, включенным в цепь отрицательной обратной связи операционного усилителя *A1*. Гармоническое напряжение питания формируется с помощью ФНЧ из меандра, вырабатываемого генератором G.

При большом коэффициенте передачи усилителя можно считать, что его входное напряжение практически равно нулю. Если входной ток усилителя пренебрежимо мал по сравнению с током I цепи, то можно записать $\mathbf{U}_{Zx} = \mathbf{I}\mathbf{Z}_x = IZ_x e^{j(\varphi_x + \varphi_I)}$, где φ_x и φ_I – фазовые углы комплексного сопротивления $\mathbf{Z}_x$ и тока **I**. Исключив ток, получим

$$\mathbf{Z}_x = -R_0 \mathbf{U}_{zx} / \mathbf{U}_{R0}, \quad (1.28)$$

откуда следует, что полное сопротивление можно определить по отношению напряжений, причем результат не зависит от амплитуды тока и от сопротивления *R*.

Процесс измерений состоит из четырех тактов, осуществляемых при положениях 1,1; 1,2; 2,1; 2,2, управляемых микропроцессорной системой переключателей SA1 и SA2.

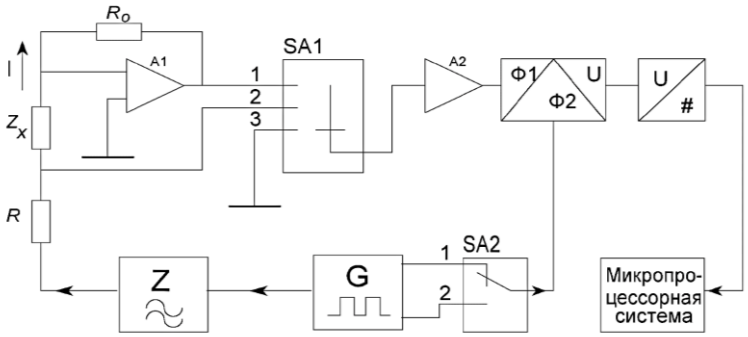


Рисунок 1.18 – Микропроцессорная система.

В первом такте ко входу фазового детектора подводят усиленное напряжение $KU_{R0} = -KIR_0 e^{j\varphi_l}$, где K – коэффициент усиления усилителя переменного напряжения А2. В качестве опорного используют напряжение в форме меандра с первого выхода генератора G с фазовым углом φ_r. Выходное напряжение фазового детектора $U_{\partial 1} = -K_1 IR_0 \cos(\varphi_l - \varphi_r)$, K_1– общий коэффициент передачи усилителя А2 и фазового детектора. С помощью АЦП напряжение $U_{\partial 1}$ преобразуют в число $n_1 = -K_2 IR_0 \cos(\varphi_l - \varphi_r)$, где K_2 – коэффициент передачи, а затем фиксируют в ОЗУ. Подобным же образом преобразуют результаты измерений и в остальных тактах. Во втором такте используют опорное напряжение со второго выхода генератора, сдвинутое на $\pi/2$ по сравнению с напряжением с первого выхода. Следовательно, $n_2 = -K_2 IR_0 \sin(\varphi_l - \varphi_r)$. В третьем и четвертом тактах также фиксируют в ОЗУ.

$$n_3 = K_2 IZ_x \cos(\varphi_l - \varphi_r + \varphi_x),$$

$$n_4 = K_2 IZ_x \sin(\varphi_l - \varphi_r + \varphi_x).$$

По окончании последнего такта микропроцессорная система (МПС) вычисляет параметры измеренного комплексного сопротивления. Например, для последовательной схемы замещения $\mathbf{Z}_x = R_x + jX_x$:

$$R_x = Z_x \cos\varphi_x = \frac{n_1 n_3 - n_2 n_4}{n_1^2 + n_2^2};$$

$$X_x = Z_x \sin\varphi_x = \frac{n_2 n_3 - n_1 n_4}{n_1^2 + n_2^2}.$$

В справедливости приведенных соотношений можно убедиться, подставив в них значения n_1-n_4. Используя соответствующие формулы, микропроцессорная система может рассчитать любые параметры комплексного сопротивления: емкость, индуктивность, добротность, тангенс угла потерь, постоянную времени, модуль и фазу комплексного сопротивления, а также параметры параллельной схемы замещения.

Применение микропроцессорной системы позволит упростить процесс измерений и устранить некоторые погрешности. Так, в

рассмотренном устройстве микропроцессор автоматически меняет сопротивление R_0 и коэффициент усиления K, что необходимо для выбора нужного диапазона измерений.

Предусмотрена компенсация ухода нуля фазового детектора и АЦП. Для этого каждое измерение дополняют пятым тактом, в котором с помощью переключателя SA1 (положение 3) заземляют вход фазового детектора и получают число n_5. Затем его вычитают из чисел n_1-n_4 и рассчитывают параметры сопротивления. Лучшие образцы приборов подобного типа работают в частотном диапазоне 100 Гц... 10 МГц и обеспечивают погрешность 0,1 % при времени одиночного измерения 0,14...0,65 с в широком диапазоне измеряемых параметров.

Таким образом, измерители с преобразованием измеряемого сопротивления в напряжение имеют погрешность, соизмеримую с погрешностью мостовых схем, но значительно проще по структуре. Очень важно, что такие измерители содержат образцовые элементы только в виде резисторов, в то время как в мостовых измерителях необходимы и образцовые конденсаторы. Для реализации метода преобразования необходима микропроцессорная система, поэтому приборы подобного типа появились значительно позже, чем мосты и измерители резонансного типа, и считаются наиболее перспективными приборами [9].

Существуют три основных метода измерений: мостовой, резонансный и преобразования параметра во временной интервал или напряжение. При мостовом методе измеряемые полные сопротивления сравнивают с образцовыми сопротивлениями, отдельно регулируя действительную и мнимую части последних. Для сравнения используют четырехплечие, шестиплечие и трансформаторные мосты. Из-за сильных паразитных связей между элементами измерительных мостов такие приборы применяют преимущественно на низких частотах, наприме, 1кГц. Основная погрешность составляет около 0,1 %. При резонансном методе измеряемые реактивные параметры замещают образцовой емкостью, а потери определяют расчетным путем по изменению добротности измерительного контура. Измерители добротности позволяют измерять параметры L, C, R в широком диапазоне частот от десятка килогерц до долей гигагерц со значительной основной погрешностью 2 – 10 % и более.

Метод преобразования измеряемого параметра в напряжение в простейшем случае реализуют с помощью цепи, состоящей из измеряемого и образцового резистора, подключенной к источнику образцового напряжения. Измеряемое сопротивление оценивают по падению напряжения на одном из резисторов. В микропроцессорных измерителях может быть получена погрешность 0,1 % в диапазоне частот 100 Гц... 10 МГц [13].

1.4 Способ определения параметров индуктивных компонентов и его реализация на основе модуля Л Кард Е 502

Как видно из рис. 1.18, прибор для реализации способа преобразования полного сопротивления в напряжение состоит из таких основных узлов, как генератор, АЦП, фазовый детектор и МПС. Все эти узлы могут быть заменены устройством сбора данных Е502 производства «Л Кард». Устройство сбора данных Л Кард Е502 представляет собой универсальный 16-битный модуль ввода/вывода до 32 аналоговых и 17 цифровых сигналов в компьютер через интерфейсы USB 2.0 (high-speed) и Ethernet (100 Мбит) с частотой преобразования до 2 МГц и возможностью их цифровой обработки в реальном времени. Кроме того, ООО «Л Кард» приветствует интеграцию своих модулей в пользовательские системы. Структурная схема предлагаемого прибора показана на рис. 1.19.

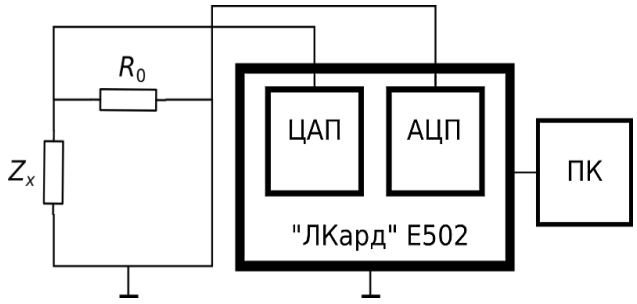


Рисунок 1.19 – Реализация измерителя параметров индуктивных компонентов магнитометров на базе платы сбора данных Л Кард Е502

Так как амплитуда напряжения ЦАП задается программно и АЦП имеет программируемые пределы измерения, необходимость в операционном усилителе отпадает, и формула (1.28) в связи с этим приобретает вид:

$$\mathbf{Z}_x = R_0(\mathbf{U}_{\mathbf{zx}} - \mathbf{U}_{\mathbf{R0}})/\mathbf{U}_{\mathbf{R0}}. \qquad (1.29)$$

Напряжение $\mathbf{U_{Zx}}$ генерируется ЦАП устройства в режиме синхронного циклического вывода с аппаратно заданной частотой дискретизации, равной 1МГц, амплитуда, частота и фаза при этом известны, т.к. задаются программно, так что:

$$U_{Zx} = A_0 \sin(2\pi\nu t_0 + \varphi_0), \qquad (1.30)$$

где A_0 – амплитуда**,** ν – частота, и φ_0 – фаза $\mathbf{U_{Zx}}$, $t_0=0, \Delta_0, 2\Delta_0, 3\Delta_0, \ldots$.

1.4.1 Фазовый детектор. Возникает задача определения амплитуды и фазы измеренного с помощью АПЦ напряжения U_{R0}, которое можно представить в виде модели*:*

$$\hat{U}_{R0}(t_1) = A_1 \sin(2\pi\nu t_1 + \varphi_1), \qquad (1.31)$$

где A_1 – амплитуда**,** ν – частота, и φ_1 – фаза напряжения $\hat{U}_{R0}$, $t_1=0, \Delta_1, 2\Delta_1, 3\Delta_1, \ldots, p\Delta_1$, где Δ_1 известная частота дискретизации АЦП.

Среднеквадратическая ошибка модели равна:

$$E_p = \frac{1}{2}\left(U_{R0,p} - \hat{U}_{R0,p}\right)^2 = \frac{1}{2}e_p^2, \tag{1.32}$$

где $p=1, 2, \ldots, n$ – количество измерений U_{R0}.

Для нахождения A_1 и φ_1 таких, чтобы ошибка (1.32) была минимальна, нужно ввести определение суммарной среднеквадратической ошибки:

$$\Omega_n = \frac{1}{n}\sum_{p=1}^{n} E_p = \frac{1}{2n}\sum_{p=1}^{n} e_p^2 = \frac{1}{2n}\sum_{p=1}^{n}\left(U_{R0} - \hat{U}_{R0}\right)^2. \tag{1.33}$$

Тогда задача определения амплитуды и фазы напряжения записывается как:

$$\Omega_n \to \min_w, \tag{1.34}$$

где $w=\{A_1$ и $\varphi_1\}$ – параметры модели (1.31) и представляет собой типичную задачу многомерной оптимизации, при этом параметры модели находятся по итерационной формуле:

$$w_{k+1} = w_k + \alpha_k \rho_k, \tag{1.35}$$

где w_k – параметры модели на предыдущей итерации k, w_{k+1} – обновленные параметры модели, α_k – длина шага, $\rho_k = -B_k^{-1}\frac{\partial L_n}{\partial w_k}$, где B_k – приближенное значение Гессиана. Параметры α_k, B_k – находятся на каждой итерации по методу Бройдена-Флетчера-Гольдфарба-Шанно.

Для использования формулы (1.35) частные производные суммарной среднеквадратической ошибки:

$$\begin{aligned}\frac{\partial L}{\partial w} &= \frac{1}{2n}\sum_{p=1}^{n}\frac{\partial E_p}{\partial w} = \frac{1}{2n}\sum_{p=1}^{n}\frac{\partial E_p}{\partial \hat{U}_{R0,p}}\frac{\partial \hat{U}_{R0,p}}{\partial w} =, \\ &= \frac{1}{n}\sum_{p=1}^{n} -\left(U_{R0,p} - \hat{U}_{R0,p}\right)\frac{\partial \hat{U}_{R0,p}}{\partial w} = \frac{1}{n}\sum_{p=1}^{n} -e_p\frac{\partial \hat{U}_{R0,p}}{\partial w}\end{aligned} \tag{1.36}$$

Частная производная по амплитуде и фазе равна:

$$\frac{\partial \hat{U}_{R0,p}}{\partial A_1} = \sin(2\pi\nu t_1 + \varphi_1),$$

$$\frac{\partial \hat{U}_{R0,p}}{\partial \varphi_1} = A_1\cos(2\pi\nu t_1 + \varphi_1).$$

Вычисления продолжаются до достижения заданной точности определения параметров A_1 и φ_1, равной $5\cdot10^{-4}$ B, то есть до тех пор, пока два последовательно полученных по итерационной формуле (1.35) значения амплитуды и фазы не будут отличаться друг от друга менее чем

на $5 \cdot 10^{-4}$ *В*. На рисунке 1.20 показан пример измерения значений U_{R0} и вычисленные с помощью модели (1.31) значения $\hat{U}_{R0}$ на частоте $\nu = 2500\,\text{Гц}$.

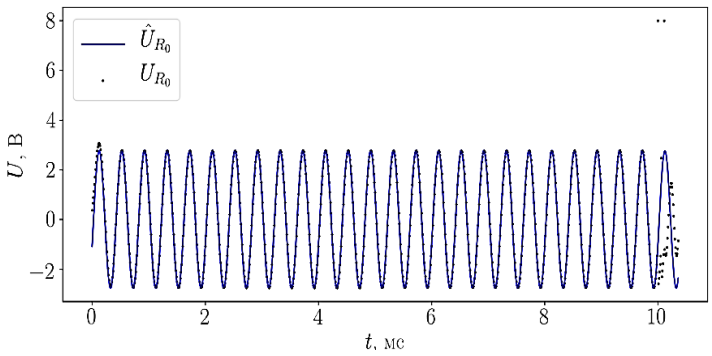


Рисунок 1.20 – Пример измерения значений U_{R0}

Вычисленные по формуле (1.35) значения фазы и амплитуды оказались равны: φ_1=0.403 *рад*, A_1=2.759 *В*. Из рисунка 1.20 можно видеть, что предложенный способ определения амплитуды и фазы с помощью оптимизационного метода Бройдена–Флетчера–Гольдфарба–Шанно хорошо справляется с различного рода помехами, вызванными переходными процессами (рис. 1.20, слева), электромагнитными возмущениями (рис. 1.20, справа), а также возможными потерями данных в канале связи (рис. 1.20, справа вверху). Кроме того, предложенный метод позволяет легко контролировать точность определения параметров за счет простого изменения условия окончания вычислений по формуле (1.35). Найдя с помощью формулы (1.35) значения A_1 и φ_1, можно вычислить $\mathbf{U}_{R0} = A_1 e^{i\varphi_1}$, и затем по формуле (1.29) найти $\mathbf{Z}_x$. Таким образом, для любой заданной частоты ν может быть найдено соответствующее значение $\mathbf{Z}_x$.

Способ определения параметров индуктивных компонентов магнитометров. Комплексное сопротивление модели, показанной на рисунке 1.19, равно:

$$\hat{\mathbf{Z}}_x(\omega) = \frac{(i\omega L + R_L)\left(\frac{1}{i\omega C} + R_C\right)}{i\omega L + R_L + \frac{1}{i\omega C} + R_C}, \tag{1.37}$$

$$\mathrm{Re}(\hat{\mathbf{Z}}_x) = \frac{(C^2L^2R_C)\omega^4 + (C^2R_C^2R_L + C^2R_CR_L^2)\omega^2 + R_L}{(C^2L^2)\omega^4 + (C^2R_C^2 + 2C^2R_CR_L + C^2R_L^2 - 2LC)\omega^2 + 1},$$

$$\mathrm{Im}(\hat{\mathbf{Z}}_x) = \frac{(C^2LR_C^2 - CL^2)\omega^3 + (L - CR_L^2)\omega}{(C^2L^2)\omega^4 + (C^2R_C^2 + 2C^2R_CR_L + C^2R_L^2 - 2LC)\omega^2 + 1},$$

где $\omega = 2\pi\nu$.

Среднеквадратическая ошибка модели равна:

$$E_s = \frac{1}{2}\left(\left[\mathrm{Re}(\mathbf{Z}_{x,s}) - \mathrm{Re}(\hat{\mathbf{Z}}_{x,s})\right]^2 + \left[\mathrm{Im}(\mathbf{Z}_{x,s}) - \mathrm{Im}(\hat{\mathbf{Z}}_{x,s})\right]^2\right) = \tag{1.38}$$

$$= \frac{1}{2}\left(\mathrm{Re}(\mathbf{e}_s)^2 + \mathrm{Im}(\mathbf{e}_s)^2\right)$$

где $s=1, 2, \ldots, m$ – количество измерений $\mathbf{Z}_x$.

$$L_m = \frac{1}{m}\sum_{s=1}^{m} E_s = \frac{1}{2m}\sum_{s=1}^{m}\left(\mathrm{Re}(\mathbf{e}_s)^2 + \mathrm{Im}(\mathbf{e}_s)^2\right) = \tag{1.39}$$

$$\frac{1}{2m}\sum_{s=1}^{m}\left(\left[\mathrm{Re}(\mathbf{Z}_{x,s}) - \mathrm{Re}(\hat{\mathbf{Z}}_{x,s})\right]^2 + \left[\mathrm{Im}(\mathbf{Z}_{x,s}) - \mathrm{Im}(\hat{\mathbf{Z}}_{x,s})\right]^2\right).$$

Тогда задача определения параметров индуктивного компонента записывается как:

$$L_m \to \min_w, \tag{1.40}$$

где $w=\{L, C, R_c, R_l\}$ – параметры модели (1.37), и представляет собой также типичную задачу многомерной оптимизации, при этом параметры модели находятся по той же итерационной формуле (1.35). При использовании формулы (1.35) частные производные суммарной среднеквадратической ошибки в этом случае равны:

$$\frac{\partial L}{\partial v} = \frac{1}{2m}\sum_{s=1}^{m}\frac{\partial E_s}{\partial v} = \frac{1}{2m}\sum_{s=1}^{m}\left(\frac{\partial E_s}{\partial \mathrm{Re}(\hat{\mathbf{Z}}_{x,s})}\frac{\partial \mathrm{Re}(\hat{\mathbf{Z}}_{x,s})}{\partial v} + \frac{\partial E_s}{\partial \mathrm{Im}(\hat{\mathbf{Z}}_{x,s})}\frac{\partial \mathrm{Im}(\hat{\mathbf{Z}}_{x,s})}{\partial v}\right) =$$

$$= \frac{1}{m}\sum_{s=1}^{m} -\left[\left(\mathrm{Re}(\mathbf{Z}_{x,s}) - \mathrm{Re}(\hat{\mathbf{Z}}_{x,s})\right)\frac{\partial \mathrm{Re}(\hat{\mathbf{Z}}_{x,s})}{\partial v} + \left(\mathrm{Im}(\mathbf{Z}_{x,s}) - \mathrm{Im}(\hat{\mathbf{Z}}_{x,s})\right)\frac{\partial \mathrm{Im}(\hat{\mathbf{Z}}_{x,s})}{\partial v}\right] = \tag{1.41}$$

$$= \frac{1}{m}\sum_{s=1}^{m} -\left(\mathrm{Re}(\mathbf{e}_s)\frac{\partial \mathrm{Re}(\hat{\mathbf{Z}}_{x,s})}{\partial v} + \mathrm{Im}(\mathbf{e}_s)\frac{\partial \mathrm{Im}(\hat{\mathbf{Z}}_{x,s})}{\partial v}\right).$$

И, в частности:

$$\frac{\partial \mathrm{Re}(\hat{\mathbf{Z}})}{\partial L} = \frac{2C\omega^2\ C^3 L R_c^{\,3}\omega^4 + C^3 L R_c^{\,2} R_l \omega^4 - C^2 L^2 R_c \omega^4 + C^2 R_c^{\,2} R_l \omega^2 + C^2 R_c R_l^{\,2}\omega^2 + CLR_c\omega^2 - CLR_l\omega^2 + R_l}{C^2 L^2\omega^4 + C^2 R_c^{\,2}\omega^2 + 2C^2 R_c R_l \omega^2 + C^2 R_l^{\,2}\omega^2 - 2CL\omega^2 + 1^{\,2}}$$

$$\frac{\partial \mathrm{Re}(\hat{\mathbf{Z}})}{\partial C} = -\frac{2\omega^2\ C^2 L^3 \mathrm{R_c}\omega^4 + C^2 L \mathrm{R_c}^{\,2}\mathrm{R_l}\omega^2 + C^2 L \mathrm{R_c}\mathrm{R_l}^{\,2}\omega^2 - CL^2\mathrm{R_c}\omega^2 + CL^2\mathrm{R_l}\omega^2 + C\mathrm{R_c}\mathrm{R_l}^{\,2} + C\mathrm{R_l}^{\,3} - L\mathrm{R_l}}{C^2 L^2\omega^4 + C^2\mathrm{R_c}^{\,2}\omega^2 + 2C^2\mathrm{R_c}\mathrm{R_l}\omega^2 + C^2\mathrm{R_l}^{\,2}\omega^2 - 2CL\omega^2 + 1^{\,2}},$$

$$\frac{\partial \mathrm{Re}(\hat{\mathbf{Z}})}{\partial R_c} = \frac{\omega^2\ C^2\mathrm{R_l}^{\,2} + 2\mathrm{R_c}C^2\mathrm{R_l}\ + C^2L^2\omega^4}{\omega^2\ C^2\mathrm{R_c}^{\,2} + 2C^2\mathrm{R_c}\mathrm{R_l} + C^2\mathrm{R_l}^{\,2} - 2LC\ + C^2L^2\omega^4 + 1} -$$

$$-\frac{\omega^2\ 2C^2\mathrm{R_c} + 2C^2\mathrm{R_l}\ \ \mathrm{R_l} + \omega^2\ C^2\mathrm{R_c}^{\,2}\mathrm{R_l} + C^2\mathrm{R_c}\mathrm{R_l}^{\,2}\ + C^2L^2\mathrm{R_c}\omega^4}{\omega^2\ C^2\mathrm{R_c}^{\,2} + 2C^2\mathrm{R_c}\mathrm{R_l} + C^2\mathrm{R_l}^{\,2} - 2LC\ + C^2L^2\omega^4 + 1^{\,2}},$$

$$\frac{\partial \mathrm{Re}(\hat{\mathbf{Z}})}{\partial R_l} = \frac{-C^4 L^2 \mathrm{R_c}^2 \omega^6 + C^4 \mathrm{R_c}^4 \omega^4 + 2C^4 \mathrm{R_c}^3 \mathrm{R_l} \omega^4 + C^4 \mathrm{R_c}^2 \mathrm{R_l}^2 \omega^4 - 2C^3 L \mathrm{R_c}^2 \omega^4}{C^2 L^2 w^4 + C^2 \mathrm{R_c}^2 w^2 + 2C^2 \mathrm{R_c} \mathrm{R_l} w^2 + C^2 \mathrm{R_l}^2 w^2 - 2CLw^2 + 1^2} +$$

$$+ \frac{-4C^3 L \mathrm{R_c} \mathrm{R_l} \omega^4 + C^2 L^2 \omega^4 + 2C^2 \mathrm{R_c}^2 \omega^2 + 2C^2 \mathrm{R_c} \mathrm{R_l} \omega^2 - C^2 \mathrm{R_l}^2 \omega^2 - 2CL\omega^2 + 1}{C^2 L^2 \omega^4 + C^2 \mathrm{R_c}^2 \omega^2 + 2C^2 \mathrm{R_c} \mathrm{R_l} \omega^2 + C^2 \mathrm{R_l}^2 \omega^2 - 2CL\omega^2 + 1^2},$$

$$\frac{\partial \mathrm{Im}(\hat{\mathbf{Z}})}{\partial L} = \frac{C^2 \mathrm{R_c}^2 - 2CL\ \omega^3 + \omega}{\omega^2\ C^2 \mathrm{R_c}^2 + 2C^2 \mathrm{R_c} \mathrm{R_l} + C^2 \mathrm{R_l}^2 - 2LC\ + C^2 L^2 \omega^4 + 1} + \frac{\omega\ L - C\mathrm{R_l}^2\ - \omega^3\ CL^2 - C^2 L \mathrm{R_c}^2\quad 2C\omega^2 - 2C^2 L\omega^4}{\omega^2\ C^2 \mathrm{R_c}^2 + 2C^2 \mathrm{R_c} \mathrm{R_l} + C^2 \mathrm{R_l}^2 - 2LC\ + C^2 L^2 \omega^4 + 1^2},$$

$$\frac{\partial \mathrm{Im}(\hat{\mathbf{Z}})}{\partial C} = \frac{\omega\ C^2 L^4 \omega^6 - C^2 L^2 \mathrm{R_c}^2 \omega^4 + 2C^2 L^2 \mathrm{R_c} \mathrm{R_l} \omega^4 + 2C^2 L^2 \mathrm{R_l}^2 \omega^4 + C^2 \mathrm{R_c}^2 \mathrm{R_l}^2 \omega^2}{C^2 L^2 \omega^4 + C^2 \mathrm{R_c}^2 \omega^2 + 2C^2 \mathrm{R_c} \mathrm{R_l} \omega^2 + C^2 \mathrm{R_l}^2 \omega^2 - 2CL\omega^2 + 1^2} +$$

$$+ \frac{\omega(2C^2 \mathrm{R_c} \mathrm{R_l}^3 \omega^2 + C^2 \mathrm{R_l}^4 \omega^2 - 2CL^3 \omega^4 - 4CL \mathrm{R_c} \mathrm{R_l} \omega^2 - 2CL \mathrm{R_l}^2 \omega^2 + L^2 \omega^2 - \mathrm{R_l}^2)}{C^2 L^2 \omega^4 + C^2 \mathrm{R_c}^2 \omega^2 + 2C^2 \mathrm{R_c} \mathrm{R_l} \omega^2 + C^2 \mathrm{R_l}^2 \omega^2 - 2CL\omega^2 + 1^2},$$

$$\frac{\partial \mathrm{Im}(\hat{\mathbf{Z}})}{\partial R_c} = \frac{2C^2\omega^3\ C^2 L^3 \mathrm{R_c} \omega^4 + C^2 L \mathrm{R_c}^2 \mathrm{R_l} \omega^2 + C^2 L \mathrm{R_c} \mathrm{R_l}^2 \omega^2 - CL^2 \mathrm{R_c} \omega^2 + CL^2 \mathrm{R_l} \omega^2 + C\mathrm{R_c} \mathrm{R_l}^2 + C\mathrm{R_l}^3 - L\mathrm{R_l}}{C^2 L^2 \omega^4 + C^2 \mathrm{R_c}^2 \omega^2 + 2C^2 \mathrm{R_c} \mathrm{R_l} \omega^2 + C^2 \mathrm{R_l}^2 \omega^2 - 2CL\omega^2 + 1^2},$$

$$\frac{\partial \mathrm{Im}(\hat{\mathbf{Z}})}{\partial R_l} = -\frac{2C\omega\ C^3 L \mathrm{R_c}^3 \omega^4 + C^3 L \mathrm{R_c}^2 \mathrm{R_l} \omega^4 - C^2 L^2 \mathrm{R} c w^4 + C^2 \mathrm{R_c}^2 \mathrm{R_l} \omega^2 + C^2 \mathrm{R_c} \mathrm{R_l}^2 w^2 + CL \mathrm{R_c} \omega^2 - CL \mathrm{R_l} \omega^2 + \mathrm{R_l}}{C^2 L^2 \omega^4 + C^2 \mathrm{R_c}^2 \omega^2 + 2C^2 \mathrm{R_c} \mathrm{R_l} \omega^2 + C^2 \mathrm{R_l}^2 \omega^2 - 2CL\omega^2 + 1^2}.$$

Таким образом, параметры индуктивного компонента вычисляются по множеству измерений полного комплексного сопротивления на различных частотах и находятся исходя из наилучшего соответствия измеренным значениям в смысле минимума суммарного квадратического отклонения, благодаря чему нивелируется воздействие случайных всплесков ЭДС в измеряемом индуктивном компоненте.

На рисунке 1.21, 1.22 показан пример определения параметров индуктивного компонента по четырехэлементной схеме замещения (см. рис. 1.12) на частотах 50–200 Гц. Среднеквадратическая ошибка, вычисленная по формуле (34) E_s = 71,595 Ом^2

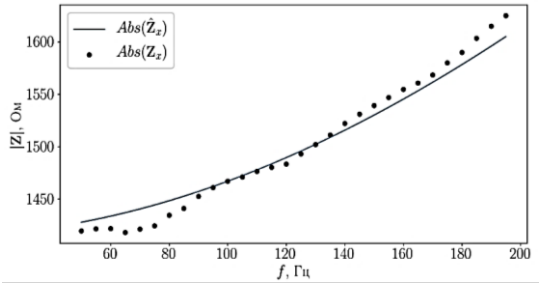


Рисунок 1.21 – Зависимость модуля полного сопротивления от частоты (50–200 Гц)

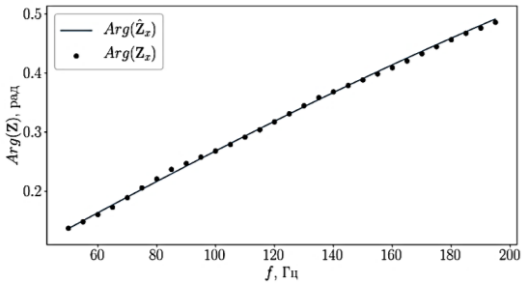


Рисунок 1.22 – Зависимость фазы полного сопротивления от частоты (50–200 Гц)

Вычисленные с помощью формулы (1.35) параметры индуктивности равны:

C = 1,18126 10^{-10} *Ф*, L = 0,618958 *Гн*, Rc = 8.2737 Ом, *Rl* = 1415,52 *Ом*.

Активное сопротивление, измеренное вольтметром В7-22А *Rl* = 1381 *Ом*, т.е. ошибка составляет не более 2,5%. Несмотря на то, что полученная

ошибка измерения достаточно велика, предложенный способ определения параметров индуктивного компонента магнитометров позволяет определить не только активное сопротивление, но и индуктивность, паразитную емкость, а также ее сопротивление переменному току, в то время как серийные приборы, предназначенные для этой цели, например цифровой измеритель L,C,R E7-12, не позволяют этого сделать ввиду того, что индуктивный компонент магнитометра с большим числом витков является активным элементом, в котором генерируется ЭДС индукции, оказывающая влияние на работу приборов [11].

1.4.2. Определение параметров индуктивного компонента магнитометра с помощью генетического алгоритма. Описанный в предыдущем разделе способ определения параметров индуктивного компонента магнитометра на основе алгоритма Бройдена–Флетчера–Гольдфарба–Шанно и данных о первых производных оптимизируемой функции (1.41) обладает одним существенным недостатком. В связи с тем, что функция (1.39) нелинейная, возникает проблема локальных минимумов, когда за верное решение алгоритмом принимается не глобальный, а локальный минимум.

Для решения этой проблемы предлагается для определения параметров индуктивных компонентов магнитометров использовать генетический алгоритм. Генетический алгоритм оперирует набором особей (так называемой популяцией). Они представляют собой записи (хромосомы), кодирующие одно из решений оптимизационной задачи (1.40). Этим генетический алгоритм отличается от алгоритма Бройдена – Флетчера – Гольдфарба – Шанно и других подобных алгоритмов оптимизации, которые работают только с одним решением, улучшая его.

В рассматриваемом случае в терминах практики применения генетических алгоритмов каждая особь кодируется одной хромосомой, содержащей 4 вещественных параметра $x = \{L, C, R_c, R_l\}$, а функция потерь имеет вид:

$$W(x) = \frac{1}{2m}\sum_{s=1}^{m}\left|\mathbf{Z}_{x,s} - \hat{\mathbf{Z}}_{x,s}\right|.$$

Задача заключается в том, что в пространстве поиска $\mathbf{X}$ требуется найти

$$x^* = \arg\min_{x \in X} W(x),$$

где $\hat{\mathbf{Z}}_x$ вычисляется по формуле (1.37).

Используемый генетический алгоритм детально выглядит следующим образом:

1. Случайным образом генерируется конечный набор пробных решений: $p^1 = \{p_1^1 \ldots p_n^1\}, p_i^1 \in \mathbf{X}$ (первое поколение, $n=200$ – размер популяции).

2. Выполняется оценка приспособленности текущего поколения: $F^k = \{f_1^k ... f_n^k\}, f_i^k = W(p_i^k)$. При этом выполняется нормировка к виду: $F^k{}' = \{f_1^k{}' ... f_n^k{}'\}, f_i^k{}' = (f_i^k - f_0)/(f_1 - f_0)$, где f_1 и f_0, соответственно, лучший и худший показатель в текущей популяции.
3. Выход, если выполняется критерий останова, иначе переход к следующему пункту.
4. Генерация нового поколения посредством операторов селекции *S*, скрещивания *C* и мутации *M:* $p^{k+1}=M \cdot C \cdot S(p^k, F^k)$ и переход к пункту 2 [14].

В процессе селекции отбирают только небольшое количество, а именно 10 % лучших пробных решений, а другие далее не используются, при этом особи с одинаковым набором генов отбрасываются. Скрещивание выполняется над половиной лучших особей попарно. Результатом являются также две особи с компонентами, взятыми от родителей $p_i^k = \{a_1, a_2, ..., a_n\}$, $p_j^k = \{b_1, b_2, ..., b_n\} \in P^k$ в *k*-й популяции являются два элемента популяции k+1, такие что $p_i^{k+1} = \{a_1, ..., a_c, b_{c+1}, b_n\}, p_j^{k+1} = \{b_1, ..., b_c, a_{c+1}, a_n\} \in P^{k+1}$, где точка с выбирается случайно. Оператор мутации изменяет 20% особей в популяции, меняя значения кодируемых в хромосомах признаков на некоторую небольшую случайную величину. В качестве критерия останова используется достижение максимального числа итераций алгоритма, описанного выше.

На рисунке 1.23 показан процесс минимизации фунции потерь (1.40) с помощью генетического алгоритма в виде зависимости значения функции потерь от номера итерации на том же примере, ранее использовавшемся для демонстрации применения алгоритма Бройдена – Флетчера – Гольдфарба – Шанно в предыдущем пункте.

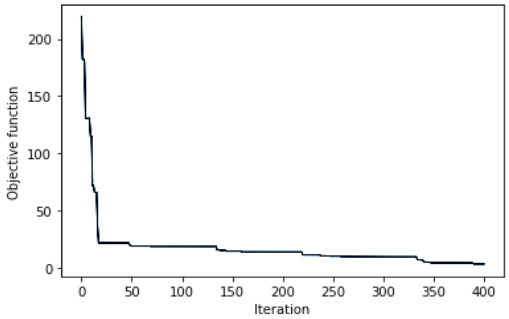


Рисунок 1.23 – Зависимость функции потерь от номера итерации алгоритма

Значение средней ошибки, достигнутой с использованием предложенных здесь гиперпараметров генетического алгоритма, таких как размер популяции, вероятность мутации, количество отбираемых для следующего поколения особей, удалось добиться ошибки, вычисленной по формуле (1.39), равной 3,2 Ом, что на порядок меньше, чем было получено с помощью алгоритма Бройдена – Флетчера – Гольдфарба – Шанно.

1.5 Программные средства для измерения параметров индуктивного компонента магнитометров

В настоящем разделе описана разработка и практическая реализация архитектуры программных средств для измерения индуктивного компонента магнитометров, реализующих описанный выше способ на базе устройства сбора данных ЛКАРД E502, выбор оптимальных методов и инструментальных средств создания программного обеспечения, позволяющих с минимальными усилиями, практически без изменения исходного кода использовать разработанные программные средства на вычислительных платформах с разнообразным аппаратным окружением, работающих под управлением различных операционных систем.

1.5.1 Архитектура программных средств. Графический интерфейс пользователя, драйвер устройств и некоторые другие части программных средств тесно связаны с оборудованием и в связи с этим должны быть соответствующим образом адаптированы для обеспечения возможности его работы на различных операционных системах Windows, Linux, Android

Схематически архитектура программных средств, способных работать на различных операционных системах, выглядит так, как показано на рисунке 1.24. В предложенной архитектуре основной код программных средств отделен от кода, зависящего от целевой рабочей платформы. Для переноса программных средств на другую платформу, например, для реализации измерителя параметров индуктивного компонента на базе смартфона или планшета, требуется лишь собрать для конкретной операционной системы и типа процессора (см. рис. 1.24) платформозависимую часть. Остальной же код программных средств, написанный на языке Python, можно будет запускать на любой платформе без изменений.

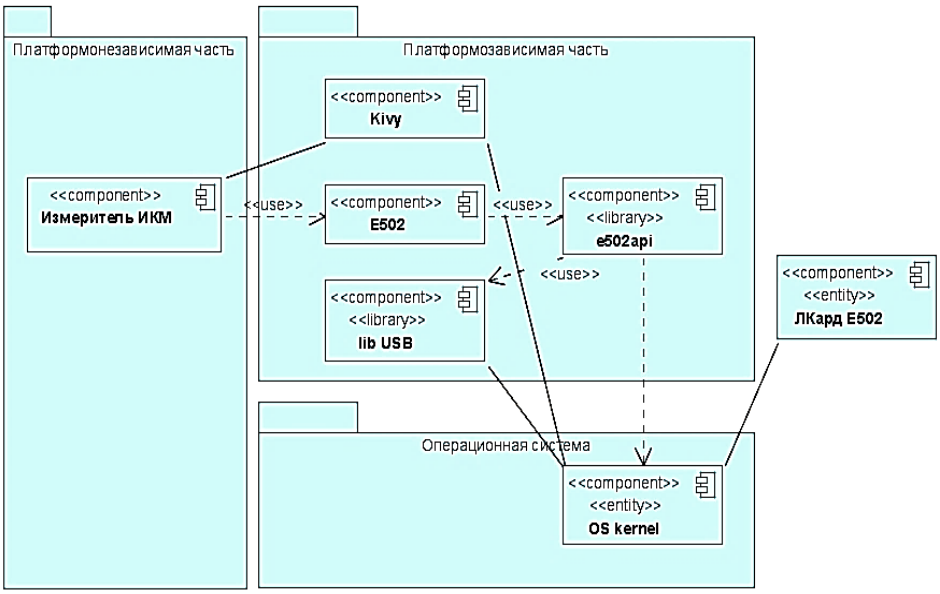


Рисунок 1.24 – Диаграмма компонентов программных средств для измерения параметров индуктивного компонента магнитометров

Плата сбора данных может подключаться к вычислительному устройству по интерфейсам USB или Ethernet. Сетевой стек TCP/IP, необходимый для связи с устройством сбора данных с помощью Ethernet или Wi-fi, в настоящее время имеется в любой современной мобильной и настольной операционной системе, однако не все функции поддерживаются одинаково хорошо. Для того чтобы обеспечить работу на различных платформах в драйвере Л КАРД Е502Б, необходимо отключить поддержку автоматического поиска устройств в локальной сети, недоступную в настоящее время в операционной системе Android, изменив в файле *CMakeLists.txt* библиотеки e502api опцию option(E502API_ENABLE_DNSSD "enable dns-sd service discovery" OFF), а в настройках программных средств в случае подключения устройства сбора данных по сети необходимо указать IP- адрес используемого устройства.

В программных средствах был использован стандартный драйвер USB, который в настоящее время поддерживается всеми популярными операционными системами. В главный CMake-файл драйвера платы сбора данных для поддержки Android устройств добавлен код, обеспечивающий корректную сборку для устройств на базе операционной системы Android. Как видно из приведенного выше кода, гарантируется работа на устройствах начиная с API 21 и выше, что соответствует версии Android не менее чем 5.0. Работа на более старых устройствах, к сожалению, невозможна, т.к. для них отсутствует поддержка стандарта POSIX Threads, необходимого для работы драйвера Л КАРД Е502, реализующего поддержку многопоточных программ, преимущественно ориентированных на исполнение на системах с общей памятью (Symmetric Multiprocessing, сокращённо SMP). Как известно, это такие системы, где установлено несколько процессоров или\и многоядерные процессоры и каждое ядро имеет доступ ко всей оперативной памяти компьютера.

Библиотека для работы с устройством ЛКАРД Е502. На рис. 1.25 показана архитектура разработанной библиотеки для работы с устройством ЛКард Е502 с помощью языка Python.

Для начала работы с модулем необходимо установить с ним связь с помощью функции open_usb. Для идентификации устройств используются их серийные номера. Получить список серийных номеров всех подключенных устройств Е502 можно с помощью get_usb_serial_list. Данная функция возвращает список, в котором сохранены найденные серийные номера, а принимает максимальное количество присоединенных модулей (по умолчанию это значение равно 16). Следует отметить, что с одним модулем одновременно может быть установлено только одно соединение. При попытке открыть модуль, с которым уже установлено соединение через другой описатель (возможно, в другой программе),

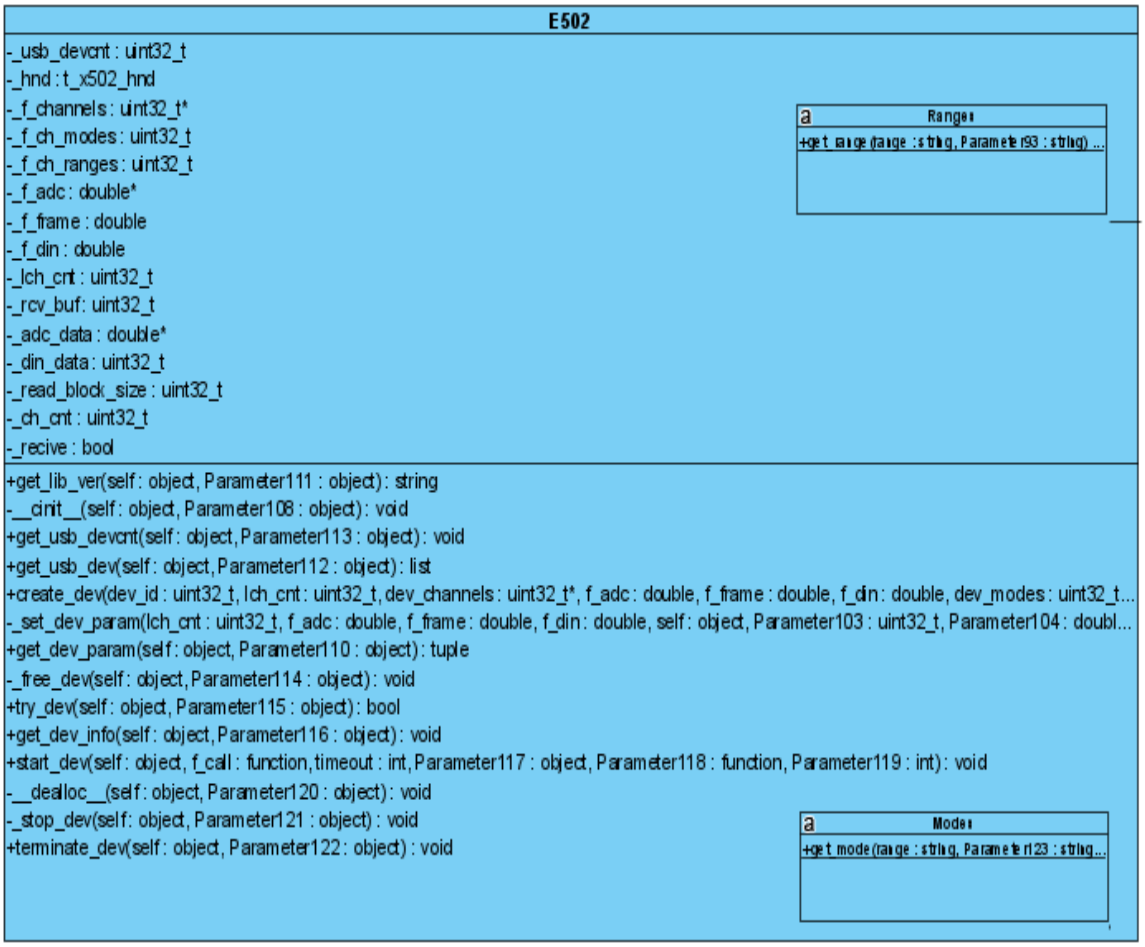


Рисунок 1.25 – Архитектура библиотеки для работы с устройством ЛКард E502

функция open_usb вернет ошибку. При этом get_usb_serial_list по умолчанию возвращает список всех серийных номеров устройств, включая те, с которыми уже установлено соединение. Если нужно получить список только тех устройств, с которыми еще не установлено соединение, то в get_usb_serial_list можно передать аргумент *only_not_opened=True* [15].

Установить связь с модулем по Ethernet можно только по явно заданному IP-адресу устройства. Следует отметить, что в отличие от интерфейса USB для корректной работы по Ethernet должны быть настроены необходимые параметры. Если USB-интерфейс в модуле E502 всегда работает и не требует дополнительной конфигурации, то для работы по интерфейсу Ethernet необходимо выполнить настройку параметров интерфейса и разрешить данный интерфейс. Следует отметить, что при разрешенном Ethernet-интерфейсе с модулем можно работать как по USB, так и по Ethernet. При этом сбор/генерация данных могут выполняться одновременно только по одному интерфейсу (именно тому, по которому пришла команда на запуск сбора/выдачи данных).

Выбор внешнего сигнала для задания опорной частоты синхронизации задается с помощью set_sync_mode, а условие запуска с помощью функции set_sync_start_mode. Следует отметить, что если задано внешнее событие запуска, то для того, чтобы модуль перешел в режим

ожидания этого события, необходимо вызвать функцию streams_start. Останов синхронного сбора/выдачи данных всегда осуществляется программно с помощью streams_stop. При использовании разъема синхронизации для организации сбора данных по принципу ведущий-ведомые для ведущего модуля источником опорной частоты синхронизации остается внутренняя частота (режим *internal*), а каждый ведомый модуль использует опорную частоту и/или признак запуска сбора от внешнего мастера, т.е. для каждого ведомого модуля должен быть установлен режим *external_master*.

При включении питания все каналы находятся в асинхронном режиме. В асинхронном режиме при вызове функции асинхронного ввода/вывода производится однократный ввод или вывод указанной информации.

При этом задержка от вызова функции до непосредственно момента измерения данных для ввода или выставления указанного значения на выходе для вывода точно не определена. Также точно не может быть определена задержка между двумя последовательными операциями ввода/вывода.

Для однократного ввода данных с АЦП используется функция async_get_adc_frame, которая выполняет ввод одного кадра данных АЦП. В отличие от других функций асинхронного ввода/вывода перед вызовом данной функции необходимо выполнить настройку модуля: необходимо задать управляющую таблицу АЦП. Измерение логических каналов внутри кадра происходит синхронно с заданной частотой сбора АЦП. Асинхронным является ввод самих кадров, то есть задержка между измерением кадров при последовательном вызове async_get_adc_frame не определена.

В синхронном режиме ввод или вывод данных осуществляется с заданной частотой, то есть время между соседними измерениями или выводом соседних отсчетов определено. Частота сбора для каждого канала задается относительно общей опорной частоты синхронизации, и запуск синхронного ввода-вывода для всех каналов осуществляется одновременно. Для запуска синхронного режима необходимо сначала с помощью функции streams_enable разрешить синхронный режим по требуемым каналам, а затем запустить синхронный ввод/вывод по всем разрешенным каналам с помощью streams_start. При синхронном вводе модуль производит измерения с заданной частотой и сам передает данные по интерфейсу в буфер. Принятые в буфер данные могут быть прочитаны программой с помощью get_data. Аналогично для синхронного вывода, модуль сам по мере необходимости считывает данные из буфера и выводит считанные отсчеты с заданной частотой. Данные в буфер драйвера должны быть предварительно записаны с помощью send_data. При этом если к моменту вывода очередного отсчета данные в буфер драйвера не поступили, то будет выведено предыдущее значение.

Библиотекой выделяется всего один буфер на прием и один на передачу данных. То есть значения для синхронного ввода с цифровых линий и отсчеты АЦП передаются одним потоком, также одним потоком поступают все данные на вывод. Каждый отсчет передается в виде 32-битного слова, содержащего дополнительную информацию, включающую в себя признак, к какому типу данных относится данный отсчет.

Далее осуществляется разбор принятых данных на отсчеты АЦП и значения цифровых выводов и перевод отсчетов АЦП в вольты. Данные от АЦП приходят в том порядке, в котором производятся измерения, т.е. сперва измерения, соответствующие всем логическим каналам первого кадра, затем второго и т.д.

При этом можно получить и нецелое количество кадров (например, если запущен синхронный ввод с цифровых линий, то заранее сложно определить, сколько в принятом блоке данных содержится отсчетов с цифровых линий, а сколько отсчетов с АЦП), в этом случае выдает все отсчеты, включая отсчеты нецелого кадра, дополняя отсутствующие значения нецелого кадра неопределенными значениями *None*, так чтобы в итоге было получено целое число кадров. Это необходимо для того, чтобы заранее можно было вычислить размер памяти, необходимый для хранения принятых данных, при дальнейшей обработке неопределенные значения могут быть отброшены.

Аналогично для формирования общего потока на вывод в требуемом формате используется функция send_data, принимающая данные из трех массивов и сохраняющая их во внешний массив. Если какой-либо из источников не должен использоваться, то в качестве массива передается нулевой указатель *None*. Для каналов, которые не были настроены на синхронный режим с помощью streams_enable, входной массив не анализируется, и данные из него не используются.

Следует отметить, что если для синхронного ввода инициализация потока передачи происходит по streams_start, так как данные начнут поступать только после запуска синхронного ввода, то с синхронным выводом дело обстоит несколько иначе. Так как по streams_start уже должна начаться выдача синхронных данных, то часть данных будет загружена в модуль. Таким образом, после разрешения синхронного вывода по нужным каналам с помощью streams_enable и до запуска синхронного вывода с помощью streams_start необходимо осуществить загрузку данных синхронного потока. Для этого следует использовать функцию send_data, по которой будет выделен буфер на передачу и инициализирован поток на передачу, если этого не сделать, то синхронный вывод начнется лишь после того, как данные будут записаны в модуль, и не будет привязан к началу синхронного сбора/выдачи данных.

Кроме того, для синхронной выдачи данных на ЦАП необходимо предварительно установить начальные значения на ЦАП с помощью функции асинхронного вывода. В противном случае при начале

синхронного вывода может быть небольшой переходный процесс от значения на ЦАП, которое было до запуска синхронного вывода, до выставления первых нужных значений, т.к. ЦАП имеет свой фильтр и ограничения на скорость изменения сигнала [16].

1.5.2 Сборка и развертывание разработанных программных средств.

Для повторяемой сборки программных средств разработан скрипт на языке Python, автоматизирующий создание контейнера Docker, и установку в него операционной системы Ubuntu 20.04 и различных инструментальных средств сборки.

В этом контейнере собирается драйвер Л КАРД E502 и библиотека для работы с платой сбора данных Л КАРД E502, а также библиотека Kivy, предоставляющая графический интерфейс пользователя.

Например, тулкит python-for-android реализует основные методы портирования приложения на операционную систему Android. В частности, он может скомпилировать интерпретатор Python, его зависимости для конкретной версии операционной системы, используемые в приложении библиотеки, и код приложения Python для устройств с операционной системой Android. Этот этап является полностью настраиваемым, при этом самыми важным параметром есть список зависимостей, т. е. модулей, которые требуются приложению для работы. В случае программных средств для измерения параметров индуктивных компонентов магнитометров это модули для работы с вычислительными методами обработки и визуализации данных, подробнее о которых говорится в работе [15]. Аналогичный тулкит существует и для операционной системы iOS – Kivy iOS.

Кроме того, для развертывания программных средств был разработан файл buildozer.spec, требующийся для сборки APK-файла для операционной системы Android с использованием сборочной системы Buildozer. В этом файле, кроме списка зависимостей и описания приложения, о которых уже говорилось выше, были указаны скомпилированные файлы провайдера E502; расширения файлов, содержащих необходимые ресурсы; минимальная версия Android API, необходимая для работы приложения; файлы драйвера Л КАРД E502 и архитектура процессора целевой платформы.

1.5.3 Графический интерфейс пользователя. Графический интерфейс пользователя построен на основе библиотеки Kivy – это современное инструментальное средство построения графического интерфейса пользователя, позволяющее создавать эргономичные интерфейсы для широкого спектра устройств. Скорость выполнения Kivy сопоставима с нативной мобильной альтернативой Java для Android или Objective C для iOS, кроме того, приложения, разработанные на основе фреймворка Kivy, могут работать и на персональных компьютерах.

Основные элементы графического интерфейса пользователя (рис. 1.26) – график, на котором отображается зависимость полного комплексного сопротивления от частоты, поля и кнопка для запуска процесса измерения.

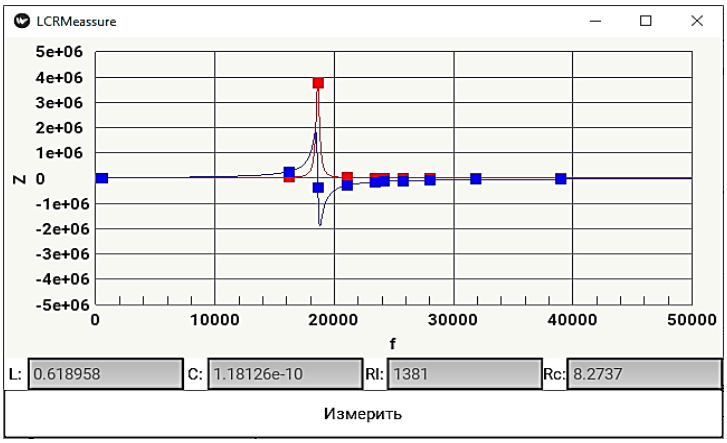


Рисунок 1.26 – Графический интерфейс программных средств для измерения параметров индуктивного компонента магнитометров

1.6 Выводы

Таким образом, предложен цифровой трёхкомпонентный индукционный вариометр высокой чувствительности с тремя буферными усилителями и двойными Т-образными фильтрами, соединёнными с помощью платы сбора данных с персональным компьютером, что позволяет в цифровой форме хранить и анализировать данные о геомагнитном поле. Использование двойных Т-образных фильтров позволяет значительно повысить отношение сигнал/шум и улучшить точность измерений. Исследована возможность использования электронного ключа для уменьшения структурной избыточности схемы цифрового магнитометра. Показано что электронные ключи могут использоваться только в том случае, если не требуется высокая частота дискретизации [8].

Показано, что индуктивный компонент, помимо основного параметра *L*, *C* и *R*, характеризуют паразитными параметрами, определяющими частотные свойства элементов. Обычно измеряют эквивалентные параметры. Рассмотрев различные методы измерения параметров индуктивности, мы предлагаем способ определения параметров индуктивных компонентов магнитометров на основе существующего метода преобразования полного комплексного сопротивления в напряжения и его реализация в виде устройства на основе платы сбора данных Л Кард E502. Рассмотрен способ определения амплитуды и фазы измеряемого напряжения, основанный на минимизации среднеквадратического отклонения между измеряемым напряжением и моделью гармонического сигнала. Показано, что этот способ хорошо справляется с различного рода помехами. Кроме того, разработаны

программные средства на базе устройства сбора данных ЛКард E502, реализующие разработанный способ, работающие на различной аппаратной платформе под управлением широкого круга операционных систем.

В результате проделанной работы решена проблема измерения параметров активного индуктивного компонента, на который воздействуют случайные фоновые электромагнитные поля. В предложенном способе это влияние нивелируется тем, что параметры индуктивности вычисляются по множеству измерений полного комплексного сопротивления на различных частотах и находятся исходя из наилучшего соответствия измеренным значениям в смысле минимума суммарного квадратичного отклонения.

2. Прогнозирование временных рядов с помощью технологий глубокого обучения

2.1 Анализ методов прогнозирования временных рядов

Поведение многих процессов реального мира представляется в виде временных рядов, т.е. последовательностей значений каких-либо параметров исследуемого процесса в дискретные, равноудаленные моменты времени. В таких случаях задача прогноза динамики процесса сводится к задаче экстраполяции временного ряда на основе модели, построенной по результатам его анализа. Стремление исследователей к повышению точности прогнозирования временных рядов обусловило существование многих методов и технологий построения моделей, аппроксимирующих исходные значения временного ряда [17].

Прогнозирование временных рядов в настоящее время – это одна из важнейших прикладных задач машинного обучения и искусственного интеллекта в целом, так как усовершенствование методов прогноза позволит более точно предсказать поведение различных факторов в экономике, технике, геоэкологии и других областях.

Основной целью любого прогноза является построение, идентификация, настройка и проверка моделей временных рядов [18]. Классической моделью для анализа и прогноза временных рядов является модель Бокса-Дженкинса ARIMA – Autoregressive Integrated Moving Average, сочетающая в себе авторегрессионную модель и модель скользящего среднего и использующая для анализа первые разности исходного временного ряда. Прогнозу временных рядов на основе этой модели посвящено множество работ, доказывающих эффективность ее применения при анализе временных рядов различной физической природы. Однако положенные в основу метода Бокса-Дженкинса или ARIMA подходы предполагают, что временные ряды генерируются линейными процессами, тогда как порождающие их механизмы реального мира часто имеют нелинейную природу. Поэтому одним из наиболее точных и широко распространенных средств для построения прогнозирующей модели стал метод прогноза временных рядов, основанный на аппарате искусственных нейронных сетей: для восстановления неизвестной аппроксимирующей функции многих переменных по набору значений, заданных историей временного ряда, используется обученная многослойная нейронная сеть с несколькими входами, соответствующими этим переменным. Нейросетевым технологиям анализа и прогноза временных рядов к настоящему времени посвящено множество работ, и эти технологии продолжают развиваться [17]. Так, например, в [18] рассмотрены различные нейронные сети, так называемые многослойные персептроны прямого и обратного распространения (рекуррентные нейронные сети). В качестве алгоритмов настройки весов используются алгоритмы сопряженных градиентов и

квази-Ньютона (BFGS- алгоритм). Предложена гомогенная ансамблевая технология: искусственная нейронная сеть обучается с каждым из рассмотренных алгоритмов, для каждого алгоритма вычисляется вес (обратно пропорционально ошибке прогноза), а финальный прогноз вычисляется как взвешенное арифметическое среднее всех прогнозов. В [22] рассматриваются два способа решения проблемы прогнозирования спроса в сезонных временных рядах с использованием искусственных нейронных сетей. Первый способ – многослойная модель персептрона, на вход которого подаются предыдущие значения временного ряда. Было оценено несколько правил обучения, используемых для корректировки весов многослойного персептрона (обратного распространения ошибки с помощью алгоритма наискорейшего спуска, адаптивный и Левенберга-Маркардта). Во втором способе был использован частный метод, основанный на искусственных нейронных сетях с применением в качестве входных переменных компонентов разложенных временных рядов (тренд, сезонная и случайная составляющие). Таким образом, большое внимание исследователей привлекают нейросетевые технологии прогнозирования, сочетающие в себе разные модели, так называемые гибридные или ансамблевые модели. Гибридные модели могут быть однородными, использующими различно настроенные нейронные сети (все многослойные персептроны, например), или гетерогенными, сочетающими как линейные, так и нелинейные модели. Основная идея этого многомодельного подхода заключается в использовании каждым компонентом уникальной способности модели улучшить захват различных паттернов в данных [23].

В последнее десятилетие были предложены специализированные нейронные сети – вейвлет-сети, являющиеся модификацией сетей на основе радиальных базисных функций, где в качестве базисных элементов используются вейвлеты («короткие волны»), хорошо локализованные как во временной, так и в частотной области. Вейвлет-сеть представляет собой трёхслойную нейронную сеть, в которой первый слой является входным слоем, второй – скрытым, а третий – выходным. Предложены различные структуры вейвлет-сетей, однако общая их идея заключается в настройке параметров сжатия и сдвига вейвлет-нейронов для наилучшего обучения данным. Так, в [24] предложена мультивейвлетная нейронная сеть, в которой используется многомерный вейвлет в качестве функции активации вейвлет-нейронов в скрытом слое, что позволяет аппроксимировать многомерные функции. В этой работе для прогноза временных рядов предложена новая структура вейвлет-сети, отличающаяся от традиционной тем, что в базовые вейвлеты введены дополнительные настраиваемые параметры. Такая сеть, названная полиморфной, имеет лучшие аппроксимирующие свойства, благодаря лучшей приспосабливаемости к характеру нестационарностей в одномерных временных рядах, для учёта инерционности во временных

рядах в структуру полиморфной вейвлет-сети введены обратные связи, учитывающие уровни временного ряда в предыдущие моменты времени [25].

В этих работах с целью повышения точности прогнозирования нестационарных временных рядов автором предложена новая модель вейвлет-сети, объединяющая преимущества предложенных ранее вейвлет-сетей – мультивейвлетная полиморфная вейвлет-сеть. Эту модель можно интерпретировать как гибрид авторегрессионной линейной модели, параметры которой определяются весами линейных связей и искусственной нейронной сети, определяемой весами нелинейных связей, в которой в качестве функции активации используются многомерные вейвлеты с дополнительным настраиваемым параметром.

В последнее время наряду с вышеизложенными методами широко применяются методы, основанные на технологиях глубокого обучения, где на первый план выходят так называемые глубокие нейронные сети. Нейронные сети уже достаточно хорошо зарекомендовали себя в тех областях, где необходимо применение человеческого интеллекта, и в частности, при решении задач классификации, распознавания образов и анализа временных рядов. При помощи нейронных сетей возможно моделирование нелинейной зависимости будущего значения временного ряда от его прошлых значений и от значений внешних факторов. Для прогноза временных рядов, как было сказано выше, предлагалось использовать полносвязные нейронные сети прямого распространения, мультивейвлетные полиморфные сети, машины экстремального обучения и сверточные нейронные сети. Однако важной особенностью всех перечисленных нейронных сетей является отсутствие памяти о временных зависимостях в данных. Каждый пакет входных данных обрабатывается ими независимо, без сохранения состояния между ними, человеческому же интеллекту свойственно воспринимать новую информацию, основываясь на предыдущей информации, и постоянно пополнять свой опыт по мере поступления новой информации. Для учета этой особенности человеческого восприятия хотя бы в упрощенном виде были предложены различные разновидности рекуррентных нейронных сетей, обрабатывающих элементы временного ряда последовательно, с использованием информации, полученной при обработке предыдущих его элементов [26].

Например, в работе [25] показано преимущество использования полиморфных вейвлет-сетей с обратными связями по сравнению с сетями без обратных связей на примере идентификации моделей нестационарных временных рядов. Недостатком этих рекуррентных нейронных сетей является то, что в каждый момент времени они должны хранить информацию о входных данных за многочисленные предыдущие интервалы времени, но на практике такие протяженные зависимости не поддаются обучению. Это связано с проблемой затухания градиента,

напоминающего эффект, наблюдающийся в сетях прямого распространения с большим количеством слоев: по мере увеличения количества слоев сеть в конечном итоге становится необучаемой. Теоретическое обоснование этого эффекта было дано Хохрейтером, Шмидхубером и Бенгио в начале 1990-х гг. Для решения этой проблемы был предложен слой долгой краткосрочной памяти (Long short-term memory, LSTM), позволяющий повторно задействовать в процессе обучения предыдущую информацию и тем самым решить проблему затухания градиента. Слой управляемых рекуррентных блоков (Gated Recurrent Unit, GRU) основан на том же принципе, что и слой LSTM, однако рекуррентные блоки представляют собой более простые структуры по сравнению с LSTM и, соответственно, менее затратные в вычислительном смысле. В традиционных рекуррентных нейронных сетях для реализации обратной связи используется комбинация скрытого состояния на предыдущем шаге и текущих входных данных в слое с нелинейной функцией активации, например, такой, как гиперболический тангенс. В LSTM-сети обратная связь реализуется аналогично, но нейронных слоев используется не один, а четыре [18].

Такие многослойные нейронные сети, содержащие слои различной структуры: полносвязные, реккурентные, свёрточные или вейвлет-слои, и называются глубокими нейронными сетями, а технологии машинного обучения, связанные с такими сетями, называются в настоящее время глубоким обучением. Технологии глубокого обучения противопоставляются поверхностному обучению сетей прямого распространения, так как с их помощью стало возможным обрабатывать гораздо больший объем входных данных, и поэтому для их использования не требуется предварительный анализ, для выделения значимых данных для получения конечного результата и их структурирование. Глубокие нейронные сети за счет своей многослойной архитектуры, в которой каждый слой выполняет свою собственную функцию, оказываются способны самостоятельно выделить и структурировать необходимые данные. Изучению возможностей глубоких нейронных сетей по сравнению с поверхностными сетями на примере прогноза временных рядов и посвящена настоящая глава.

На рисунке 2.1 показана схема работы ячейки LSTM. На горизонтальной линии сверху показано состояние ячейки *c*, оно представляет внутреннюю память ячейки.

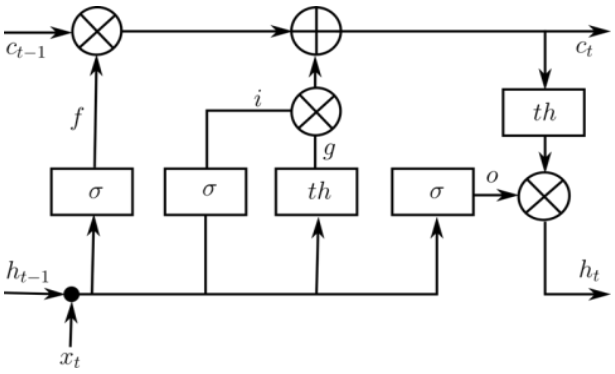


Рисунок 2.1 – Схема работы ячейки LSTM

На линии снизу показано скрытое состояние, а вентили *i, f, o и g* – это механизмы, благодаря которым решается проблема затухания градиента, их значения находятся в процессе обучения сети.

Скрытое состояние LSTM-сети h_t в момент t по состоянию h_{t-1} на предыдущем шаге находится по формулам:

$$i = \sigma(W_i h_{t-1} + U_i x_t),$$
$$f = \sigma(W_f h_{t-1} + U_f x_t),$$
$$o = \sigma(W_o h_{t-1} + U_o x_t),$$
$$g = th(W_g h_{t-1} + U_g x_t),$$
$$c_t = (c_{t-1} \otimes f) \oplus (g \otimes i),$$
$$h_t = th(c_t) \otimes o.$$

Здесь i, f и o – входной вентиль, вентиль забывания и выходной вентиль. Все они вычисляются по одним и тем же формулам, но разными матрицами весов W_i, W_f, W_o и U_i, U_f, U_o. Сигмовидная функция σ модулирует выход вентилей, приводя его к диапазону от 0 до 1, так что порождаемый выходной вектор можно умножить поэлементно на другой вектор, чтобы определить, какая часть второго вектора может пройти через первый. Вентиль забывания определяет, какую часть предыдущего состояния h_{t-1} желательно пропустить дальше. Входной вентиль определяет, какую часть внутреннего состояния передать следующему слою. Внутреннее скрытое состояние g вычисляется на основе текущего входа x_t и предыдущего скрытого состояния h_{t-1}. Зная i, f, o и g, можно вычислить состояние ячейки c_t в момент t как сумму произведений c_{t-1} на вентиль f и g на входной вентиль i. Наконец, скрытое состояние h_t в момент t вычисляется путем умножения памяти c_t на значение выходного вентиля. GRU – это вариант LSTM слоя, также обладающий устойчивостью к проблеме затухания градиента, но структура его проще (рис. 2.2), а потому и обучается он быстрее.

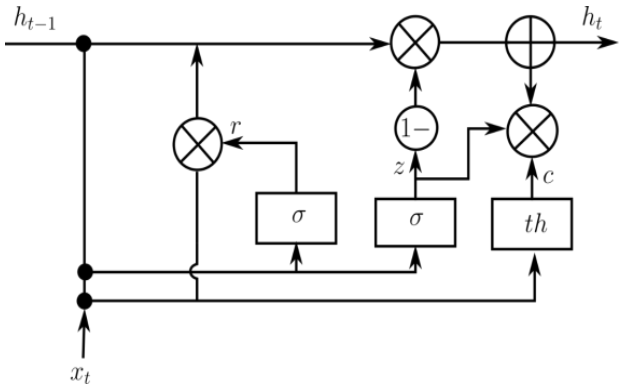


Рисунок 2.2 – Схема работы ячейки GRU-сети

Вместо трех вентилей в ячейки LSTM – входного забывания и выходного, в ячейке GRU всего два вентиля: обновления z и сброса r. Вентиль обновления определяет, какую часть предыдущего запомненного значения сохранять, а вентиль сброса – как смешивать новый вход с предыдущей памятью. Механизм работы GRU описывается формулами:

$$z = \sigma(W_i h_{t-1} + U_z x_t),$$
$$r = \sigma(W_r h_{t-1} + U_r x_t),$$
$$c_t = th(W_c(h_{t-1} \otimes r) + U_c x_t),$$
$$h_t = (z \otimes c) \oplus ((1-z) \otimes h_{t-1}).$$

Кроме рекуррентных нейронных сетей, для прогнозирования временных рядов можно использовать сверточные нейронные сети (convolutional neural network, CNN), благодаря их способности к свертыванию параметров, извлечению признаков из локальных входных шаблонов, получению эффективных и модульных представлений данных несмотря на то, что основная сфера их применения – распознавание образов на изображениях. Время можно рассматривать как пространственное измерение, подобно высоте или ширине двумерного изображения. Такие одномерные сверточные нейронные сети с успехом могут состязаться с рекуррентными сетями в некоторых задачах обработки последовательностей, как правило, требуя меньше вычислительных ресурсов. Кроме того, известно, что небольшие одномерные сверточные нейронные сети могут служить быстрой альтернативой рекуррентным сетям в таких задачах, как классификация текста и прогнозирование временных рядов [26].

В работе [27] высказана идея использования для прогноза временных рядов гибридной сети, состоящей из полносвязной сети прямого распространения и простой рекуррентной нейронной сети, так как авторами было показано, что рекуррентные нейронные сети дают лучший результат при долгосрочном прогнозировании, а сети прямого распространения при краткосрочном. Также различными авторами предлагалось для этой цели использовать и другие архитектуры нейронных

сетей как в комбинации с традиционным методом прогноза ARIMA, так и по отдельности.

Таким образом, появление в последнее время более специализированных нейросетевых слоев, таких как LSTM, GRU, CNN и технологий глубокого обучения, заставляет многих исследователей создавать новые архитектуры глубоких нейронных сетей для прогноза временных рядов и ряда других задач. Глубокое обучение – это особый раздел машинного обучения: новый подход к поиску способов обработки информации, делающий упор на обучение последовательных слоев (или уровней), все более значимых преобразований исходных данных. Под глубиной в таком обучении не подразумевается более глубокое понимание, достигаемое этим подходом; идея заключается в многослойном представлении. Современное глубокое обучение часто вовлекает в процесс десятки и даже сотни последовательных слоев представления – и все они автоматически настраиваются под воздействием обучающих данных. Между тем традиционные подходы к машинному обучению ориентированы на изучении одного-двух слоев представления данных; по этой причине их иногда называют методами поверхностного обучения.

2.2 Модель прогноза одномерного временного ряда на основе поверхностных нейронных сетей

В основе предложенного в работе [18] метода прогноза, как, впрочем, и большинства методов, связанных с обработкой временных рядов, лежит построение векторов задержек:

$$\mathbf{x}_n = (x_n, x_{n+1}, ..., x_{n+p-1})^T, n = 1,2,..., N - p$$

и построение целевого вектора

$$\mathbf{y} = (x_{p+1}, x_{p+2}, ..., x_N),$$

где N – количество отсчетов временного ряда, p – количество задержек.

Векторы задержек и целевой вектор используются для обучения, например, мультивейвлетной полиморфной сети: векторы задержек $\mathbf{x}_n$ подаются на вход мультивейвлетной полиморфной сети, в результате формируется вектор ответов сети $\hat{y}_n(\mathbf{x}_n) = g_\lambda(\mathbf{x}_n, \mathbf{w}_v)$.

Для прогноза временного ряда с помощью обученной мультивейвлетной полиморфной сети используется итеративный способ. Сначала строится исходный вектор задержек $\hat{\mathbf{x}}_1 = (x_{N-p+1}, x_{N-p+2}, ..., x_N)^T$, на основании которого с помощью мультивейвлетной полиморфной сети делается прогноз на один шаг: $\hat{x}_{N+1} = g_\lambda(\hat{\mathbf{x}}_1; \hat{\mathbf{w}})$, затем полученное значение добавляется к исходному вектору задержек, строится новый вектор $\hat{\mathbf{x}}_2 = (x_{N-p+2}, x_{N-p+3}, ..., \hat{x}_{N+1})^T$, и делается прогноз ещё на один шаг $\hat{x}_{N+2} = g_\lambda(\hat{\mathbf{x}}_2; \hat{\mathbf{w}})$ и так далее:

$$\hat{x}_{(k+1)} = g_\lambda(\hat{\mathbf{x}}_k; \hat{\mathbf{w}}),$$

где $\hat{\mathbf{w}}$ – параметры обученной мультивейвлетной полиморфной сети, $k=1, 2, ..., K,$ и K – количество прогнозируемых отсчетов временного ряда.

Выход мультивейвлетной сети определяется уравнением:

$$\hat{y}(\mathbf{x}) = g_{\lambda}(\mathbf{x};\mathbf{w}) = w_{\lambda+1}^{[2]} + \sum_{j=1}^{\lambda} w_j^{[2]} \cdot \boldsymbol{\Psi}_j(\mathbf{x}) + \sum_{i=1}^{m} w_i^{[0]} \cdot x_i \quad (2.1)$$

,

где $\boldsymbol{\Psi}_j(\mathbf{x})$ – многомерный вейвлет, задающийся как произведение m скалярных вейвлетов, $\mathbf{x}$ – вектор входных данных, m – количество входов, λ – количество скрытых вейвлет-нейронов и $\mathbf{w}$ – параметры сети: $w^{[2]}{}_{\lambda+1}$ – вес смещения, $w_i^{[0]}$ – веса линейных связей , $w_j^{[2]}$ – веса нелинейных связей. Многомерный вейвлет в формуле (2.1) вычисляется как

$$\boldsymbol{\Psi}_j(\mathbf{x}) = \prod_{i=1}^{m} \psi(z_{ij}) \quad (2.2)$$

,

где ψ – материнский вейвлет, и

$$z_{ij} \frac{x_i - w_{(\xi)ij}^{[1]}}{w_{(\zeta)ij}^{[1]}} \quad (2.3)$$

.

В выражении (2.3) $i=1, ..., m, j=1, ..., \lambda+1$, $w^{[1]}{}_{(\xi)ij}$ – параметры сдвигов, $w^{[1]}{}_{(\zeta)ij}$ – параметры масштабов вейвлетов.

Выбор материнского вейвлета зависит от решаемой задачи, это могут быть: первая или вторая производные Гауссианны, вейвлет Морле, ортогональные вейвлеты и вейвлет-фреймы:

первая производная Гауссианны т.н. «WAVE-вейвлет»:

$$\psi(z_{ij}) = z_{ij} e^{-0.5 z_{ij}^2}, \quad (2.4)$$

вторая производная Гауссианны, т.н. «Мексиканская шляпа»:

$$\psi(z_{ij}) = (1 - z_{ij}^2) e^{-0.5 z_{ij}^2} \quad (2.5)$$

или вейвлет Морле:

$$\psi(z_{ij}) = \cos(5 z_{ij}) e^{-0.5 z_{ij}^2}. \quad (2.6)$$

В мультивейвлетной полиморфной сети (рис. 2.3) используются материнские вейвлеты с дополнительным настраиваемым параметром, изменяющим форму вейвлета способом, отличным от сжатия и сдвига. Например, можно использовать полиморфный материнский вейвлет Superposed LOGistic functions («суперпозиция логистических функций»), известный, как SLOG:

$$\psi(z_{ij}, w_{(\rho)ij}^{[1]}) = \frac{1}{1+e^{-z_{ij}+w_{(\rho)ij}^{[1]}}} - \frac{1}{1+e^{-z_{ij}+3w_{(\rho)ij}^{[1]}}} - \frac{1}{1+e^{-z_{ij}-3w_{(\rho)ij}^{[1]}}} + \frac{1}{1+e^{-z_{ij}-w_{(\rho)ij}^{[1]}}},$$

где $w^{[1]}{}_{(\rho)ij}$ – дополнительный настраиваемый параметр формы вейвлета, определяющий скорость его затухания.

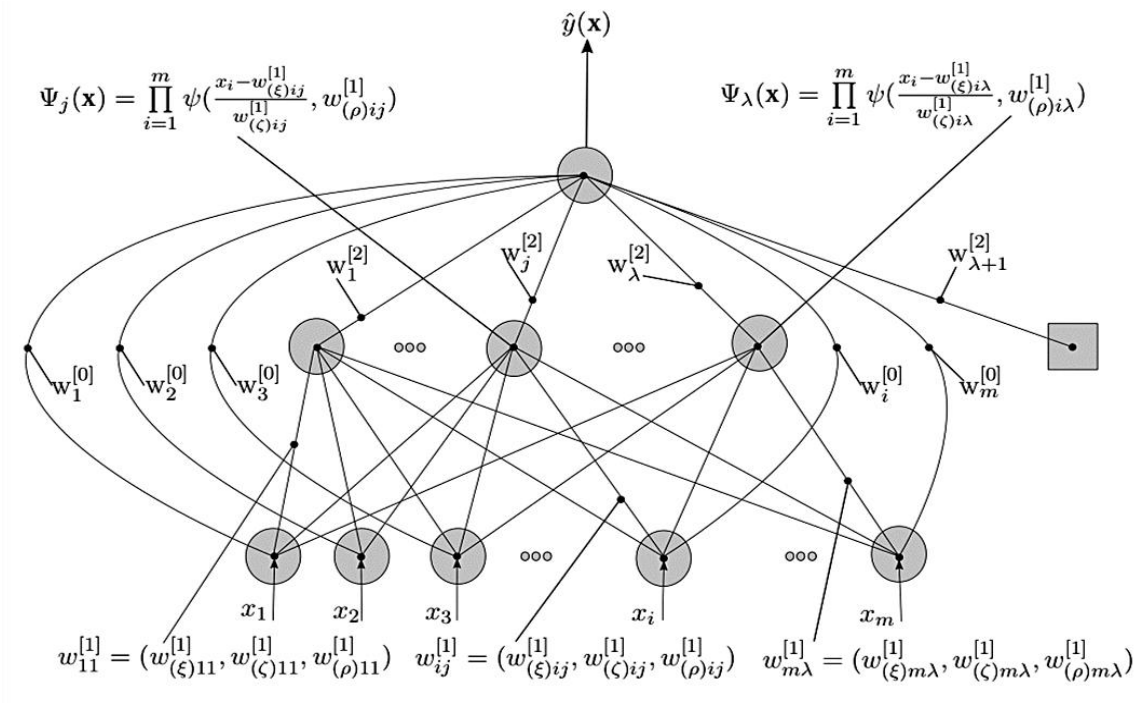


Рисунок 2.3 – Мультивейвлетная полиморфная сеть

Тогда (2.2) будет выглядеть как:

$$\mathbf{\Psi}_j(\mathbf{x}) = \prod_{i=1}^{m} \psi(z_{ij}, w^{[1]}_{(\rho)ij})$$

Полный вектор параметров сети включает $\mathbf{w}=(w_i^{[0]}, w_j^{[2]}, w^{[2]}{}_{\lambda+1}, w^{[1]}{}_{(\xi)ij}, w^{[1]}{}_{(\zeta)ij}, w^{[1]}{}_{(\rho)ij})$. Эти параметры настраиваются в процессе обучения сети путём минимизации среднеквадратической ошибки, построенной на разности требуемого y_p и действительного $\hat{y}_p$ значений сети, по настраиваемым параметрам сети. В результате получаются формулы для пересчета параметров сети на каждой итерации:

$$\mathbf{w}_{v+1} = \mathbf{w}_v - \eta \frac{\partial E}{\partial w_v} + \kappa(\mathbf{w}_v - \mathbf{w}_{v-1}) \qquad (2.7)$$

где $E = \frac{1}{2n}\sum_{p=1}^{n}(y_p - \hat{y}_p)^2$ – среднеквадратическая ошибка сети, v – номер итерации, η – параметр скорости обучения и к – параметр момента.

Затем находится среднеквадратическая ошибка сети, исходя из минимума которой на основе одного из итерационных методов многомерной оптимизации (в приведенных примерах – метода Бройдена–Флетчера–Гольдфарба–Шанно) подстраиваются параметры сети, используя обучающее правило (2.7). Конкретные значения параметров η и κ зависят от используемого итерационного метода многомерной оптимизации.

Пример прогноза. Для анализа используется временной ряд вариаций продолжительности суток [19]. Как видно из скользящего среднего:

$$m_t = \frac{1}{n}\sum_{i=0}^{n-1} l_{(t-i)}$$

и скользящего среднеквадратического отклонения:

$$\sigma_t = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n-1}(l_{(t-i)} - m_t)^2}$$

со сглаживающим интервалом $n = 100$, временной ряд (рис. 2.4. а)

$$l(t) = \sum_{i=1}^{M} a_i t^{i-1} + \sum_{j=1}^{L} b_j \cos(c_j t + d_j) + x_t$$

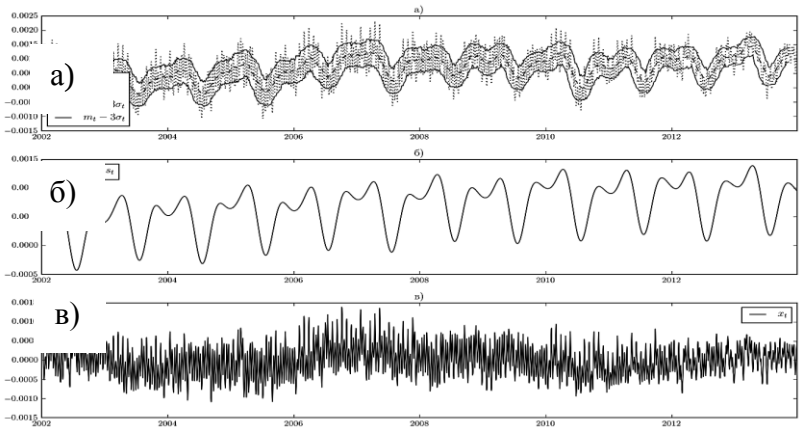


Рисунок 2.4 – Временные ряды: а – временной ряд вариаций ППС; б – тренд и сезонность; в – остатки

Этот временной ряд имеет тренд и периодически изменяющееся среднеквадратическое отклонение, а значит, является нестационарным.

Предположим, что временной ряд вариаций ПС может быть представлен в виде аддитивной модели как:

$$l_t = d_t + s_t + x_t,$$

где d_t – трендовая составляющая (тренд), s_t – сезонная составляющая (сезонность), x_t – хаотическая составляющая (остатки). Причём тренд с достаточной точностью можно аппроксимировать полиномом, а сезонность – рядом Фурье (см. рис. 2.4. б): где параметры **a**, **b**, **c**, **d** определяются по методу наименьших квадратов, а *M, L* подбираются на основании графика временного ряда (см. рис. 2.4.а) и частотных спектров временного ряда, показанных на рис. 2.5.

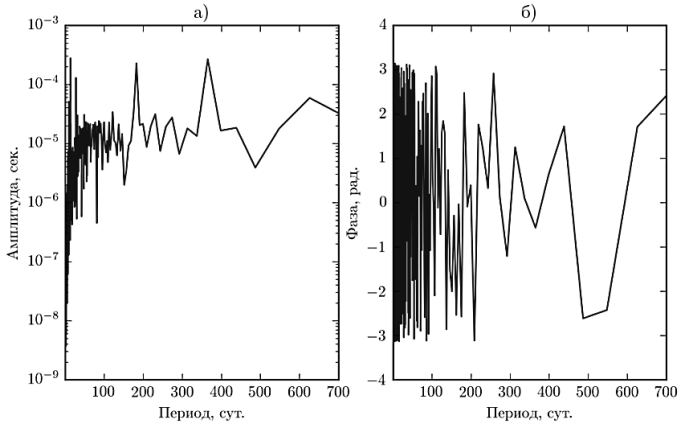


Рисунок 2.5 – Спектры временного ряда вариаций ППС: а – частотный, б – фазовый

Исходя из них, можно заключить, что временной ряд содержит линейный тренд и постоянную составляющую (т.е. $M = 2$), две длиннопериодические компоненты (приблизительно 625, 365 дней), две среднепериодические (приблизительно 183, 121 день), а также несколько короткопериодических компонент, связанных с вращением Земли вокруг Солнца и с вращением Луны вокруг Земли (см. рис. 2.5. а). Длинно- и среднепериодические компоненты могут быть аппроксимированы четырьмя членами ряда Фурье (т.е. $L = 4$). Однако фаза короткопериодических компонентов не остаётся постоянной во времени (см. рис. 2.5 б), и для их аппроксимации оказывается невозможным использовать ряд Фурье. Поэтому их следует отнести к хаотической составляющей временного ряда (рис. 2.4 в), и для её прогнозирования предлагается использовать мультивейвлетную полиморфную сеть с полиморфным вейвлетом SLOG в качестве материнского.

Вейвлет SLOG даёт лучшие результаты прогноза хаотической составляющей вариаций ППС по сравнению с вейвлетами (2.4)-(2.6), что обусловлено схожестью формы этого вейвлета и прогнозируемого временного ряда, как видно из рисунков 2.4 и 2.6.

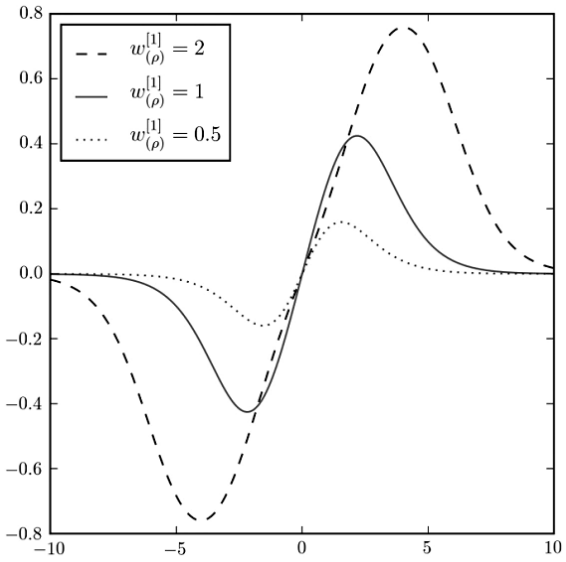


Рисунок 2.6 – Полиморфный SLOG вейвлет

На рис. 2.7 показаны результаты прогноза на 120 суток хаотической составляющей ряда вариаций ППС мультивейвлетной полиморфной сетью со 120 входами, содержащей 10 вейвлет-нейронов с многомерным полиморфным SLOG вейвлетом в качестве материнского вейвлета.

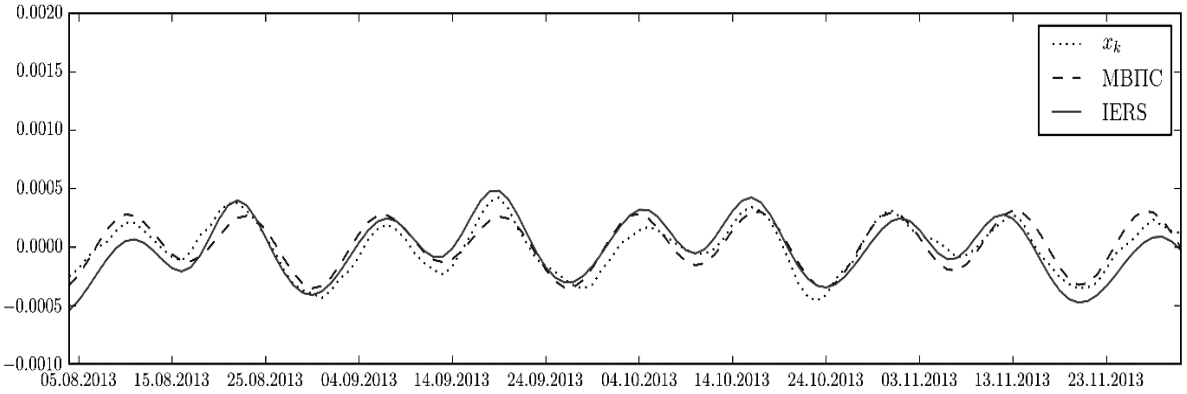


Рисунок 2.7 – Прогноз хаотической составляющей временного ряда вариаций ППС с помощью мультивейвлетной полиморфной сети

Для сравнения показан также прогноз с помощью общепринятой модели IERS [20], учитывающей лунные и солнечные приливные взаимодействия. Среднеквадратическая ошибка прогноза, рассчитанная как:

$$RMS = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(x_k - \hat{x}_k)^2} \ ,$$

(где $k = 120$ – интервал прогнозирования, x_k – хаотическая составляющая временного ряда вариаций ППС, $\hat{x}_k$ – прогноз этого ряда) для прогноза, полученного с помощью мультивейвлетной полиморфной сети, равна $9.87 \cdot 10^{-5}$, а для прогноза, полученного с помощью модели IERS – $1.1 \cdot 10^{-4}$.

2.3 Сравнительный анализ прогностических возможностей моделей поверхностного обучения на примерах одномерных временных рядов

В настоящем разделе для исследования работоспособности и эффективности предложенной в мультивейвлетной полиморфной нейросетевой модели (Multiwavelet Polymorphic Neural Network MWNN) проведены вычислительные эксперименты с тремя хорошо известными в статистике временными рядами:

- ✓ количество солнечных пятен (Wolf's sunspot data),
- ✓ численность популяции канадской рыси (Canadian lynx data)
- ✓ курс британского фунта к доллару США (British pound/US dollar exchange rate data).

Представлено также сравнение полученных результатов с результатами прогноза на других поверхностных моделях: ARIMA, нейронной сети прямого распространения и гибридной модели Чанга, показавшей, что предложенная модель имеет преимущества перед известными моделями в плане уменьшения ошибок прогноза применительно к одномерным временным рядам. Все эти модели содержат 1-3 последовательных слоя структур, обрабатывающих данные, и поэтому

называются в настоящее время моделями, основанными на поверхностном обучении. Для сравнения возможностей различных моделей для прогнозирования временных рядов были рассмотрены следующие модели: ARIMA-модель, искусственная нейронная сеть, гибридная модель Чанга, объединяющая модели ARIMA и нейронной сети и рассмотренная выше мультивейвлетная полиморфная сеть.

В качестве экспериментальных данных были взяты три временных ряда (ВР) (табл. 2.1) из открытой библиотеки данных Time Series Data Library (TSDL) repository, обладающих разными статистическими характеристиками и, хорошо известные в статистических исследованиях, количество солнечных пятен (Wolf's sunspot data), численность популяции канадской рыси (Canadian lynx data) и курс британского фунта к доллару США (British pound/United States dollar exchange rate data).

Таблица 2.1 – Наборы данных

ВР	Полный размер выборки	Обучающая выборка (размер)	Тестовая выборка (размер)
Wolf's Sunspot data	288	1700 –1920 (221)	1921–1987 (67)
Canadian Lynx data	114	1821–1920 (100)	1921–1934 (14)
Exchange rate data	731	1980 –1992 (679)	1993 (52)

Эффективность прогнозирования оценивалась средней абсолютной ошибкой (MAE, Mean Absolute Error) и средней квадратической ошибкой (MSE, Mean Squared Error):

$$MAE = \frac{1}{K}\sum_{i=1}^{K}|e_i|, \qquad MSE = \frac{1}{K}\sum_{i=1}^{K}(e_i)^2, \tag{2.8}$$

где $e_i = y - \hat{y}_f$, K– интервал прогнозирования, y – действительное значение уровня ряда, $\hat{y}_f$ – прогнозное значение уровня ряда.

В табл. 2.2 и на рис. 2.8 приведены результаты прогнозирования данных Wolf's Sunspot Data. Были использованы два периода прогнозирования: 35 и 67 лет.

Таблица 2.2 – Ошибки прогнозов Wolf's Sunspot Data

Модель прогноза	Упреждение прогноза 35		Упреждение прогноза 67	
	MSE	MAD	MSE	MAD
ARIMA	216.965	11.319	306.08217	13.033739
ИНС	205.302	10.243	351.19366	13.544365
Гибридная модель Чанга	186.827	10.831	280.15956	12.780186
MWNN	151.825	9.1351	272.49066	12.420121

Для прогноза были использованы: ARIMA- модель 9×0×0 (как показано в работе [27]), эта модель – наиболее минималистичная среди всех ARIMA-моделей, порождающих близкие по величине ошибки прогноза и часто используемая во многих других исследованиях, нейросетевая модель, содержащая 4×4×1 нейронов, архитектура которой также обоснована в [27], гибридная модель Чанга и мультивейвлетная

полиморфная нейронная сеть (MWNN), содержащая 9×4×1 вейвлет-нейронов.

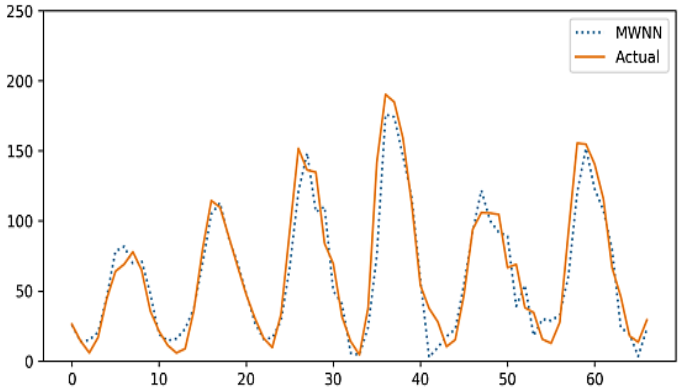


Рисунок 2.8 – Прогнозирование количества солнечных пятен (Wolf's Sunspot data) мультивейвлетной полиморфной сетью

Сравнение результатов показывает, что MSE-прогнозирование с использованием модели MWNN меньше MSE гибридной модели Чанга [27] на 18,735% и 2,74% соответственно.

Аналогичным образом были обработаны данные Canadian Lynx Data. Прогноз с помощью ARIMA выполнен на модели порядка 12×0×0. Эта минималистичная модель также использовалась в работе [22]. ИНС, содержащая 7×5×1 нейронов, дает несколько лучший прогноз по сравнению с ARIMA-моделью. На рис. 2.9 приведены результаты прогнозирования с использованием мультивейвлетной полиморфной сети MWNN, содержащей 2×7×1 нейронов.

Сравнение результатов (табл. 2.3) показывает, что MSE-прогнозирование мультивейвлетной полиморфной сети меньше MSE гибридной модели Чанга [27] на 54,89%. Здесь при построении модели использовался натуральный логарифм исходных данных.

Таблица 2.3 – Ошибки прогнозов Canadian Lynx Data

	MSE	MAD
ARIMA	0.020486	0.112255
ИНС	0.020466	0.112109
Гибридная модель Чанга	0.017233	0.103972
MWNN	0.007774	0.063614

Прогноз курса британского фунта к доллару США (Exchange rate Data) выполнен для интервала прогнозирования в 1, 6 и 12 месяцев с использованием модели ARIMA 0×1×0, нейросетевой модели ИНС, содержащей 7×6×1 нейронов, гибридной модели Чанга [27] и мультивейвлетной полиморфной сети MWNN, содержащей 1×7×1 нейронов (рис. 2.10).

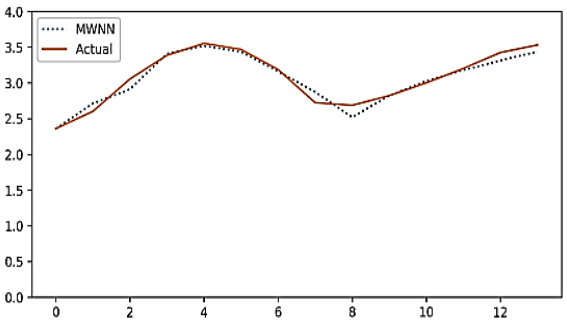


Рисунок 2.9 – Прогнозирование Canadian Lynx Data мультивейвлетной полиморфной сетью

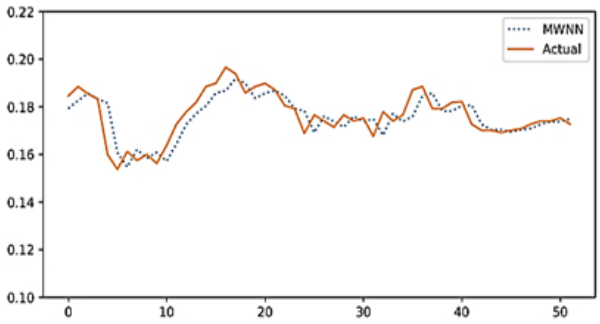


Рисунок 2.10 – Прогнозирование курса британского фунта к доллару США (Exchange rate Data) мультивейвлетной полиморфной сетью

Сравнение результатов (табл. 2.4) показывает, что использование мультивейвлетной полиморфной сети MWNN дает меньшее MSE-прогнозирование по сравнению с MSE гибридной модели Чанга на 41,77%, 16,50% и 22,92% соответственно [17].

Таблица 2.4 – Ошибки прогнозов Exchange rate Data (все значения MSE должны быть умножены на 10^{-5}).

	1 месяц		6 месяцев		12 месяцев	
	MSE	MAD	MSE	MAD	MSE	MAD
ARIMA	3.68493	0.005016	5.65747	0.0060447	4.52977	0.0053597
ИНС	2.76375	0.004218	5.71096	0.0059458	4.52657	0.0052513
Гибридная модель Чанга	2.67259	0.004146	5.65507	0.0058823	4.35907	0.0051212
MWNN	1.55629	0.002873	4.72234	0.0052145	3.31015	0.0042310

2.4 Модель прогноза многомерного временного ряда на основе глубоких нейронных сетей

Рассматривается многомерный временной ряд $\mathbf{x}_t$ размерностью S. Известны K значений этого временного ряда, взятые через равные промежутки времени: $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_N$ и необходимо прогнозировать значения одного из элементов этого ряда в будущий момент времени с упреждением прогноза f: $x_{1,N+f}$, используя для этого p его предыдущих отсчетов. Для получения прогноза необходимо построить матрицы входных данных:

$$\mathbf{x}_n = \begin{pmatrix} x_{1,n} & x_{1,n+1} & \cdots & x_{1,n+p-1} \\ x_{2,n} & x_{2,n+1} & \cdots & x_{2,n+p-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{S,n} & x_{S,n+1} & \cdots & x_{S,n+p-1} \end{pmatrix}^T, n = 1,2,...,K-p,$$

последовательность которых задает тензор обучающих данных:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{N-p-f} \\ \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_{N-p-f+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_p & \mathbf{x}_{p+1} & \cdots & \mathbf{x}_{N-f} \end{pmatrix},$$

и целевой вектор

$$\mathbf{y} = (x_{1,p+f}, x_{1,p+f+1}, ..., x_{1,N}).$$

Тензор обучающих данных и целевой вектор необходимо использовать для настройки параметров глубокой нейронной сети. Если на вход обученной нейронной сети подать матрицу

$$\begin{pmatrix} x_{1,N-p-f+1} & x_{1,N-p-f+2} & \cdots & x_{1,N} \\ x_{2,N-p-f+1} & x_{2,N-p-f+2} & \cdots & x_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{S,N-p-f+1} & x_{s,N-p-f+2} & \cdots & x_{s,N} \end{pmatrix}^T,$$

на выходе должен получиться искомый прогноз x^1_{N+f}, аналогично можно получить прогнозы x^1_{N+f+1}, x^1_{N+f+2} и так далее, соответствующим образом формируя целевой вектор. Все входы и цели в нейронной сети должны быть тензорами чисел с плавающей точкой (или в особых случаях тензорами целых чисел).

Данные нормируются с помощью *z*-оценок, чтобы обеспечить обучение нейронной сети:

$$s' = \frac{s - \mu}{\sigma},$$

где s – исходное значение признака, s' – нормированное значение, μ – среднее значение признака, а σ – стандартное отклонение. Величины μ и σ должны вычисляться по обучающей выборке. При этом большая часть входных данных принимает значения от -3 до +3. Веса сети инициализируются небольшими случайными значениями, не слишком далекими от оптимальных, что и обеспечивает возможность обучения глубокой нейронной сети с помощью алгоритмов многомерной оптимизации RMSprop, Adam или Nadam.

Архитектура используемых LSTM и GRU сетей показана на рис. 2.11, они состоят из LSTM и полносвязного слоя (Dense) с функцией активации *softmax*, необходимого для

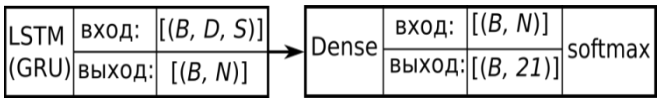


Рисунок 2.11 – Архитектура используемых глубоких нейронных сетей GRU и LSTM

получения итогового прогноза, D – количество предыдущих отсчетов временного ряда, необходимых для прогноза, это значение в несколько раз превышает дальность прогноза, а N – количество LSTM или GRU ячеек в

сети. При большом значении этого параметра после нескольких эпох обучения возникает эффект переобучения, т.е. ошибка на тестовой выборке становится намного больше, чем на обучающей. При малом значении сеть не обладает достаточной репрезентативной емкостью, в результате значение ошибки остается очень большим даже после множества эпох обучения. Наконец, S – количество признаков в одной выборке.

На рис. 2.12 показана архитектура других используемых глубоких нейронных сетей.

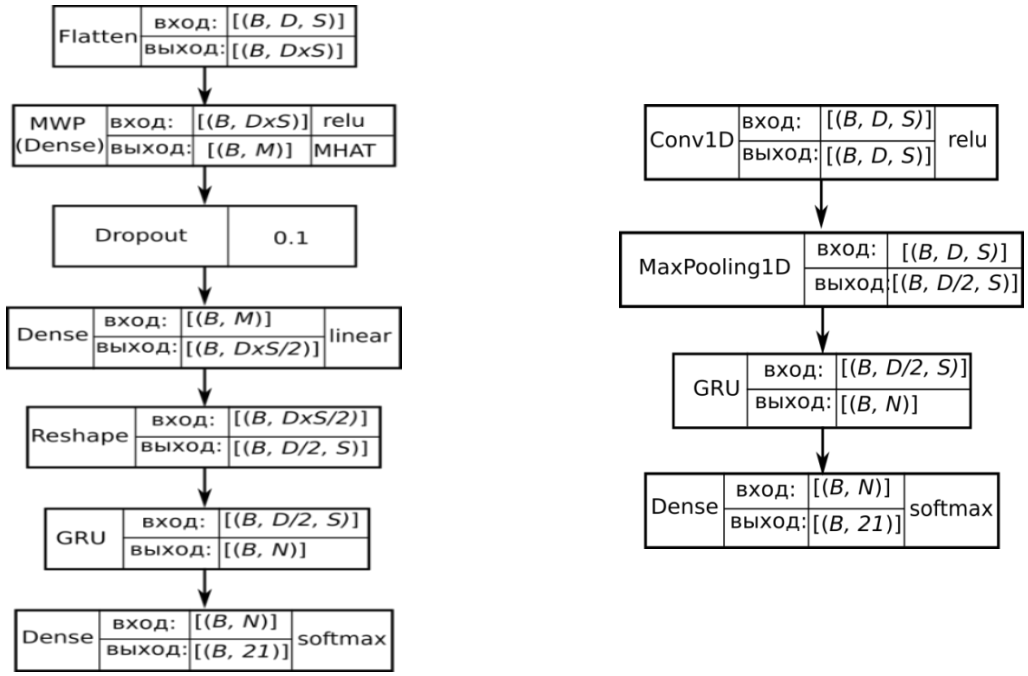


Рисунок 2.12 – Архитектуры используемых глубоких нейронных сетей

Основная идея, положенная в их основу, описана в работе [28] и заключается в том, чтобы объединить скорость и легкость сетей прямого распространения с временной памятью рекуррентных нейронных сетей.

Сеть прямого распространения (полносвязная, вейвлет-сеть или сверточная) превращает длинную входную последовательность в более короткую последовательность высокоуровневых признаков, а затем последовательность выделенных признаков подается на вход рекуррентной части глубокой нейронной сети (рис. 2.13). Этот прием оказывается особенно выгодным, когда имеющиеся последовательности настолько длинны, что их затруднительно обработать с помощью рекуррентной сети.

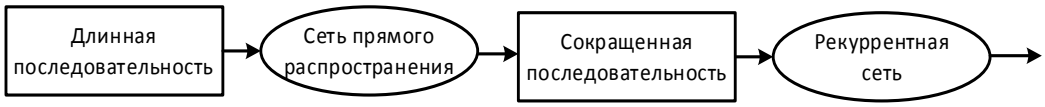


Рисунок 2.13 – Объединение сети прямого распространения и рекуррентной сети для обработки длинных последовательностей

Слой Flatten, показанный на рис. 2.13, преобразует входной тензор с формой [B, D, S] в матрицу с формой [B, D×S], каждая строка которой

затем поступает на вход сверточного полносвязного или мультивейвлетного полиморфного слоя (Multi Wavelet Polymorhic, MWP), схема работы которого показана на рис. 2.14.

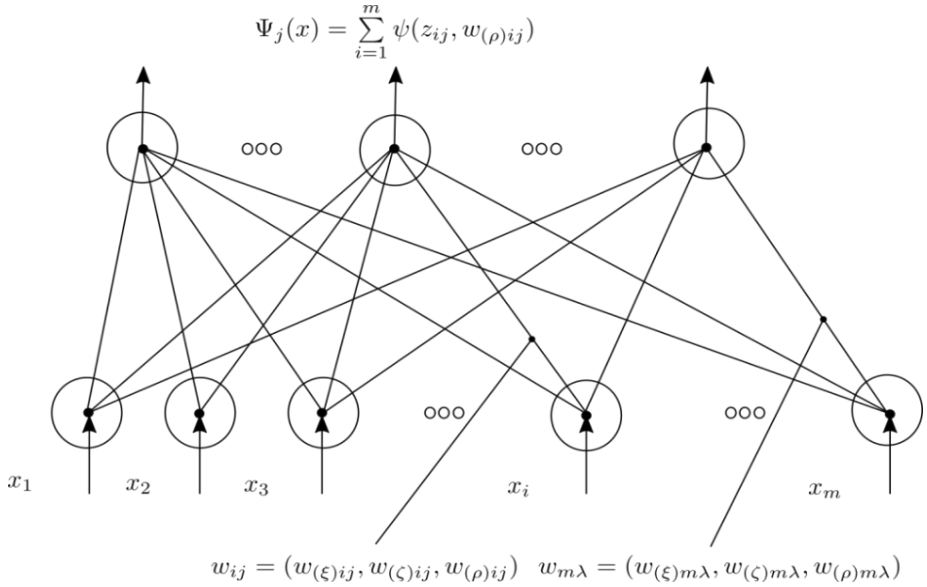


Рисунок 2.14 – Схема работы мультивейвлетного полиморфного слоя

Для прогнозирования использовался слой, являющийся упрощенным для более быстрых вычислений вариантом мультивейвлетной полиморфной сети, описанной выше. Выходы этого слоя определяются как:

$$\mathbf{\Psi}_j(\mathbf{x}) = \sum_{i=1}^{m} \psi(z_{ij}),$$

где $\mathbf{\Psi}_j(\mathbf{x})$ – j-й выход сети, многомерный вейвлет, задающийся как сумма m скалярных вейвлетов, $\mathbf{x}$ – вектор входных данных, m – количество входов, ψ – материнский вейвлет, и

$$z_{ij} = \frac{x_i - w_{(\xi)ij}}{w_{(\zeta)ij}}.$$

В этом выражении $i=1, \ldots, m$, $j=1, \ldots, \lambda$, где λ – число вейвлет-нейронов слоя, равное количеству выходов, $w_{(\xi)ij}$ – параметры сдвигов и $w_{(\zeta)ij}$ – параметры масштабов вейвлетов. В качестве материнского вейвлета наиболее часто используется полиморфный вейвлет MHAT:

$$\psi(z_{ij}, w_{(\rho)_{ij}}) = (w_{(\rho)_{ij}} - z_{ij}^{\ 2}) e^{-\frac{z_{ij}^{\ 2}}{2}},$$

где $w_{(\rho)ij}$ – дополнительный настраиваемый параметр формы вейвлета (рис.2.15).

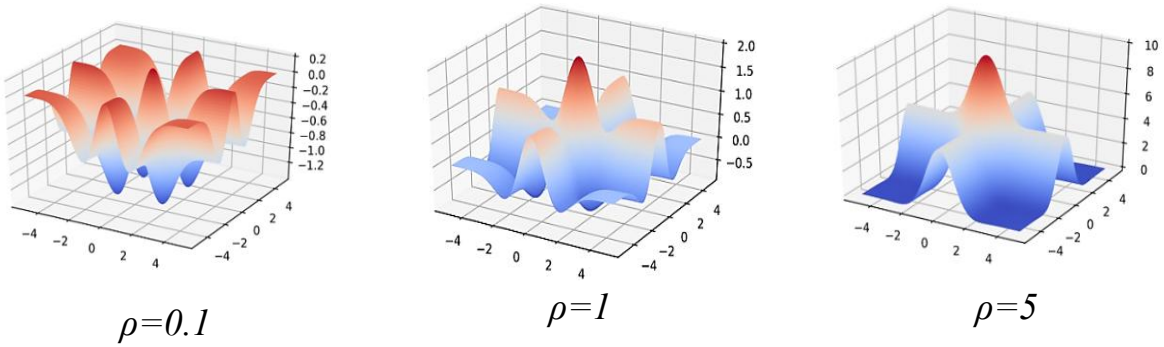


Рисунок 2.15 – Полиморфный MHAT-вейвлет

Кроме полносвязной и вейвлет-сети, для сокращения длины входной последовательности используется также одномерная сверточная сеть. Схема работы одномерного сверточного слоя (Conv1D) показана на рис. 2.16.

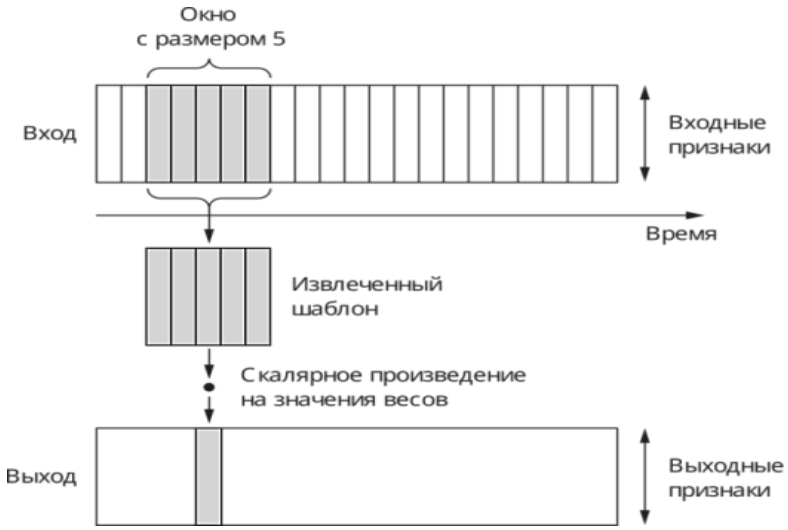


Рисунок 2.16 – Схема работы одномерного сверточного слоя

Сети, содержащие такой слой, способны распознавать локальные шаблоны во временном ряду. Поскольку к каждому шаблону применяются одни и те же преобразования, тот или иной шаблон, найденный в некоторой части временного ряда, может быть опознан и в другом месте, что делает преобразования, выполняемые одномерными сверточными сетями, инвариантными во времени. Слой MaxPooling1D используется для снижения разрешения входной последовательности, возвращая максимальное значение из соседних входных значений. Архитектура всех описанных выше полносвязных, вейвлет- и сверточных нейронных слоев выбрана таким образом, чтобы сократить входную последовательность в 2 раза.

Для борьбы с переобучением применяется прореживание (dropout), заключающееся в удалении (обнулении) некоторого процента случайно выбираемых выходов предыдущего слоя. На этапе тестирования прореживание не производится; вместо этого выходные значения слоя делятся на коэффициент, зависящий коэффициента прореживания, чтобы компенсировать разницу на этапах тестирования и обучения. Как было

показано в работе [26], применение одной той же маски прореживания к каждому интервалу времени позволяет сети правильно распространить свою ошибку обучения во времени и добиться тем самым более точного прогноза.

2.5 Сравнительный анализ прогностических возможностей моделей глубокого обучения

2.5.1 Прогноз дальности видимости в аэропорту «Манас» с использованием данных SYNOP. Прогноз погодных факторов, являясь, наверное, одной из наиболее старейших задач прогноза, до сих пор привлекает внимание многих ученых ввиду большого влияния климата на различные стороны человеческой жизни. В настоящее время в свободном доступе находятся большие объемы данных, полученные с помощью глобальной сети метеостанций, что позволяет исследовать скрытые в них закономерности с применением методов глубокого обучения для получения более точных прогнозов погодных факторов. Особенно большое значение имеет прогноз погоды в зоне полетной навигации, где одним из важнейших факторов на всех этапах полета является дальность видимости.

По данным ИКАО (международной организации гражданской авиации), более 90% аварий и катастроф происходит в условиях ограниченной видимости, из них 80% – на посадке. Полеты в горной местности относятся к полетам в особых условиях. Горы оказывают значительное влияние на аэродинамические характеристики воздушных судов, работу средств навигации и метеорологические условия. Основным явлением, затрудняющим или исключающим выполнение полетов горной местности, является туман. Возможность выполнения посадки в условиях тумана определяется эксплуатационными минимумами – высотой принятия решения и вертикальной видимостью на взлетно-посадочной полосе, которые зависят от типа светотехнического оборудования аэродрома. То есть возможность взлета и посадки самолета ограничена не только погодными факторами, но и оборудованием, установленным в аэропорту и на воздушном судне. Однако в условиях очень плохой горизонтальной видимости может оказаться невозможным управлять самолетом даже на взлетно-посадочной полосе и рулежных дорожках, ведущих к стоянке самолетов.

Среднее в году число дней с туманами в центральной по высоте зоне Бишкека равно 29, а максимально возможное 45. Эти показатели можно распространить и на всю северную зону города, так как метеостанция Чуйская, расположенная примерно на 150 м ниже, имеет такие же данные наблюдений. Следовательно, туманы в зоне аэропорта «Манас» являются достаточно обыденным явлением в холодную половину года, поздней осенью, зимой и ранней весной они представляют вполне ощутимые сложности для авиации, связанные с возможной отменой и переносом

рейсов, а также для работы наземного транспорта, что приводит к большим финансовым и временным потерям. В настоящее время вопрос прогноза дальности видимости исследуется многими учеными, например, в работе, показана возможность прогнозирования видимости в аэропорту Урумчи с помощью многослойной нейронной сети.

Таким образом, прогноз видимости в аэропорту «Манас» в связи с его географическим расположением, особенностями климата является актуальной задачей. В настоящем разделе исследуются различные архитектуры глубоких нейронных сетей для прогнозирования временного ряда горизонтальной дальности видимости.

Исходные данные для исследования были получены с сайта rp5.ru. Данные наблюдений в формате SYNOP (Surface SYNOPtic Observations – формат для оперативной передачи данных приземных гидрометеорологических наблюдений с сети станций гидрометслужбы, расположенных на суше) поступают на сайт восемь раз в сутки, через каждые три часа. В настоящее время сайт предоставляет прогнозы погоды для 172500 населённых пунктов и данные наблюдений, выполненных на 10400 метеостанциях. Из всего набора доступных данных для проведения эксперимента были выбраны временные ряды горизонтальной видимости, температуры, давления, направления ветра, скорости ветра и влажности за 14 лет с апреля 2005 г. по апрель 2019 года. В целом рассматривались данные 5 станций, расположенных примерно на одинаковой высоте над уровнем моря (максимальная разность высот составила немногим более 200 м) в радиусе 150 км от аэропорта «Манас» (табл. 2.5). Обучающая выборка составила 30000 отсчетов, тестовая – 10000, оставшиеся данные (рис. 2.17) использовались для построения контрольных прогнозов.

Отсутствующие по тем или иным причинам значения были заполнены предыдущими существующими значениями. Затем данные, содержащие качественные оценки и представленные в текстовом виде, такие как направление ветра (например, ветер, дующий с востока), были векторизованы путем их кодирования целыми числами в порядке их появления в исходных данных.

Таблица 2.5 – Станции-источники данных

Станция	Страна	Широта, град.	Долгота, град.	Высота н.у. м, м.	Удаление, км
Манас	Кыргызстан	43.067	74.483	637.0	0
Бишкек	Кыргызстан	42.850	74.533	760.0	24
Токмак	Кыргызстан	42.833	75.283	817.0	70
Отар	Казахстан	43.533	75.250	743.0	81
Хантау	Казахстан	44.217	73.800	504.0	139

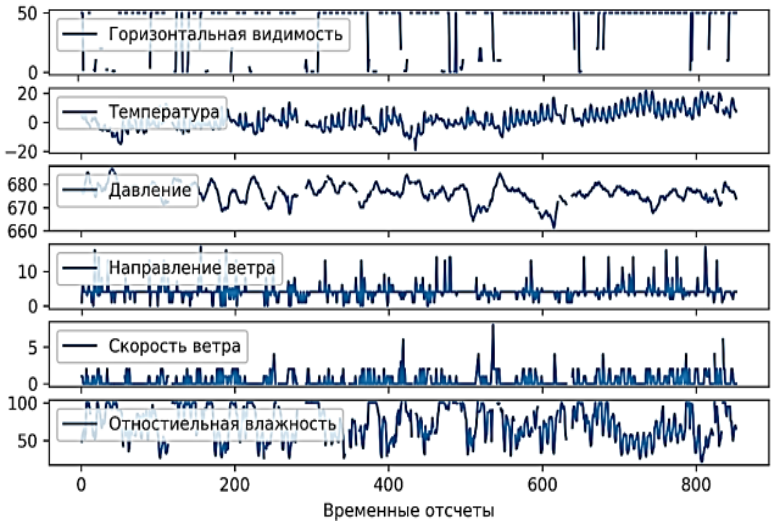


Рисунок 2.17 – Данные о погоде в аэропорту «Манас» в период с 15 декабря 2018 по 1 апреля 2019 г. (формат SYNOP)

Так как дальность видимости в формате SYNOP может принимать 21 дискретное значение, эти значения также были закодированы целыми числами, а ошибка сети вычислялась по формуле, категориальной перекрестной энтропии (cross entropy, CE):

$$CE = -\frac{1}{B}\sum_{i=1}^{B}\sum_{c=1}^{C} 1(y_i \notin C_c)\ \ln p(y_i \in C_c), \tag{2.9}$$

где B – количество выборок исходных данных на каждом шаге обучения – параметр, от которого зависит скорость и точность обучения, его значение принято равным 128, так как при этом значении получается меньшая ошибка прогноза на тестовой выборке; C – количество возможных дискретных значений; $1(y_i \in C_c)$ – функция, возвращающая 1, если i-е наблюдение равно дискретному значению C_c, а иначе 0; $p(y_i \in C_c)$ – вероятность, предсказанная моделью того, что i-е наблюдение принимает дискретное значение C_c.

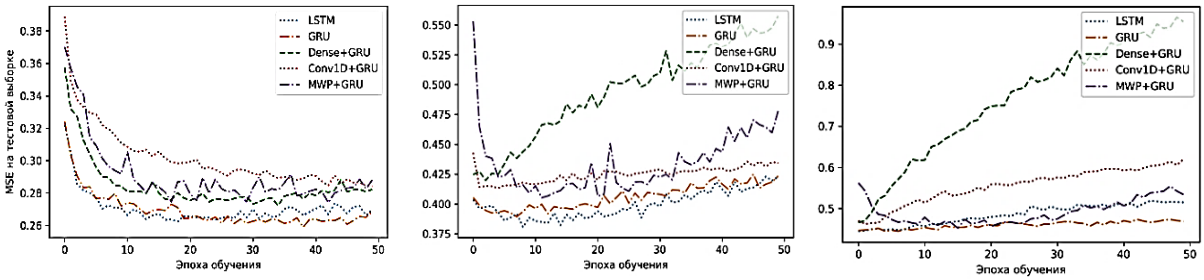
После перебора множества возможных вариантов количество нейронов N принято равным 32. Для регуляризации выходов, сформированных рекуррентными слоями, такими как GRU и LSTM, к внутренним рекуррентным слоям блоков применяется постоянная во времени маска прореживания с коэффициентом 0,2. Для сравнения предлагается использовать также базовый метод прогноза, заключающийся в том, что через 3 часа дальность видимости будет такой же, как и сейчас, а через 12 и 24 часа будет равна 50 км – самому часто встречающемуся возможному значению. Рассматривалась возможность прогноза на 1 шаг вперед, с дальностью 3, 12, 24 часа. При этом использовались данные 1, 3 и 5, станций из таблицы 2.5 в порядке их удаления от аэропорта «Манас». В таблице 2.6 показан процент верно выполненных прогнозов сетями с различными архитектурами, обученными по алгоритму RMSProp на

протяжении 50 эпох по 500 шагов, в каждом из которых ошибка вычислялась по формуле (2.8).

Таблица 2.6 – Результаты вычислительного эксперимента с использованием данных SYNOP

Упреждение прогноза	3 часа			12 часов			24 часа		
Модель	Количество станций								
	1	3	5	1	3	5	1	3	5
LSTM	93.36	92.73	89.48	89.87	89.78	89.59	89.57	89.57	89.57
GRU	93.41	92.91	89.48	89.97	89.64	89.59	89.57	89.57	89.57
Dense+GRU	93.39	92.37	89.48	89.59	89.59	89.59	89.57	89.57	89.57
Conv1D+GRU	92.19	91.60	89.48	89.59	89.59	89.59	89.57	89.57	89.57
MWP+GRU	93.21	92.62	89.48	89.59	89.59	89.59	89.57	89.57	89.57
Базовый метод	93.59			89.59			89.57		

Из табл. 2.6 видно, что для прогноза с упреждением 3 часа процент верно выполненных с помощью глубоких сетей прогнозов немного ниже, чем у прогноза, полученного с помощью базового метода, а для упреждения 24 часа равен ему. Другими словами, обученные нейронные сети практически точно воспроизводят предложенный базовый метод прогноза. Значение верности 89,48%, полученное при обучении сетей по данным 5 метеостанций, соответствует предположению, что дальность видимости будет всегда принимать значение 50 км. Как показывают результаты вычислительного эксперимента, использование данных нескольких метеостанций не является целесообразным, так как при этом точность прогноза только ухудшается. На рисунке 2.18 показано изменение CE, найденного по формуле (2.9) в процессе обучения сетей.



а) б) в)

Рисунок 2.18 – Зависимость CE глубоких нейронных сетей от эпохи обучения на тестовой выборке для прогноза по данным одной станции с дальностью: а)3 часа, б) 12 часа, в) 24 часа

Как можно видеть из графиков, уже при упреждении прогноза в 12 часов начинает проявляться эффект переобучения, так как с увеличением упреждения прогноза увеличивается количество предыдущих отсчетов, необходимых для его получения, а, следовательно, уменьшается количество образцов в обучающей выборке. Это говорит о том, что использование сетей с одинаковой архитектурой при разной дальности прогноза не является оптимальным вариантом. При увеличении дальности

прогноза емкость сетей, т.е. количество нейронов в скрытых слоях, необходимо уменьшать для того, чтобы избежать переобучения.

На рисунке 2.19 показаны результаты прогнозов с упреждением 12 часов, полученных с применением нейронных сетей GRU и LSTM, обученных до достижения минимального значения CE.

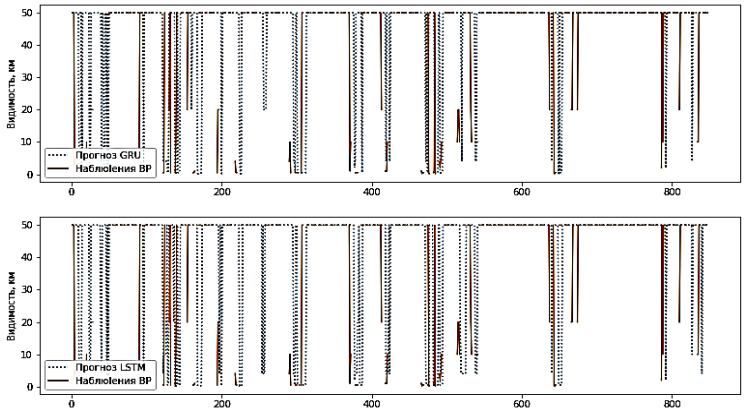


Рисунок 2.19 – Прогноз дальности видимости в аэропорту «Манас» с упреждением 12 часов с помощью GRU и LSTM нейронных сетей (период 01.2018 – 04.2019 г.)

Как можно видеть, при упреждении прогноза 12 часов нейронные сети хотя и не комбинируют два варианта, предложенных в базовом методе, но в целом не дают прогноз, значительно превосходящий по точности базовый метод. Это можно объяснить тем, что временной интервал между возникновением совокупности погодных условий, являющихся причиной ухудшения или улучшения дальности видимости и возникновением этого явления, составляет менее 3 часов. Следовательно, для прогноза дальности видимости необходимо использовать данные о погоде с меньшим шагом по времени, чем были использованы в данном случае.

2.5.2 Прогноз дальности видимости в аэропорту «Манас» с использованием данных METAR. Кроме данных наблюдений о погоде в формате SYNOP, сайт rp5.ru для аэропорта «Манас» и других аэропортов представляет данные в формате METAR, (METeorological Aerodrome Report) – авиационный метеорологический формат для передачи сводок о фактической погоде на аэродроме.

Из всего набора доступных с 2014 года данных для проведения эксперимента были выбраны временные ряды этих параметров за 6 лет и 9 месяцев – с марта 2014 г. по ноябрь 2019 года. Данные за этот период даны с интервалом дискретизации 30 мин, тогда как в период с августа 2013 года по февраль 2014-го данные собирались с интервалом в 1 час. Обучающая выборка составила 68175 отсчетов, тестовая – 22725, оставшиеся данные – 900 отсчетов. Данные, содержащие качественные оценки, были векторизованы, как и ранее, путем их кодирования целыми числами в порядке их появления в исходных данных (рис. 2.20) и затем нормированы с помощью *z*-оценок.

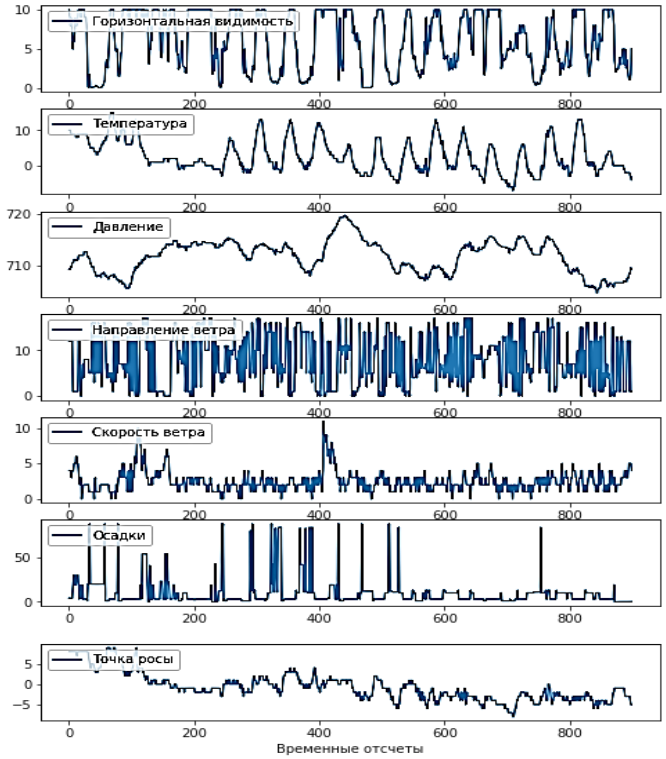


Рисунок 2.20 – Данные о погоде в аэропорту «Манас» в период со 2 по 21 ноября 2019 г. (формат METAR)

Дальность видимости в формате METAR может принимать 57 дискретных значений, эти значения были закодированы целыми числами в порядке увеличения дальности видимости, а ошибка сети вычислялась с помощью «шарнирной» функции ошибки:

$$H(\hat{\mathbf{y}}, \mathbf{y}) = \max(0, 1 - \mathbf{y} \cdot \hat{\mathbf{y}}), \tag{2.10}$$

где $\hat{\mathbf{y}}$ – выход нейронной сети, а $\mathbf{y}$ – закодированное в виде вектора значение класса:

$$y_i = \begin{cases} 1, C_i = C_c, \\ 0, C_i \neq C_c \end{cases}, i = 1..57,$$

где C_c – наблюдаемое значение дальности видимости.

В табл. 2.7 показано минимальное значение функции ошибки, достигнутое сетями с различными архитектурами, обученными по алгоритму RMSProp на протяжении 50 эпох по 500 шагов, в каждом из которых ошибка вычислялась по формуле (2.10).

Таблица 2.7 – Результаты вычислительного эксперимента с использованием данных METAR

Модель	Упреждение прогноза, ч					
	0.5	1	1.5	2	2.5	3
MWP	0.4578	0.4823	0.6027	0.6245	0.6465	1.9943
MWP+GRU	0.4346	0.4561	0.4778	0.4936	0.5090	0.5110
Conv1D+GRU	0.4402	0.4571	0.4714	0.4854	0.4984	0.5036
GRU	0.4383	0.4547	0.4711	0.4831	0.4972	0.5092

Как видно из таблицы, ошибка достаточно быстро увеличивается с увеличением упреждения прогноза. На рис. 2.21 показаны результаты прогнозов с упреждением 30 мин, полученных с применением различных нейронных сетей, обученных до достижения минимального значения *H*.

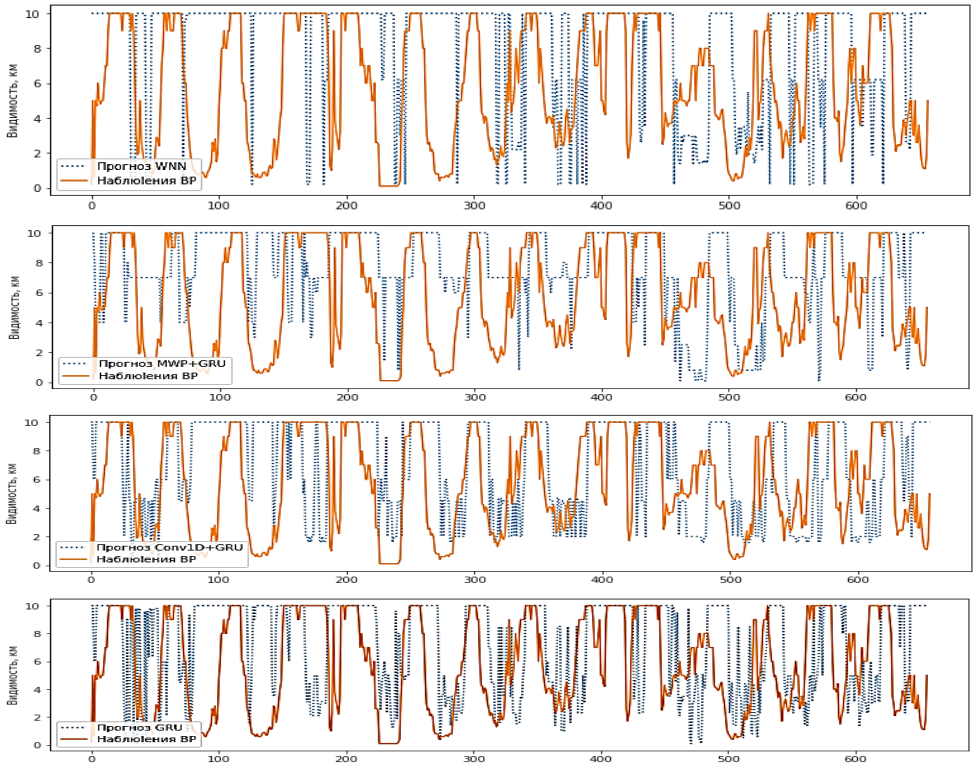


Рисунок 2.21 – Прогноз дальности видимости в аэропорту «Манас» с упреждением 30 минут (период 1.10.2019 – 21.10 2019 г.)

В случае применения MWP-сети – MAE, найденное по формуле (7.5), равно 1,114 км; MWP+GRU-сети – MAE= 0,8770 км; Conv1D+GRU-сети – MAE 0,7345 км; GRU-сети – MAE= 0,7311 км.

Для сравнения MAE базового прогноза, основанного на предположении о том, что дальность видимости через 30 мин. будет такой же, как сейчас, составляет 1,569 км. Следовательно, использование GRU глубокой нейронной сети позволяет уменьшить ошибку прогноза более чем в 2 раза, по сравнению с базовым методом прогноза и 1,5 раза по сравнению с поверхностной нейронной сетью [18].

2.6 Модели прогноза уровня загрязненности атмосферного воздуха г. Бишкек

2.6.1 Эколого-климатические особенности г. Бишкек. В последнее время в литературе большое внимание уделяется исследованиям процессов загрязнения атмосферного воздуха городов в связи с ухудшающейся экологической ситуацией.

Отмечено, что «физико-географические и климатические условия г. Бишкек, а также относительная замкнутость Чуйской долины способствуют возникновению интенсивных приземных и приподнятых

инверсий, что ведет к формированию высокого потенциала загрязнений» [29]. В последние годы из-за быстрого роста плотности населения, плотности хаотичной застройки, возросшего потребления энергии на отопление, увеличения количества автотранспорта качество атмосферного воздуха в городе ухудшилось. Поэтому задача анализа динамики изменения концентраций вредных веществ, в частности твердых частиц PM2.5 и индекса качества воздуха (Air Quality Index, AQI) города Бишкек, а также построения моделей для прогноза их уровней, представляет особый интерес.

Универсальных моделей для прогноза концентрации PM2.5 и AQI быть не может, поскольку в них необходимо учитывать региональные природные, экономические, антропогенные и климатические особенности территории. В литературе опубликовано немало работ, связанных с построением моделей прогноза качества воздуха для различных регионов и городов. Для г. Бишкек таких работ практически нет, главным образом, ввиду того, что открытая информация об уровне загрязненности атмосферного воздуха в городе стала публиковаться на сайте AirNow (https://www.airnow.gov/) лишь с февраля 2019 г. Основываясь на этой информации, авторами выполнен ряд работ, связанных с оценкой временной изменчивости концентраций в атмосферном воздухе твердых частиц PM2.5 и индекса качества воздуха AQI. На основе данных наблюдений для различных периодов 2019–2020 гг. разработаны интегрированные модели авторегрессии – скользящего среднего (AutoRegressive Integrated Moving Average model, ARIMA-модели) для краткосрочного прогноза концентраций в атмосферном воздухе твердых частиц PM2.5 и индекса качества воздуха AQI [30, 31], линейные мультирегрессионные модели процессов загрязнения на основе регрессионного анализа метеорологических факторов и концентраций частиц PM2.5 [31], модель краткосрочного прогноза концентраций PM2.5 с учетом метеорологических факторов на основе обобщенно-регрессионной нейронной сети GRNN [31], модель среднесрочного прогноза класса AQI на основе классификатора индекса качества воздуха на базе LSTM-нейронных сетей [32]. В настоящей работе обсуждены некоторые особенности и возможности каждой из перечисленных моделей, а также приведены результаты прогнозирования PM2.5 и AQI на их основе.

2.6.2 Модели и методы.

ARIMA-модель. ARIMA-модели – наиболее популярные и эффективные статистические модели для прогноза временных рядов (ВР). В основу этих моделей положена идея, что «будущие значения временного ряда можно представить как некоторую линейную функцию прошлых наблюдений, дополненную случайной ошибкой (белым шумом).

В общем виде ARIMA(p,d,q) - модель имеет вид:

$$\phi(B)(1-B)^d y_{ft} = \theta(B)\varepsilon_t ,$$

где y_{ft} – значения модельного ряда в момент времени t; ε_t – случайная ошибка с нулевым средним и постоянной дисперсией; $\phi(B)$, $\theta(B)$ – полиномы степени p и q, B – лаговый оператор, с учетом которого $By_{ft} = y_{ft-1}$, $B^2y_{ft} = y_{ft-2}$, … , $B^dy_{ft} = y_{ft-d}$, $B^j\varepsilon_{ft} = \varepsilon_{ft-j}$, $j=0,1,\ldots$; d – порядок взятия последовательной разности $\Delta y_t = y_{ft-1} - y_{ft}$, которая вычисляется для того, чтобы сделать ряд стационарным» [34]. Для того чтобы определить порядок (p, d, q) ARIMA-модели, обычно используются автокорреляционная (ACF) и частичная автокорреляционная (PACF) функции ВР- наблюдений. После выбора порядка модели, продолжая следовать методологии Бокса-Дженкинса, «оценивают параметры модели, исходя из минимума «разногласий» между модельным ВР- и ВР-наблюдений. Затем адекватность модели проверяется на основе статистической обработки прогнозных значений ряда и анализа остатков, т.е. разницы между значением ряда y_t и его предсказанным на модели значением y_{ft} : $e_t = y_t - y_{ft}$. Если модель не адекватна, должна быть определена новая предварительная модель. Этот трехэтапный процесс построения модели обычно повторяется несколько раз, пока удовлетворительная модель будет окончательно выбрана» [34]. В силу того, что инерционность во ВР загрязнений велика, ARIMA-модели широко используют для прогноза загрязнений атмосферного воздуха.

Мультирегрессионная модель прогноза. ARIMA-модели позволяют учитывать динамику процесса загрязнения, но не учитывают метеорологические факторы. Поэтому интересным представляется использование модели множественной регрессии, позволяющей учитывать большое количество факторов, характеризующих причинно-следственные связи, имеющие место в объекте исследования (в процессе загрязнений воздуха в нашем случае).

Как показано в [34], «общее уравнение линейной множественной регрессии может быть записано как:

$$y_{ft}=b+k_1x_{1t}+ k_2x_{2t}+ \ldots + k_mx_{mt}+e_t,$$

где b – константа регрессии, k_1, k_2,…, k_m – коэффициенты регрессии, m – количество независимых переменных (факторов, предикатов). Значения константы и коэффициентов определяются с использованием метода наименьших квадратов, который минимизирует ошибку $e_t=(y_t - y_{ft})$ или остатки модели. Для построения регрессионной модели обычно используют метод ступенчатой (пошаговой) регрессии, суть которого заключается в отборе из большого количества предикатов x небольшой подгруппы переменных, которые вносят наибольший вклад в вариацию зависимой переменной» [35]. При построении мультирегрессионной модели в работе [31] концентрации PM2.5 рассматривались как зависимые переменные, а метеопараметры (температура воздуха, температура точки росы, влажность воздуха, скорость ветра, давление, интенсивность осадков) рассматривались как независимые переменные.

Модель прогноза на основе обобщенно-регрессионной нейронной сети. Модели на основе искусственных нейронных сетей (ИНС) активно используются в прогнозировании, поскольку они обладают способностью находить зависимости между данными путем анализа большого количества подобных примеров или паттернов. Выбор обобщенно-регрессионной нейронной сети (Generalized Regression Neural Network, GRNN) [36] как варианта ИНС для прогнозирования степени загрязненности атмосферного воздуха обусловлен следующими ее преимуществами: архитектура сети фиксирована и не нуждается в определении, и эта сеть характеризуется достаточно высокой скоростью обучения. В литературе достаточно подробно описана архитектура этой сети: «GRNN-сеть содержит два слоя: радиально-базисный слой с числом нейронов, равным числу элементов обучающего множества, и линейный слой, а окончательная выходная оценка сети получается как взвешенное среднее выходов по всем обучающим наблюдениям, где величины весов отражают расстояние от этих наблюдений до той точки, в которой производится оценивание (и, таким образом, более близкие точки вносят больший вклад в оценку)» [36].

Модель прогноза на основе LSTM-нейронной сети. На степень загрязнения воздуха влияют метеорологические условия не только на текущий момент, но и история изменения этих условий. Это говорит об инерционности процессов загрязнения. Очевидно предположить, что сети, учитывающие историю наблюдений, будут работать лучше, чем сети, основанные на оценке исключительно текущих данных. Для анализа исторических данных, как правило, используются рекуррентные нейронные сети. Однако степень загрязнения воздуха может реагировать на различные факторы с разной скоростью. Поэтому история каждого фактора должна быть оценена по-разному – для каких-то факторов важны только последние данные, влияние других может сказываться продолжительное время. Учитывая этот факт, для прогнозирования степени загрязненности воздуха интересным представляется использование LSTM-сетей (Long Short-Term Memory, LSTM – долгая краткосрочная память). Их архитектура содержит так называемые «фильтры, которые в процессе обучения настраиваются сохранять/забывать информацию выборочно о различных факторах и таким образом могут обнаруживать как длинные, так и короткие шаблоны в данных» [26].

Структура используемой LSTM-сети представлена в [32]. «Сеть содержит: 1) входной слой длиной 6 (6 признаков входного вектора ВВ), который принимает последовательность длиной *S* векторов признаков; 2) первый скрытый слой – слой прямого распространения с числом нейронов 100 и тангенциальной функцией активации, отображает входные векторы в векторы большей длины, для дальнейшего внесения шума в данные; 3) второй скрытый слой – слой регуляризации, который меняет некоторый

процент значений выхода предыдущего слоя для предотвращения переобучения (вносит шумы); 4) третий скрытый слой – LSTM-сеть с 50 нейронными модулями и тангенциальной функцией активации как основной классифицирующий слой; 5) четвертый скрытый слой – слой прямого распространения с числом нейронов 10 и тангенциальной функцией активации; 6) выходной слой – слой с 2 нейронами и функцией активации *SOFTMAX*. Функция взвешивает входы и предсказывает вероятности активации каждого нейрона. При этом сумма выходов нейронов всегда равна 1. Все слои сети полносвязные, т.е. каждый нейрон имеет связь с каждым предыдущим нейроном, а для рекуррентных слоев (LSTM-сеть) каждый вход слоя также связан с каждым выходом слоя» [32].

2.6.3 Некоторые решения задач прогнозирования.

Данные для исследования. Данные, представленные в настоящей работе и использованные в работах [30-32] – данные о концентрациях PM2.5 в mkg/m^3 и индексе качества воздуха AQI г. Бишкек, размещенные на сайте https://www.airnow.gov/. Этот сайт публикует данные начиная с 06.02.2019 по настоящее время (период измерения – 1 час) в CSV-формате.

На рис. 2.22 представлено изменение содержания PM2.5 в атмосферном воздухе и индекс качества воздуха г. Бишкек за период с 09.02.2019 по 31.03.2020 г. с интервалом в 3 часа. На графиках отчетливо отмечаются два скачка в значениях PM2.5 и AQI: 23.03.2019 г. значения резко падают и 01.11. 2019 г. значения резко возрастают. Указанные даты – конец и начало отопительного сезона в городе. Среднее (математическое ожидание) *mean* и стандартное (среднеквадратическое) отклонение σ за весь период наблюдений, а также за периоды 09.02.2019–23.03.2019, 24.03.2019–31.10.2019, 1.11.2019–31.03.2020 представлены в табл. 2.8.

Временные ряды (ВР) на рис. 2.22 явно нестационарные. Это подтверждают и проведенные с использованием функции MatLab *adftest* расширенные тесты Дики-Фуллера.

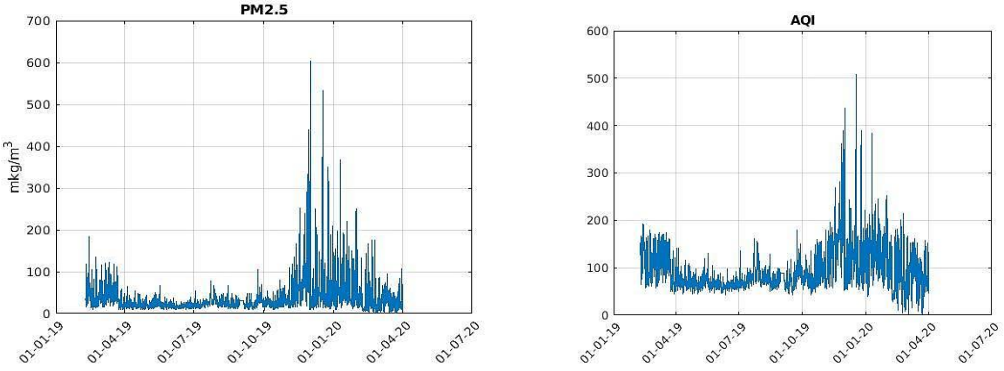


Рисунок 2.22 – Значения концентраций PM2.5 и индекса качества воздуха в г. Бишкек за период 09.02.2019-31.03.2020г.

Для анализа исходных рядов построены автокорреляционные и частичные автокорреляционные функции (ACF и PACF). Анализ ACF

PM2.5 (рис. 2.23) показывает, что они не затухают быстро, что также характерно для нестационарных ВР, и содержат периодичность в 8 лагов, что соответствует 24 часам. Последний значимый лаг на PACF PM2.5 равен 80, что соответствует 10 суткам и не согласуется с данными [36], согласно которым время оседания частиц PM2.5 составляет 5 суток. ACF AQI показывает, что ВР также содержит периодичность в 8 лагов.

Таблица 2.8 – Средние значения и стандартные отклонения концентраций PM2.5 и AQI

Период наблюдений	PM2.5		AQI	
	mean, mkg/m^3	σ, mkg/m^3	*mean*	σ
09.02.2019-25.11.2019	29.6322	21.9179	85.2881	32.1227
09.02.2019-23.03.2019	43.9942	23.8089	115.1541	32.9580
24.03.2019-31.10.2019	22.2688	10.4384	69.9941	21.1958
01.11.2019-31.03.2020	40.7214	122.0221	102.2294	128.8958

Уровень загрязненности атмосферного воздуха в немалой степени зависит от метеорологических факторов. Исследования, широко представленные в литературе, показывают, что «высокие уровни содержания загрязняющих веществ наблюдаются в период длительных неблагоприятных метеорологических условий, которым свойственны температурные инверсии, слабые ветры, туманы» [31].

Для учета влияния метеорологических факторов на уровень загрязнения атмосферного воздуха авторами использованы данные с сервера международного обмена NOAA (США) в формате SYNOP. На этом сервере хранятся измерения метеопараметров, произведенные более чем 90000 наземными станциями мира, в том числе метеостанцией «Бишкек» (WMO_ID=38353). Данные получены через сайт [38] в виде Excel-файла за период с 09.02.2019 по 31.03.2020 г. В [39] «выполнена оценка влияния метеорологических факторов (таких, как скорость ветра, температура, относительная влажность воздуха, температура точки росы, интенсивность осадков и атмосферное давление) на процесс загрязнения воздуха г. Бишкек частицами PM2.5 в период с февраля по ноябрь 2019 г. и выявлены умеренные корреляции (как положительные, так и отрицательные) между концентрациями PM2.5 и метеорологическими параметрами, измеренными в текущий и прошлые сроки измерений» [39]. Эти оценки учтены при разработке моделей прогноза уровня загрязненности воздуха с учетом влияния метеофакторов.

Прогнозирование на ARIMA-модели. Для иллюстрации прогностических возможностей ARIMA-моделей в [30] представлены результаты прогнозирования индекса качества воздуха AQI на основе ARIMA-модели, построенной на данных за период с 24.03.2019 по 29.08.2019г. Поскольку исходный ВР нестационарный, была произведена декомпозиция исходного ряда на трендовую, сезонную и недетерминированную составляющую. Недетерминированная составляющая ВР была разделена на 2 части: обучающую выборку и

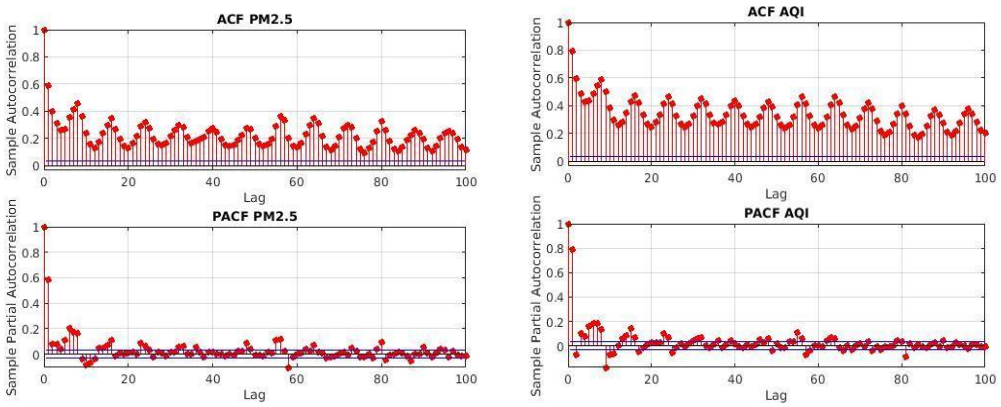


Рисунок 2.23 – Автокорреляционные (ACF) и частичные автокорреляционные функции исходных ВР

тестовую. В результате анализа частичной автокорреляционной функции PACF и после проведения вычислений с различными значениями коэффициентов *p, d* и *q* в ARIMA-моделях получены различные варианты прогнозов. Наименьшая среднеквадратическая ошибка *MSE*=17.14 и наилучший коэффициент детерминации R^2=0.88 соответствуют варианту модели *p=4, d=1, q=0* [30], результат прогноза с использованием которой также представлен на рис. 2.24.

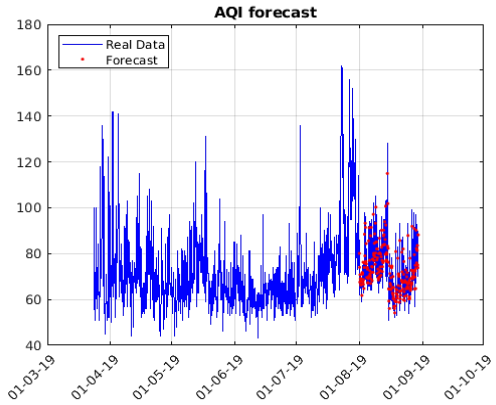


Рисунок 2.24 – Динамика реальных и прогнозных AQI

Прогнозирование на линейной мультирегрессионной модели. Возможности прогнозирования загрязнений воздуха на мультирегрессионных моделях показаны в [31]. Архивные данные были разделены на 2 части: обучающее множество – данные измерения Pm2.5 и метеорологических параметров по г. Бишкек за период с 24.03.2019 по 31.07.2019г; тестовое множество – данные измерения за период с 01.08.2019 по 18.08.2019г. В [15] показано, что для летнего периода времени (24.03.2019–31.10.2019) корреляция между концентрациями PM2.5 и давлением несущественная, а «наилучшие взаимосвязи обнаружены между концентрациями PM2.5 и измерениями температуры воздуха T в предшествующий срок измерений *i-1* (т.е. тремя часами ранее), температуры точки росы Td в срок измерений *i-2* (шестью часами ранее), интенсивностью осадков PR в срок измерений *i-2*, влажностью RH в срок

измерений *i-1*, скоростью ветра Ws в срок измерений *i-5»* [15] . Используя эту информацию, на обучающей выборке 24.03.2019 – 31.07.2019г. была построена линейная мультирегрессионная модель (табл. 2.9), учитывающая все перечисленные относительно существенные факторы.

Таблица 2.9 – Линейные мультирегрессионные модели

Teaching set: 24.03.2019 – 31.07.2019									
$PM2.5(i)= b+ k_1*Td(i-2)+ k_2*T(i-1)+ k_3*PR(i-2)+ k_4*Ws(i-5)+ k_5*RH(i-1)$									
b	k_1	k_2	k_3	k_4	k_5	R	MAPE	MAE	MSE
13.1054	-0.4092	0.4569	-1.9980	-0.0671	0.0550	*0.3183*	0.284	7.440	134.4
$PM2.5(i)= b+ k_1*Td(i-2)+ k_2*T(i-1)+ k_3*PR(i-2)+ k_4*PM2.5(i-1)+ k_5*PM2.5(i-2)$									
b	k_1	k_2	k_3	k_4	k_5	R	MAPE	MAE	MSE
10.4621	-0.4235	0.4930	-2.9192	0.0537	0.0432	0.3514	0.2772	5.867	70.11
Testing set: 01/08/2019-18/08/2019									
$PM2.5(i)= b+ k_1*Td(i-2)+ k_2*T(i-1)+ k_3*PR(i-2)+ k_4*PM2.5(i-1)+ k_5*PM2.5(i-2)$									
b	k_1	k_2	k_3	k_4	k_5	R	MAPE	MAE	MSE
10.4621	-0.4235	0.4930	-2.9192	0.0537	0.0432	0.3514	0.2087	4.395	43.57

В табл. 2.9 также представлены коэффициенты множественной корреляции *R* между концентрациями PM2.5 и метеорологическими параметрами, средняя квадратическая ошибка модели *MSE*, средняя абсолютная ошибка *MAE,* средняя абсолютная процентная ошибка *MAPE*.

Затем, используя метод пошаговой множественной линейной регрессии, получена итоговая регрессионная модель, также представленная в табл. 2:

$$PM2.5(i)= b+ k_1*Td(i-2)+ k_2*T(i-1)+ k_3*PR(i-2)+ k_4*PM2.5(i-1)+ k_5*PM2.5(i-2).$$

В этой модели исключены факторы: скорость ветра *Ws* и относительная влажность *RH* как имеющие наименьшие регрессионные коэффициенты. Для учета инерционности процесса загрязнения в модель включены измерения PM2.5 в *(i-1)*- й и *(i-2)*- й сроки измерения.

Полученная модель применена для восстановления концентрации PM2.5 на тестовой выборке 05.09.2019 – 15.09.2019. На Рисунок 2.25 представлены вычисленные на основе этой модели и измеренные (наблюденные) значения концентраций.

Прогнозирование на нейросетевой модели GRNN. Возможности прогнозирования загрязнений воздуха на нейросетевой модели GRNN показаны в [5]. На первом этапе были исследованы различные варианты обучения и моделирования прогнозных значений PM2.5 с использованием вышеприведенных параметров (исключая давление как несущественный фактор) с различной историей и прогнозом метеослужбы на всем диапазоне сроков измерения *i* (полная выборка, *i*=3, 6, 9, 12, 15, 18, 21, 24 час). Данные были разделены на 2 части: обучающее множество – данные измерения Pm2.5 и метеорологических параметров по г. Бишкек за период 24.03.2019 – 31.07.2019г. и тестовое множество – данные измерения за

период 01.08.2019 –18.08.2019г. С целью повышения точности обучения была проведена декомпозиция измеренных значений PM2.5 по срокам измерения: 03 ч., 06 ч., 09 ч., 12ч., 15 ч., 18 ч., 21 ч., 24 ч. и для каждого срока была получена своя модель.

Сравнение полученных мультирегрессионной и нейросетевой моделей прогноза производилось на выборке 5.09.2019 – 15.09.2019г. и показало, что средняя абсолютная ошибка MAE и среднеквадратическая ошибка MSE, рассчитанные для нейросетевой модели, меньше, чем соответствующие ошибки мультирегрессионной модели [31]. На графиках (рис. 2.25) представлены наблюденные (измеренные) значения концентраций PM2.5 и их прогнозные значения на 3 часа вперед (что соответствует одному сроку наблюдений).

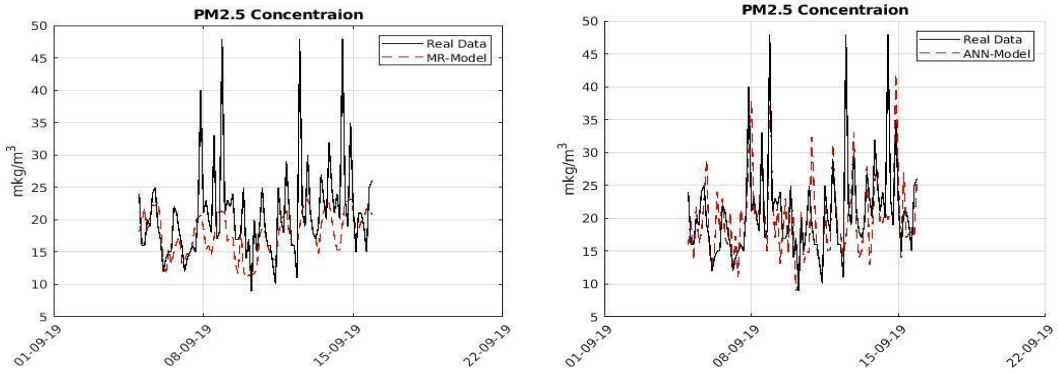


Рисунок 2.25 – Сравнение мультирегрессионной и нейросетевой GRNN моделей

В [31] также показано, что идея декомпозировать данные концентраций PM2.5 и метеопараметры по срокам измерений оправдала себя, поскольку позволила снизить ошибку прогноза.

Прогнозирование класса AQI на основе LSTM-классификатора Анализ распределения AQI по классам в период 06.02.2019 – 31.03.2020г. показал, что 65% наблюдений принадлежит классу «Умеренный» рис. 2.26(а). Количества наблюдений AQI, принадлежащих классам «Хороший», «Нездоровый для чувствительных групп», «Нездоровый», «Очень нездоровый», «Опасный», недостаточно для решения задачи классификации по указанным классам. Поэтому было решено объединить наблюдения, принадлежащие классам «Хороший» и «Умеренный», а также классам «Нездоровый для чувствительных групп», «Нездоровый», «Очень нездоровый» и «Опасный» (рис. 2.26(б)).

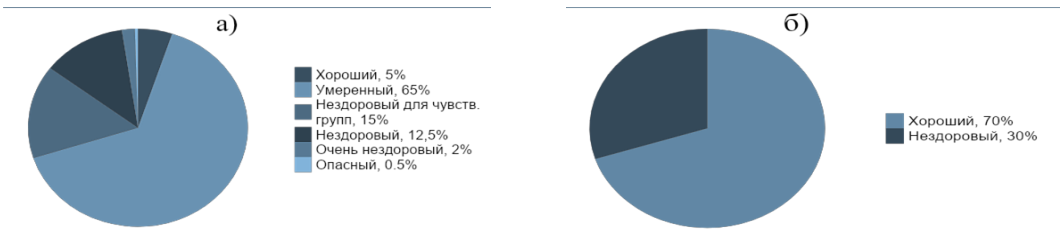


Рисунок 2.26 – Распределение наблюдений по классам AQI (а) и распределение наблюдений по объединенным классам AQI (б)

Это позволило рассмотреть задачу классификации AQI в зависимости от метеофакторов по двум интегрированным классам, условно названным: «Хороший» и «Нездоровый». При этом если всегда предсказывать «Хороший» класс, то точность прогноза составит 70%, поэтому будем считать классификатор приемлемым, если точность прогноза класса AQI на его основе превысит точность 70%.

Применение различных нейросетевых классификаторов (многослойный перцептрон, обычная рекуррентная сеть и LSTM-сеть) выявило, что наилучшую точность в прогнозировании класса AQI показала LSTM-сеть, поскольку она учитывает исторические данные и оценивает их в зависимости от временного удаления вектора входа до прогнозируемого выхода [26].

Как показано в [32], «входной вектор LSTM-классификатора определяют 6 метеофакторов (температура воздуха, атмосферное давление, относительная влажность, скорость ветра, температура точки росы, показатель интенсивности осадков) и значение AQI. При этом данные о температуре воздуха, атмосферном давлении, температуре точки росы нормализуются с помощью *Z*-нормы. Выходной вектор, к которому должен приближаться выход классификатора, определяется 2 параметрами – вероятность отнесения выхода классификатора к классу «Хороший», равная 1 для AQI <= 100, и вероятность отнесения выхода классификатора к классу «Нездоровый», равная 0 для AQI > 100» [32]. Проводя эксперименты по прогнозированию класса AQI, было решено варьировать следующие параметры: *S* – длину последовательности векторов исторических данных – входных векторов (ВВ) классификатора; *P* – глубину прогноза (на сколько шагов вперед прогнозируется AQI). Шаг прогноза – 3 часа. Эксперименты с LSTM-сетью показали [32], что лучшую точность прогнозирования дали классификаторы, учитывающие историю данных глубиной 12–16 шагов (1,5 – 2 дня), при этом прогноз AQI возможен до 4 дней вперед с точностью 88–90%» [32].

2.7 Выводы

Таким образом, в данном разделе рассмотрены различные способы прогноза геофизических временных рядов. Показано, что в связи с тем, что многие временные ряды, встречающиеся при изучении геофизических процессов, являются нестационарными и в той или иной мере хаотическими, для их прогноза целесообразно использовать методы, основанные на вейвлет- и нейросетевых технологиях.

Представлена структура мультивейвлетной полиморфной сети с настраиваемым параметром материнского вейвлета, объединяющая в себе преимущества мультивейвлетной и полиморфной вейвлет-сети. Предложено для прогноза временного ряда превышения продолжительности суток использовать мультивейвлетную полимофную сеть, содержащую полиморфный SLOG вейвлет в качестве материнского

вейвлета. Показано, что точность краткосрочного прогноза, полученного с помощью мультивейвлетной полиморфной сети, превышает точность прогноза, полученного с помощью общепринятого в настоящее время метода [21].

Представлены структура мультивейвлетной полиморфной сети и основанная на ней технология прогнозирования нестационарных временных рядов.

На примере краткосрочных прогнозов, известных в статистике временных рядов, показано, что точность прогнозов, полученных с помощью мульти-вейвлетной нейросетевой модели, превышает точность прогнозов, полученных с помощью ARIMA–модели, ИНС и гибридной модели Чанга, объединяющей модели ARIMA и ИНС. Это достигнуто благодаря введению дополнительных настраиваемых параметров и, как результат, лучшей приспосабливаемости модели к характеру нестационарности временных рядов.

Предложенная технология может быть применена для прогнозирования временных рядов, порожденных динамическими процессами различной природы. Рассмотрены различные архитектуры глубоких нейронных сетей, применяемых для прогноза временных рядов. Показано, что прогноз дальности видимости в аэропорту является актуальной задачей, и предложен базовый метод прогноза такого временного ряда. Выполнено сравнение результатов прогноза, полученных с помощью различных архитектур глубоких нейронных сетей по сравнению с базовым методом прогноза. Показано, что использование данных нескольких метеостанций не является целесообразным для прогноза дальности видимости. Также показано, что использование одинаковых методов прогноза и идентичных архитектур нейронных сетей для различных значений упреждения прогноза с использованием одинаковых данных не является оптимальной стратегией прогноза.

Исходя из полученных результатов, можно сделать вывод о том, что для прогноза дальности видимости необходимы данные о погоде с временным шагом менее 3 часов. Использование GRU глубокой нейронной сети позволяет уменьшить ошибку прогноза более чем в 2 раза по сравнению с базовым методом прогноза и в 1,5 раза по сравнению с поверхностной нейронной сетью.

Прогнозирование значений концентраций частиц PM2.5 в атмосферном воздухе и индекса качества воздуха AQI − непростая задача, поскольку на уровень загрязненности воздуха влияют многие факторы. Известны различные модели для прогнозирования: от классических статистических моделей до моделей глубокого обучения.

В настоящей работе для прогнозирования уровня загрязненности воздуха представлены четыре варианта моделей: ARIMA-модели (интегрированные модели авторегрессии − скользящего среднего) для краткосрочного прогноза PM2.5 и AQI с использованием суточной

истории наблюдений; линейные мультирегрессионные модели на основе регрессионного анализа метеорологических факторов и концентраций частиц PM2.5 в предшествующие сроки наблюдений, модель краткосрочного прогноза концентраций PM2.5 с учетом метеорологических факторов на основе обобщенно-регрессионной нейронной сети GRNN, модель среднесрочного прогноза двух интегрированных классов AQI («Хороший»/«Нездоровый») в зависимости от метеопараметров на базе LSTM-нейронных сетей. Указанные модели использованы для построения прогноза уровня загрязненности воздуха г. Бишкек в различные периоды 2019−2020 гг. При разработке нейросетевой модели предложено декомпозировать ряды наблюдений в соответствии со сроками наблюдений, что позволило повысить точность прогноза. Задача прогнозирования AQI в зависимости от метеопараметров рассмотрена как задача нейросетевой классификации. Последующее накопление данных позволит проводить классификацию AQI на большее число классов. Представленные здесь модели позволяют заключить, что затруднительно однозначно рекомендовать определенную модель для прогноза уровня загрязненности атмосферного воздуха, поскольку выбор должен определяться спецификой временного ряда наблюдений, размером выборки и целью прогнозирования.

3. Задачи медицинской геоэкологии

3.1 Особенности диагностики коронавирусной инфекции с помощью рентгеновских снимков

После вспышки коронавирусной инфекции COVID-19 и ее объявления ВОЗ пандемией в марте 2020 года во всем мире были приняты беспрецедентные меры для сдерживания распространения болезни. При этом было доказано, что одним из ключевых факторов сдерживания болезни является быстрая и точная диагностика инфицированных пациентов, их адекватное лечение и изоляция от остального населения. Коронавирусная болезнь приобрела исключительно большое значение, что объясняется тем, что COVID-19 характеризуется относительно низкой общей летальностью (1–3,5%), которая, однако, резко возрастает до >30% в возрастной группе старше 70 лет. В целом пневмония развивается у 15–20% заболевших, а от 5 до 30% больных требуют лечения в условиях отделения реанимации и интенсивной терапии.

Со стороны системы государственного здравоохранения ответом на пандемию COVID-19 являются комплексные действия, направленные на снижение уровня заболеваемости и смертности, минимизацию передачи заболевания, защиту медицинского персонала и сохранение бесперебойного функционирования системы здравоохранения. Диагностика COVID-19 проводится с помощью совокупной оценки эпидемиологического анамнеза, клинической картины, результатов лучевых и лабораторных исследований. Верификация болезни подразумевает получение положительного результата лабораторного исследования на наличие РНК SARS-CoV-2 с применением методов амплификации нуклеиновых кислот вне зависимости от клинических проявлений. Вместе с тем, по данным многочисленных источников, точность данного метода не превышает 70%. Из-за этого значительное количество пациентов с развернутой клинической и рентгенологической картиной не получают своевременной целевой терапии, а также оказываются вне действия нужных мер инфекционного контроля [40].

Однако, кроме COVID-19, другие пневмонии представляют собой одну из актуальных проблем медицины нашего времени. Это связано с их распространенностью, а также сильным влиянием на качество жизни человека и на его социальную жизнь. По данным ВОЗ, пневмония является причиной смертности 15% у детей и 7% у взрослых во всем мире. 808 694 ребенка умерли от пневмонии различной этиологии только в 2018 году. В этом же году было проведено исследование, в котором проанализировано 2968 протоколов патологоанатомических вскрытий. Было выявлено 297 умерших с диагнозом «пневмония», что составило 10,7% от всех умерших. Была выявлена бактериальная пневмония в 40% случаев, вирусно-бактериальная пневмония – 34%, пневмоцистная пневмония – 13%, сочетание пневмоцистной и вирусно-бактериальной пневмонии – 13% [41].

В настоящее время полимеразная цепная реакция (ПЦР) с обратной транскрипцией остается основным диагностическим инструментом, но роль рентгенографических методов исследования грудной клетки как дополнительных методов диагностики или даже надежной альтернативы получает все большее значение. Объясняется это тем, что получение результатов ПЦР-теста может занять несколько дней, напротив, информация, полученная при рентгенографическом исследовании, может быть незамедлительно использована для быстрой диагностики и, что важнее, при сравнительно небольших затратах [40].

Тем не менее интерпретация рентгеновских снимков затруднительна, так как для того, чтобы обнаруживать тонкие визуальные особенности, присутствующие на изображениях, требуется достаточно высокая квалификация врачебного персонала. Напротив, система искусственного интеллекта (СИИ) может обнаруживать закономерности на рентгенограммах грудной клетки, обычно с трудом распознающиеся врачами-радиологами, и в различных источниках [42, 43] было опубликовано множество исследований о новых разработках СИИ с использованием сверточных нейронных сетей (convolutional neural network, CNN) с целью дифференцирования COVID-19 от других заболеваний с использованием общедоступных баз данных рентгеновских снимков грудной клетки. Как показано в этих работах, СИИ на основе сверточных нейронных сетей позволила бы улучшить возможности рентгенографии с точки зрения распознавания типичных признаков COVID-19 и дифференциацию его от других типов пневмонии и, кроме того, могла бы ускорить диагностику больных и улучшить степень определения риска тяжелого течения болезни в случаях с неопределенными результатами при отсутствии других диагностических методов.

Основное отличие сверточных нейронных сетей заключается в том, что полносвязные нейронные сети изучают глобальные шаблоны в пространстве входных признаков, тогда как сверточные слои изучают локальные шаблоны: в случае с рентгеновскими снимками – шаблоны в небольших двумерных окнах во входных данных. Например, в сети ResNet, используемой работе [43], все такие окна имели размеры 3 × 3. Эта ключевая характеристика наделяет сверточные нейронные сети двумя важными свойствами:

- Шаблоны, которые они изучают, являются инвариантными в отношении переноса. После изучения определенного шаблона в правом нижнем углу снимка сверточная нейронная сеть сможет распознавать его повсюду: например, в левом верхнем углу. Полносвязной сети пришлось бы изучить шаблон заново, если он появляется в другом месте. Это увеличивает эффективность сверточных сетей в задачах обработки изображений (потому что видимый мир по своей сути является инвариантным в отношении

переноса): таким сетям требуется меньше обучающих образцов для получения представлений, обладающих возможностью обобщения.

- Они могут изучать пространственные иерархии шаблонов. Первый сверточный слой будет изучать небольшие локальные шаблоны, такие как края, второй – более крупные шаблоны, состоящие из признаков, возвращаемых первым слоем, и т. д. Это позволяет сверточным нейронным сетям эффективно изучать все более сложные и абстрактные визуальные представления (потому что видимый мир по своей сути является пространственно-иерархическим).

Свертка применяется к трехмерным тензорам, называемым картами признаков, с двумя пространственными осями (высота и ширина), а также с осью глубины (или осью каналов). Для рентгеновских снимков в формате RGB размерность оси глубины равна 3, потому что имеется три канала цвета: красный (red), зеленый (green) и синий (blue). Для черно-белых изображений грудной клетки, используемых, например, в наборе [44], ось глубины имеет размерность 1 (оттенки серого). Операция свертывания извлекает шаблоны из своей входной карты признаков и применяет одинаковые преобразования ко всем шаблонам, производя выходную карту признаков. Эта выходная карта признаков также является трехмерным тензором: она имеет ширину и высоту. Ее глубина может иметь любую размерность, потому что выходная глубина является параметром слоя, и разные каналы на этой оси глубины больше не соответствуют конкретным цветам, как во входных данных в формате RGB, скорее, они соответствуют фильтрам. Фильтры представляют собой конкретные аспекты входных данных: на верхнем уровне, например, фильтр может соответствовать понятию присутствие «матового стекла» на рентгеновском снимке.

Свертки определяются двумя ключевыми параметрами:

- ✓ Размер шаблонов, извлекаемых из входных данных, – обычно 3 × 3 или 5 × 5. В сети ResNet, рассматриваемой в работе [43], используется размер 3 × 3, что является наиболее распространенным выбором.
- ✓ Глубина выходной карты признаков – количество фильтров, вычисляемых сверткой. В данном примере свертка начинается с глубины 64 и заканчивается глубиной 512.

Свертка работает методом скользящего окна: она двигает окно с размером 3 × 3 или 5 × 5 по трехмерной входной карте признаков, останавливается в каждой возможной позиции и извлекает трехмерный шаблон окружающих признаков (с формой: высота окна, ширина окна, глубина входа). Каждый такой трехмерный шаблон затем преобразуется (путем умножения тензора на матрицу весов, получаемую в ходе обучения, которая называется ядром свертки) в одномерный вектор с формой (выходная глубина). Все эти векторы затем собираются в трехмерную выходную карту с формой (высота, ширина, выходная глубина). Каждое

пространственное местоположение в выходной карте признаков соответствует тому же местоположению во входной карте признаков (например, правый нижний угол выхода содержит информацию о правом нижнем угле входа). Упрощенная схема работы сверточной сети изображена на рисунке 3.1.

Рисунок 3.1 – Принцип работы сверточной нейронной сети

При этом выходные ширина и высота могут отличаться от входных. На то есть две причины:

- эффекты границ, которые могут устраняться дополнением входной карты признаков;
- использование шага свертки, о чем подробнее говорится в [26].

Ранее разными авторами уже предпринимались попытки разработать СИИ для этой цели на основе сверточных нейронных сетей. Например, сверточная сеть Bayes-SqueezeNet на основе метода байесовской оптимизации была использована для классификации рентгеновских снимков, разделенных на 3 класса: норма, вирусная пневмония и COVID-19. Благодаря широко известному методу расширения данных авторам удалось частично решить проблему переобучения сети на небольшом объеме данных.

СИИ CoroNet также была разработана для обнаружения инфекции COVID-19 по рентгеновским снимкам грудной клетки. Эта модель основана на предварительно обученной CNN Xception с добавлением слоя прореживания и двух полносвязных слоев в конце. Сеть применялась для классификации по 3 классам (COVID-19, пневмония и норма, а также для классификации по 4 классам (COVID-19, бактериальная пневмония, вирусная пневмония и норма).

СИИ CovidGAN с использованием генеративной состязательной сети была построена на основе предварительно обученной CNN VGG-16, дополненной четырьмя настраиваемыми сверточными слоями в конце и выбора среднего значения из соседних, за которым следовал полносвязный слой с 64 нейронами и слой прореживания с вероятностью 0,5 в конце. Также авторами была использована генеративно-состязательная сеть для создания синтетических рентгеновских изображений грудной клетки с целью расширения имеющегося объема данных и повышения эффективности классификации.

СИИ DarkCovidNet, основанная на модели DarkNet, является еще одной моделью CNN, предложенной для обнаружения COVID-19 с помощью рентгеновских снимков грудной клетки. DarkCovidNet состоит из меньшего по сравнению с DarkNet количества слоев и постепенно увеличивающихся фильтров выбора максимального значения из соседних. Эта СИИ была протестирована на 2 классах (COVID-19 и норма) и 3 классах (COVID-19, норма и пневмония) [44].

В работе [45] рассмотрена проблема диагностики COVID-19 по рентгеновским снимкам с помощью предварительно обученной сверточной нейронной сети. Классификация выполнялась по двум классам — COVID-19 и норма, при этом остальные возможные классы, например, пневмония, туберкулёз и т.д., в этом исследовании не учитывались. Однако если рассматривать задачу диагностики COVID-19, то следует учитывать и другие заболевания с похожей рентгенологической клинической картиной, такие, как бактериальная и вирусная пневмония и, учитывая традиционно высокую распространенность, по данным ВОЗ в Кыргызстане и Центральной Азии, туберкулез [46].

В работе [47] были реализованы VGG-19, MobileNet-v2, Inception, Xception и Inception ResNet-v2, а в работе [48] сети AlexNet, GoogLeNet, and SqueezeNet, в качестве предварительно обученных CNN для обнаружения COVID-19 по рентгеновским снимкам. Эти СИИ были использованы для классификации на 2 и 3 класса с помощью наборов данных, состоящих из изображений COVID-19, бактериальной пневмонии, вирусной пневмонии и нормы.

Целью настоящего исследования является разработка надежного классификатора на основе технологий глубокого обучения с помощью предварительно обученной глубокой сверточной нейронной сети и полносвязных нейронных слоев, дообученных на сравнительно небольшом объеме данных, а также сравнение различные вариантов предварительно обученных нейронных сетей на базе: DenseNet, EfficientNet, InceptionResNetV2, InceptionV3, MobileNet, MobileNetV2, MobileNetV3, NASNet, ResNe, ResNetV2, VGG, Xception.

Эти предварительно обученные CNN выбраны по той причине, что они все входят в состав свободно распространяемого фреймворка Keras. Классификация выполнялась отдельно по двум классам: COVID-19 и

норма и по 6 классам: норма, COVID-19, бактериальная пневмония, вирусная пневмония, туберкулез и класса «затемнение легких», включающего в себя различные неуточненные болезненные состояния неизвестной природы, отражающиеся на рентгеновских снимках.

3.2 Выбор и обоснование предварительных функциональных процедур

3.2.1 Предварительно обученные сверточные основы. Наиболее эффективным подходом к глубокому обучению на небольших наборах изображений является использование предварительно обученной сети.

Предварительно обученная сеть — это сохраненная сеть, прежде обученная на большом наборе данных, обычно в рамках масштабной задачи классификации изображений. Если этот исходный набор данных достаточно велик и достаточно обобщен, тогда пространственная иерархия признаков, изученных сетью, может эффективно выступать в роли обобщенной модели видимого мира и быть полезной во многих разных задачах распознавания образов, даже если эти новые задачи будут связаны с совершенно иными классами, отличными от классов в оригинальной задаче.

Другими словами, можно обучить сеть на изображениях из сайта ImageNet (где подавляющее большинство классов — животные и бытовые предметы) и затем использовать эту обученную сеть для идентификации чего-то иного, например, предметов мебели на изображениях. Такая переносимость изученных признаков между разными задачами – главное преимущество глубокого обучения перед многими более старыми приемами поверхностного обучения, которое делает глубокое обучение очень эффективным инструментом для решения задач с малым объемом данных [26].

Основные параметры некоторых используемых предварительно обученных нейронных сетей приведены в таблице 3.1. Полностью гиперпараметры всех используемых нейронных сетей даны в источнике [49]. Исходные рентгеновские снимки грудной клетки были приведены к формату PNG и преобразованы таким образом, чтобы их размер соответствовал входному размеру предварительно обученных CNN – 224x224 точек.

Для дополнительного обучения полносвязных нейросетевых классификаторов по двум классам использовался оптимизатор Adam, основанный на стохастическом градиентном спуске и адаптивной оценке моментов первого и второго порядков с параметром начальной скорости обучения L_r=0,001 и постоянной скоростью уменьшения до нуля на протяжении всего процесса обучения эпох, с размером пакетов при обучении, равном 8 и экспоненциальной скорости затухания для 1-го момента $\beta_1 = 0,9$, для 2-го момента $\beta_2=0,999$ и константой численной стабильности $\varepsilon=10^{-7}$. Этот вариант настройки оптимизатора показал

наилучший результат из множества опробованных. Для дополнительного обучения нейросетевых классификаторов по шести классам использовался оптимизатор Adam с теми же параметрами, но с размером пакетов при обучении равным 32, так как в этом случае использовалось гораздо большее количество исходных данных.

Таблица 3.1 – Основные параметры предварительно обученных CNN

№ п./п.	Модель	Размер	Топ-1 по точности	Топ-5 по точности	Количество параметров	Глубина
1	Xception	88 MB	0.790	0.945	22,910,480	126
2	VGG16	528 MB	0.713	0.901	138,357,544	23
3	VGG19	549 MB	0.713	0.900	143,667,240	26
4	ResNet50	98 MB	0.749	0.921	25,636,712	-
5	ResNet101	171 MB	0.764	0.928	44,707,176	-
6	ResNet152	232 MB	0.766	0.931	60,419,944	-
7	ResNet50V2	98 MB	0.760	0.930	25,613,800	-
8	ResNet101V2	171 MB	0.772	0.938	44,675,560	-
9	ResNet152V2	232 MB	0.780	0.942	60,380,648	-
10	InceptionV3	92 MB	0.779	0.937	23,851,784	159
11	InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
12	MobileNet	16 MB	0.704	0.895	4,253,864	88
13	MobileNetV2	14 MB	0.713	0.901	3,538,984	88
14	DenseNet121	33 MB	0.750	0.923	8,062,504	121
15	DenseNet169	57 MB	0.762	0.932	14,307,880	169
16	DenseNet201	80 MB	0.773	0.936	20,242,984	201
17	NASNetMobile	23 MB	0.744	0.919	5,326,716	-
18	NASNetLarge	343 MB	0.825	0.960	88,949,818	-
19	EfficientNetB0	29 MB	-	-	5,330,571	-
20	EfficientNetB1	31 MB	-	-	7,856,239	-
21	EfficientNetB2	36 MB	-	-	9,177,569	-
22	EfficientNetB3	48 MB	-	-	12,320,535	-
23	EfficientNetB4	75 MB	-	-	19,466,823	-
24	EfficientNetB5	118 MB	-	-	30,562,527	-
25	EfficientNetB6	166 MB	-	-	43,265,143	-

26	EfficientNetB7	256 MB	-	-	66,658,687	-

При дополнительном обучении веса предварительно обученных CNN оставались зафиксированными, т.е. не изменялись, в процессе дополнительного обучения полносвязных классификаторов настраивались только веса двух добавленных полносвязных слоев.

3.2.2 Исходные данные. В настоящей работе использовались 4 общедоступные базы данных рентгеновских снимков грудной клетки:

- Chest X-Ray Images (Pneumonia) [50].
- COVID-19 image data col lection [51].
- COVID-19_Radiography_Dataset [52].
- TB_Chest_Radiography_Database [53].

Набор данных [50] состоит из рентгеновских снимков грудной клетки здоровых людей и пациентов с бактериальной и вирусной пневмонией. Всего имеется 5856 рентгенограмм в передней прямой проекции, примерно 2/3 из них принадлежит пациентам, больным пневмонией, приблизительно поровну вирусной и бактериальной, а остальная часть здоровым пациентам.

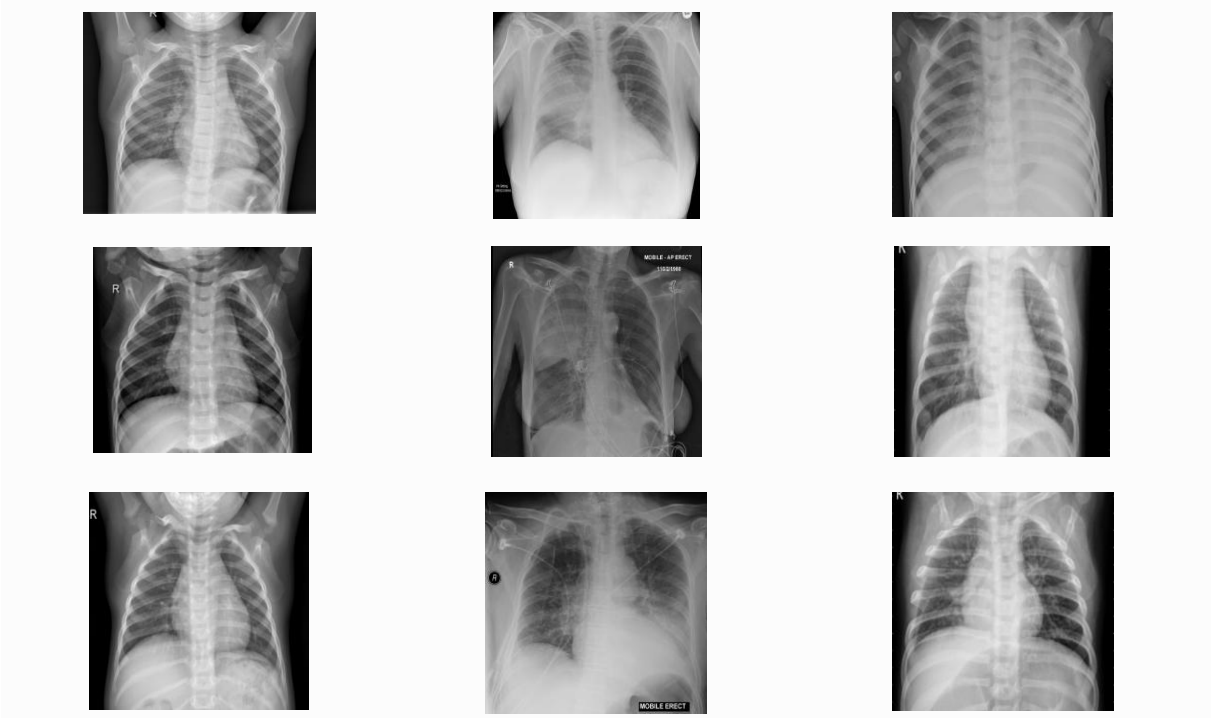
Набор [51] содержит 468 рентгеновских снимков грудной клетки пациентов с COVID-19, 38 снимков с вирусной пневмонией, 46 снимков с бактериальной пневмонией, 26 снимков с грибковой пневмонией, 9 снимков с пневмонией другого типа и 59 изображений другого типа (рисунок 3.2).

Набор [52] содержит 3616 рентгеновских снимков грудной клетки пациентов с COVID-19, 1345 снимков с вирусной пневмонией, 10192 снимка грудной клетки здоровых людей и 6012 снимков с затемнениями на легких, вызванными неизвестными причинами.

Набор [53] содержит 700 рентгеновских снимков пациентов с туберкулезом и 3500 снимков грудной клетки здоровых людей. Для классификации по двум классам из первого набора был выбран 191 рентгеновский снимок грудной клетки здоровых людей в передней прямой проекции (рис. 3.2а), а из второго набора 191 снимок грудной клетки пациентов, больных COVID-19 (рис. 3.2б), в передней и задней прямой проекции с дополнением данных путем случайного поворота на величину до 15° и случайного отражения в горизонтальной плоскости. Дополнение отражением необходимо, так как в данных из первого набора присутствуют снимки только в передней проекции, во втором наборе как в передней, так и в задней проекции, а небольшой поворот изображения часто встречается при вводе изображения с помощью сканера или камеры.

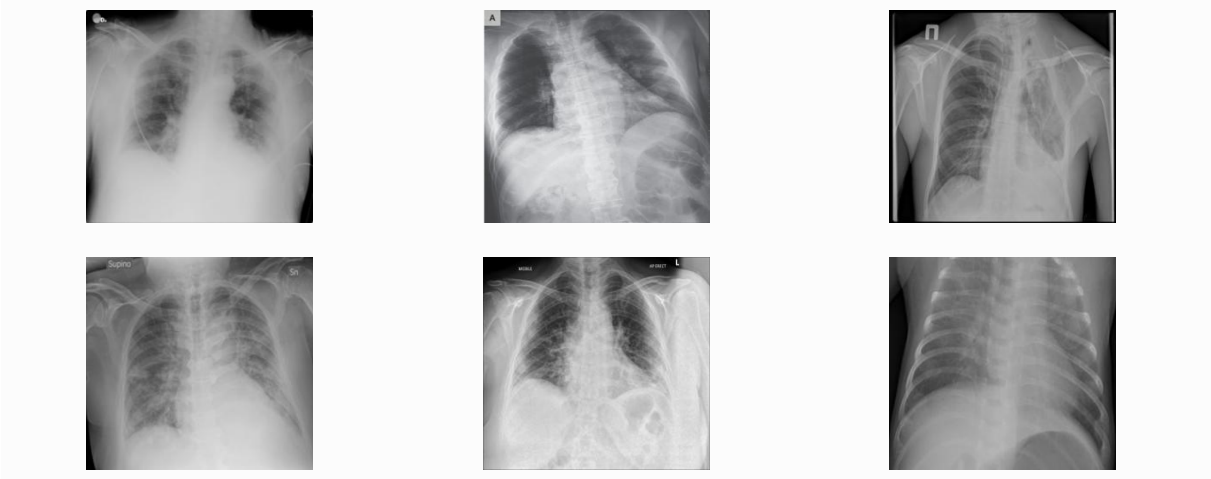
Для сравнения с другими работами [47, 48] использовался отдельный набор данных, включающих 403 рентгеновских снимка грудной клетки с COVID-19 и 721 рентгеновский снимок грудной клетки здоровых людей без дополнения. Для классификации по 6 классам из всех вышеперечисленных наборов данных были выбраны 3616 рентгеновских

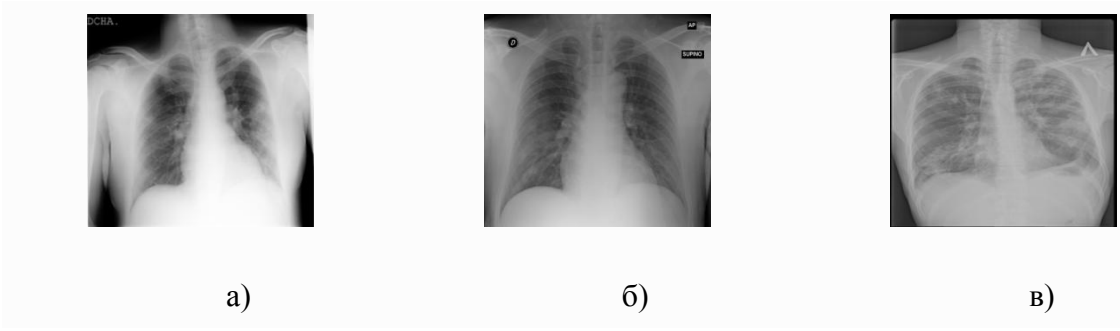
снимков пациентов с COVID-19, 1345 снимков с вирусной пневмонией, 2780 снимков с бактериальной пневмонией, 6012 снимков с затемнением легких с невыясненной причиной, 700 снимков с туберкулезом и 10192 снимка грудной клетки здоровых людей, всего 27645 изображений (рис. 3.3)



а) б) в)

Рисунок 3.2 – Наборы данных: а) Chest X-Ray Images (норма); б) COVID-19 image data collection (бактериальная пневмония); в) COVID-19_Radiography_ Dataset (вирусная пневмония)





а) б) в)

Рисунок 3.3 – Наборы данных: а) COVID-19_Radiography_ Dataset (COVID-19); б) COVID-19_Radiography_ Dataset (затемнение легких); в)TB_Chest_Radiography_ Database (туберкулез)

3.3 Аппаратное и программное обеспечение исследований

3.3.1 Методы оценки обучения нейронных сетей. Как и в работе [48], в случае 2 классов использовались 6 характеристик для сравнения результатов дополнительного обучения полносвязных классификаторов глубоких нейронных сетей на основе различных вариантов CNN – верность, чувствительность, специфичность, точность, оценка F_1 и площадь под ROC-кривой (receiver operating characteristic, рабочая характеристика приемника), т.н. AUC (area under ROC curve).

Чувствительность S_e определяется как процент пациентов с COVID-19, у которых инфекция правильно диагностирована, и выражается как:

$$Se = Tp/P \times 100\% = Tp/(Tp + Fn) \times 100\%, \quad (3.1)$$

где T_p – количество пациентов с COVID-19, правильно диагностированных как инфицированные; F_n – количество пациентов с COVID-19, ошибочно диагностированных как не инфицированные и P – общее количество больных COVID-19.

Специфичность S_p определяется как процент людей, не инфицированных COVID-19, правильно диагностированных как не инфицированные:

$$Sp = Tn/N \times 100\% = Tn/(Tn + Fp) \times 100\%, \quad (3.2)$$

где T_n – количество людей, не инфицированных COVID-19, правильно диагностированных как не инфицированные; F_p – количество людей, не инфицированных COVID-19, ошибочно диагностированных как инфицированные и N – общее количество людей, не инфицированных COVID-19.

Точность A_c классификации определяется как:

$$Ac = (Tp + Tn)/(P + N) \times 100\%. \quad (3.3)$$

Верность P_r классификации определяется как:

$$Pr = Tp/(Tp + Fp) \times 100\%. \quad (3.4)$$

Оценка F_1 определяется как среднее гармоническое между точностью и чувствительностью:

$$F_1 = 2Tp/(2Tp + Fp + Fn). \quad (3.5)$$

ROC представляет собой график, отображающий соотношение между T_p и F_p при варьировании порога решающего правила, а *AUC* представляет собой меру производительности классификатора. Чем выше *AUC*, тем лучше модель диагностирует COVID-19, отличая его от случаев, не связанных с ним. Для идеального классификатора, например, *AUC=1*, а *AUC=0,5* показывает, что классификатор работает случайным образом.

В случае многоклассовой классификации использовалась только точность классификации A_c, так как остальные характеристики плохо экстраполируются на этот вариант. Она определяется как: $Ac = T/N \times 100\%$, где T – количество правильно классифицированных пациентов, N – общее количество людей.

3.3.2 Аппаратное и программное обеспечение исследований. Для проведения вычислительных экспериментов по обучению нейронных сетей использовался суперкомпьютер, предоставленный в рамках договора о сотрудничестве между ИМА НАН КР и КРСУ им. Б.Н.Ельцина.

Основные характеристики используемого программного и аппаратного обеспечения приведены в таблице 3.2

Таблица 3.2 – Основные характеристики используемого аппаратного обеспечения

Операционная система	Ubuntu 18.04.5 LTS
Материнская плата	ASUSTeK COMPUTER INC. TUF Z270 MARK 2
Процессор	Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz
Твердотельный накопитель	ATA ADATA SP580
Память	32880 Мб
Графические адаптеры (GPU)	2 X NVIDIA Corporation GP104 [GeForce GTX 1080]
Суммарная производительность	17.7 TFLOPS

Как видно из таблицы 3.2, основную вычислительную мощность суперкомпьютера обеспечивают два GPU NVIDIA GeForce GTX 1080, имеющие 2560 ядер CUDA каждая, работающие на частоте 1607 MHz в режиме SLI, что позволяет задействовать одновременно оба графических адаптера. В качестве основного программного обеспечения вычислительного эксперимента использовался контейнер на основе

официального Docker-образа фреймворка машинного обучения TensorFlow версии tensorflow:2.4.1-gpu-jupyter.

Для достижения максимальной производительности использовалась “зеркальная” стратегия распределения работы, в равной мере разделяющая текущую задачу между всеми GPU и обеспечивающая синхронизацию с помощью алгоритма All-Reduce [54]. Это позволяет в среднем примерно на 70% задействовать вычислительную мощность обоих графических адаптеров (рисунок 3.4).

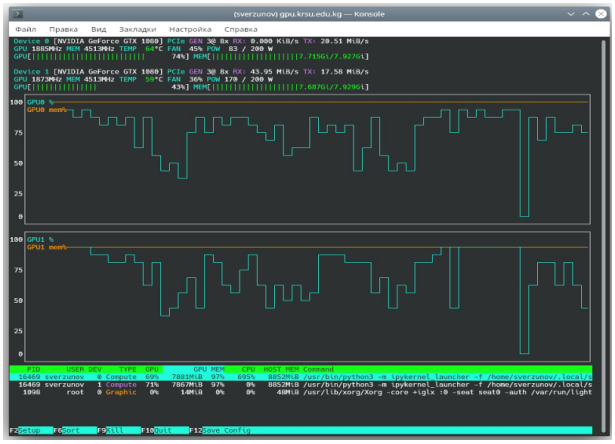


Рисунок 3.4 – График загрузки GPU во время обучения нейронной сети

Полностью используемая конфигурация контейнера показана в листинге ниже в формате docker-compose yaml-файла.

```
version: "2.4"
services:
  tensorflow:
    image: tensorflow/tensorflow:2.4.1-gpu-jupyter
    build:
      ./docker
    deploy:
      resources:
        reservations:
          devices:
          - driver: nvidia
            count: 2
            capabilities: [gpu]
    container_name: x-ray2
    ports:
      - 10000:8888
    volumes:
      - ./home:/home/sverzunov/
      - ./src:/tf
    user: sverzunov:sverzunov
```

Для получения необходимой информации о ходе обучения использовались также некоторые дополнительные библиотеки, такие как, например, numpy, matplotlib и другие. Полностью программная конфигурация в формате Dockerfile, использовавшаяся при обучении нейронных сетей, показана в листинге ниже.

Следует отметить, что проброс домашней директории пользователя в контейнере в папку на хост-системе позволил сохранять между сеансами

важные настройки библиотеки matplotlib и используемой облачной интерактивной среды разработки Jupyter.

```
FROM tensorflow/tensorflow:2.4.1-gpu-jupyter
RUN apt-get update
RUN apt install -y python3-pip
RUN python3 -m pip install --upgrade pip
RUN python3 -m pip install \
numpy \
pandas \
matplotlib
RUN groupadd -f sverzunov --gid 1000
RUN id -u sverzunov >/dev/null 2>&1 || useradd -G sverzunov --uid 1000 sverzunov
RUN mkdir -p /home/sverzunov
RUN chown -R sverzunov:sverzunov /home/sverzunov
```

Таким образом, описанная здесь программно-аппаратная конфигурация позволила в большей степени задействовать все имеющиеся в нашем распоряжении вычислительные ресурсы и достичь скорости обучения до 350 изображений в секунду, при использовании, например, предварительно обученной сверточной нейронной сети EfficientNetB2, что на порядок больше того, что можно было бы достичь, если бы использовались только вычислительные мощности CPU.

3.4 Система искусственного интеллекта для диагностики COVID-19 по рентгеновским снимкам грудной клетки

В настоящем разделе из-за ограниченного на момент проведения исследования объема доступных данных не рассматривается подробная классификация на вирусную, бактериальную пневмонии и туберкулёз не учитывался, а использовалось только 2 класса – COVID-19 и норма.

Другая проблема заключается в том, что для каждого пациента имеется только одномоментный набор рентгеновских снимков грудной клетки. Это ограничение приводит к тому, что оказывается невозможным определить, появились ли у пациентов новые рентгенологические особенности по мере прогрессирования болезни.

В предлагаемых архитектурах предварительно обученных сверточных сетей последний слой с настраиваемыми весами является полносвязным слоем. Этот полносвязный слой был заменен новой последовательностью слоев, состоящих из слоя выбора максимального значения из соседних размером 4X4, промежуточного слоя для согласования размеров тензоров на выходе предыдущего слоя с входом последующего, полносвязного слоя, состоящего из 128 нейронов, слоя прореживания с вероятностью 0,6 и выходного слоя из 2 нейронов, так как классификация производится по двум классам.

Результаты обучения различных сверточных нейронных сетей представлены в [49] в виде средних значений и доверительного интервала

всех вышеперечисленных характеристик (1–5) для трёх случайных разбиений данных на обучающие и проверочные.

Пятнадцать дополнительно обученных нейросетевых классификторов достигли достаточно высокой точности, превышающей 92% чувствительности, специфичности, верности, показателя F_1 и AUC при двух различных пропорциях разбиения данных на обучающие и проверочные наборы. В таблице 2 в работе [49] их названия отмечены жирным шрифтом. Для некоторых из этих сетей изменение точности на протяжении 200 эпох для наглядности показано на рисунке 3.5.

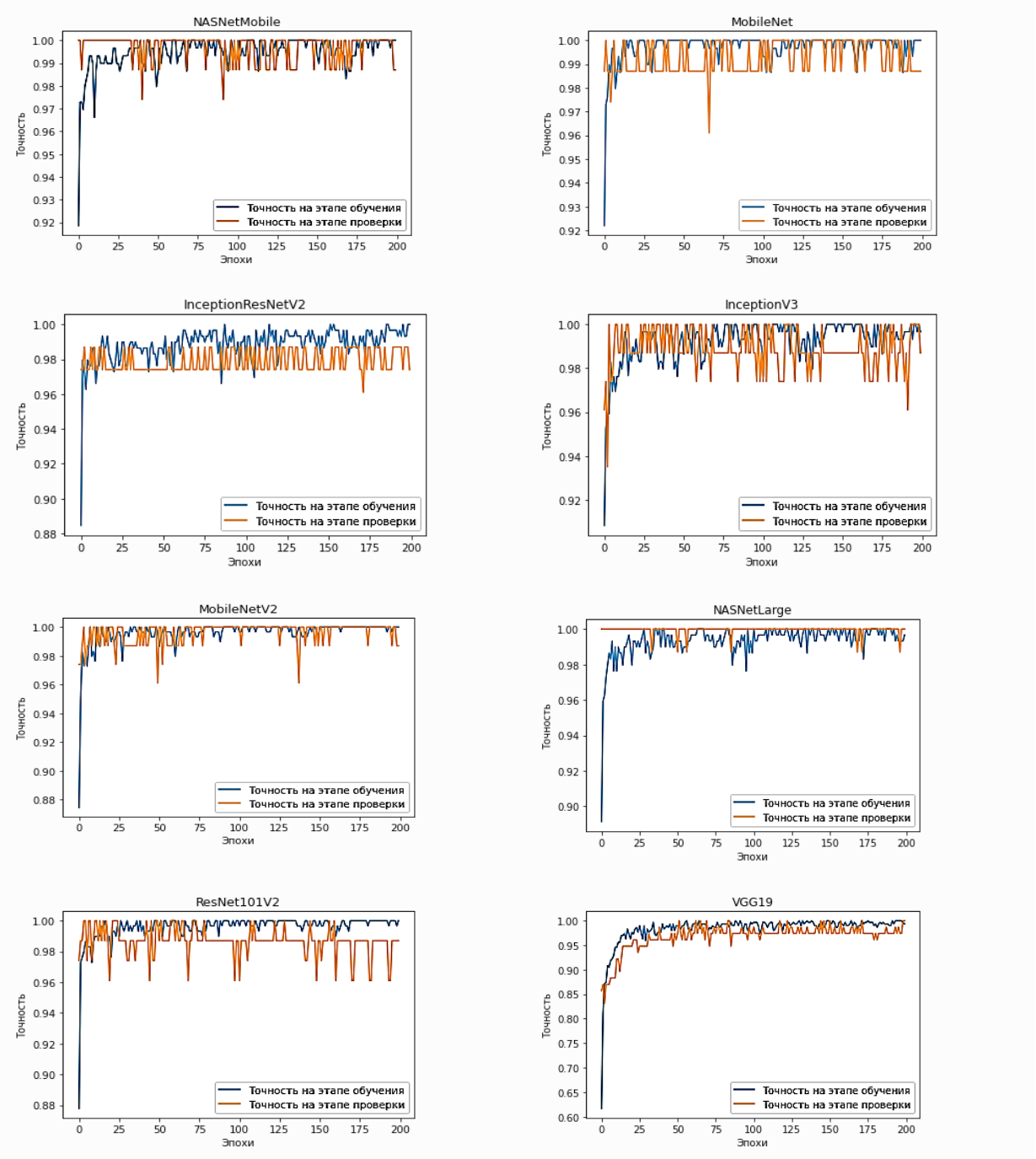


Рисунок 3.5 – Точность классификации на этапе обучения и проверки нейронных классификаций на проверочном наборе данных в процессе обучения

В первом случае обучающие данные составляли 80%, а проверочные 20 %, во втором случае – по 50%. Для восьми из сетей, показавших наиболее стабильное обучение, а именно построенных на основе предварительно обученных CNN DenseNet121, DenseNet169, MobileNetV2, NASNetLarge, ResNet152V2, VGG16, VGG19, Xception и дополнительно обученного на протяжении 50 эпох полносвязного классификатора, созданы интерпретации взаимосвязи входных изображений с результатами классификации по методу интегрированных градиентов [55] с количеством шагов, равных 100.

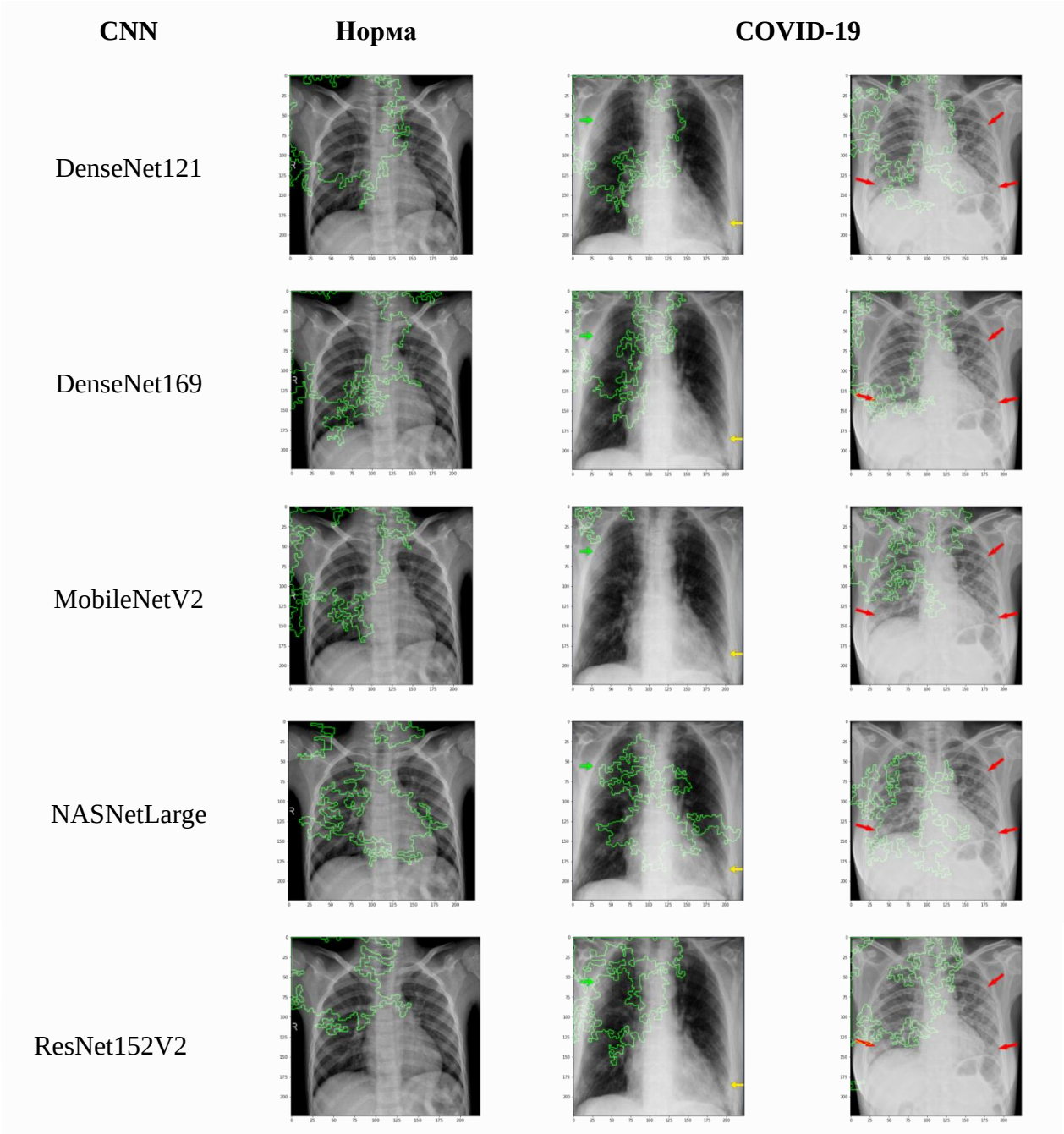


Рисунок 3.6 – Интерпретация результатов классификации с помощью сетей на основе различных CNN (DenseNet121, DenseNet169, MobileNetV2, NASNetLarge, ResNet152V2), полученная на основе метода интегрированных градиентов

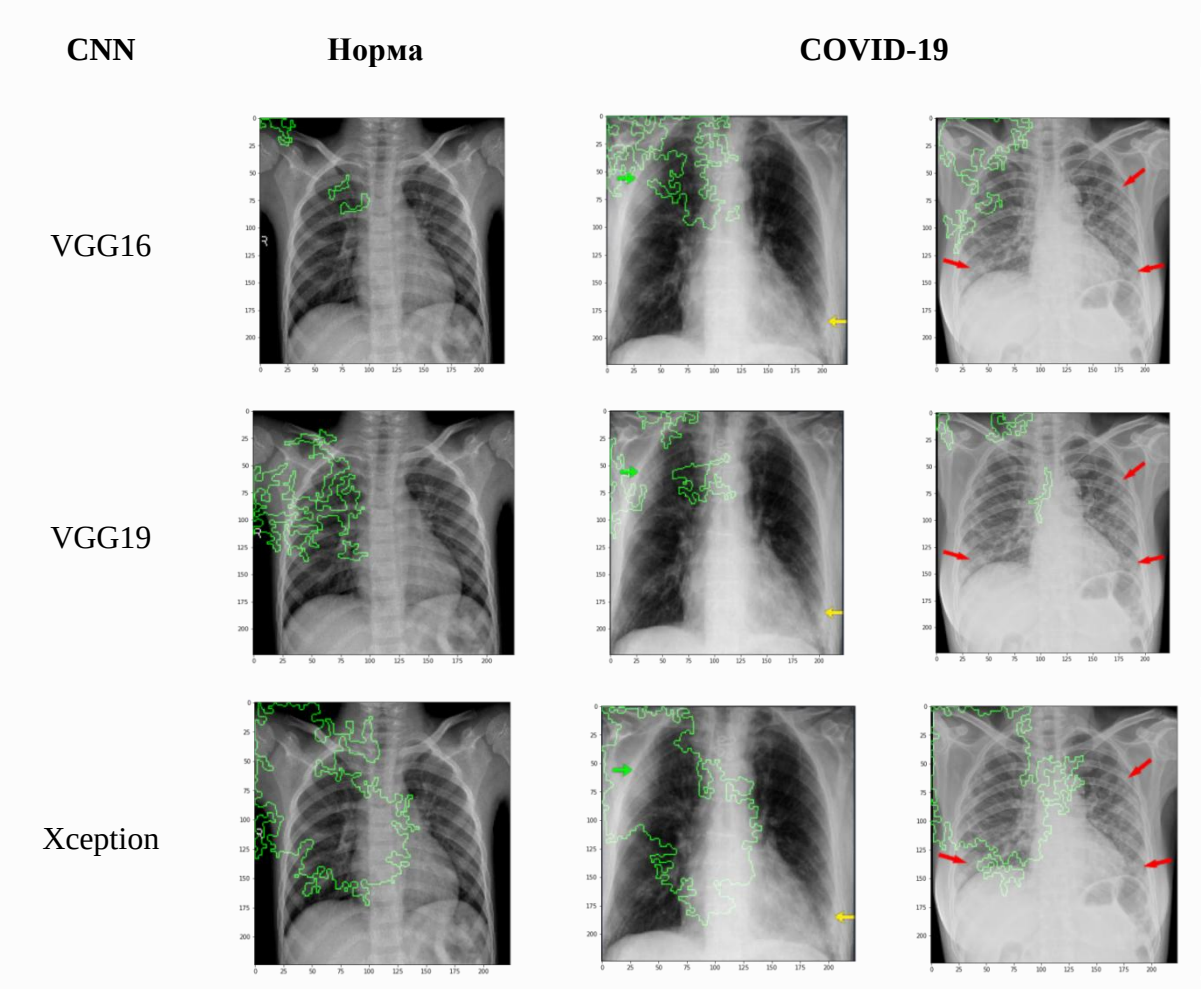


Рисунок 3.7 – Интерпретация результатов классификации с помощью сетей на основе различных CNN (VGG16, VGG19, Xception), полученная на основе метода интегрированных градиентов

Результаты обучения наглядно показывают, какие участки изображения в большей степени используются для классификации. Для построения визуализаций интегрированного градиента (рис. 3.6, 3.7) использовались пиксели с уровнем значений использования сетью от 30% до 95% с применением трех изображений из наборов [56, 57], два из которых принадлежат больным COVID-19, а одно – здоровому человеку.

Как можно заметить, результаты, полученные с помощью дополнительного обучения нейронных сетей на основе предварительно обученных CNN DenseNet169, NASNetLarge, ResNet152V2, наиболее согласуются с мнением экспертов, в смысле соответствия участков изображения, использованных ими для классификации, и участков легких, пораженных COVID-19.

Таким образом, результаты, полученные при дополнительном обучении сетей на основе DenseNet169, NASNetLarge, ResNet152V2, демонстрируют высокую эффективность этих сверточных основ для диагностики COVID-19. Однако из-за постоянных обновлений базы данных рентгеновских снимков и общедоступности других коллекций

данных невозможно провести точное сравнение результатов, представленных здесь и в других работах, посвященных данной проблеме.

Сравнение с некоторыми работами [47, 48] при использовании идентичного набора данных, приведенное в таблице 3.3, убедительно свидетельствует о том, что обученные до достижения максимальной точности нейронные сети на основе предварительно обученных CNN DenseNet169 и ResNet152V2 достигли такого же или даже лучшего результата, чем несколько других сетей с точки зрения точности классификации и разных соотношений объема данных для обучения и проверки.

Таблица 3.3 – Сравнение результатов дополнительного обучения нейронных сетей на основе некоторых предварительно обученных CNN с другими моделями

Модели	Обучающие данные (%)	Тестовые данные (%)	Точность (%)
CovidGAN [8]	80	20	95
AlexNet [12]	80	20	99
GoogLeNet [12]	80	20	100
SqueezeNet [12]	80	20	100
DenseNet169	80	20	100
NASNetLarge	80	20	99
ResNet152V2	80	20	100
AlexNet [12]	50	50	99
GoogLeNet [12]	50	50	99
SqueezeNet [12]	50	50	98
DenseNet169	50	50	100
NASNetLarge	50	50	99
ResNet152V2	50	50	100

Таким образом, в разделе были проанализированы результаты дополнительного обучения нейронных сетей на основе 28 часто используемых предварительно обученных CNN, доступных с помощью фреймворка Keras, для классификации рентгеновских снимков грудной клетки пациентов, больных COVID19, и здоровых людей с использованием общедоступных наборов данных. Результаты классификации, полученные с помощью различных наборов данных для обучения и проверки, позволили продемонстрировать высокую эффективность 8 из 28 исследованных предварительно обученных CNN. Эти результаты свидетельствуют, что правильный выбор предварительно обученной CNN

важен, поскольку он помогает избежать усилий по разработке более сложных моделей, в то время как уже существующие предварительно обученные CNN позволяют добиться такого же или даже лучшего результата.

3.5 Результаты разработки системы искусственного интеллекта для диагностики различных видов пневмонии по рентгеновским снимкам грудной клетки

В настоящем разделе используются все собранные на момент проведения исследования доступные данные и рассматривается подробная классификация пневмонии на вирусную, бактериальную и вызванную туберкулёзом. Так выделяется 6 классов: COVID-19, норма, бактериальная пневмония, вирусная пневмония, туберкулёз и отдельный класс для заболеваний невыясненной природы, который мы для краткости назовем «затемнение легких». К сожалению, здесь также для большинства пациентов имеется только одномоментный набор рентгеновских снимков грудной клетки.

Это ограничение приводит к тому, что оказывается невозможным определить, появились ли у пациентов рентгенологические особенности по мере прогрессирования болезни. В используемых здесь архитектурах предварительно обученных сверточных сетей последний слой с настраиваемыми весами по-прежнему является полносвязным слоем. Этот полносвязный слой, как и в предыдущей главе, был заменен новой последовательностью слоев, состоящей теперь из слоя выбора среднего значения из соседних по всем каналам (слоя глобального пуллинга), для согласования размеров тензоров на выходе предыдущего слоя с входом последующего полносвязного слоя, состоящего из 128 нейронов, слоя прореживания с вероятностью 0,5 и выходного слоя из 6 нейронов, так как классификация производится по шести классам. Сравнение результатов приведено в табл. 3.4.

Сравнение убедительно свидетельствует о том, что обученные до достижения максимальной точности на протяжении 5 эпох нейронные сети на основе предварительно обученных CNN EfficientNetB3 и ResNet152 достигли наилучшего результата, чем 28 других моделей с точки зрения точности классификации и разных соотношений объема данных для обучения и проверки.

Таблица 3.4 – Результаты вычислительного эксперимента по диагностике различных видов пневмонии по рентгеновским снимкам грудной клетки

№	Модели	Точность, %	
		Обучающие данные 80%	Обучающие данные 50%
1	DenseNet121	65	66
2	DenseNet169	56	62
3	enseNet201	67	69

4	EfficientNetB0	79	80
5	EfficientNetB1	82	83
6	EfficientNetB2	82	80
7	EfficientNetB3	83	82
8	EfficientNetB4	82	80
9	EfficientNetB5	82	82
10	EfficientNetB6	82	81
11	EfficientNetB7	83	81
12	InceptionResNetV2	41	41
13	InceptionV3	41	41
14	MobileNet	67	69
15	MobileNetV2	68	67
16	MobileNetV3Large	81	81
17	MobileNetV3Small	80	77
18	NASNetLarge	63	63
19	NASNetMobile	60	60
20	ResNet101	82	81
21	ResNet101V2	41	41
22	ResNet152	82	83
23	ResNet152V2	41	41
24	ResNet50	80	82
25	ResNet50V2	41	41
26	VGG16	82	79
27	VGG19	82	78
28	Xception	41	61

Так как предварительно обученная сверточная нейронная сеть EfficientNetB3 занимает меньший объем памяти, всего 48 MB, по сравнению 232 MB у ResNet152, для практического применения, в частности в мобильных приложениях, мы рекомендуем использовать именно эту сеть.

Таким образом, архитектура сети имеет следующий вид, показанный в табл. 3.5.

Таблица 3.5 – Архитектура нейронной сети на основе CNN EfficientNetB3

Layer (type)	Output Shape	Param #
efficientnetb3 (Functional)	(None, 7, 7, 1536)	10783535
global_average_pooling2d_4	(None, 1536)	0
dense_8 (Dense)	(None, 128)	196736
dropout_4 (Dropout)	(None, 128)	0
dense_9 (Dense)	(None, 6)	774

Сверточная основа EfficientNetB3 имеет 10 783 535 параметров, что представляет собой достаточно большое число. Классификатор, добавленный сверху, имеет около 200 тысяч параметров. Поэтому перед

обучением нейронной сети очень важно заморозить сверточную основу. Замораживание одного или нескольких слоев предотвращает изменение весовых коэффициентов в них в процессе обучения.

Если этого не сделать, тогда представления, прежде изученные сверточной основой, изменятся в процессе обучения на новых данных. Так как слои Dense сверху инициализируются случайными значениями, в сети могут произойти существенные изменения весов, фактически разрушив представления, полученные ранее.

На рис. 3.8 показано изменение точности классификации на проверочном наборе данных в процессе обучения на протяжении 200 эпох.

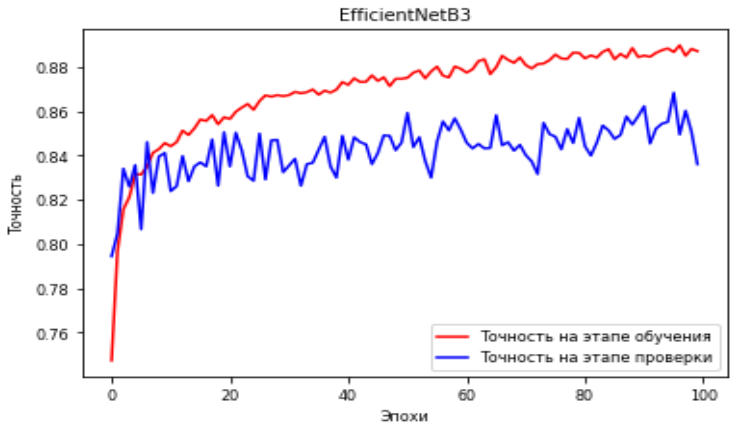


Рисунок 3.8 – Точность классификации на этапе обучения и проверки нейронной сети на основе предварительно обученной основы EfficientNetB3

Как видем из рисунка, обучая сеть на протяжении 97 эпох, можно достичь точности почти 87% на этапе проверки за счет тонкой настройки количества эпох обучения сети.

Результаты, полученные по методу интегрированных градиентов и показанные на рис. 3.9, дают представление об участках изображения, использующихся нейронной сетью для классификации изображений.

Для построения показанных на этом рисунке визуализаций интегрированного градиента использовались пиксели с уровнем значений использования сетью от 20 до 80% и нейронная сеть на базе сверточной основы EfficientNetB3.

Сеть сначала изучила внешние края и текстуру, а затем более абстрактные свойства изображений на более высоких уровнях абстракции, что, на наш взгляд, привело к нахождению отличительных особенностей, эффективных для классификации рентгеновских снимков.

Для практической проверки части СИИ был разработан прототип мобильного приложения для операционной системы Android с помощью языка JAVA на базе фреймворка TensorFlow Lite, предоставляющего набор инструментов, обеспечивающих реализацию машинного обучения на различных устройствах, позволяя использовать нейронные сети на мобильных, встроенных устройствах и устройствах Интернета вещей. Ключевыми особенностями этого фреймворка являются следующие:

- Оптимизация для работы на мобильных и встроенных устройствах.
- Поддержка нескольких платформ, включая устройства на базе Android и iOS , Linux и микроконтроллеров, например ESP32.
- Поддержка разнообразных языков, включая Java, Swift, Objective-C, C ++ и Python.
- Высокая производительность за счет использования доступного аппаратного ускорения.
- Подробная документация и большое количество примеров для часто встречающихся задач машинного обучения.

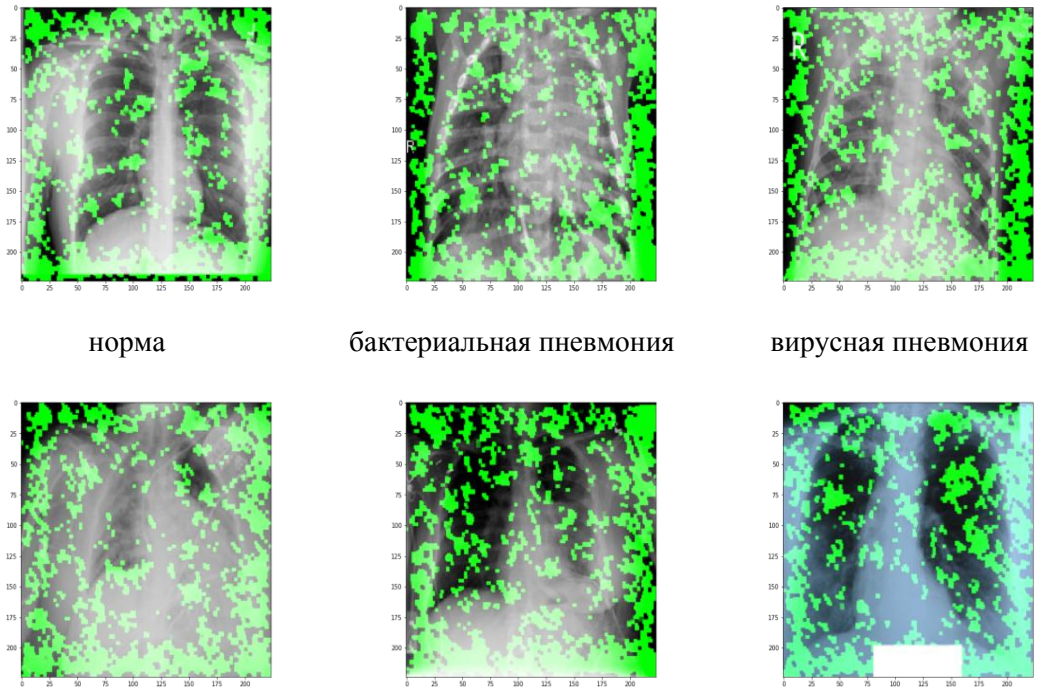


Рисунок 3.9 – Интерпретация результатов классификации различных изображений, показывающая, какие части изображения используются для классификации

Разработанные выше нейронные сети были конвертированы в специальный переносимый формат, принятый в библиотеке TensorFlow Lite, так называемый FlatBuffers, и сохранены в файле с расширением tflite. Этот формат дает несколько преимуществ по сравнению с форматом, принятым в библиотеке Keras, таких, как уменьшенный размер (что важно в мобильных приложениях) и более быстрый доступ, который осуществляется напрямую, без дополнительного этапа синтаксического анализа и распаковки, что позволяет СИИ эффективно работать на переносных устройствах с ограниченными вычислительными ресурсами и ресурсами памяти.

Полученные файлы в формате TensorFlow Lite дополнительно включают метаданные, содержащие описание нейронной сети, используемые ее классы и машиночитаемые данные для автоматического

создания конвейеров предварительной и постобработки во время работы на устройстве.

Разработанный прототип мобильного приложения реализует такие функции, как (рис. 3.10):

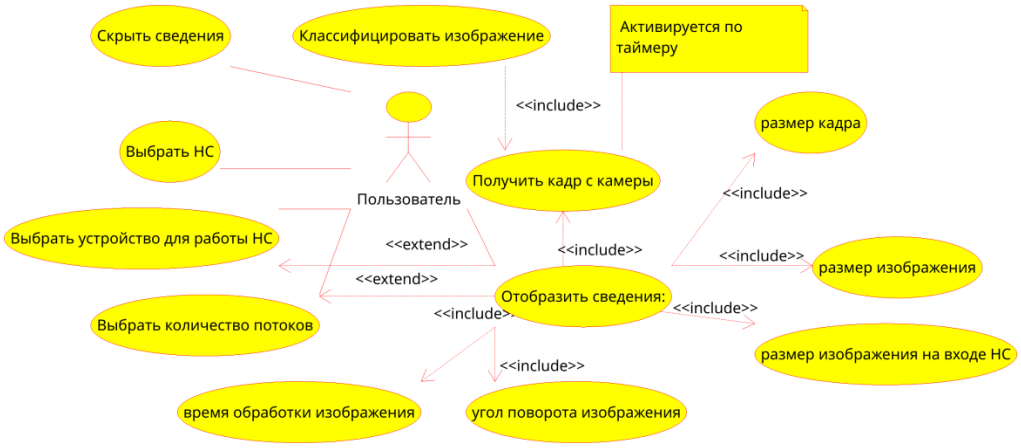
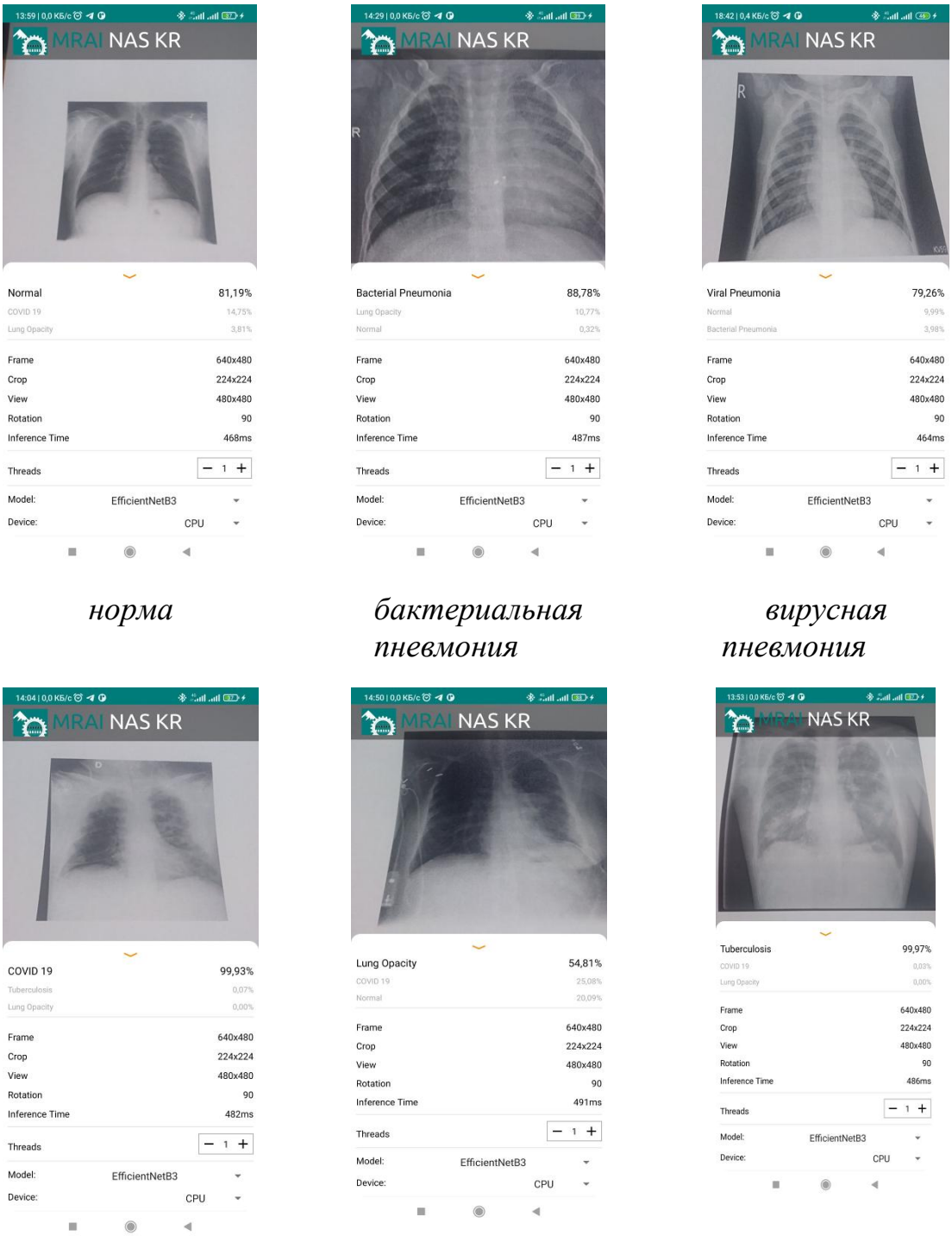


Рисунок 3.10 – Варианты использования прототипа программных средств

- Классификация изображения.
- Отображение сведений о режиме работы нейронной сети.
- Выбор количества потоков, используемых для работы нейронной сети.
- Выбор используемой для классификации нейронной сети (DenseNet169 илиEfficientNetB3).
- Выбор используемого для работы нейронной сети устройства (CPU или GPU).

Через определенные промежутки времени зависящее от времени обработки изображения приложение получает “сырые” кадры с камеры. Нейронная сеть выполняет классификацию изображения по двум или шести классам в зависимости от выбранной модели сети. Пользователь может узнать распознанный сетью класс изображения, скрыть или отобразить дополнительные сведения о работе приложения, такие как: размер кадра; размер и угол поворота изображения, используемого сетью; время обработки изображения; количество используемых потоков процессора. Дополнительные сведения могут быть использованы для отладки программных средств (рисунок 3.11).

Кроме того, пользователь может выбрать используемую для классификации нейронную сеть для диагностики по двум или шести классам, количество используемых потоков процессора, а также используемое нейронной сетью для работы устройство: CPU или GPU. В дальнейшем планируется дополнить разработанную программу базой данных изображений, присланных пользователями, что позволит улучшить точность работы СИИ.



норма *бактериальная пневмония* *вирусная пневмония*

COVID-19 *“затемнение легких”* *туберкулез*

Рисунок 3.11 – Интерфейс и примеры диагностики различных заболеваний с помощью разработанного прототипа программных средств

3.6 Выводы

Таким образом, в результате исследований было подтверждено, что рентгеновские снимки грудной клетки могут иметь большое значение для постановки диагноза пациентам с COVID-19 и также могут быть полезны

для диагностики различных видов пневмонии. Были собраны и проанализированы имеющиеся в сети Интернет рентгеновские снимки для диагностики COVID-19. Выполнен обзор и анализ литературы в области диагностики коронавирусной инфекции. Сформулированы критерии успешного решения задачи диагностики коронавирусной инфекции по рентгеновским снимкам в виде 6 характеристик для сравнения результатов обучения глубоких нейронных сетей. Выполнена предварительная подготовка найденных изображений для построения нейросетевого классификатора на базе предварительно обученных сверточных нейронных сетей, выделены уникальные рентгеновские снимки грудной клетки по каждому классу: 10200, 1345, 2538, 3616, 700, 6012. В результате анализа изображений сформулирована задача диагностики коронавирусной инфекции, как задача бинарной (COVID-19, норма) и многоклассовой классификации изображений грудной клетки по 6 классам: норма, вирусная пневмония, бактериальная пневмония, COVID-19, туберкулез, неуточненное заболевание легких. Разработаны и протестированы архитектуры на базе предварительно обученных сверточных нейронных сетей для дифференциальной диагностики коронавирусной инфекции и различных видов пневмонии. Выполнена регуляризация и настройка гиперпараметров предлагаемой нейронной сети: параметров прореживания, количества слоев, числа нейронов на слой, шага обучения оптимизатора. В результате найдены наиболее оптимальные настройки нейронной сети. Разработан прототип мобильного приложения для диагностики различных видов пневмонии, который используется для тестирования и усовершенствования разработанной СИИ.

4. Мониторинг подземных силовых кабельных линий

4.1 Разработка программного компонента системы мониторинга подземных силовых кабельных линий

Поиск линий связи и определение мест повреждений – традиционная задача, решаемая посредством известных приборов, но далеко не все эти приборы отвечают современным требованиям [58]. В настоящее время высокая плотность расположения подземных кабельных линий и огромное количество источников электромагнитных помех, таких как сети связи, воздушные ЛЭП и т.д., требуют значительного повышения точности и надежности. Поэтому существует необходимость разработки нового устройства – легкого, многофункционального, эргономичного и интеллектуального.

Рядовыми работниками применяются трассоискатели различных типов, но до сих пор это чаще всего устаревший комплект КИ-4П, КИ-4ПГ, КИ-4ПИ, который был разработан в 80-х гг. прошлого века в Центральном научно-исследовательском институте связи (ЦНИИС). Трассоискатель КИ-4П дает возможность определять трассу и глубину залегания кабеля в условиях сильных помех при небольшой мощности генератора – всего 2 Вт. Недостатки трассоискателя КИ-4: сложная супергетеродинная схема приемного устройства, большие габариты и вес приемника, генератора и антенны – в общем более 10 кг. В настоящее время широко используется цифровая модификации этого прибора со стрелочной и акустической индикацией КИ-7 [59], а также ПОИСК-210Д-2, -310, -410, ЛИДЕР, SG-600, SG-80 и другие. Среди основных требований, предъявляемых к современному трассоискателю, можно выделить в первую очередь его массогабаритные характеристики, в связи с тем, что линейный персонал, кроме них, как правило, носит с собой и другое необходимое оборудование, и расходные материалы. Другим немаловажным требованием является удобство и простота использования, из чего следует, что в перспективе традиционные аналоговые стрелочные и акустические системы индикации и механические органы управления должны быть заменены на более удобные цифровые. Последним немаловажным требованием является цена. Снизить цену кабелепоискового комплекса может использование в качестве средств и индикации и управления, современных широко распространённых мобильных и планшетных персональных компьютеров, оснащенных достаточно мощным процессором, сенсорным дисплеем и средствами мобильной связи, а также модулями Wi-Fi и Bluetooth.

Таким образом, современная кабелепоисковая система должна быть снабжена цифровой системой обработки данных и индикации результатов, что позволит ей эффективнее бороться с помехами и точнее определять расположение кабельных трасс, снизит ее вес и габариты, увеличив при этом удобство и упростив ее использование. Достижение этих результатов

возможно за счет сокращения и упрощения аналоговой части трассоискателя, при этом часть ее функций возьмет на себя цифровая часть системы, осуществляющая дополнительную фильтрацию, анализ данных и индикацию результатов.

Возможная структура такой трассопоисковой системы показана на рисунке 4.1. Неотъемлемой частью современной трассопоисковой системы, как системы, производящей обработку сигналов, является звено, обеспечивающее возможность передачи информации в персональную или портативную ЭВМ [59].

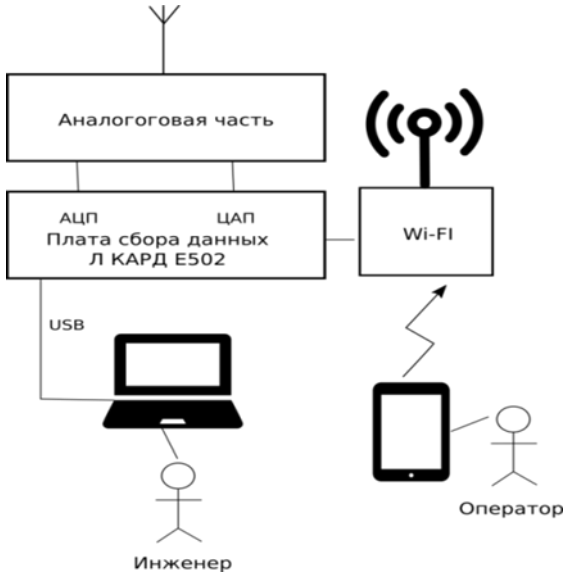


Рисунок 4.1 - Структура системы мониторинга

Для этой цели могут использоваться различные аппаратно-программные средства. Как правило, программное обеспечение трассоискателей поставляется в составе аппаратно-программного комплекса и используется для контроля над распространением продукта, обновления прошивки трассоискателя и настройки ее конфигурации, настройки дополнительных пользовательских частот, на которых может работать трассоискатель. В качестве примера можно привести такие программы, как RD Manager [60] , LOGICAT. Эти программы кроме того позволяют проводить тест самодиагностики трассопоисковой системы, а также управлять информацией о результатах сканирования и об обнаруженных коммуникациях, в том числе проводить быстрый поиск информации в сохраненных данных, просматривать результаты сканирования рабочих участков и осуществлять автоматический контроль исправности трассоискателя [61].

Как недостаток этого программного обеспечения может быть отмечено то, что оно, само собой разумеется, жестко привязано к аппаратному обеспечению одного определенного производителя, имеет закрытый исходный код и поэтому не может быть использовано для создания собственного трассопоискового комплекса.

В качестве аппаратных средств передачи сигналов благодаря своей универсальности может использоваться устройство сбора данных Л Кард E502. Эта плата представляет собой универсальный 16- битный модуль ввода/вывода до 32 аналоговых и 17 цифровых сигналов в компьютер через интерфейсы USB 2.0 (high-speed) и Ethernet (100 Мбит) с частотой преобразования до 2 МГц и возможностью их цифровой обработки в реальном времени. Кроме того, ООО «Л Кард» приветствует интеграцию своих модулей в пользовательские системы. [16].

Аналого-цифровые преобразователи (АЦП) и цифро-аналоговые преобразователи (ЦАП) этой платы могут быть использованы для соединения с аналоговой частью трассопоисковой системы. Плата сбора данных Л Кард E502 может быть подключена к ноутбуку или персональному компьютеру инженеров-разработчиков системы с помощью интерфейса USB для обновления прошивки или загрузки специальных программ для цифрового сигнального процессора ADSP-BF523, которым оснащена эта плата [62]. С помощью интерфейса Ethernet плата сбора данных может быть подключена к Wi-Fi роутеру и через него к мобильным и планшетным компьютерам операторов (рис. 0).

С модулем Л Кард E502 поставляется драйвер, написанный на языке С и библиотеки для языков программирования C++, C# и LabVeiw. Однако в области анализа технических данных с интерактивной визуализацией результатов в настоящее время используются такие предметно-ориентированные языки программирования и инструменты – с открытым исходным кодом и коммерческие, как R, MATLAB, SAS, Stata и Python.

Важно, что драйвер, предоставляемый компанией Л Кард вместе с исходным кодом [63], работает по стандартному протоколу шины USB 2.0, реализованному для практически всех операционных систем, что позволяет скомпилировать и использовать драйвер для мобильных и планшетных персональных компьютеров, оснащенных процессорами с архитектурой ARM на базе операционной систем Android. Однако особенности реализации драйвера, в частности использованием им стандарта POSIX Threads для синхронизации потоков сбора данных и управления устройством, несколько ограничивает возможность его использования. Реализации данного API существуют для большого числа UNIX-подобных ОС (GNU/Linux, Solaris, FreeBSD, OpenBSD, NetBSD, OS X), а также для Microsoft Windows, но в операционной системе Android поддержка этого стандарта в полной мере присутствует только начиная с версии Lollipop, т.е. Android 5.0.

В комплекте с устройством сбора данных поставляется также бесплатно распространяемая программа LGraph2, предназначенная для регистрации, визуализации и обработки аналоговых сигналов, записанных с помощью измерительных устройств или модулей АЦП производства ООО «Л Кард». Однако возможности анализа данных этой программой весьма ограничены, в частности, с помощью этой программы нельзя

выполнить вейвлет-анализ данных и привязку результатов измерения к карте местности, что необходимо для фиксации расположения кабельных линий. Кроме того, это программное обеспечение жестко привязано к операционной системе Windows, поставляется без исходного кода, и поэтому ее использование в мобильных и планшетных компьютерах с операционной системой Android оказывается затруднительным.

В научном парке МГУ разработана виртуальная измерительная лаборатория PowerGraph, предоставляющая пользователю широкие возможности регистрации, визуализации, редактирования, обработки и анализа данных, полученных в том числе и с помощью устройства сбора данных Л Кард E502 [64]. Однако это программное обеспечение поставляется с аппаратным USB-ключом и имеет достаточно высокую стоимость, что не позволяет использовать его в собственных разработках.

С момента своего появления в 1991 году Python стал одним из самых популярных динамических языков программирования наряду с Perl, Ruby и другими. Из всех интерпретируемых динамических языков Python выделяется большим и активным сообществом среди исследователей по всему миру, использующих его для научных и инженерных расчетов. Применение Python для этой цели в промышленных и технических кругах значительно расширилось с начала 2000-х годов. Сравнительно недавнее появление библиотек различных математических методов обработки данных для Python (прежде всего NumPy, pandas) сделало его серьезным конкурентом в решении задач манипулирования данными [65].

В сочетании с достоинствами Python как универсального языка программирования это делает его лучшим на сегодняшний день выбором для создания программного обеспечение устройств, для которых требуется реализация сложных алгоритмов обработки данных.

Таким образом, в настоящее время не существует какого-либо готового программного обеспечения для создания новой трассопоисковой системы на базе модуля Л Кард E502, требующей сложной цифровой обработки сигналов и управления данными. Поэтому актуальной задачей является разработка библиотеки для языка Python, позволяющей работать с устройством сбора данных Л Кард E502, и создание программного компонента трассоискателя на языке Python для цифровой обработки и регистрации полученных данных, позволяющего эффективнее бороться с помехами и надежнее определять расположение кабельных линий.

4.1.1 Программный компонент системы мониторинга. Разрабатываемый программный компонент системы должен решать задачи сбора, регистрации, визуализации, фильтрации и анализа данных, позволяя при этом выбрать: модуль сбора данных; тип подключения датчика к устройству сбора данных; канал и пределы измерения; частоту сбора данных; тип и параметры фильтра, а также параметры вейвлет-анализа.

Кроме того, необходимо визуализировать координаты найденных кабельных линий с возможностью просматривать и сохранять результаты

работы на карте. Важно еще и то, что разрабатываемое приложение должно быть кроссплатформенным, т.е. должно запускаться на различных операционных системах, в том числе Linux, Windows, OS X, Android, iOS, и операционных системах Raspberry Pi, кроме того, оно должно работать на различных архитектурах, по крайней мере на ARM и x86-64. Это требуется для того, чтобы программное обеспечение трассоискателя могло работать на широком круге мобильных и планшетных ПК операторов, а также на стационарных компьютерах инженеров.

Диаграмма вариантов использования программных средств показана на рис. 4.2.

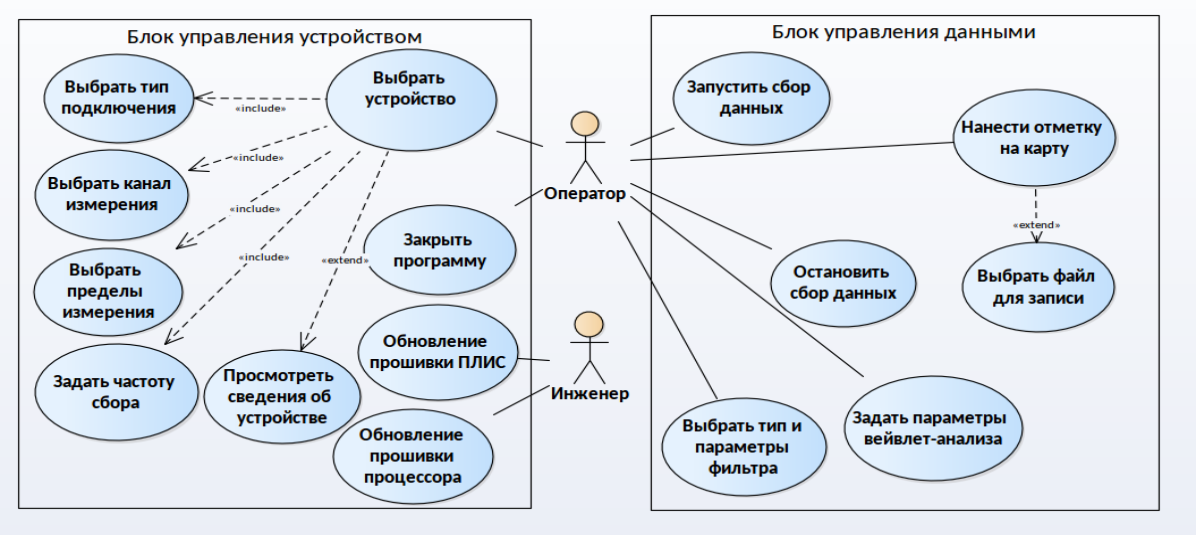


Рисунок 4.2 – Диаграмма вариантов использования программных средств

С помощью разработанного программного компонента возможно использовать все распространенные типы цифровых фильтров: фильтр верхних частот, фильтр нижних частот, полосовой фильтр и фильтр-пробку, так как вблизи ЛЭП и трубопроводов сигналы промышленной частоты 50 Гц и ее гармоники затрудняют поиск неисправности на линии [58]. И, наконец, собранные данные должны быть привязаны к карте местности и сохранены в файл в формате PNG (Portable Network Graphics – портативная сетевая графика). Для привязки к карте местности можно использовать встроенные GPS-приемники, имеющиеся сейчас практически во всех мобильных и планшетных компьютерах, а также встроенные в операционную систему Android средства управления ими.

Важная функция программы – визуализация данных в виде временных рядов. В режиме реального времени можно видеть график исходных данных, поступающих с датчика, график отфильтрованных данных, а также результат вейвлет-преобразования исходного временного ряда в виде двумерной скалограммы.

Обычно для построения оценок спектра мощности временных рядов используют преобразование Фурье. Оно обладает способностью фокусировать в точку, «размазанную» по времени, информацию о периодичности функции при переходе из временной области в частотную.

Достигается это за счет того, что ядро преобразования Фурье не локализовано во времени, но имеет предельную локализацию в частотной области. Это обстоятельство и делает преобразование Фурье хорошим инструментом для изучения процессов, характеристики которых не меняются со временем. Напротив, вейвлет-анализ основан на использовании локализованных во времени ядер преобразования, размеры которых согласованы с масштабом изучаемых компонентов ряда. Основная идея вейвлет-преобразования отвечает специфике временных рядов, получаемых путем оцифровки сигнала датчика кабелеискателя, демонстрирующих эволюцию во времени своих основных характеристик – среднего значения, дисперсии, периодов, амплитуд и фаз гармонических компонентов.

В программном компоненте трассопоисковой системы для обработки сигнала трассоискателя используется непрерывное вейвлет-преобразование, которое для функции $f(t) \in L^2(R)$ задается следующим образом:

$$W(a,b) = \frac{1}{|a|^{1/2}} \int_{-\infty}^{\infty} f(t)\psi^*\left(\frac{t-b}{a}\right),$$

где $a.b \in R, a \neq 0$, a – масштаб вейвлета, b – сдвиг вейвлета, $\psi(t)$ – материнская вейвлет-функциия или вейвлет-базис, а символом * обозначена процедура комплексного сопряжения. В табл. 3.6 приведены используемые в программном компоненте вейвлет-базисы.

Таблица 3.6 – Используемые вейвлет-базисы

Название вейвлет-базиса	Определение	Фурье-образ
DOG (derivative of a Gaussian – производная функции Гаусса)	$\psi_m(t) = (-1)^m \partial_t^m \left[e^{-\frac{t^2}{2}}\right]$, где $\partial_t^m = \partial^m[...]/\partial t^m, m \geq 1$	$\hat{\psi}_m(k) = m(ik)^m e^{-\frac{k^2}{2}}$
Морли	$\psi(t) = e^{ik_0 t} e^{-\frac{t^2}{2}}$	$\hat{\psi}(t) = \Theta(k) e^{-\frac{(k-k_0)^2}{2}}$, где $\Theta(k)$ – функция Хевисайда
Пауля	$\psi(t) = \Gamma(m+1)\frac{i^m}{(1-it)^{m+1}}$	$\hat{\psi}(t) = \Theta(k) k^m e^{-k}$

Так как сигнал, получаемый с платы сбора данных, уже дискретизирован и оцифрован, его можно представить значениями функции, следующими друг за другом с постоянным шагом Δt:

$$f_k = f(t_k), t_k = k\Delta t, k = 0,1,..., N-1$$

тогда вейвлет-преобразование дискретной функции f_k:

$$W(a,b) = \frac{1}{n(a,b)} \sum_{k=0}^{N-1} f_k \psi^*\left(\frac{t_k - b}{a}\right), \quad (4.1)$$

где $n(a,b) = \sum_{k=0}^{N-1} e^{-\frac{1}{2}\left(\frac{t_k - b}{a}\right)^2}$.

Так же используется оценка локального спектра энергии, так называемая скалограмма [66]:

$$S(a_i, b_j) = \left|W_A(a_i, b_i)\right|^2, \quad (4.2)$$

Выбор функции $\psi(t)$, базисного вейвлета, как правило, определяется тем, какую информацию необходимо извлечь из сигнала. Каждый вейвлет имеет характерные особенности во временном и в частотном пространстве, поэтому иногда, с помощью разных вейвлетов, можно полнее выявить и подчеркнуть те или иные свойства анализируемого сигнала. Например, МНАТ-вейвлет (Mexican hat – «Мексиканская шляпа») получается из DOG при *m=2*. Он имеет узкий энергетический спектр и два равных нулю момента, хорошо приспособлен для анализа сложных сигналов, так как коэффициенты *W(a, b)* зависят от малого интервала области частот вейвлета. Вейвлет Морли представляет собой плоскую волну, модулированную гауссианом единичной ширины. С увеличением k_0 возрастает частотная избирательность базиса, но ухудшается временное разрешение вейвлет-преобразования.

Необходимо отметить также, что у вейвлета Морле равен нулю только нулевой момент, поэтому он чувствителен к линейному тренду. Другим часто применяемым базисом является вейвлет Пауля. Чем больше m , тем больше нулевых моментов имеет этот вейвлет [67]. Таким образом, каждый вейвлет имеет свои характерные особенности, и поэтому программные средства предоставляют возможность выбора базового вейвлета.

4.1.2 Структура программного компонента системы мониторинга. Своим успехом в качестве платформы для технических расчетов Python отчасти обязан простоте интеграции с кодом на C, C++ и FORTRAN. Во многих современных вычислительных средах применяется общий набор унаследованных библиотек, написанных на FORTRAN и C, содержащих реализации алгоритмов линейной алгебры, оптимизации, интегрирования, быстрого преобразования Фурье и других. Поэтому многочисленные компании и исследовательские институты используют Python как средство для интегрирования написанных за 30 лет программ [68].

Использование в качестве языка программирования Python позволяет задействовать описанные ниже внешние компоненты без изменения исходного кода (рис. 4.3). NumPy реализует численные алгоритмы и работу с матрицами. NumPy, сокращение от Numerical Python – основной пакет для выполнения научных и технических расчетов на Python. Большая часть вычислений в разработанной программе базируется на NumPy и построенных поверх него библиотек. В числе прочего он предоставляет:

- быстрый и эффективный объект многомерного массива;
- функции для выполнения вычислений над элементами одного

массива или математических операций с несколькими массивами;

- средства для чтения и записи на диски наборов данных, представленных в виде массивов;
- операции линейной алгебры, преобразование Фурье;
- средства для интеграции с кодом, написанным на C, C++ или Fortran.

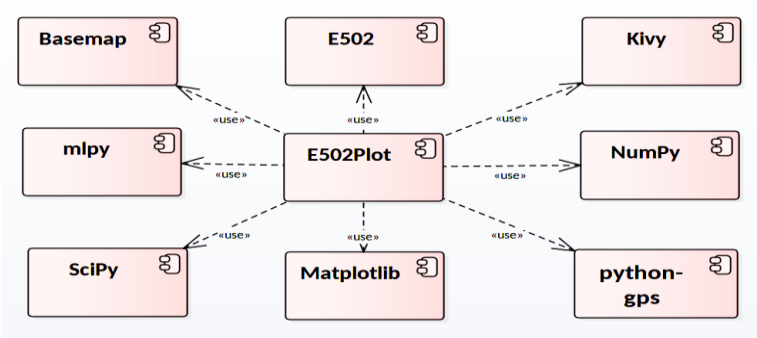


Рисунок 4.3 Структурная схема программы

Помимо быстрых средств работы с массивами, одной из основных целей NumPy в части анализа данных является организация контейнера для передачи данных между алгоритмами. Как средство хранения и манипуляции данными массивы NumPy куда эффективнее встроенных в Python структур данных.

SciPy – библиотека научных алгоритмов для Python, из этого модуля импортируются функции, реализующие фильтрацию данных. SciPy используется в программе для решения различных стандартных вычислительных задач. В программе задействованы такие компоненты, как:

• scipy. signal, содержащий средства для обработки сигналов;

• scipy. special: обертка вокруг SPECFUN, написанной на Fortran библиотеке, содержащей реализации многих стандартных математических функций, в том числе гамма-функции, которая необходима для вычисления некоторых вейвлетов.

Библиотека Matplotlib – самый популярный в Python инструмент для создания графиков и других способов визуализации двумерных данных. Первоначально она была написана Джоном Д. Хантером (John D. Hunter, JDH), а теперь сопровождается большой группой разработчиков и поэтому имеет достаточно хорошую документацию и сопровождение. Basemap используется для нанесения результатов работы на карту. Пакет поддерживает несколько географических проекций и средства для преобразования широты и долготы в двумерный график Matplotlib.

Из библиотеки машинного обучения mlpy импортируются функции вейвлет-анализа данных, реализующие формулы (4.1), (4.2). Модуль python-gps предоставляет вспомогательные функции определения географического местоположения для привязки данных к карте местности,

Kivy применяется для построения графического интерфейса пользователя (GUI – Graphic User Interface).

4.1.3 Архитектура программного компонента системы мониторинга. Для упрощения разработки и сопровождения код программы был разбит на несколько классов (рис. 4.4), объединенных в модуль E502Plot.

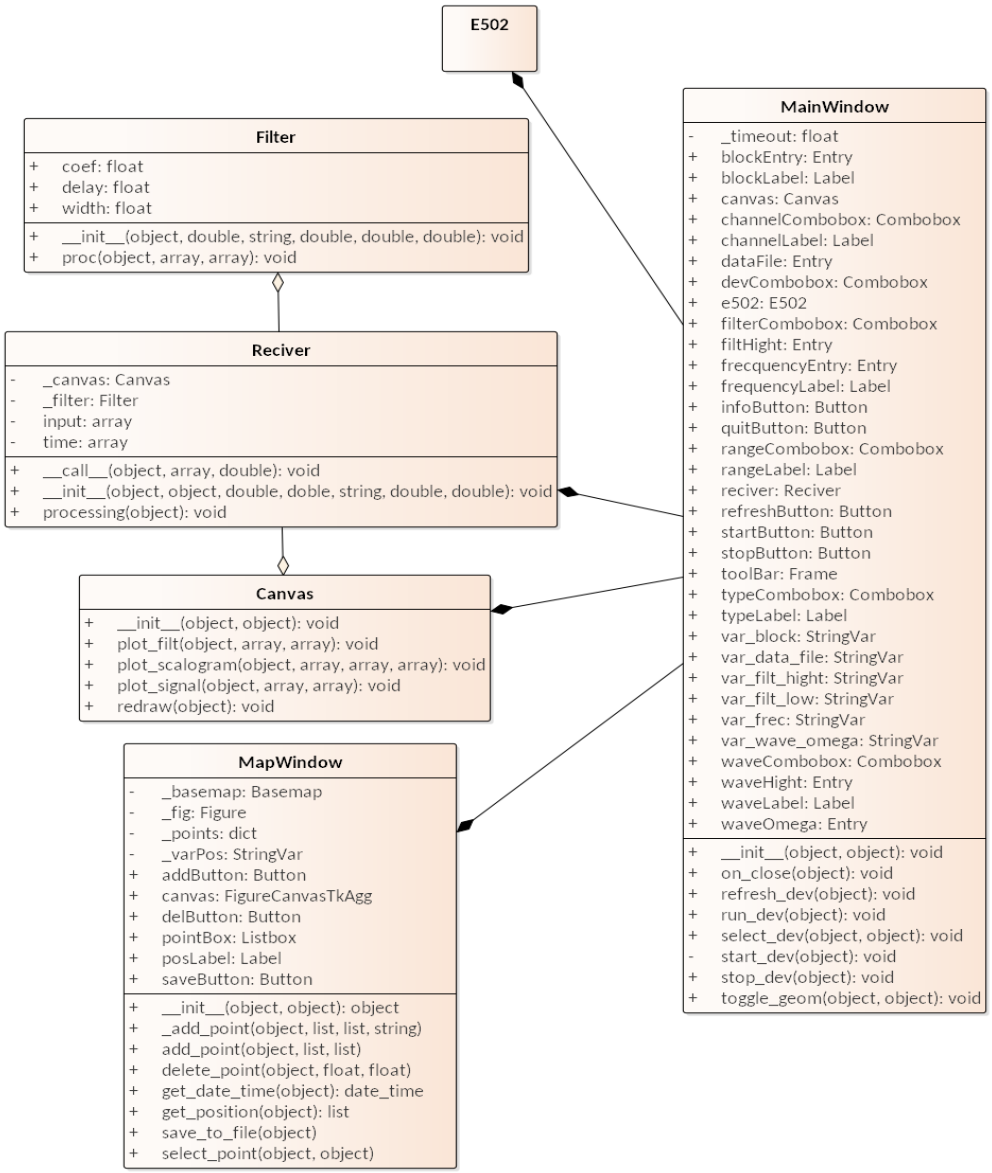


Рисунок 4.4 – Диаграмма классов программного компонента системы мониторинга

Класс MainWindow реализует основное окно программы и предоставляет пользователю интерфейс для взаимодействия с устройством сбора данных. В главном окне расположены графики исходного и отфильтрованного временного ряда, а также его скалограмма. Вывод графических данных реализуется посредством класса Canvas. Непосредственное взаимодействие с устройством сбора данных осуществляется с помощью класса Reciver, который использует модуль mlpy и класс Filter для обработки и фильтрации данных.

Класс MapWindow реализует вспомогательное окно, в котором отображается карта местности, текущее положение, а также все отмеченные пользователем пункты на карте с возможностью добавить новые или удалить ошибочно отмеченные точки на карте.

Класс E502 (рис. 4.5) представляет собой программную абстракцию устройства сбора данных, реализованную на языке Cython, вышедшего за последние несколько лет на одно из первых мест в области создания быстрых компилируемых расширений Python и организации интерфейса с кодом на C и C++.

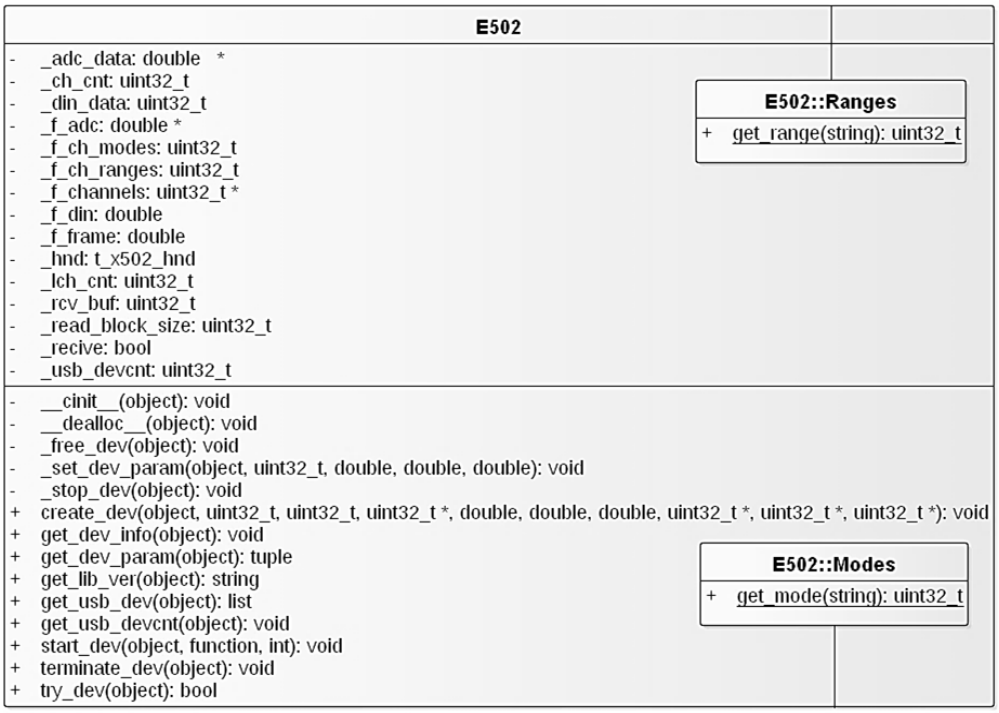


Рисунок 4.5 – Классы библиотеки для устройства сбора данных Л Кард E502

В настоящее время Cython стал излюбленным инструментом многих разработчиков, пишущих на Python код инженерных приложений, который должен взаимодействовать с библиотеками на C или C++. Поэтому и интерфейс библиотеки платы сбора данных написан на языке Cython и оформлен в виде классов, показанных на рис. 4.5. В коде на языке Cython возможно организовать прямой вызов C-функций, использующих такие низкоуровневые C-конструкции, как указатели, структуры, динамические массивы др. Так как Cython занимает промежуточное положение между языками C и Python, он обеспечивает полный контроль над всеми аспектами взаимодействия двух языков, оставаясь при этом совместимым с Python, что делает интерфейсный код на Cython более удобным для понимания и отладки по сравнению с другими инструментами (такими, как SWIG, SIP, Boost.Python, ctypes, cffi), предназначенными для этой цели. Благодаря такой организации функции, импортированные с помощью Cython, имеют производительность на уровне C, за вычетом минимальных накладных расходов на организацию совместимого с Python интерфейса,

так как код на Cython сначала транслируется на язык C, а затем компилируется в бинарный код. В частности, написанная на Cython функция get_usb_dev находит все подключенные по интерфейсу USB устройства, возвращая список устройств и полную информацию о них, включая название, серийный номер и основные характеристики каждого устройства, а функция get_usb_devcnt возвращает количество устройств, подключенных по интерфейсу USB. С помощью функции create_dev можно инициализировать параметры устройства сбора данных, такие как используемый тип подключения, канал, частоту сбора данных и пределы для измерения, с помощью функции get_dev_param можно получить установленные с учетом технических ограничений параметры устройства. Функция try_dev позволяет проверить, были ли параметры устройства успешно проинициализированы, а функция get_dev_info узнать полные технические характеристики устройства.

Наконец функции start_dev и terminate_dev позволяют запустить и остановить сбор данных. Классы Modes и Ranges позволяют преобразовать строковое представление возможных типов подключений и пределов измерений платы сбора данных во внутреннее, используемое в драйвере устройства.

Общий алгоритм работы программного компонента трассоискателя на базе устройства сбора данных Л Кард E502 (рис.4.6) включает в себя следующие основные пункты:

1. При работе с платой сбора данных по интерфейсу USB необходимо получить список серийных номеров с помощью функции get_usb_dev и их количество с помощью get_usb_devcnt. При работе по Wi-Fi должен быть известен IP-адрес устройства и необходимо перейти сразу к пункту 3.
2. Если в системе присутствует нужный модуль, требуется создать описатель модуля с помощью create_dev. При этом устанавливается соединение с платой сбора данных.
3. Чтобы установить связь с модулем по Wi-Fi с использованием IP-адреса, необходимо использовать функцию create_dev_byipaddr.
4. При необходимости получить дополнительную информацию о устройстве с помощью get_dev_info (в частности, для проверки необходимости и выполнения обновления микропрограмм).
5. Проверка параметров работы платы сбора данных с помощью get_dev_param.
6. Проверка инициализации устройства и его готовности к работе с помощью функции try_dev. Если устройство не готово к работе, то необходимо устранить неисправности и перейти к пункту 2.
7. Сбор данных в синхронном режиме работы.
8. Построение графика временного ряда.
9. Построение скалограммы временного ряда.

10. Цифровая фильтрация и построение графика отфильтрованного временного ряда.
11. Нанесение найденной точки трассы кабеля на карту.
12. Повторение шагов 7–10 до тех пор, пока оператор не прервет сбор данных.
13. Закрытие связи с модулем с помощью функции free_dev.

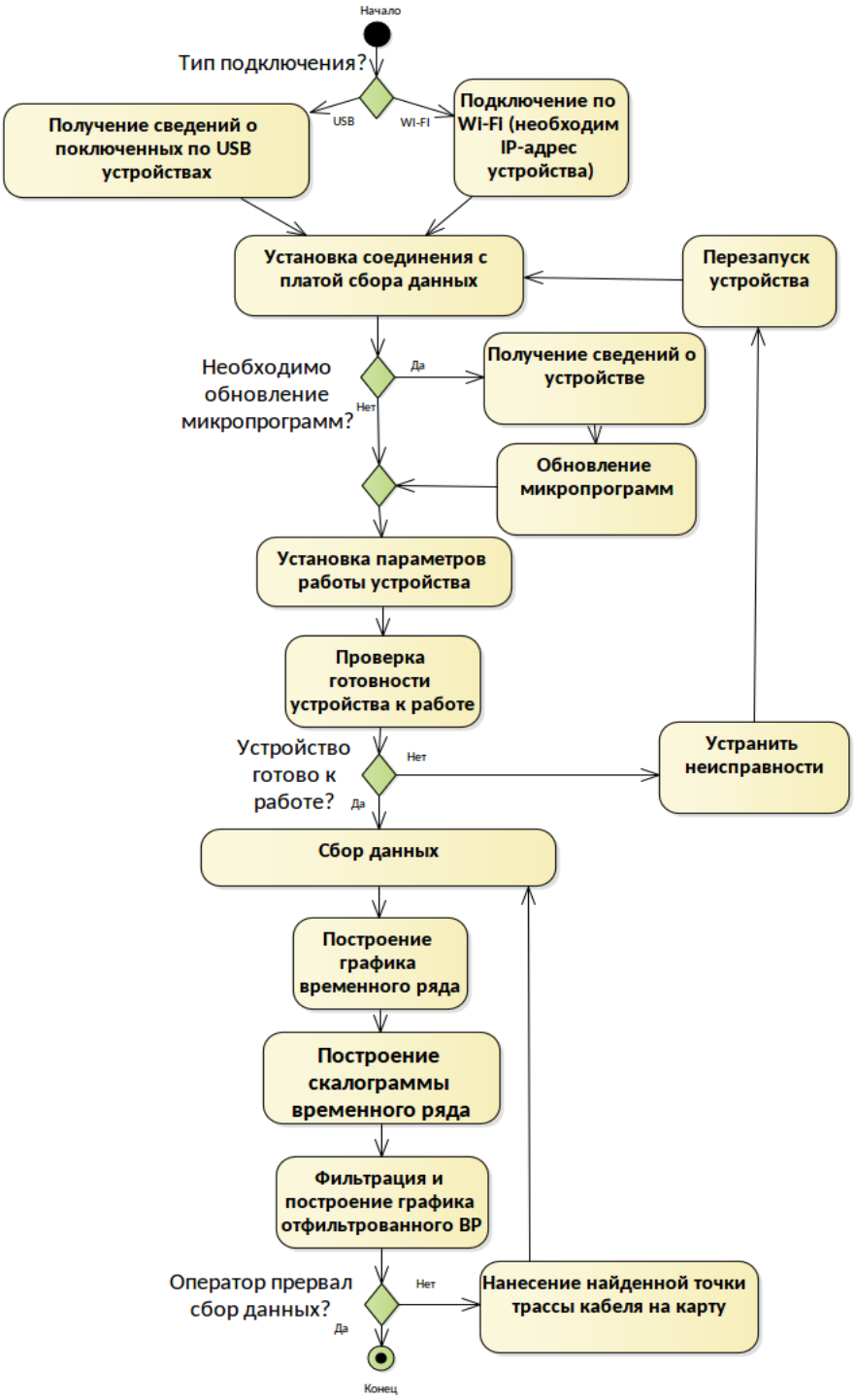


Рисунок 4.6 – Алгоритм работы программного компонента

4.1.4 Описание интерфейса пользователя. Интерфейс пользователя состоит из главного окна и вспомогательного окна с картой для визуализации найденной трассы кабеля. В главном окне расположены элементы интерфейса, позволяющие выбрать и настроить устройство сбора данных (рис. 4.7), также здесь находятся графики исходного и отфильтрованного временного ряда и скалограмма исходного ряда данных.

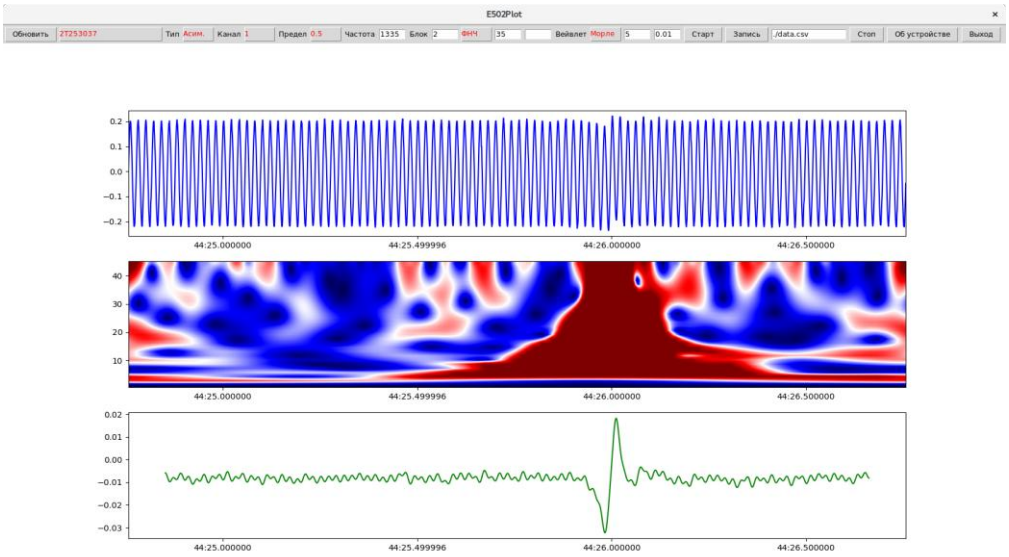


Рисунок 4.7 – Главное окно программы

Для визуализации координат найденных кабельных линий предусмотрено специальное окно (рис. 4.8), позволяющее по отдельным точкам измерений создавать и редактировать найденную трассу кабеля, а также экспортировать результаты работы в графический файл в формате PNG (Portable Network Graphics – портативная сетевая графика)

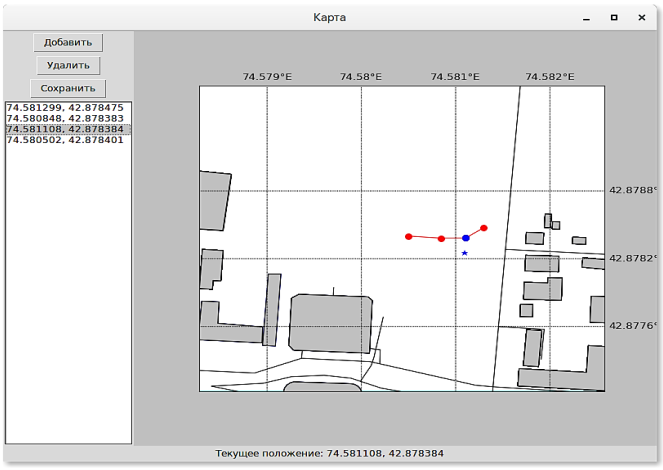


Рисунок 4.8 – Диалоговое окно карты с нанесенной на нее трассой кабеля

Для того чтобы можно было узнать технические характеристики устройства, предусмотрено диалоговое окно свойств устройства сбора данных (рис. 4.9), в котором, кроме технической информации, отображаются версии микропрограмм основного, сигнального процессоров и ПЛИС устройства.

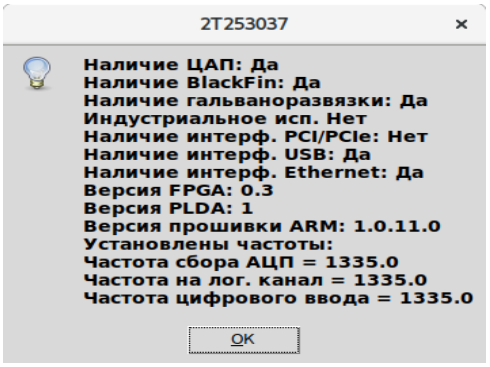


Рисунок 4.9 – Диалоговое окно свойств платы сбора данных

4.2 Мобильный программный компонент системы мониторинга

Выше разработана архитектура программного компонента системы мониторинга на базе устройства сбора данных ЛКАРД E502, основанная на использовании языка Python, и модулей, предназначенных для обработки и визуализации данных. Как показали лабораторные испытания программной реализации данной архитектуры, важнейшим требованием, предъявляемым к такому программному обеспечению, является его кроссплатформенность, то есть способность работать на стационарных и портативных вычислительных устройствах с процессорами различной архитектуры под управлением разнообразных операционных систем. Это необходимо как в целях увеличения потенциального круга пользователей, так и для расширения функциональных возможностей трассоискателя, связанных с применение встроенного во многие мобильные устройства gps-приемника и магнитного компаса, а также удобства его практического использования. Трассоискатель с кроссплатформенным программным компонентом можно использовать на стационарных компьютерах с архитектурой процессора x86-64 для отладки и настройки аналоговой части трассоискателя, так и без изменения исходного кода на смартфонах и планшетных компьютерах с архитектурой процессора с архитектурой ARM под управлением операционных систем Android, iOS, и Windows 10 Mobile. В разработанном ранее варианте программного компонента кроссплатформенность по большей части уже была достигнута за счет применения языка программирования Python, и поэтому разработанный программный компонент может успешно работать под управлением операционных систем Windows и Linux, и, в частности, возможна реализация трассоискателя в виде независимого устройства на базе одноплатного компьютера Raspberry Pi. Однако некоторые части программного компонента трассоискателя, такие как графический интерфейс пользователя, библиотека визуализации данных matplotlib и драйвер платы сбора данных, тесно связаны с оборудованием и поэтому так или иначе должны быть портированы, т.е. адаптированы соответствующим образом, чтобы успешно взаимодействовать с различными устройствами используемой программно-аппаратной среды.

Задачей настоящего исследования является разработка и практическая реализация кроссплатформенной архитектуры программного компонента трассоискателя, выбор оптимальных методов и инструментальных средств обеспечения портирования программного компонента трассоискателя, позволяющих с минимальными усилиями, без изменения исходного кода использовать его на вычислительных устройствах с разнообразным аппаратным окружением, управляемым различными операционными системами.

4.2.1 Используемые методы и инструментальные средства. Kivy – это современное инструментальное средство (toolkit, тулкит) построения графического интерфейса пользователя, позволяющее создавать эргономичные интерфейсы для широкого спектра устройств. Скорость выполнения Kivy сопоставима с нативной мобильной альтернативой Java для Android или Objective C для iOS [3]. Основная идея, являющаяся ключевой для понимания всех преимуществ данного тулкита, – это модульность и абстракция. Архитектура Kivy абстрагирована от таких базовых задач, как открытие окна, отображение графики и текста, воспроизведение звука, получение и отправка информации с устройств ввода/вывода, и так далее. Это делает API (application programming interface, программный интерфейс приложения) расширяемым и простым в использовании. И, что важнее, это позволяет использовать низкоуровневые модули операционной системы, в которой запускается приложение. Например, в iOS, Andoroid и Windows существуют собственные API для подобного рода базовых задач.

Часть кода, которая использует один из этих конкретных API-интерфейсов для связи с операционной системой с одной стороны и с Kivy с другой (выступая в качестве промежуточного уровня, обеспечивающего совместимость) – называется в терминах библиотеки Kivy провайдером (поставщиком). Это такие модули, как Mouse, FFMpeg, Cairo (Рисунок 4.10).

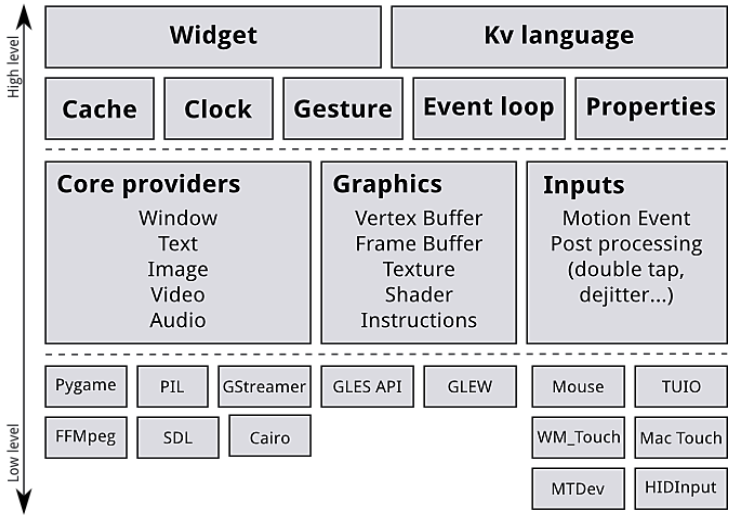


Рисунок 4.10 – Архитектура тулкита Kivy

Провайдер – это фрагмент кода, который предоставляет поддержку для определенного устройства, например, сенсорного экрана или эмулятора мыши. Для того чтобы добавить поддержку нового устройства ввода, такого, как плата сбора данных, достаточно разработать новый класс, который считывает входные данные с этого устройства и преобразует их в формат, принятый в библиотеке Kivy. Преимущество использования специализированных провайдеров для каждой платформы состоит в том, что это позволяет максимально эффективно использовать функциональность, предоставляемую операционной системой. Кроме того, за счет использования модулей, поставляемых с платформой, значительно уменьшается размер дистрибутива Kivy и упрощается упаковка. Это также упрощает перенос Kivy на другие платформы. Та же концепция используется при обработке ввода и вывода данных с помощью устройства сбора данных ЛКард E502. Разработка провайдера для этого устройства является основной целью настоящего исследования.

Графический API Kivy (Graphics) – представляет собой абстракцию OpenGL, позволяющую кроссплатформенно использовать графическое ускорение на мобильных устройствах. Все элементы графического интерфейса используют этот графический API, реализованный по соображениям производительности на языке C.

Базовый пакет (Core) предоставляет часто используемые функции, такие, как часы, чтобы запланировать события таймера. Поддерживаются как одноразовые, так и периодические таймеры. Таймеры используются в программном компоненте трассоискателя для реализации сбора данных пакетами с определенной частотой дискретизации. В базовый пакет входят также функции кэширования часто используемых данных, также применяемые в программном компоненте при обработке полученных данных.

Модуль (Inputs) предоставляет функции распознавания жестов, вводимых с помощью сенсорного экрана и используемых для управления трассоискателем. Язык Kivy используется для простого и эффективного описания графического интерфейса пользователя. Модуль UIX содержит часто используемые виджеты и макеты, которые применяются для создания пользовательского интерфейса. Виджеты (Widgets) – это элементы пользовательского интерфейса, добавляемые в программу для обеспечения требуемой функциональности. В программном компоненте трассоискателя используются такие виджеты, как файловый браузер, кнопки, ползунки, списки и др. Макеты применяются в графическом интерфейсе для размещения виджетов, так как их позиции зависят от размера экрана устройства, и рассчитать их самостоятельно оказывается довольно затруднительным. Поэтому используются предоставляемые тулкитом Kivy макеты сетки или макеты блоков, а также вложенные макеты. Кроме того, Kivy предоставляет модуль, позволяющий отображать некоторую текстовую информацию в удобном для восприятия на

мобильных устройствах виде, позволяющий, например, отображать количество кадров данных, получаемых с платы сбора данных. Наконец Kivy предоставляет абстракцию различных устройств и источников ввода, таких как касания сенсорного дисплея, мыши, трекбола и т.п. Поддерживаются другие устройства ввода, такие, как акселерометры, модули gps, значительно увеличивающие удобство работы с ГИС программного компонента трассоискателя.

Таким образом, в настоящее время Kivy активно развивается и имеет большое количество пользователей и разработчиков по целому ряду причин:

- ✓ Kivy имеет удобную в использовании встроенную поддержку мультитач-устройств с сенсорным экраном;
- ✓ Kivy специально создан для разработки кроссплатформенных графических приложений на языке Python и имеет более лучшую поддержку этого языка, по сравнению с адаптированными для него Qt и GTK +3;
- ✓ Kivy предоставляет более удобные API по сравнению с ранними инструментами, такими как HTML и CSS) для построения графического интерфейса. Язык разметки интерфейса Kivy представляет собой адаптированный для использования совместно с Python вариант языка HTLM, отличающийся лаконичным, и поэтому более удобным синтаксисом.
- ✓ Kivy позволяет разрабатывать и поддерживать приложение без изменения исходного кода для множества операционных систем.

Последний пункт реализуется благодаря python-for-android – инструменту для сборки с открытым исходным кодом, позволяющим упаковывать код на языке Python в отдельные APK (Android Package – формат архивных исполняемых файлов-приложений) для Android. Их можно передавать, устанавливать или загружать в магазин приложений Play Store, как и в любое другое приложение для Android. Этот инструмент изначально был разработан для кроссплатформенного графического тулкита Kivy, но в настоящее время поддерживает множество различных опций упаковки и может быть применен для создания любых приложений, написанных на языке Python.

Тулкит python-for-android реализует основные методы портирования приложения на операционную систему Android. В частности, он может скомпилировать интерпретатор Python, его зависимости для конкретной версии операционной системы, используемые в приложении библиотеки, и код приложения Python для устройств с операционной системой Android. Этот этап является полностью настраиваемым, при этом самыми важным параметром приходится список зависимостей, т. е. модулей, которые требуются приложению для работы. В случае программного компонента трассоискателя это модули для работы с ГИС (геоинформационной системой), вычислительными методами обработки и визуализации данных,

подробнее о которых говорится в работе [69]. Аналогичный тулкит существует и для операционной системы iOS – Kivy iOS.

Для удобного управления всеми параметрами компиляции и сборки приложения под конкретную платформу используется такое инструментальное средство, как Buildozer. Это инструмент, позволяющий быстро упаковать мобильное приложение. Он автоматизирует весь процесс сборки, загружает необходимые компоненты, такие, как python-for-android, Android SDK, NDK, и т.д. В настоящее время Buildozer поддерживает упаковку для таких операционных систем, как:

- ✓ Android: с помощью python-for-android (необходим компьютер с Linux или OSX, чтобы компилировать приложения для операционной системы Android);
- ✓ iOS: с помощью Kivy iOS (для этого необходим компьютер с операционной системой OSX).

Поддержка других платформ включена в план (например, .exe для Windows, dmg для OSX и т.д.) и будет реализована в ближайшее время. Процесс упаковки с помощью Buildozer управляется правилами, описанными в файле с именем buildozer.spec в каталоге приложения, определяя необходимые для его работы условия и такие параметры, как заголовок, значок, используемые ресурсы (например, изображения, электронные подписи и др.) и библиотеки. Один и тот же файл спецификации используется для создания пакета для Android, iOS и т. д. Таким образом реализуется возможность работы приложения на различных операционных системах без изменения исходного кода. В этом файле необходимо указать необходимые версии SDK, NDK и другие зависимости приложения от среды, в которой оно будет работать. В частности, для портируемого программного компонента необходимо в качестве зависимостей указать модуль pyler, из которого импортируется набор функций, необходимых для работы со встроенным во многие устройства gps-датчиком, который в свою очередь необходим для работы модуля ГИС mapview – виджета Kivy для отображения интерактивных карт. Кроме того, импортируется и модуль graph, используемый в программном компоненте трассоискателя для отображения интерактивных графиков.

Другой важной частью, зависимой от аппаратного обеспечения, является драйвер платы сбора данных Л КАРД E502. В целом он предоставляет собой три библиотеки:

- ✓ x502api – содержит общие функции для обоих модулей. Должна включаться в любой проект, работающий с одним из модулей, за исключением проектов, написанных только для L502 до появления библиотеки x502api, которые могут использовать только l502api;

- ✓ l502api – содержит специфические функции для модуля L502, а также функции, оставленные для совместимости с проектами, написанными до появления x502api;
- ✓ e502api – содержит специфические функции для модуля E502 [9].

Для портирования программного компонента необходимо адаптировать две библиотеки – x502api и e502api (т. к. в нем в настоящее время используется только модуль E502). Для ОС Windows предоставляется общий установщик «L-Card L502/E502 SDK», автоматически устанавливающий все необходимые драйвера, динамические библиотеки в системную директорию, а также все файлы, необходимые для подключения библиотеки к проекту приложения и примеры работы с ним в указанную директорию. Установка для ОС Linux так же не предоставляет особых усилий. Можно, например воспользоваться готовыми собранными пакетами, предоставляемыми «ЛКард». Это рекомендованный способ для дистрибутивов, для которых предоставляются собранные пакеты.

Тем не менее предоставляемые производителем исходные коды драйвера не являются полностью кроссплатформенными, так как ООО «Л Кард» не предоставляет поддержку операционных систем Android и iOS. Однако анализ исходных кодов показывает, что для них используется система сборки CMake. Эта система является свободным инструментом с открытым исходным кодом, основным разработчиком которого выступает компания Kitware. Название системы расшифровывается как cross-platform make (кроссплатформенная система сборки). Разработка инструмента ведётся с 1999 г., в качестве прототипа была использована утилита pcmaker, написанная в 1997 г. одним из авторов CMake. В настоящее время инструмент внедряется в процесс разработки многих программных продуктов, в качестве примеров широко известных проектов с открытым кодом можно привести KDE , MySQL , Blender , LLVM + clang и многие другие [11]. Принцип работы инструмента CMake напоминает принцип работы Buildozer: из каталога исходных кодов считывается файл CMakeLists.txt с описанием проекта, на выходе инструмент генерирует файлы проекта для одной из множества целевых операционных систем. Требования для сборки самого CMake включают наличие утилиты make и интерпретатора сценариев на языке bash либо скомпилированного инструмента CMake одной из предыдущих версий, а также компилятора C++ . При этом в исходных кодах CMake преднамеренно используются только возможности языка и стандартной библиотеки, поддерживаемые достаточно старыми версиями компиляторов. Таким образом, CMake является кроссплатформенным инструментом, который переносим на большое количество платформ, что и позволяет модифицировать и скомпилировать драйвер устройства сбора данных Л КАРД E502 для мобильных операционных систем.

Для создания провайдера устройства ввода данных с платы Л КАРД Е 502 для библиотеки Kivy требуется SWIG – инструмент разработки программного обеспечения, который связывает программы, написанные на C и C ++, с различными языками программирования высокого уровня. SWIG используется с различными типами целевых языков, включая распространенные языки сценариев, такие как Javascript, Perl, PHP, Python, Tcl и Ruby. Список поддерживаемых языков также включает языки, не относящиеся к сценариям, такие как C#, D, Go, Java, включая Android, Lua, OCaml, Octave, Scilab и R. Также поддерживаются несколько интерпретируемых и скомпилированных реализаций Scheme (Guile, MzScheme / Racket).

И, наконец для того, чтобы обеспечить повторяемую сборку программного компонента «Перспектива», результаты которой бы не зависели от используемой операционной системы и программного окружения, необходимо использовать инструментальное средство Docker. Контейнеры Docker предоставляют простые быстрые и надёжные методы разработки, распространения и запуска программного обеспечения, особенно в динамических и распределённых средах [70]. Docker позволяет создать контейнер, содержащий новейшие версии операционной системы и всех необходимых инструментов: Python, Kivy, Buildozer, CMake и др.

Таким образом, для портирования программного компонента необходимо решить целый ряд задач: разработать общую архитектуру, объединяющую графический интерфейс пользователя, драйвер платы сбора данных, тулкит Kivy и провайдер платы сбора данных в единую систему, отделив, однако, при этом платформозависимые и платформонезависимые части. Для реализации этой архитектуры потребуется доработать драйвер платы сбора данных и процесс его сборки так, чтобы его можно было использовать на мобильных операционных системах, разработать архитектуру провайдера платы сбора данных и реализовать ее, обеспечив при этом повторяемую сборку платформозависимой части программного компонента трассоискателя для различных операционных систем.

Кроссплатформенная архитектура программного компонента трассоискателя.

Как уже было сказано выше, графический интерфейс пользователя, драйвер платы сбора данных и некоторые другие части программного компонента тесно связаны с оборудованием и в связи с этим должны быть соответствующим образом адаптированы для обеспечения возможности его работы на операционных системах Android, iOS или Windows 10 Mobile. Схематически кроссплатформенная архитектура программного компонента, способного работать на мобильных операционных системах, выглядит так, как показано на рис. 4.11. В предложенной архитектуре основной код программного компонента трассоискателя отделен от кода, зависящего от целевой рабочей платформы. Для портирования

программного компонента требуется лишь собрать для конкретной операционной системы и типа процессора показанную на рис. 2 платформозависимую часть. Тогда основной код программного компонента на языке Python можно будет запускать на любой платформе без изменений.

Сетевой стек TCP/IP, необходимый для связи с устройством сбора данных с помощью Ethernet или Wi-fi, в настоящее имеется в любой современной мобильной и настольной операционной системе, однако не все функции поддерживаются одинаково хорошо. Для того чтобы обеспечить работу на различных платформах в драйвере Л КАРД E502 необходимо отключить поддержку автоматического поиска устройств в локальной сети, недоступную в настоящее время в операционной системе Android, изменив в файле *CMakeLists.txt* библиотеки e502api опцию

```
option(E502API_ENABLE_DNSSD "enable dns-sd service discovery" OFF),
```

а в настройках программного компонента «Перспектива» в случае подключения устройства сбора данных по сети необходимо указать IP-адрес используемого устройства.

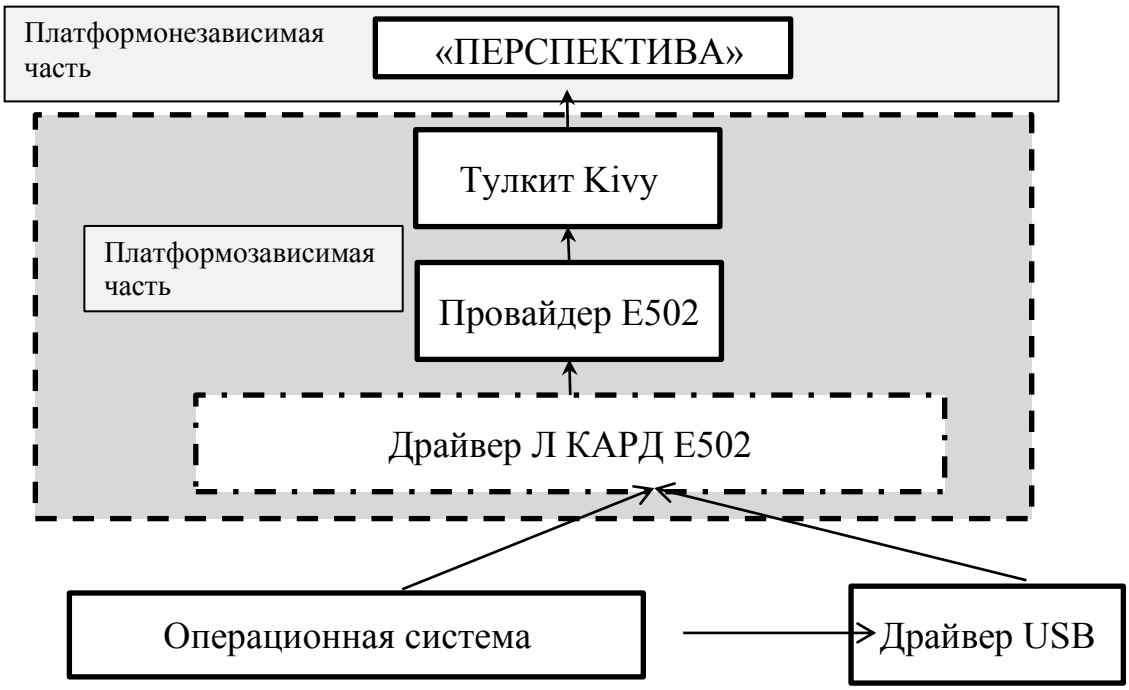


Рисунок 4.11 – Кроссплатформенная архитектура программного компонента трассоискателя на базе устройства сбора данных Л КАРД E 502

Кроссплатформенный драйвер USB можно найти по адресу [71] и добавить код:

```
if(android)
        include_directories(libusb-1.0 libusb-1.0/android)
…
if(linux)
        include_directories(libusb-1.0)
…
if(WIN32)
```

```
include_directories(libusb-1.0 libusb-1.0/msvc)
```
…

в файл *CMakeLists.txt* библиотеки e502api. В настоящее время кроссплатформенным драйвером USB поддерживаются такие операционные системы, как Linux, macOS, Windows, OpenBSD/NetBSD и Haiku.

В главный CMake-файл драйвера платы сбора данных для поддержки Android устройств следует добавить код:

```
if(${CMAKE_SYSTEM_NAME} MATCHES Android)
    set(CMAKE_SYSTEM_VERSION 21) # уровень API
    set(CMAKE_ANDROID_ARCH_ABI armeabi)
    set(CMAKE_ANDROID_STL_TYPE gnustl_static)
endif(),
```

обеспечивающий корректную сборку для устройств на базе операционной системы Android. Как видно из приведенного выше кода, гарантируется работа на устройствах с API 21 и выше, что соответствует версии Android не менее чем 5.0. Работа на более старых устройствах, к сожалению, невозможна, т.к. для них отсутствует поддержка стандарта POSIX Threads, необходимого для работы драйвера Л КАРД E502, реализующего поддержку многопоточных программ, преимущественно ориентированных на исполнение на системах с общей памятью (Symmetric Multiprocessing, сокращённо SMP). Как известно, это такие системы, где установлено несколько процессоров или\и многоядерные процессоры, и каждое ядро имеет доступ ко всей оперативной памяти компьютера.

Кроме того, в файл /lib/osspec/osspec.c [72] необходимо внести исправление для компиляции драйвера платы сбора данных для операционной системы Android:

```
if (0) {/*timeout != OSSPEC_TIMEOUT_INFINITY*/
  struct timespec timeToWait;
  f_get_abs_time(timeout, &timeToWait);
  /*wt_res = pthread_timedjoin_np(thread, NULL,
&timeToWait);*/
} else {
        wt_res = pthread_join(thread, NULL);
```

заменив тем самым неподдерживаемую в Android функцию стандарта POSIX Threads pthread_timedjoin_np на поддерживаемую во всех операционных системах функцию pthread_join.

На рис. 4.12 показана архитектура разработанного провайдера устройства ЛКард E502. Для начала работы с модулем необходимо установить с ним связь с помощью функции `open_usb`. Для идентификации устройств используется их серийные номера. Получить список серийных номеров всех подключенных устройств E502 можно с помощью `get_usb_serial_list`. Данная функция возвращает список, в котором сохранены найденные серийные номера, а принимает максимальное количество присоединенных модулей (по умолчанию это значение равно 16). Следует отметить, что с одним модулем одновременно

может быть установлено только одно соединение. При попытке открыть модуль, с которым уже установлено соединение через другой описатель `_hnd` (возможно, в другой программе) `open_usb`, вернет ошибку. При этом `get_usb_serial_list` по умолчанию возвращает список всех серийных номеров устройств, включая те, с которыми уже установлено соединение. Если нужно получить список только тех устройств, с которыми еще не установлено соединение, то в `get_usb_serial_list` можно передать аргумент *only_not_opened=True*.

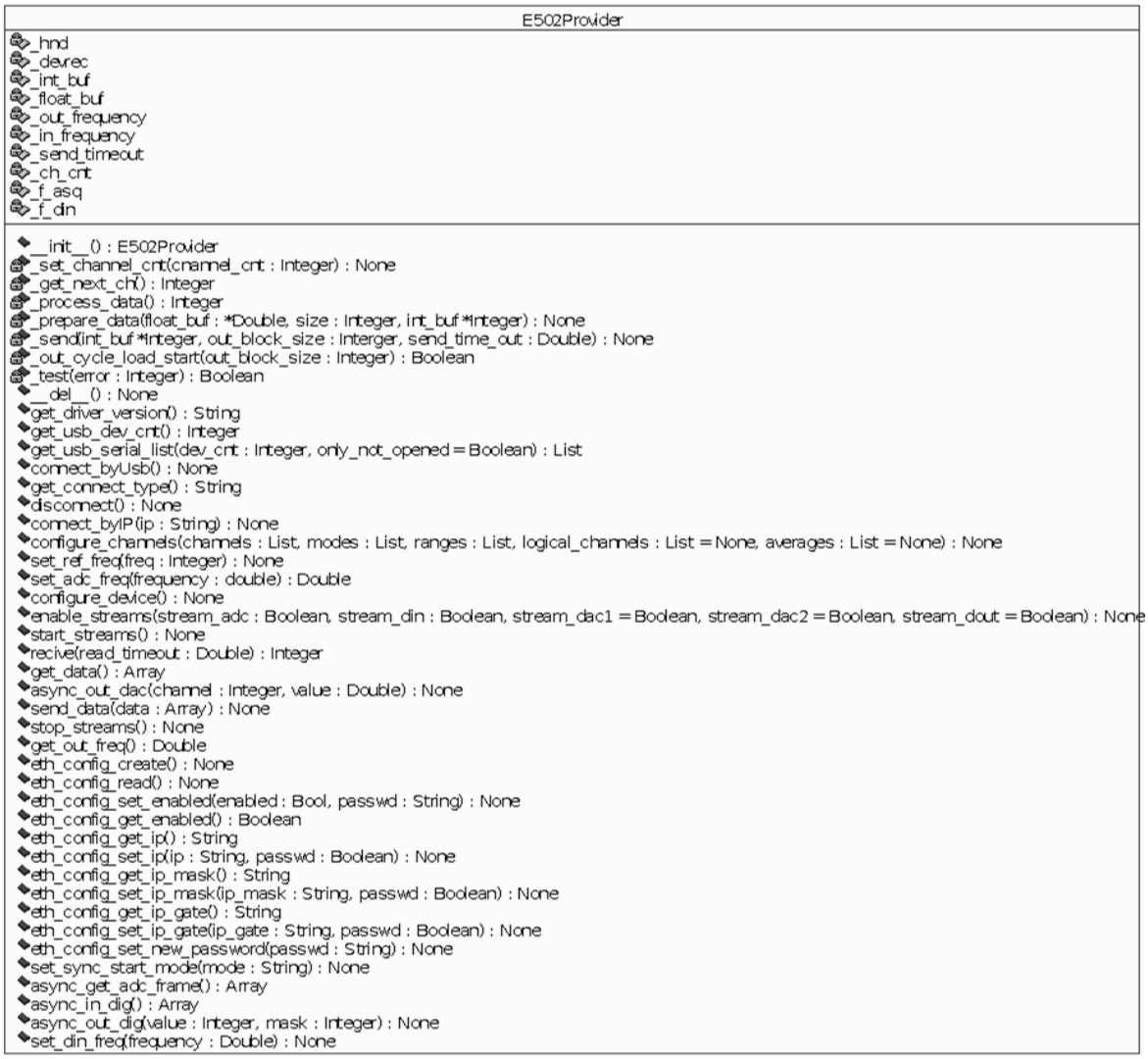


Рисунок 4.12 – Архитектура провайдера устройства ЛКард E502

Установить связь с модулем по Ethernet можно только по явно заданному IP-адресу устройства. Следует отметить, что в отличие от интерфейса USB для корректной работы по Ethernet должны быть настроены необходимые параметры. Если USB-интерфейс в модуле E502 всегда работает и не требует дополнительной конфигурации, то для работы по интерфейсу Ethernet необходимо выполнить настройку параметров интерфейса и разрешить данный интерфейс. Следует отметить, что при разрешенном Ethernet-интерфейсе с модулем можно работать как по USB, так и по

Ethernet. При этом сбор/генерация данных могут выполняться одновременно только по одному интерфейсу (по которому пришла команда на запуск сбора/выдачи данных). Основными параметрами для работы по Ethernet являются:

- ✓ IP-адрес устройства. Записывается как 4 цифры от 0 до 255 (модуль поддерживает только протокол IPv4), разделенные точками (например, 192.168.0.10). Состоит из адреса подсети и адреса устройства внутри подсети. Последний должен быть уникальным внутри подсети.
- ✓ Маска подсети. Определяет, какая часть адреса относится к адресу подсети, а какая к адресу устройства. Маска 255.255.255.0 означает, что первые 3 цифры (192.168.0) обозначают адрес подсети, последняя цифра (10) – адрес устройства в подсети.
- ✓ IP-адрес шлюза. Используется, только когда модуль E502 и хост, с которого выполняется управление модулем, находятся в разных подсетях. Модуль передает пакеты по адресу шлюза, если адрес назначения находится не в той же подсети, что и модуль. При работе в локальной сети не используется, но необходим для передачи данных по сети Интернет.
- ✓ MAC-адрес модуля (6 цифр от 0 до 255, которые записываются в 16-ричном формате). Физический адрес устройства, который должен быть уникален внутри локальной сети. В «Л Кард» для каждого модуля прописывается свой заводской MAC-адрес, который нельзя изменить. Однако при необходимости пользователь может задать свой пользовательский MAC-адрес и разрешить его использование вместо заводского. При этом всегда есть возможность вернуться к заводскому MAC-адресу, запретив пользовательский.

Для работы в первую очередь необходимо задать правильные IP-параметры (адрес, маску и при необходимости адрес шлюза). IP-параметры модуля E502 могут быть установлены 3 способами:

- ✓ Заданы вручную (статические параметры). Пользователь сам должен позаботиться о том, чтобы адрес принадлежал нужной подсети и был уникальный в ней.
- ✓ Получены автоматически от DHCP-сервера. Если задано автоматическое получение адреса и в локальной сети присутствует DHCP-сервер, то модуль делает запрос к нему и использует выделенные DHCP-сервером IP-адреса.
- ✓ Может использоваться автоматически получаемый локальный (link-local) адрес (в соответствии с RFC3927[17]). Адрес выбирается случайным образом в диапазоне от 169.254.1.0 до 169.254.254.255 и проверяется, что в сети нет другого устройства с таким адресом (если есть, то идет попытка выбора следующего адреса и т.д.). Это делает возможным подключение к устройству в

локальной сети без специальной конфигурации. Следует отметить, что так как адрес действителен только в одной сети, то два разных устройства в разных сетях могут иметь одинаковый link-local адрес, что приводит к тому, что если у ПК несколько активных интерфейсов (и на обоих используется link-local адрес или ,наоборот, обычный адрес), то хост не знает, на каком интерфейсе искать нужное устройство. Т.е. для подключения по link-local адресу на ПК должен быть либо один активный сетевой интерфейс, либо на нужном интерфейсе должен использоваться link-local адрес, а на другом статический или полученный по DHCP адрес.

При включении автоматического получения адреса модуль выбирает себе link-local адрес (и проверяет его уникальность), параллельно выполняя поиск в сети DHCP-сервера. Если DHCP сервер не обнаружен, то используется link-local адрес. Как только будет обнаружен DHCP-сервер, то предпочтение отдается полученному от него адресу (в частности, при старте в сети с DHCP сервером модуль может некоторое время до получения адреса от него использовать link-local адрес). Подобный алгоритм используется при автоматическом получении адреса, в частности в ОС Windows, а также и во многих дистрибутивах Linux (иногда предоставляя возможность отдельного разрешения DHCP и link-local адреса). Следует также иметь в виду, что автоматически получаемый адрес модуль проверяет на уникальность в сети, поэтому существует задержка в несколько секунд от подключения к сети модуля до назначения адреса. Автоматическое получение адреса не требует дополнительных настроек, однако при этом неизвестен адрес устройства со стороны ПК для установления соединения. Для решения этой проблемы возможно использование функций получения настроек интернет-соединения по интерфейсу USB.

Изменить сетевые настройки можно с помощью программы L-Card Measurement Studio[72]. Также изменение сетевых настроек модуля возможно через функции провайдера устройства ЛКард E502. Для этого существует отдельное поле описателя конфигурации `_eth_config_hnd`. Для изменения конфигурации нужно выполнить следующие шаги:

1. Создать описатель конфигурации с помощью `eth_config_create`.
2. Прочитать текущую конфигурацию устройства с помощью `eth_config_read`. (соединение с модулем должно быть установлено).
3. Можно получить нужные параметры с помощью функций `eth_config_get` и/или установить новые значения с помощью функций `eth_config_set`.

После завершения изменений можно записать измененную конфигурацию в модуль с помощью `eth_config_write`. Модуль сохранит новую конфигурацию в энергонезависимой памяти, запрещает Ethernet-интерфейс, после чего снова его переинициализирует уже с

новыми параметрами. При этом если соединение с устройством было выполнено по Ethernet, то для дальнейшей работы нужно разорвать соединение и установить заново (используя новые параметры) Для избежания непреднамеренного изменения конфигурации по сети конфигурация может быть защищена паролем. Пока пароль не установлен, в качестве пароля нужно передавать пустую строку. Установка нового пароля выполняется аналогично любым другим изменениям параметров конфигурации (с помощью `eth_config_set_net_password`). В случае если пароль забыт, то можно установить соединение по USB и изменить конфигурацию (включая пароль), введя в качестве текущего пароля серийный номер модуля.

Перед использованием модуля, как правило, необходимо выполнить настройку его параметров. Сперва все настройки записываются в поля структуры описателя модуля с помощью функций, начинающихся с `set_`, которые будут описаны ниже, после чего установленные параметры передаются в модуль с помощью функции `configure_device`.

Плата E502 представляет собой АЦП с последовательной коммутацией каналов. То есть измерение нескольких каналов происходит последовательно, путем переключения входного коммутатора АЦП, последовательность опроса каналов задается с помощью управляющей таблицы логических каналов АЦП. Всего таблица может содержать от одного до 256 логических каналов. Каждый логический канал задает следующие параметры:

- ✓ номер физического канала, с которого производится измерение. Номер физического канала задается, считая от 0, то есть 0 означает первый канал, 1 – второй и т.д. Таким образом в дифференциальном режиме номер канала может быть от 0 до 15, а в режиме измерения с общей землей – от 0 до 31.
- ✓ режим измерения АЦП – измерение напряжения относительно общей земли (comm), диференциальное измерение напряжения *(diff*), измерение собственного нуля *(zero*);
- ✓ используемый диапазон измерения в вольтах +/- 10.0, 5.0, 2.0, 1.0, 0.5, 0.2;
- ✓ коэффициент усреднения по заданному логическому каналу;
- ✓ задать параметры логических каналов с нужными номерами можно с помощью функции `configure_channels`, при этом будет задано количество логических каналов в управляющей таблице с помощью приватной функции `_set_channel_cnt`.

Реально микросхема АЦП всегда работает на частоте, равной опорной частоте синхронизации. В случае если частота сбора АЦП установлена меньше, чем опорная частота синхронизации, то на одно измерение значения логического канала приходится *n* измерений АЦП ($n = f_{acq}/f_{ref}$ – отношение установленной частоты сбора АЦП к опорной частоте дискретизации). При отключенном усреднении первые *n-1* измерений

отбрасывается, что увеличивает время установления сигнала. При необходимости можно использовать несколько последних отсчетов (n_{avg}) для получения результирующего значения. Тогда результирующего значения будет являться средним между n_{avg} последними измерениями, однако это сокращает соответственно время на установление сигнала. Естественно n_{avg} всегда меньше либо равно n. Кроме того, n_{avg} не может превышать максимального значения, равного 128.

Все частоты потокового сбора и выдачи данных основываются на опорной частоте синхронизации. В качестве опорной частоты может использоваться внутренний источник частоты или внешний. В первом случае опорная частота может быть 2МГц либо 1,5МГц. По умолчанию используется 2МГц. Изменить ее можно с помощью функции set_ref_freq.

При внешней опорной частоте может использоваться сигнал с произвольной частотой до 1,5 МГц. При этом, для того, чтобы функции библиотеки могли оптимально подобрать настройки передачи данных (если они не задаются вручную), а также для того, чтобы корректно работали функции, подбирающие делители (см. ниже), для получения нужных частот ввода/вывода необходимо задать значение внешней опорной частоты, которая будет подана. Это значение можно задать с помощью функции `set_ext_ref_freq_value`.

Частота сбора АЦП получается с помощью деления значения опорной частоты на установленный коэффициент, который может быть в диапазоне от 1 до 2^{20}. Кроме того, между измерением последнего логического канала одного кадра и началом следующего кадра может быть добавлена межкадровая задержка. Межкадровая задержка задается в виде количества периодов опорной частоты синхронизации. Функцией `set_adc_freq` можно передать значения частоты сбора АЦП и частоты кадров в Герцах, а функция сама рассчитает нужный делитель и значение межкадровой задержки, чтобы полученные частоты были наиболее близки к указанным. При этом функция вернет реально установившиеся значения частот.

Под частотой сбора АЦП (f_{acq}) понимается величина, обратная времени одного преобразования, соответствующего одному логическому каналу. Под частотой кадров (f_{frame}) понимается величина, обратная времени от начала измерения первого логического канала одного кадра до начала измерения первого логического канала следующего кадра. Эта частота соответствует частоте сбора для одного логического канала.

На рис. 4.13 приведена диаграмма, иллюстрирующая на примере сбора при заданных трех логических каналах, как определяются упомянутые выше частоты.

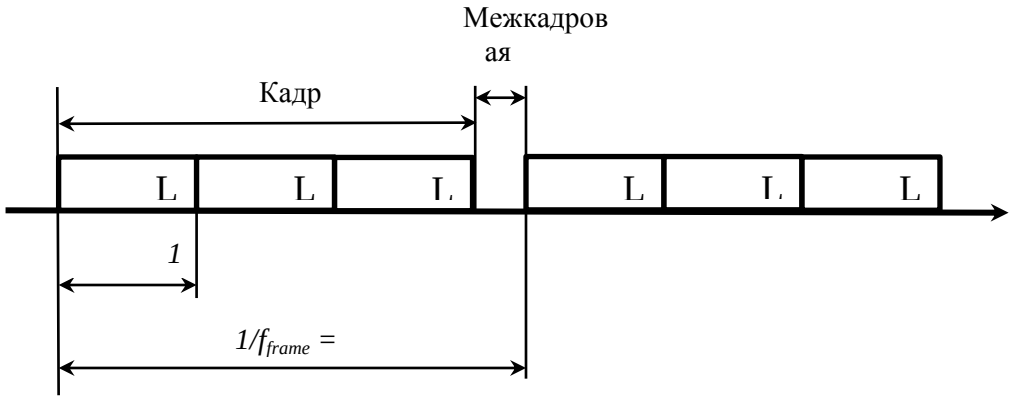


Рисунок 4.13 – Диаграмма сбора данных АЦП для трех логических каналов

Если межкадровая задержка не нужна, то можно передать нулевой указатель None в качестве второго параметра `set_adc_freq`, тогда будет использоваться всегда нулевая межкадровая задержка (т.е. измерение следующего кадра начнется сразу после завершения предыдущего).

Помимо синхронного ввода с АЦП, модули L502 и E502 позволяют осуществлять синхронный ввод с цифровых входов. Так же, как и для синхронного сбора данных АЦП, частота синхронного цифрового ввода определяется как опорная частота, деленная на коэффициент, который можно установить с помощью функцию `set_din_freq`, чтобы она рассчитала этот коэффициент для получения наиболее близкой частоты к указанной. Для синхронного ввода цифровых линий нет ни логической таблицы (так как каждый раз считывается значение всех цифровых входов и передается в виде одного слова), ни межкадровой – при запуске синхронного ввода все измерения выполняются через одинаковые промежутки времени. При этом частота для ввода с цифровых линий может отличаться от частоты сбора АЦП.

Также плата E502 позволяет осуществить синхронный вывод параллельно на один или два канала ЦАП и цифровые выводы. При этом максимальная частота вывода каждого канала в два раза меньше значения опорной частоты. Необходимо вызвать функцию `set_out_freq`, чтобы она подобрала делитель так, чтобы частота была наиболее близка к заданной. При этом частота задается общая для всех потоков вывода.

По умолчанию в качестве опорной частоты синхронизации используется внутренняя частота модуля, а запуск всех синхронных измерений осуществляется при выполнении функции `streams_start`. Однако при необходимости возможно задать как внешний источник опорной частоты, так и внешний сигнал запуска синхронного сбора/выдачи данных. Для этого могут быть использованы входы цифрового разъема (может использоваться как фронт, так и спад одного из этих сигналов), либо также может использоваться разъем синхронизации для организации синхронного сбора данных по принципу ведущий-ведомые(рис. 4.14) [16].

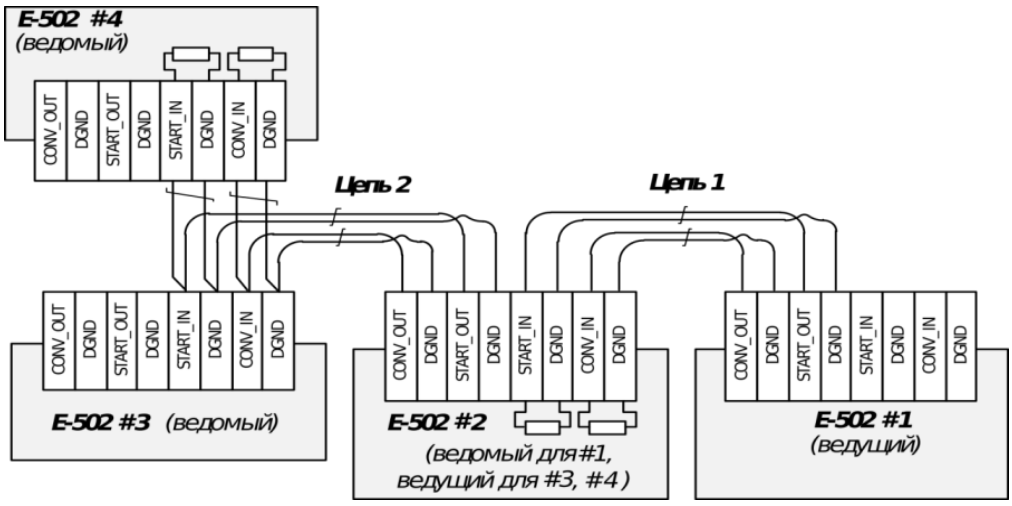


Рисунок 4.14 – Объединение нескольких устройств для организации синхронного сбора данных

Выбор внешнего сигнала для задания опорной частоты синхронизации задается с помощью `set_sync_mode`, а условие запуска с помощью функции `set_sync_start_mode`. Следует отметить, что если задано внешнее событие запуска, то для того, чтобы модуль перешел в режим ожидания этого события, необходимо вызвать `streams_start`. Останов синхронного сбора/выдачи данных всегда осуществляется программно с помощью `streams_stop`. При использовании разъема синхронизации для организации сбора данных по принципу ведущий-ведомые для ведущего модуля источником опорной частоты синхронизации остается внутренняя частота (режим *internal*), а каждый ведомый модуль использует опорную частоту и/или признак запуска сбора от внешнего мастера, т.е. для каждого ведомого модуля должен быть установлен режим *external_master*.

При включении питания все каналы находятся в асинхронном режиме. В асинхронном режиме при вызове функции асинхронного ввода/вывода производится однократный ввод или вывод указанной информации. При этом задержка от вызова функции до непосредственно момента измерения данных для ввода или выставления указанного значения на выходе для вывода точно не определена. Также точно не может быть определена задержка между двумя последовательными операциями ввода/вывода.

Плюсом асинхронного режима является простота его использования – достаточно одного вызова требуемой функции:

- `async_in_dig` – асинхронный ввод значений цифровых линий,
- `async_out_dig` – асинхронный вывод значений на цифровые линии,
- `async_out_dac` – асинхронный вывод значения на один из каналов ЦАП.

Для однократного ввода данных с АЦП используется функция `async_get_adc_frame`, которая выполняет ввод одного кадра данных АЦП. В отличие от других функций асинхронного ввода-вывода перед вызовом данной функции необходимо выполнить настройку модуля: необходимо задать управляющую таблицу АЦП. Измерение логических каналов внутри кадра происходит синхронно с заданной частотой сбора АЦП. Асинхронным является ввод самих кадров, то есть задержка между измерением кадров при последовательном вызове `async_get_adc_frame` не определена.

В синхронном режиме ввод или вывод данных осуществляется с заданной частотой, то есть время между соседними измерениями или выводом соседний отсчетов определено. Частота сбора для каждого канала задается относительно общей опорной частоты синхронизации, и запуск синхронного ввода-вывода для всех каналов осуществляется одновременно. Для запуска синхронного режима необходимо сперва с помощью функции `streams_enable` разрешить синхронный режим по требуемым каналам, а затем запустить синхронный ввод/вывод по всем разрешенным каналам с помощью `streams_start`. При синхронном вводе модуль производит измерения с заданной частотой и сам передает данные по интерфейсу в буфер. Принятые в буфер данные могут быть прочитаны программой с помощью `get_data`. Аналогично для синхронного вывода, модуль сам по мере необходимости считывает данные из буфера и выводит считанные отсчеты с заданной частотой. Данные в буфер драйвера должны быть предварительно записаны с помощью `send_data`. При этом если к моменту вывода очередного отсчета данные в буфер драйвера не поступили, то будет выведено предыдущее значение.

В провайдере выделяется всего один буфер на прием и один на передачу данных. То есть значения для синхронного ввода с цифровых линий и отсчеты АЦП передаются одним потоком, также одним потоком передаются все данные на вывод. Каждый отсчет представлен в виде 32-битного слова, содержащего дополнительную информацию, включающую в себя признак, к какому типу данных относится данный отсчет. Далее провайдер осуществляет разбор принятых данных на отсчеты АЦП и значения цифровых выводов и перевод отсчетов АЦП в Вольты. Данные от АЦП приходят в том порядке, в котором производятся измерения, т.е. сперва измерения, соответствующие всем логическим каналам первого кадра, затем второго и т.д. При этом можно получить и нецелое количество кадров (например, если запущен синхронный ввод с цифровых линий, то заранее сложно определить, сколько в принятом блоке данных содержится отсчетов с цифровых линий, а сколько отсчетов с АЦП), в этом случае выдает все отсчеты, включая отсчеты нецелого кадра, дополняя отсутствующие значения нецелого кадра неопределенными значениями *None*, так чтобы в итоге было получено целое число кадров. Это

необходимо для того, чтобы заранее можно было вычислить размер памяти, необходимый для хранения принятых данных, при дальнейшей обработке неопределенные значения могут быть отброшены.

Аналогично для формирования общего потока на вывод в требуемом формате используется функция `send_data,` принимающая данные из трех массивов и сохраняющая их во внешний массив. Если какой-либо из источников не должен использоваться, то в качестве массива передается нулевой указатель *None*. Для каналов, которые не были настроены на синхронный режим с помощью `streams_enable`, входной массив не анализируется и данные из него не используются.

Следует отметить, что если для синхронного ввода инициализация потока передачи происходит по `streams_start`, так как данные начнут поступать только после запуска синхронного ввода, то с синхронным выводом дело обстоит несколько иначе. Так как по `streams_start` уже должна начаться выдача синхронных данных, то часть данных уже должна быть загружена в модуль. Таким образом, после разрешения синхронного вывода по нужным каналам с помощью `streams_enable` и до запуска синхронного вывода с помощью `streams_start` необходимо осуществить загрузку данных синхронного потока. Для этого следует использовать функцию `send_data`, по которой в провайдере будет выделен буфер на передачу и инициализирован поток на передачу, если этого не сделать, то синхронный вывод начнется лишь после того, как данные будут записаны в модуль и не будет привязан к началу синхронного сбора/выдачи данных.

Кроме того, для синхронной выдачи данных на ЦАП необходимо предварительно установить начальные значения на ЦАП с помощью функции асинхронного вывода. В противном случае при начале синхронного вывода может быть небольшой переходный процесс от значения на ЦАП, которое было до запуска синхронного вывода, до выставления первых нужных значений, т.к. ЦАП имеет свой фильтр и ограничения на скорость изменения сигнала.

Далее была разработана программа на языке Python, автоматизирующая создание контейнера Docker и установку в него операционной системы Ubuntu 18.04 и все вышеперечисленные инструментальные средства сборки. В этом контейнере был собран драйвер Л КАРД E502 и провайдер платы сбора данных Л КАРД E502 для библиотеки Kivy, состоящий из динамически разделяемой библиотеки и модуля для интерпретатора языка Python, созданных с помощью SWIG. Для этого в файл CMakeLists.txt библиотеки платы сбора данных e502api был включен CMake-файл следующего содержания:

```
cmake_minimum_required(VERSION 2.6)
project(lcarde502)
set(LIB_MAJOR_VERSION "0")
set(LIB_MINOR_VERSION "0")
set(LIB_PATCH_VERSION "1")
```

```
    set(LIB_VERSION_STRING
"${LIB_MAJOR_VERSION}.${LIB_MINOR_VERSION}.${LIB_PATCH_VERSION}")
    INCLUDE_DIRECTORIES(${X502API_INCLUDE_DIR})
    #Генерация модуля для Python и связанной с ним динамически разделяемой
    #библиотеки
    find_package(SWIG 3.0)
    include(UseSWIG)
    find_package(PythonLibs REQUIRED)
    include_directories(${PYTHON_INCLUDE_DIRS}
${CMAKE_CURRENT_SOURCE_DIR}/src)
    swig_add_library(x502api        LANGUAGE        python        SOURCES
${CMAKE_CURRENT_SOURCE_DIR}/x502api.i)
    SWIG_LINK_LIBRARIES(x502api                              ${PYTHON_LIBRARY}
${CMAKE_BINARY_DIR}/libx502api.so)
    file(COPY ${CMAKE_CURRENT_SOURCE_DIR}/e502.py
    DESTINATION ${CMAKE_CURRENT_BINARY_DIR})
```

Для генерации кода, связывающего провайдер E502 с драйвером платы сбора данных, разработан интерфейсный файл x502api.i следующего содержания:

```
    /* File : x502api.i */
    %module x502api
    %include <stdint.i>
    %include <typemaps.i>
    %{
    #include "../src/x502api.h"
    #include "../devs/e502/e502api.h"
    %}
    #define __attribute__(x)
    %inline %{
        double * _malloc_double(int32_t size) {
            return (double *) malloc(size*sizeof(double));
        }
        uint32_t *_malloc_uint(int32_t size) {
            return (uint32_t *) malloc(size*sizeof(uint32_t));
        }
        void _free_double(double * buf) {
            free(buf);
        }
        void _free_uint(uint32_t* buf) {
            free(buf);
        }
    %}
    %apply uint32_t *INOUT { uint32_t *  }
    %apply double *INOUT {double *}
    %include "../src/x502api.h"
    %include "../devs/e502/e502api.h,
```

в котором импортируются библиотеки *stdint.i* целочисленных типов ISO C99, использующиеся в драйвере платы сбора данных, и *typemaps.i*, обеспечивающая прямой доступ к низкоуровневому генератору кода SWIG с помощью двух инструкций:

```
    %apply uint32_t *INOUT {uint32_t *}
    %apply double *INOUT {double *},
```

первая из которых служит для преобразования целочисленного беззнакового типа языка C99 в соответствующий Python объект типа *int*, а вторая для преобразования типа *double* в соответствующий ему тип *float*. Это необходимо для передачи различных параметров настойки в драйвер устройства Л Кард E502 и получения результатов об успешной установке этих параметров или кодов ошибок.

Функции `_malloc_double` и `_free_double` используются в провайдере для выделения и освобождения памяти для хранения исходных данных, предназначенных для отправки на устройство, или обработанных данных принятых от него, а функции `_malloc_uint` и `_free_uint` нужны для управления памятью буферов, необходимых для хранения данных, подготовленных для отправки на устройство, или «сырых» данных, принятых от него.

Наконец, был разработан файл buildozer.spec, требующийся для сборки APK-файла для операционной системы Android. В этом файле, кроме списка зависимостей и описания приложения, о которых уже говорилось выше, были указаны скомпилированные файлы провайдера E502; расширения файлов, содержащих необходимые ресурсы; минимальная версия Android API, необходимая для работы приложения; файлы драйвера Л КАРД E502 и архитектура процессора целевой платформы:

```
source.include_patterns = aarch64/python/*.so, aarch64/python/*.py
source.include_exts = py,kv,so
android.minapi = 21
android.add_libs_armeabi_v7a = aarch64/*.so, aarch64/devs/e502/*.so
android.arch = armeabi-v7a
garden_requirements = mapview, graph.
```

Получившийся в результате сборки с помощью Buildozer APK-файл для проверки был установлен на планшет Google Nexus 7 3G (2012) и запущен (рис. 4.15) на операционной системе Android 7.2 с использованием ГИС OpenStreetMap.

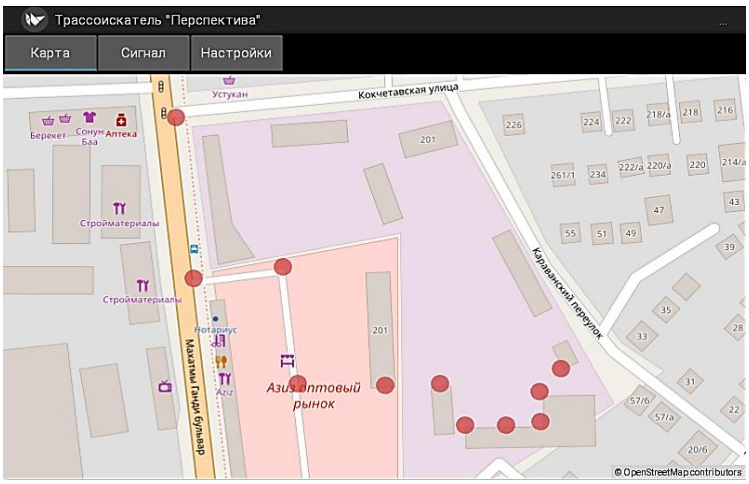


Рисунок 4.15 – Графический интерфейс кроссплатформенного программного компонента трассоискателя для работы с картой

Так же могут быть использованы и другие поставщики географических данных, например, Thunderforest или Google Maps. Портированный программный компонент трассоискателя позволяет наносить на карту найденные трассы кабелей на карту, выделяя их различными цветами, масштабировать карту до любых требуемых размеров и перемещать фокус ввода в любую необходимую для работы локацию.

Тестирование показало полную работоспособность портированного программного компонента трассоискателя и на смартфоне Meizu M2 Note с операционной системой Android 5.1. На рис. 4.16 показан графический интерфейс программного компонента трассоискателя для отображения принятых с помощью платы ЛКард E502 аналоговых и цифровых сигналов.

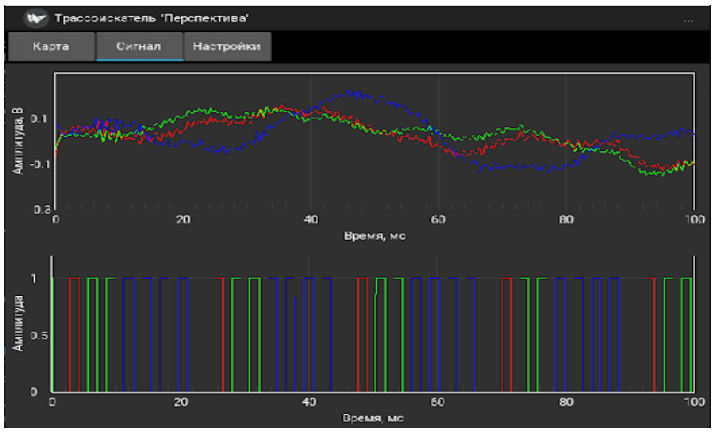


Рисунок 4.16 – Графический интерфейс кроссплатформенного программного компонента трассоискателя для отображения принятых сигналов

4.3 Разработка подсистем хранения и визуализации данных для системы мониторинга кабельных линий

Выше разработана архитектура мобильного программного компонента трассоискателя, позволяющего хранить и отображать информацию о трассе кабеля на плоскости, а также считывать данные с датчиков с помощью платы сбора данных Л-КАРД E502. Однако высокая частота дискретизации устройства сбора данных, работающего на частоте до 2 МГц и собирающего данные по нескольким каналам, требует программной реализации, способной быстро накапливать, обрабатывать и передавать данные для визуализации с минимальной задержкой, то есть возникает необходимость работать с большими наборами данных.

Проблему представляют сбор, очистка, хранение, поиск, доступ, передача, анализ и визуализация таких наборов, как целостной сущности, а не локальных фрагментов. В качестве определяющих характеристик для больших данных отмечают «три V»: объём (англ. volume, в смысле величины физического объёма), скорость (англ. velocity, означающее в данном контексте скорость прироста и необходимость высокоскоростной

обработки и получения результатов), многообразие (англ. variety, в смысле возможности одновременной обработки различных типов структурированных и частично структурированных данных) [74]. Ведущей характеристикой в данной задаче является скорость прироста данных, которая будет рассмотрена в аспекте хранения, обработки и визуализации данных на мобильных системах.

4.3.1 Особенности хранения и визуализации больших данных.

Для решения этой проблемы предлагается использовать оптимизированные библиотеки компании Л-КАРД для сбора данных, после чего информация об измерениях передается по сетевому UDP-протоколу в базу данных временных рядов (Time Series Database, TSDB) InfluxDB для временного хранения перед визуализацией.

На графике рис. 4.16 приведен пример набора исходных данных, полученных по методу, описанному в работе [75] с помощью трехкомпонентного индукционного датчика, который позволит определить точное положение неисправности кабеля в трехмерных координатах.

Полученные после оцифровки данные сохраняются в базе данных InfluxDB, являющейся не реляционной TSDB базой данных (Time series database – база данных временных рядов), оптимизированной для хранения временных последовательностей. TSDB, такие как InfluxDB, значительно превосходят классические реляционные СУБД по скорости работы с большими временными рядами (рис. 4.17). Например, InfluxDB поддерживает UDP-протокол для передачи информации по сети. Сетевой протокол UDP в отличие от протокола TCP, традиционно использующегося реляционными системами управления базами данных, позволяет передавать пакеты данных с минимальной задержкой, так как протокол не устанавливает двунаправленное соединение, не требует буферизации, а также передачу пакетов подтверждения успешной доставки. Протокол UDP делает акцент на минимизацию задержки ценой возможной потери пакетов. В настоящей задаче потеря нескольких замеров из миллионов, приходящих каждую секунду, не является серьезной проблемой.

Рисунок 4.17 – Скорость выполнения запросов InfluxDB и Microsoft SQL Server на одноядерных и многоядерных машинах в разработанной подсистеме хранения данных

Кроме оптимизаций сетевого взаимодействия, InfluxDB может эффективно использовать память системы. База данных может проводить группировку и обобщение данных на лету, может хранить только указанное количество последних нескольких записей, может автоматически сжимать временные последовательности, используя алгоритмы, специально адаптированные для многомерных временных рядов. Для выборки и обработки данных есть возможность использовать ограниченный SQL-подобный набор запросов.

Для проведения промежуточного анализа и построения запросов для базы InfluxDB может быть использована кроссплатформенная библиотека Grafana, имеющая визуальный конструктор для запросов, что позволяет проводить предварительную обработку данных без детального изучения языка запросов InfluxDB.

Для наглядного представления трассы кабеля и демонстрации процесса его построения была разработана графическая система рендеринга графики для мобильных устройств. Использование мобильных устройств, таких как планшетные компьютеры или смартфоны, позволяет проводить проверку и трассировку кабеля в полевых условиях.

Многие мобильные устройства содержат программные интерфейсы графического адаптера, недостаточно адаптированные к особенностям мобильных графических устройств, кардинально отличаются своей архитектурой от адаптеров персональных компьютеров. Мобильные графические процессоры используют архитектуру отложенного расчёта и обрабатывают изображение отдельными регионами фиксированного размера (т.н. тайлами) для минимизации потребления электроэнергии устройства.

Системы мобильной графики построены на основе архитектуры отложенного расчёта и обрабатывают изображение отдельными регионами фиксированного размера (тайлами) для минимизации потребления электроэнергии устройства. Архитектуре отложенного расчёта требуется, чтобы центральный процессор подготовил информацию обо всех запросах на рисование объектов сцены в буфере для проверки данных, распределения информации на тайлы и передачи, позже целиком в память графического ускорителя. Такой подход взаимодействия между процессором и графическим ускорителем позволяет, с одной стороны, упростить схему мобильного графического процессорного устройства, минимизировать его потребление памяти в системе и уменьшить потребление энергии устройства. С другой стороны, этот подход требует выполнять дополнительную работу по сбору данных и поддержке буферов для каждого последующего кадра. Это накладывает серьезные ограничения на максимальное количество объектов, которые могут быть представлены на экране одновременно из-за ограниченного размера таких буферов.

При этом большинство реализаций процессоров используют размер 16 на 16 и 32 на 32 пикселей для каждого тайла. Тайловое графическое процессорное устройство (ГПУ) имеет схемотехническое решение (рис. 4.18), использующее специально подготовленный буфер для хранения графических данных только в пределах одного тайла. Из-за небольшого размера тайла такое аппаратное решение может уместить все промежуточные данные и данные обработанной части изображения в регистровой памяти графического процессора. Это позволяет ускорить расчет, минимизировать энергопотребление, упростить схемотехническое решение и получить горизонтально-масштабируемую систему, проводя обработку нескольких тайлов одновременно несколькими ядрами графического процессора [76].

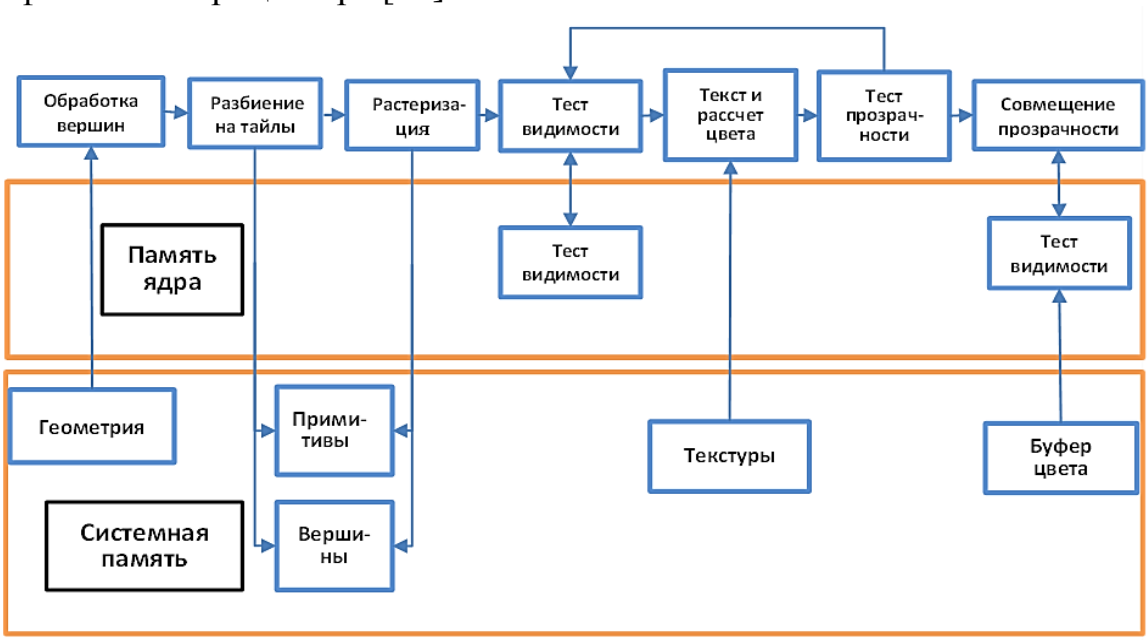


Рисунок 4.18 – Конвейер архитектуры тайлового ГПУ

Однако отложенные системы требуют от разработчиков тратить драгоценное время процессора на поддержку графической подсистемы, а не на выполнение полезной работы их программ. Кроме того, архитектуре отложенного расчёта требуется, чтобы центральный процессор подготовил информацию обо всех запросах на рисование объектов сцены в специальных буферах на передачу. Такой подход взаимодействия между центральным и графическим процессором позволяет упростить схемотехническое решение графического устройства, минимизировать потребление памяти в системе и уменьшить потребление энергии устройством. К сожалению, этот подход требует выполнять дополнительную работу по сбору данных и поддержке буферов для каждого последующего кадра. Это приводит к серьезным ограничениям на максимальное количество объектов, которые можно рисовать на экране одновременно из-за ограниченного размера программных буферов.

Несмотря на многие преимущества тайловых процессоров отложенного рисования, они имеют набор существенных недостатков. Одним из них является ограничение на количество объектов рисования по причине наличия фиксированного размера параметрического буфера. Другим ограничением являются высокие временные затраты на подготовку и проверку данных этого буфера, которую необходимо проводить на центральном процессоре перед каждым просчетом кадра. Это еще более усугубляет проблему рисования большого количества объектов на тайловых архитектурах. Независимо от сложности 3D-модели, будь это всего один треугольник, попытка нарисовать сцену в 1000 объектов, совершая 1000 отдельных запросов на рисование без потери производительности, практически невозможна в ГПУ такой архитектуры. Другая проблема отложенного рисования заключается в обработке прозрачных объектов. Тайловая архитектура создавалась для ускорения общего случая, когда больше половины объектов сцены непрозрачны. К сожалению, специализированный конвейер этой архитектуры (рис. 4.19) не приводит к ускорению, а иногда и замедляет процесс рисования объектов при наличии прозрачности.

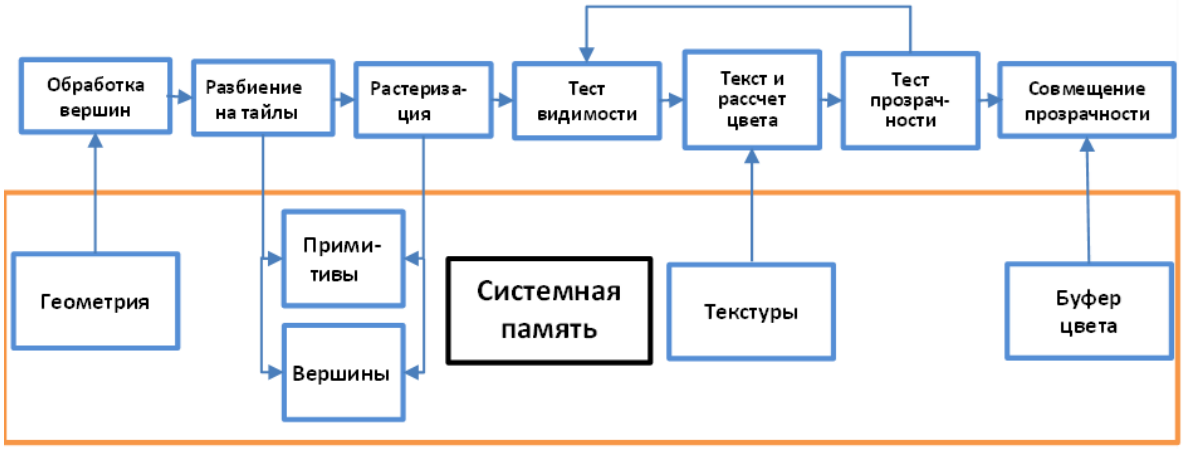


Рисунок 4.19 – Конвейер архитектуры ГПУ немедленного режима расчёта

Это вызвано тем, что по рекомендации дизайнеров системы для минимизации потери скорости обработки данных необходимо передавать запросы на рисование в определенном порядке. Сначала нужно нарисовать все непрозрачные объекты, а после рисовать прозрачные, в порядке от дальних к самым близким, исходя из позиции виртуальной камеры [77].

На базе тайловой архитектуры ключевые игроки рынка стали вводить модификации для последующего увеличения производительности. Британская компания Imagination Tech PowerVR, которая занималась производством ГПУ для мобильных продуктов компании Apple, использовала архитектуру отложенного рисования. Архитектура отложенного рисования (TBDR, Tile-Based-Deferred-Rendering, рис. 4.20) также позволяет проводить сбор всех данных объектов сцены в специальном параметрическом буфере. После сбора информации или при

переполнении буфера система командует ГПУ нарисовать все объекты за один проход.

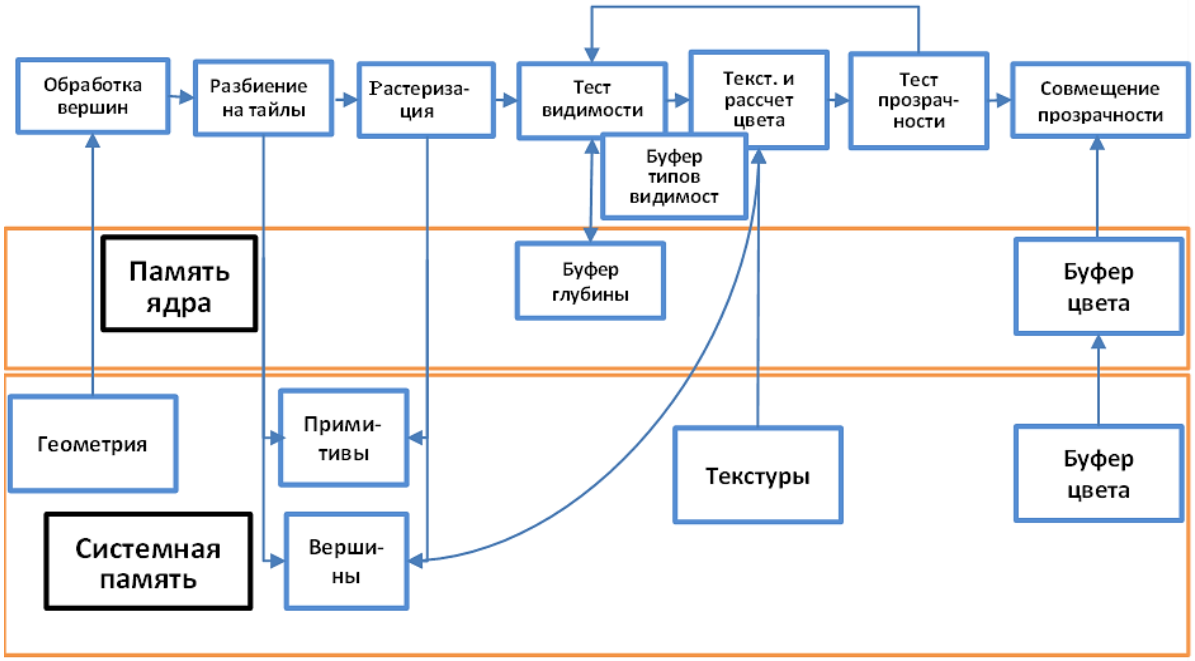


Рисунок 4.20 – Конвейер архитектуры тайлового ГПУ с отложенной системой рисования

Информация обо всех или большей части объектов сцены позволяет тайловому ГПУ произвести более эффективную сегментацию данных между тайлами, тем самым значительно ускорить работу. В частности, специализированная часть ядра может удалить все полигоны геометрии, которые были перекрыты полигонами других объектов сцены, стоящими ближе к виртуальной камере. Это позволит проводить дорогую операцию просчета цвета для непрозрачных объектов всего лишь один раз для каждого фрагмента (пикселя) многоугольника.

Реализация первой версии системы визуализации данных (рис. 4.21), лежащей в основе программных средств, описанных ранее [78], состояла из двух ключевых частей: базы данных объектов сцены и системы передачи информации базы данных на ГПУ. База данных объектов сцены представляла из себя дерево объектов, где отношение родителя и дочернего узла состояло из отношения локальных матриц трансформации. Перемножение матрицы каждого дочернего узла с матрицами предков позволяло получить финальную трансформацию мира 3D-сцены. Отношение матриц позволяло соединить несколько объектов сцены вместе и, трансформируя корневой объект, получать верную позицию и ориентацию подчиненного относительно изменения положения и поворота, корневого пространства сцены. В это же время задачи системы передачи данных заключались в том, чтобы проходить по всем узлам дерева объектов, рассчитывать финальную матрицу, передавать всю информацию объекта на ГПУ и давать команду на его рисование.

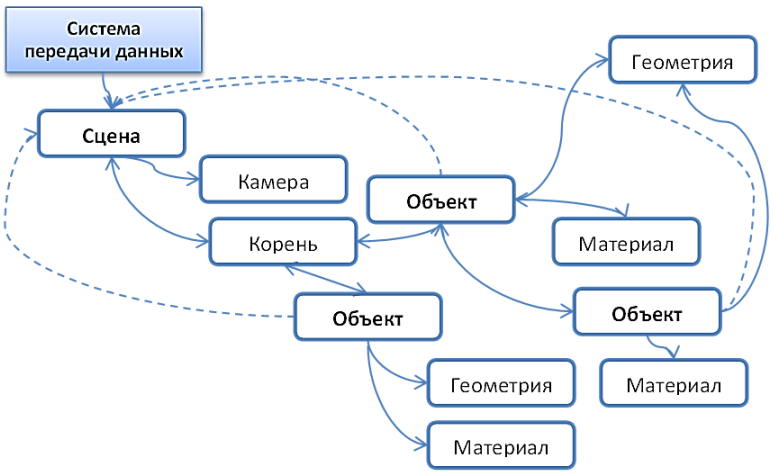


Рисунок 4.21 – Дерево сцены (база данных состояний) и система передачи данных на ГПУ не оптимизированной версии графического движка

Для решения этих проблем предлагается использовать усовершенствованную, по сравнению с предложенной ранее, архитектуру графической подсистемы, имеющую несколько оптимизаций для сохранения достаточной скорости прорисовки графических объектов. При этом используется набор технических решений для минимизации размера параметрического буфера, а также обходы системы проверки параметрического буфера группировкой данных, не отражающей действительное количество видимых объектов (рис. 4.22).

Из набора оптимизаций можно отметить следующие ключевые моменты:

- поддержка дерева обновлений объектов;
- объединение геометрии и параметров нескольких объектов;
- дублирование геометрии однотипных объектов;
- вычисление разности геометрии и частичное обновление буферов графического процессора.

Первая важная оптимизация – уменьшение параметрического буфера данных. Для каждого объекта базы данных сцены хранится специальный экземпляр класса флагов и кэша изменений (Renderable). Каждый объект Renderable имеет набор чисел, используемых как битовые карты флагов, которые могут ответить на вопрос – изменился ли определенный параметр связанного объекта мира с момента предыдущего вызова на рисование или нет. Каждый объект базы данных сцены теперь имеет набор сеттеров для обновления соответствующих битов флагов Renderable. В будущем планируется заменить сеттеры на шаблон наблюдения (Observable pattern) при наличии поддержки библиотеками или хост-средствами среды исполнения языка. Изменения параметров прозрачности приводит к тому, что объект самостоятельно перемещает себя в одноуровневый список прозрачных или непрозрачных объектов системы передачи данных.

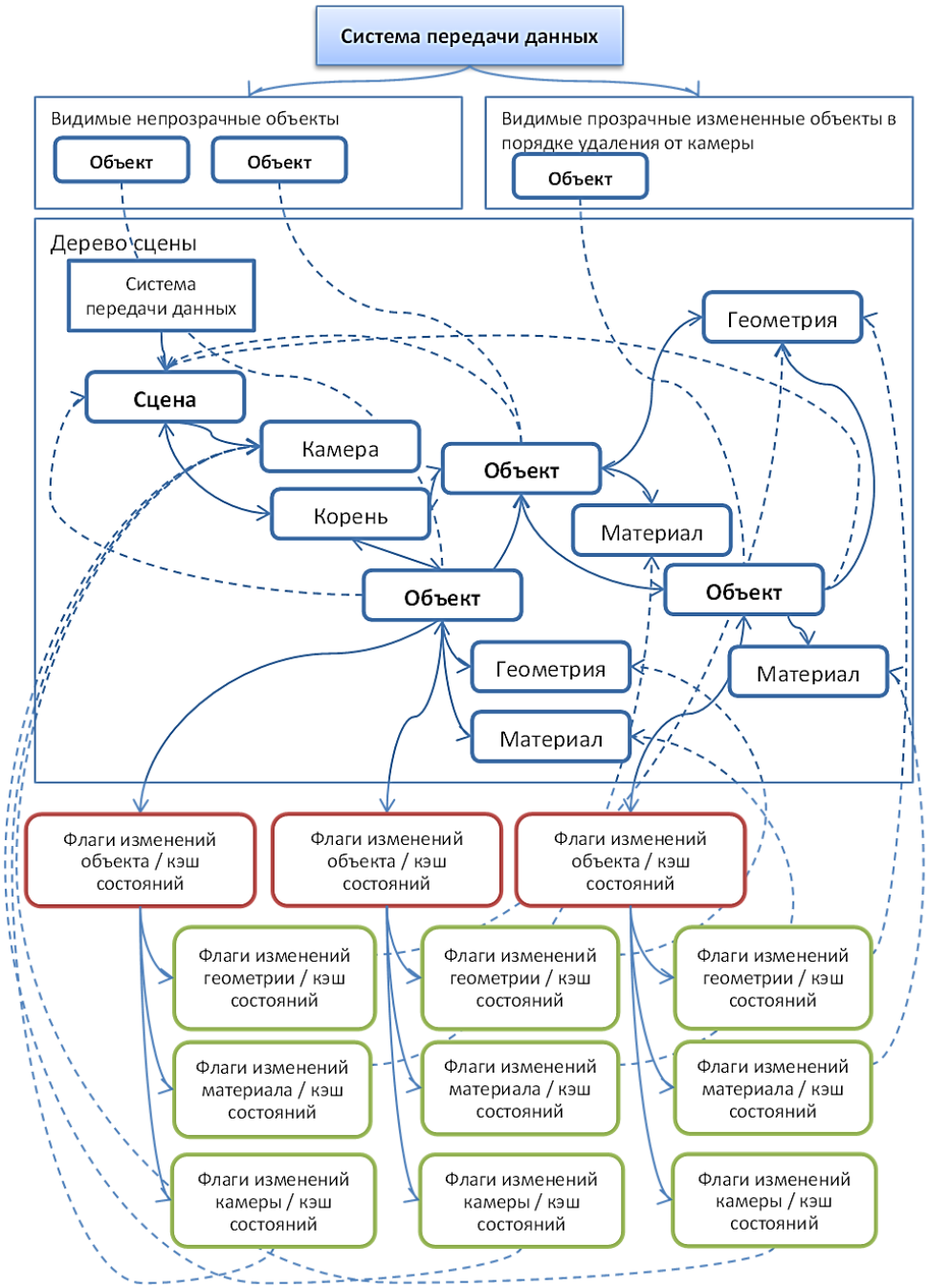


Рисунок 4.22 – Система передачи данных на ГПУ оптимизированной версии графического движка

Система передачи данных из первых версий системы также была изменена, и теперь производит анализ двух списков на рисование (прозрачные объекты и непрозрачные объекты) и выполняет необходимые сортировки и комбинации объектов по мере необходимости. Далее система проходит по списку непрозрачных и далее – по списку прозрачных объектов. На каждой итерации система анализирует связанный объект флагов Renderable, вычисляет промежуточные данные (по необходимости) и кэширует их на случай отсутствий изменений при повторном использовании в следующем кадре.

Затем каждый объект флагов делает запрос к интерфейсу драйвера ГПУ на передачу данных только, если флаг изменений для определенного параметра был установлен. В завершении система обнуляет все флаги для анализа изменений следующего кадра. Нужно отметить, что такой подход передает только те данные, которые были изменены. Кроме того, система следует рекомендациям тайловых архитектур отложенного расчета и рисует непрозрачные и прозрачные объекты в порядке, помогающем ГПУ максимизировать прирост производительности при удалении невидимых поверхностей [79]. Система сначала рисует все непрозрачные объекты, а после начинает работать с прозрачными, если они имеются в наличии.

Системы визуализации данных часто требуют демонстрации простых примитивов схематично при дальних расстояниях удаления от объектов. Количество примитивов в таких случаях может достигать миллионов фигур, одновременно видимых на экране. Средства минимизации передачи данных никак не помогут преодолеть ограничения тайловых архитектур.

Так как подобного рода примитивы не требуют большого количества вершин, предоставляются механизмы, комбинирующие геометрию объектов со всеми связанными данными на центральном процессоре. Комбинирование можно делать как статически перед началом цикла рисования, так и динамически между последующими кадрами. Функции комбинирования используют вставки на ассемблере для разных целевых семейств процессоров (x86, ARM) для ускорения процесса. В будущем планируется заменить ассемблерные вставки на встроенные (intrinsic) функции компилятора для улучшения переносимости кода.

Для модификации положения объектов на ГПУ предоставляется возможность передавать вспомогательные данные трансформации методом упаковки в текстуры. Для этого в нашем графическом движке есть специальный класс, позволяющий упростить трудоемкий процесс подготовки на стороне ЦПУ и распаковки на стороне шейдерных программ. Такой подход сложен в реализации, но тем не менее позволяет получить портативное решение для дешевых или старых устройств без обновлений ОС и новых графических устройств.

Подход комбинирования хорошо подходит для простых примитивов из-за низкого количества вершин многоугольников. К сожалению, он не применим для объектов с большим количеством вершин. В данной системе визуализации многие тела вблизи камеры отображены в виде сфер. Для демонстрации плавных переходов на видимых границах необходимо иметь более 1000 вершин для каждого объекта. Комбинирование такого количества данных на ЦПУ приведет к значительной потере производительности. Для решения этой проблемы используется интерфейс дублирования геометрии (geometry instancing API), доступный на большинстве целевых устройств. Идея дублирования заключается в том, что мы передаем геометрию однотипных объектов всего лишь один раз.

Кроме того, подготавливается специальный буфер дублирования с глобальными параметрами трансформации, цвета и материалов для каждого отдельного объекта (рис. 4.22).

Отметим, что второй буфер значительно меньше и часто состоит из нескольких векторов для каждого объекта. В заключение, используя специализированные вызовы драйвера ГПУ, мы командуем нарисовать определенный объект *N* раз, используя одинаковую геометрию, но передавая разные глобальные параметры (трансформации, цвета, материалов) из буфера дублирования в шейдерную микропрограмму графического адаптера.

Таким образом, мы получаем *N* объектов с одинаковой геометрией, но отличительными свойствами позиции, ориентации, цвета, текстурирования и материалов. Используя вершинный шейдер, можно также изменить геометрию отдельных объектов. Такой подход кардинально разгружает шину передачи данных для рисования тысяч однотипных объектов на существующих графических адаптерах.

Таким образом, мобильный программный компонент трассоискателя был дополнен подсистемой промежуточного хранения и выборки данных, подготовки запросов и визуализации данных. Для того чтобы обеспечить повторяемую сборку мобильного программного компонента, результаты которой бы не зависели от используемой операционной системы и программного окружения, применяется инструментальное средство Docker.

4.3.2 Преимущества разработанной системы визуализации данных.

Для замера производительности новой системы была использована искусственно созданная тестовая схема расположения кабельной линии. Исходная версия графического движка [79] не могла справиться со сценами такой симуляции с более чем 800 объектов (при частоте обновления в 30 кадров в секунду), состоящей из простых геометрических примитивов, например, прямоугольников и сфер. Однопоточная версия новой системы с применением описанных оптимизаций позволила увеличить количество сфер до 2000, а количество прямоугольников до 260 000 (рис. 4.23, 4.24).

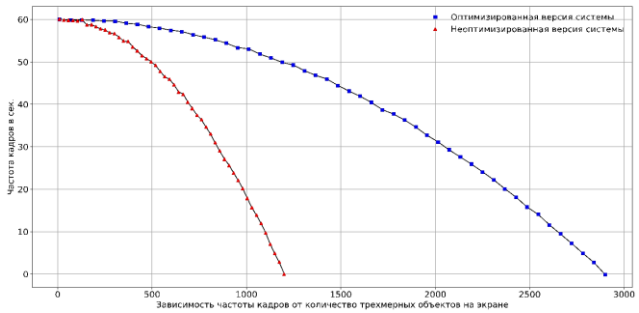


Рисунок 4.23 – Частота обновления экрана при различных количествах трехмерных объектов

Прирост в количестве трехмерных объектов в основном получен методом дублирования. Прирост в количестве простых двумерных примитивов получен путем комбинирования. Другие оптимизации имеют меньше влияния на ускорение, но, безусловно, их значение будет возрастать по мере усложнения схемы расположения кабельной линии.

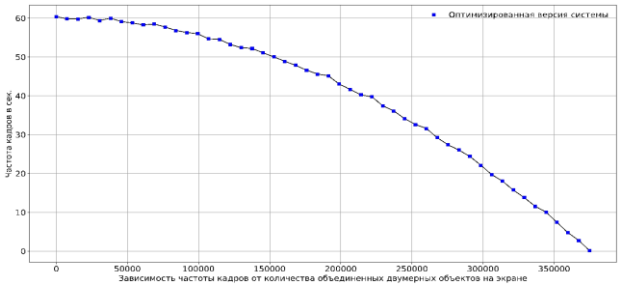


Рисунок 4.24 – Частота обновления экрана при различных количествах объединенных двумерных объектов (неоптимизированная версия не имела подсистемы объединения)

Полученная подсистема может демонстрировать положение трассируемого участка кабеля относительно первой зафиксированной точки.

4.4 Разработка системы искусственного интеллекта для мониторинга подземных силовых кабельных линий

Подземные кабельные сети в современных городах получили широкое распространение благодаря их многочисленным преимуществам. В то же время из-за их конструктивных особенностей часто оказывается невозможным локализовать возникающие неисправности путем визуального осмотра. Решить эту проблему позволила бы система мониторинга неисправностей в режиме реального времени для их обнаружения и классификации. Такая система может быть основана, например, на применении метода кривых Лиссажу [80] и использовании машины опорных векторов, одного из методов поверхностного машинного обучения для классификации неисправностей. Известно большое количество работ, посвященных системам классификации и обнаружения неисправностей, возникающих в системе передачи, например [81, 82], основанных на схожих принципах.

Рекуррентные нейронные сети, широко использующиеся для обработки временных рядов, обрабатывают элементы ряда последовательно, с применением информации, полученной при обработке предыдущих его элементов, также могут использоваться в таких системах. Кроме рекуррентных нейронных сетей, для обработки временных рядов можно использовать сверточные нейронные сети (Convolutional Neural Network, CNN) благодаря их способности к свертыванию параметров, извлечению признаков из локальных входных шаблонов, получению эффективных и модульных представлений данных. Объединение одномерной CNN и сети LSTM позволяет сочетать преимущества обоих

методов глубокого обучения, что показано в работах [83, 18], в которых гибридные CNN-LSTM-сети превзошли альтернативные технологии машинного обучения в решении некоторых задач классификации и регрессии. Однако в области локализации повреждений в подземных кабельных линиях эти технологии глубокого обучения пока еще не нашли широкого применения.

Цель настоящего раздела – разработать систему искусственного интеллекта (СИИ) для обнаружения, классификации и локализации неисправностей в трехфазной силовой подземной кабельной линии среднего напряжения с использованием глубоких нейронных сетей на основе CNN и LSTM- моделей.

Задачами настоящего раздела являются:

– моделирование неисправностей подземных силовых кабельных линий с использованием специализированного программного обеспечения PSCAD/EMTDC (https://www.pscad.com/) для получения большого набора данных для обучения глубоких нейронных сетей и разработки способа предварительной обработки полученных данных;

– разработка архитектур глубоких нейронных сетей, входящих в состав предлагаемой СИИ;

– оценка точности определения неисправностей с помощью обученных нейронных сетей.

4.4.1 Моделирование подземной кабельной линии. Рассмотрим, например, процесс, сопровождающий возникновение короткого замыкания в подземной кабельной линии. Построим имитационную модель этой линии (рис. 4.25) в PSCAD/EMTDC, состоящую из трехфазного неразветвленного кабеля.

Рисунок 4.25 – Схема подземной кабельной линии

Данное программное обеспечение позволяет задать рабочее напряжение, частоту, материал, сечение кабеля и параметры нагрузки, такие, как, например, ее активная и реактивная мощность. С помощью этой модели можно имитировать 10 различных неисправностей (см. табл.4.1), например, трехфазное замыкание на землю (ФФФЗ), замыкание одной из фаз на землю (ФЗ), межфазное замыкание (ФФ), межфазное замыкание на землю (ФФЗ) с различными условиями возникновения неисправности для того, чтобы получить большой набор данных для обучения глубоких нейронных сетей.

Генерирование данных

Точность работы любой СИИ очень сильно зависит от качества и количества образцов в обучающем наборе данных. Для того чтобы получить статистически значимый набор данных, все параметры

рассматриваемой модели варьируются в некоторых пределах, и для каждой возможной комбинации генерируется соответствующая выборка данных. Это позволяет получить достаточное количество разнообразных данных для обучения нейронных сетей. Пусть, например, неисправность возникает в интервале времени моделирования *t=200–220 мс* в зависимости от заданной фазы напряжения, а предохранитель срабатывает через 150 *мс* после возникновения короткого замыкания. Генерируется два набора данных: первый для обнаружения и классификации неисправности, а второй для локализации неисправностей. Параметры модели, используемые для генерации данных, приведены в таблице 4.1.

Таблица 4.1 – Параметры модели, используемые для генерации наборов данных

Параметр модели	**Значения параметров для первого набора данных**	**Значения параметров для второго набора данных**
Фаза напряжения при возникновении неисправности, градусы	0, 18, 36, 54,.....288, 306, 324, 342	0, 36, 72, 108, 144, 160, 192, 228, 324
Импеданс короткого замыкания, Ом	0.01, 0.1, 1, 5, 10, 15	0.01, 1, 5, 10
Тип неисправности	ФЗ(АЗ, БЗ, ВЗ), ФФЗ(АБЗ, БВЗ, АВЗ, АБВЗ, ФФ(АБ, БВ, АВ),	ФЗ (АЗ, БЗ, ВЗ)
Местоположение (удаление от источника) неисправности, км	2, 4, 6, 8, 10, 12, 14, 16, 18, 20	0.2, 0.4, 0.6, 0.8,.....19.4, 19.6, 19.8, 20

Значения параметров, приведенных в табл. 4.1, должны быть выбраны таким образом, чтобы с определенным шагом перекрывать значения, которые могут встретиться в реальных условиях, так как обученные нейронные сети могут правильно распознать и обработать только такие входные данные, встречавшиеся в обучающей выборке.

Таким образом, при использовании всех возможных комбинаций значений параметров модели генерируется 12000 выборок данных. Все сгенерированные данные имеют частоту дискретизации 8 *кГц*. Для имитации реальных условий добавляется аддитивный белый гауссовский шум с отношением сигнал/шум от 20 до 60 *дБ*, его амплитуда выбирается случайным образом для каждого образца в наборах данных. Из сгенерированных выборок с помощью скользящего окна вырезаются участки шириной 100 *мс* для формирования двух наборов данных – для сетей, отвечающих за классификацию и локализацию неисправностей. Затем наборы данных случайным образом перемешиваются и разделяются в отношении 5:1 для обучения и проверки работы глубоких нейронных сетей.

4.4.2 Архитектура предлагаемой СИИ. Для решения поставленной задачи предлагается использовать глубокие нейронные сети, основанные на комбинации одномерных сверточных сетей Conv-1D и LSTM-сетей для классификации и локализации неисправностей в подземных кабельных линиях.

Выход Conv-1D сети h_c после операции свертки может быть описан формулой:

$$h_c(\mathrm{c}_i) = b(c_i) + \sum_{j=0}^{l_c-1} w(c_i, j) \odot x(j). \quad (4.3)$$

где x – входной тензор, w – матрица весов, b – вектор смещения, c_i –i- й выходной канал; l_c – размерность входа; $\odot$ – оператор, означающий кросс-корреляцию.

К результату свертки затем применяется нелинейная функция активации, такая, как, например, RelU, и результат преобразуется с помощью операции одномерного пулинга (снижение разрешения входной последовательности путем взятия максимального значения h_p из соседних в обрабатываемой последовательности):

$$h_p(\mathrm{c}_i, j) = max_{m=0,\dots,lk-1}\, h_c(c_i, (l_s * j) + m), \quad (4.4)$$

где l_k и l_s означают длину ядра пулинга и длину шага соответственно.

Далее результат преобразуется в вектор и передается в рекуррентную LSTM-сеть (см. разд. 2.4).

В целом СИИ для обнаружения, локализации и классификации неисправностей с применением предобработки скользящим окном показана на рис. 4.26.

В процессе работы системы мгновенные значения токов и напряжений сохраняются в буфере соответствующего размера. Скользящее окно требуется для того, чтобы уменьшить длину данных на входе нейронной сети, разбив многомерные входные данные напряжения и тока на более мелкие фрагменты, которые можно быстро обработать с помощью нейронных сетей.

Возьмем для определенности скользящее окно длиной в 800 отсчетов (100 *мс*). Сеть для классификации неисправностей использует, в отличие от сети, описанной в работе [84], трехмерные сигналы тока и напряжения, так же, как и сеть для определения местоположения неисправности, измеряемые со стороны источника с помощью специальных высоковольтных измерительных средств. Таким образом, окно размером 6x800 «перемещается» по сигналу (рис. 4.26), сохраненному в буфере, и каждый участок передается в первую сеть для классификации неисправности, имеющую два выхода, соответствующих определенной неисправности и времени ее возникновения.

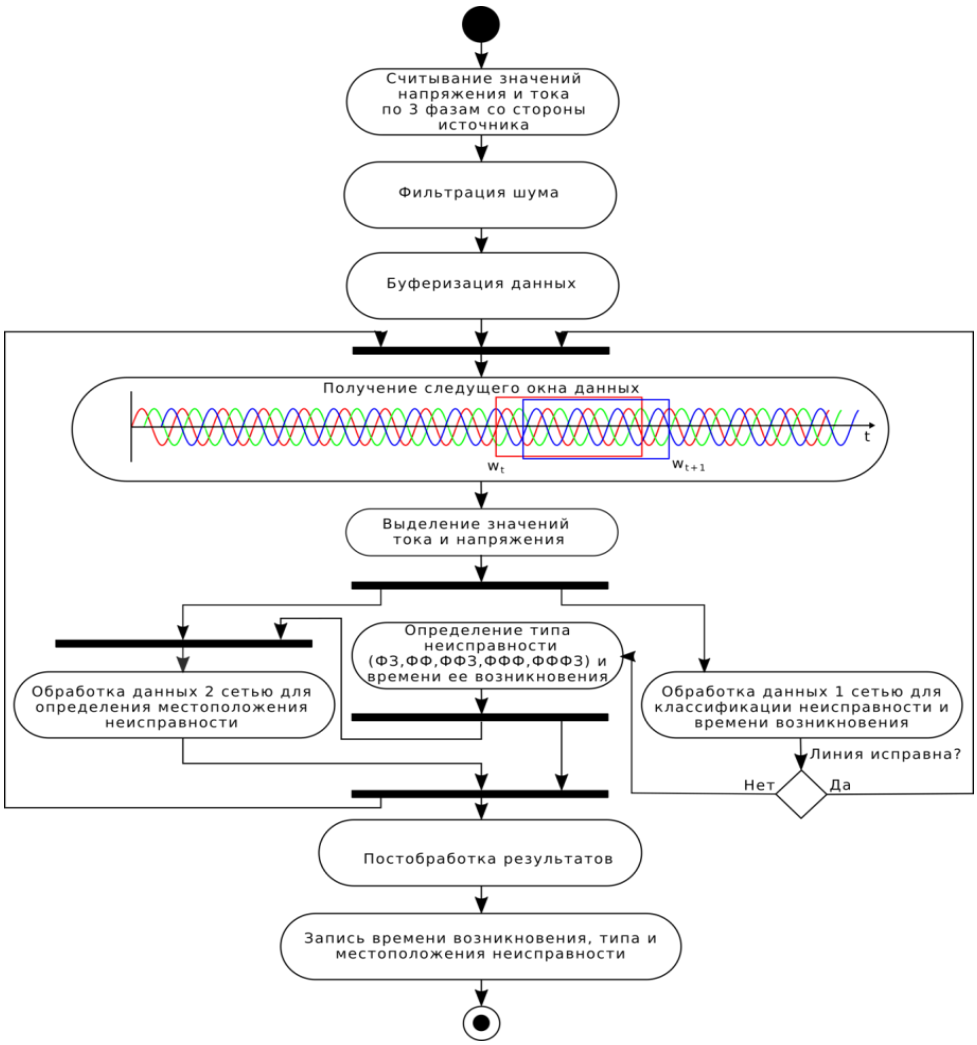


Рисунок 4.26 – Схема работы предлагаемой системы мониторинга

Если прогнозируемый класс неисправности не соответствует классу «исправно», то используется вторая сеть для локализации неисправности (ФЗ, ФФ, ФФЗ, ФФФ). Окно размером 6x800 с трехмерными сигналами тока и соответствующими сигналами напряжения передается во вторую сеть для определения местоположения неисправности, имеющую один выход, соответствующий расстоянию до места возникновения неисправности. В онлайн-системе мониторинга этот процесс повторяется бесконечно, где текущее окно обозначается как w_t, включающее данные между текущим временным шагом *t-799 и t* включительно. После того, как сети обработают эти данные, процесс повторяется для следующего окна w_{t+1}, содержащего данные между временным шагом *t-798* и *t+1* и включительно. Постобработка результатов необходима для получения окончательной оценки времени возникновения, типа и местоположения неисправности из множества выходных данных, генерируемых по нескольким окнам. Для окон, не содержащих данные о возникновении неисправности, первая сеть должна возвращать ноль, для остальных окон должен возвращаться номер отсчета, соответствующий времени начала

неисправности, относительно начала окна – целое число от 1 до 800. Тогда время возникновения неисправности может быть вычислено как:

$$t(w_i) = t_i + \frac{t_i - 799}{f_s} t_f \neq 0, \tag{4.5}$$

где t_f – номер отсчета, соответствующий возникновению неисправности и f_s – частота дискретизации. Среднее всех полученных значений и используется для окончательной оценки времени возникновения неисправности.

Тип неисправности будет оставаться постоянным большую часть времени, однако может колебаться в окнах, где происходит переход между различными возможными значениями. Чтобы избавиться от подобной неопределенности, необходимо выбрать класс, возникающий наибольшее количество раз после предполагаемого начала неисправности. Местоположение неисправности определяется только для окон данных, определенных как содержащих информацию о неисправности. Среднее значение всех полученных на выходе сети значений используется для получения окончательной оценки расстояния до места возникновения неисправности.

4.4.3 Архитектура глубокой нейронной сети для определения типа неисправности. На рис. 4.27 показана возможная архитектура нейронной сети для обнаружения и классификации неисправностей в подземной кабельной линии.

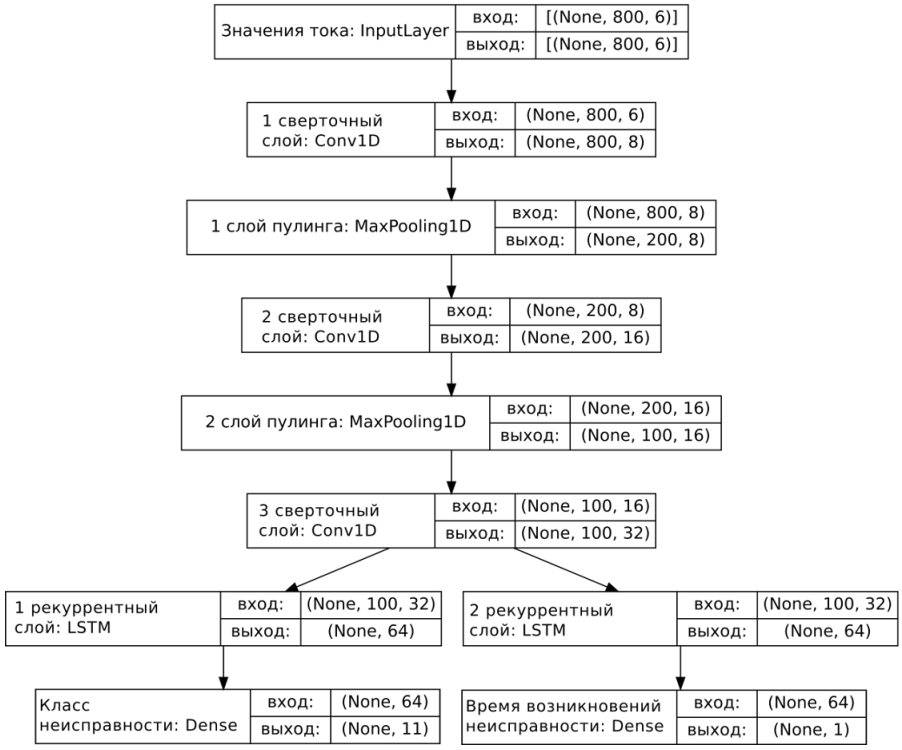


Рисунок 4.27 – Архитектура глубокой нейронной сети для определения класса неисправности и времени ее возникновения

Помимо 10 типов возможных неисправностей, имеется дополнительный класс «исправно» для предотвращения появления случайных выходных значений сети при отсутствии неисправности. По сравнению с определением местоположения неисправности их классификация является более простой задачей. Но для повышения точности эта сеть использует и значения тока и напряжения. Чтобы получить набор данных для этой сети, окно размером 100 мс скользит по каждой выборке данных, и для каждой выборки вырезаются два участка. Обрезка выполняется таким образом, чтобы в результате получилось не менее 20 % данных после возникновения неисправности. Должно получиться 24000 образцов по 2400 на каждый тип неисправности. Образцы, соответствующие классу «исправно», можно получить путем случайной обрезки участков с менее чем 20 % данных после возникновения неисправности. Важно сделать это таким образом, чтобы количество образцов, соответствующих классу ИСП («исправно»), было таким же, как и в других классах, что необходимо для получения сбалансированного набора данных. В общей сложности должно получиться 26400 образцов, из которых 21120 будут использоваться для обучения, а 5280 для тестирования нейронной сети. Обнаружение возникновения неисправности с использованием перехода от класса «исправно» к одному из других классов может оказаться сложным из-за возникновения множества таких переходов в какой-то момент времени.

В работе [81] был предложен фильтр, устраняющий эту неопределенность. Однако он может вызвать значительную задержку в обнаружении неисправности, даже несмотря на относительно высокую частоту дискретизации. Поэтому участки с меньшим количеством данных после возникновения неисправности (<20%) не используются для обучения этой сети. Тем не менее сложность в определении точного времени возникновения неисправности сохраняется, а это является важным для данной СИИ. В связи с этим предлагается использовать нейронную сеть Conv1D-LSTM с двумя выходами, способную не только классифицировать неисправность, но и определять время ее возникновения. Это можно использовать для улучшения работы второй сети, определяющей местоположение неисправности, так, например, и для управления силовым предохранителем.

Сеть состоит из трех одномерных сверточных слоев с размером ядра 41 и 8, 16, 32 фильтрами в каждом слое с функцией активации ReLU. Слой пулинга с размером ядра, равным 4, используется после первого сверточного слоя, в отличие от [84], и, кроме того, для сверточных слоев используется каузальное дополнение по краям, не разрушающее временной порядок данных [9]. Полученный результат затем дублируется, и каждая копия подается на свой LSTM-слой с 64 ячейками без преобразования в вектор, в отличие от [8], так как LSTM-слой способен обрабатывать временные ряды любой размерности. Полносвязные слои

следуют за обоими LSTM-слоями. Первый слой с функцией активации softmax состоит из 11 нейронов, на выходе которых получаются вероятности принадлежности входного образца к одному из классов. Второй слой содержит 1 нейрон с линейной функцией активации, на выходе которого получается номер отсчета, соответствующий началу возникновения неисправности в текущем входном образце. Для образцов, не содержащих неисправности, этот выход должен быть равен 0. Для обучения сети можно использовать оптимизатор ADAM [85] со скоростью обучения $l_r = 10^{-5}$. Для выходного слоя в качестве функции потерь lf_1 используется мультикатегориальная кроссэтропия:

$$lf_1 = -\sum_{i=1}^{n}\sum_{j=0}^{10} y_{ij}\, ln\, \widehat{y_{ij}}. \tag{4.6}$$

где y_{ij} – вероятность i- го обучающего образца, соответствующего j- му классу, $\widehat{y_{ij}}$– вероятность того, что i- й обучающий образец на выходе сети будет соответствовать j- му классу и n – общее количество обучающих образцов. Для выходного слоя другой сети необходимо использовать среднеквадратическую ошибку lf_2:

$$lf_2 = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2, \tag{4.7}$$

где y_i и $\widehat{y_i}$ – фактический и определяемый сетью номер отсчета, соответствующий началу возникновения неисправности.

4.4.4 Архитектура глубокой нейронной сети для определения местоположения неисправности. Архитектура сети для определения местоположения неисправности в кабельной линии показана на рис. 4.28.

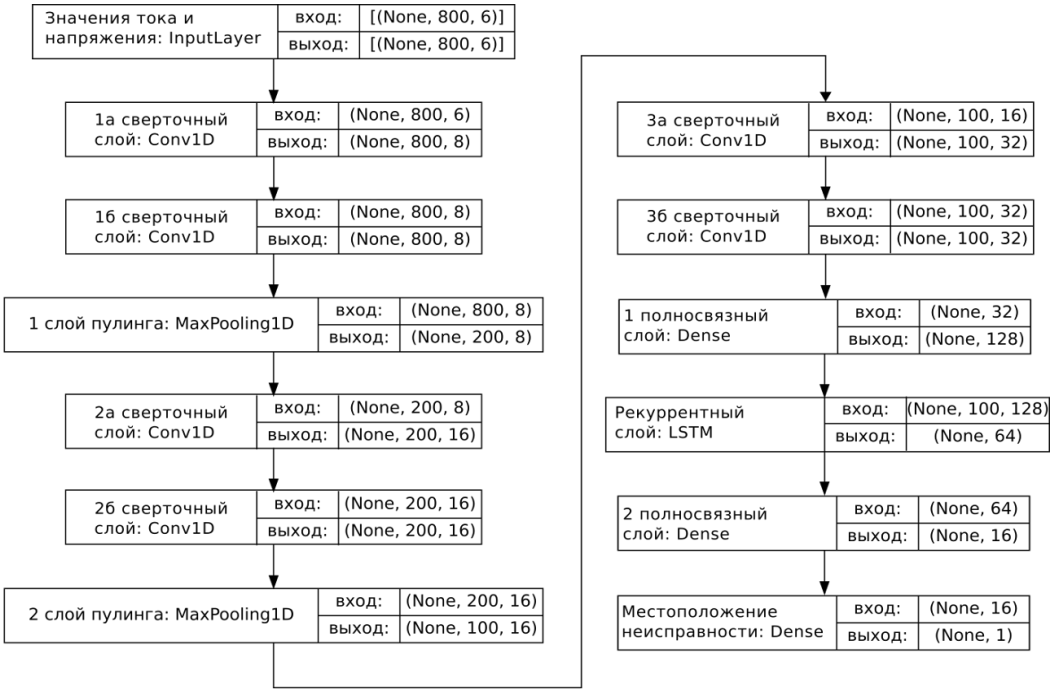


Рисунок 4.28 – Архитектура глубокой нейронной сети для определения местоположения неисправности

На вход сети поступают 6-мерные данные для значений тока и напряжения по 3 фазам. Так как эта задача является более сложной, по сравнению с предыдущей, для ее решения необходимо использовать более глубокую нейронную сеть. Поэтому можно поступить так, как описано в работе [84], взяв 2 подряд идущих сверточных слоя, за которыми следует слой пулинга, и повторить этот шаблон 3 раза.

Далее следует полносвязный слой для сжатия размерности данных, а потом LSTM- слой и еще один полносвязный слой перед выходным слоем, состоящим из одного нейрона с линейной функцией активации для определения местоположения неисправности. Вход этой сети также является двухмерным с формой 800x6 отсчетов из образцов, обрезанных таким образом, чтобы присутствовало не менее 20% данных после возникновения неисправности, а данные, не содержащие информации о неисправности, гарантированно не использовались для обучения этой сети, так как эта сеть задействуется только в случае возникновения неисправности. Из каждой выборки модельных данных нужно взять по 5 фрагментов, что в сумме дает 60000 образцов, 48 000 для обучения и 12000 для тестирования нейронной сети соответственно.

Таким образом, эта сеть состоит из 3 стеков свёрточных слоев, по 2 слоя в каждом, аналогичным тем, что использовались в первой сети. После каждого слоя, кроме первого, следует слой пулинга с размером ядра, равным 2, после первого слоя используется пуллинг с размером ядра 4, для того чтобы сократить количество входных данных и ускорить работу нейронной сети.

Затем следует полносвязный слой со 128 нейронами и слой LSTM из 64 ячеек. И еще один полносвязный слой с 16 нейронами идет перед выходным слоем с одним нейроном и линейной функцией активации. Для более стабильного обучения глубокой сети можно использовать оптимизатор ADAM с $l_r = 10^{-5}$ и уменьшающейся в 2 раза скоростью обучения, если обнаруживается, что функция потерь изменятся незначительно (к примеру, меньше чем на 5% за 5 эпох обучения). В качестве функции потерь используется средняя квадратическая ошибка, рассчитываемая по формуле (11), где y_i и $\widehat{y_i}$ – фактическое и определяемое сетью местоположение неисправности в i- м обучающем образце.

Результаты работы обученной сети для определения типа неисправности на тестовых модельных данных показаны в табл. 4.2. Общая точность на модельных данных составила 99,71%, по каждому классу минимальная и максимальная точность составляет 99,31% и 100% соответственно. Таким образом, сеть способна обнаружить и классифицировать неисправность с достаточно высокой точностью, причем небольшое количество ошибок связано в основном с неисправностями типов ФЗ и ФФЗ из-за схожести их сигналов.

Авторы работы [11] провели исследование различных способов обработки данных и их влияние на точность классификации. Ими было

обнаружено, что использование сигналов напряжения и тока повышает точность по сравнению с использованием только сигналов тока или напряжения.

Таблица 4.2 – Матрица ошибок и точность определения каждого класса неисправностей

Определенный сетью тип неисправности	Действительный тип неисправности										
	АЗ	БЗ	ВЗ	АБЗ	БВЗ	АВЗ	АБ	БВ	АВ	АБВЗ.	ИСП
АЗ (100%)	581	0	0	0	0	0	0	0	0	0	0
БЗ (100%)	0	603	0	0	0	0	0	0	0	0	0
ВЗ (99.84%)	0	0	616	0	1	0	0	0	0	0	0
А БЗ (99.33%)	1	0	0	591	0	0	3	0	0	0	0
БВЗ (99.51%)	0	0	0	0	605	0	0	3	0	0	0
АВЗ (99.31)	0	0	0	0	0	578	0	0	4	0	0
АБ (99.65%)	0	0	0	2	0	0	575	0	0	0	0
БВ (99.52%)	0	0	0	0	3	0	0	628	0	0	0
АВ (99.82%)	0	0	0	0	0	1	0	0	564	0	0
АБВЗ (99.84%)	0	0	0	0	0	0	0	0	1	615	0
ИСП (100%)	0	0	0	0	0	0	0	0	0	0	619

Однако они показывают, что это улучшение незначительно, так как на низких частотах значения напряжения не содержат достаточной информации относительно класса неисправности. Поэтому в предложенной СИИ используется частота дискретизации 8 кГц, что достаточно для того, чтобы получить большую точность классификации по сравнению с результатами, полученными в работе [84].

Средняя абсолютная ошибка при определении времени возникновения неисправности время равна 0,35 мс. С учетом наличия аддитивного шума и относительно низкой частоты дискретизации 8 кГц (0,125 мс) такую ошибку можно считать вполне удовлетворительной.

На рис. 4.29 представлен график абсолютной погрешности определения времени возникновения неисправности в зависимости от образцов входных данных. Образцы отличаются начальными отсчетами внутри окна, соответствующими началу возникновения неисправности.

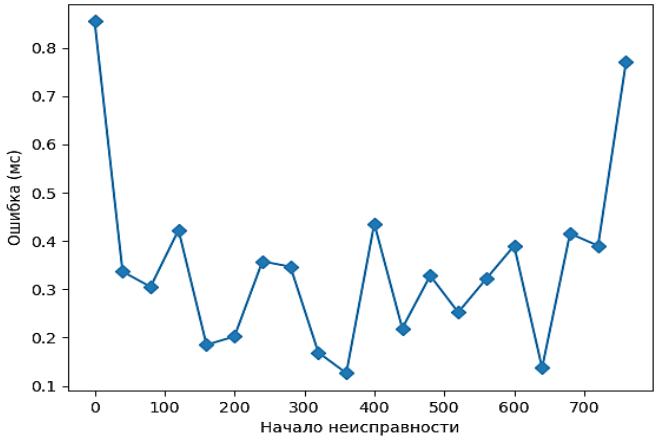


Рисунок 4.29 – Зависимость ошибки определения времени возникновения неисправности от ее положения в образце данных

Например, первая слева точка на рис. 4.29 соответствует среднему абсолютному значению ошибки, в которых неисправность возникает в пределах первых 40 точек окна (или в течение первых 5 мс) данных.

Можно заметить, что средняя абсолютная ошибка для образцов, в которых неисправность, возникает на краях, относительно больше. Это можно объяснить тем, что сеть не имеет достаточно данных для обнаружения неисправности в таком небольшом интервале времени.

Результаты работы сети для определения местоположения неисправности показаны на рис. 4.30. На этом графике показаны средние абсолютные ошибки определения

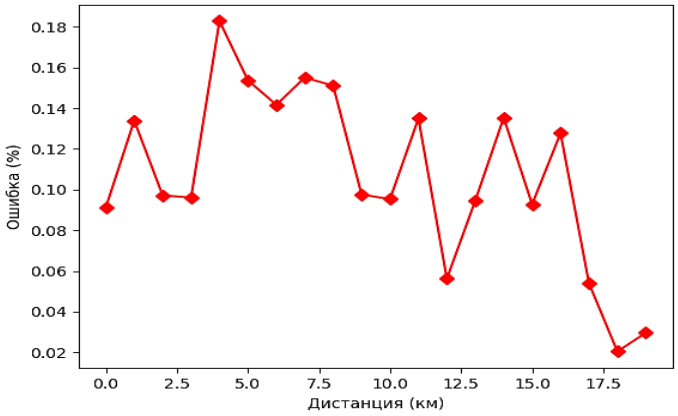


Рисунок 4.30 – Зависимость ошибки определения местоположения неисправности в зависимости от расстояния до нее

местоположения неисправности в зависимости от расстояния до ее локализации. Например, первая точка соответствует среднему значению абсолютной ошибки местоположения неисправности для образцов исходных данных, где неисправность возникает на первом километре кабеля. В целом средняя абсолютная ошибка при определении местоположения неисправности составляет 0,11% (или 21,4 м), что немного меньше, чем получено в работе [8]. Улучшение можно объяснить увеличенной в два раза частотой дискретизации исходных данных. Таким образом, СИИ позволяет определить местонахождение неисправности с достаточно высокой точностью, что может значительно облегчить ручной поиск точного места расположения неисправности.

4.5 Выводы

Таким образом, проведен анализ требований, предъявляемых к современным кабелеискателям, и предложена концепция современного кабелепоискового комплекса, снабженного цифровой системой обработки данных и индикации результатов, что позволяет ему эффективнее бороться с помехами и точнее определять расположение кабельных трасс, снижает вес и габариты, увеличивает при этом удобство и упрощает его использование. Достижение этих результатов достигнуто за счет разгрузки аналоговой части трассоискателя, при этом часть ее функций берет на себя

цифровая часть системы, осуществляющая дополнительную цифровую фильтрацию, анализ данных и индикацию результатов.

Проведен обзор и анализ возможностей программного обеспечения трассоискателей, показавший, что возможностей существующего программного обеспечения недостаточно для реализации предложенной концепции кабелепоискового комплекса. Поэтому была создана библиотека для языка Python, позволяющая работать с устройством сбора данных Л Кард E502, разработаны алгоритмы и программный компонент трассоискателя, позволяющий решить задачи сбора, регистрации, визуализации, фильтрации и вейвлет-анализа данных. Выбор Python в качестве языка программирования позволил применить при проектировании программных средств готовый набор проверенных временем библиотек, написанных на FORTRAN и C, содержащих реализации алгоритмов линейной алгебры, фильтрации, визуализации, привязки данных к местности, быстрого преобразования Фурье и других [69].

В целях увеличения потенциального круга пользователей и для расширения функциональных возможностей трассоискателя, связанных с применением встроенного во многие мобильные устройства gps-приемника и магнитного компаса, а также для удобства его практического использования была разработана и протестирована кроссплатформенная архитектура программного компонента трассоискателя и выбраны инструментальные средства ее реализации. Предложенная архитектура основана на разделении кода, зависимого от целевой платформы, от кода, который может без каких-либо изменений запускаться на любой платформе. С помощью таких тулкитов, как Kivy, и инструментальных средств сборки кроссплатформенных приложений CMake и Buildozer удалось портировать программный компонент трассоискателя на мобильную операционную систему Android, сохранив при этом его работоспособность и на десктопных операционных системах Windows и Linux. В будущем возможно портирование и на другие мобильные устройства на базе операционных систем iOS и Windows 10 Mobile [73].

Разработаны улучшенная реализация и четыре способа оптимизации графической системы для визуализации большого количества двухмерных и трехмерных примитивов для мобильных тайловых процессоров. Следует отметить, что особенности рассматриваемой архитектуры требуют минимизировать количество данных, передаваемых на графический адаптер, и количество запросов на рисование. Предложенная система достигает этих целей, храня дерево изменений состояния каждого объекта сцены, для подготовки и минимизации списка объектов на рисование. При этом так же уменьшается количество объектов передачи путем динамического объединения их геометрии. Кроме того, используется API для дублирования геометрии при повторной прорисовке, передается только разница между конечными сетками геометрии.

Произведена оценка влияния предложенных оптимизаций на примере визуализации искусственно сгенерированной тестовой кабельной линии. Результаты измерений на наборе целевых устройств демонстрируют прирост производительности в 2,5 раза для трехмерных объектов и более 2 порядков для двухмерных объектов по сравнению со стандартной системой визуализации. Такой прирост делает систему рентабельной на переносных устройствах с ограниченным профилем энергопотребления.

В целом разработанная подсистема осуществляет сбор и обработку результатов измерения трехкомпонентного магнитометра с частотой дискретизации до 2 МГц с последующей возможной визуализацией кабельной линии в 3-D- пространстве.

Визуализация может производиться на мобильных устройствах для проведения полевых работ с аппаратурой и работает с частотой обновления до 30–60 кадров в секунду с большим количеством 3-D-примитивов на экране. Графическая подсистема оптимизирована для мобильных тайловых графических процессоров отложенного рисования. Все подсистемы в целом используют специализированную базу данных InfluxDB для сбора временных рядов вместе с промежуточной подсистемой графического конструирования запросов [76].

Предложена СИИ для онлайн-мониторинга неисправностей подземных кабельных линий, использующая специализированное программное обеспечение для получения большого набора данных для обучения глубоких нейронных сетей, и изложена процедура подготовки данных, необходимых для обучения. Кроме того, предложены архитектуры глубоких нейронных сетей на основе Conv1D и LSTM-слоев. Был собран большой набор данных по описанному способу и использован для обучения глубоких нейронных сетей. Показаны результаты работы предложенной СИИ на модельных данных.

ЗАКЛЮЧЕНИЕ

Таким образом, спектр применения машинного обучения достаточно широк, что требует тщательного выбора того или иного метода и технологии в зависимости от особенностей решаемой задачи. В частности, применение эволюционных вычислений позволяет значительно повысить скорость и точность определения параметров индуктивного компонента магнитометров по сравнению с традиционными методами многомерной оптимизации.

В настоящее время среди методов машинного обучения все большое значение приобретают методы глубокого обучения, основанные на применении современных графических процессоров, для обработки больших объемов данных. Изложенная методология прогнозирования временных рядов на основе глубоких рекуррентных нейронных сетей позволяет более полно учесть временные зависимости в данных. Сети на основе долгой краткосрочной памяти LSTM и сети GRU позволяют повторно задействовать в процессе обучения предыдущую информацию и тем самым решить проблему затухания градиента. Архитектуры глубоких нейронных сетей, предложенные в данной работе, прекрасно демонстрируют преимущества таких нейронных сетей на примере прогнозирования метеорологических показателей в г. Бишкек и других геоэкологических данных.

Особое внимание среди других технологий глубокого обучения уделяется сверточным сетям CNN. Причина в том, что CNN доказали свою эффективность в приложениях компьютерного зрения, и поэтому они являются приоритетным выбором в системах, осуществляющих измерения на основе изображений. Тот факт, что в последнее время в исследовательских целях стали доступны обширные наборы данных, и в частности, изображения рентгеновских снимков грудной клетки, стимулирует использование ИИ и, в частности, CNN, для решения задач медицинской геоэкологии. Кроме того, более широкого использования заслуживают одномерные сверточные сети Conv-1D, особенно в комбинации в рекуррентными слоями. Сверточная часть позволяет превратить длинную входную последовательность в более короткую последовательность высокоуровневых признаков (уменьшив ее разрешение). А затем последовательность выделенных признаков подается на вход рекуррентной части сети. Преимущества такого подхода были показаны на примере СИИ для мониторинга кабельных линий.

Представленная интерпретация применения современных методов применения машинного обучения в различных информационно-измерительных системах, основанная на ключевых технологиях глубокого обучения, позволяет существенно сократить объем необходимых для обучения нейронных сетей данных и повысить их информационную эффективность. Знание и правильное применение этих методов делает работу исследователя в области анализа данных более целенаправленной и организованной, существенно повышает его производительность и надежность получаемых им результатов практических исследований.

ЛИТЕРАТУРА

1. C. Alippi et al., "Artificial intelligence for instruments and measurement applications," IEEE Instrum. Meas. Mag., vol. 1, no. 2, pp. 9-17, Jun. 1998.
2. S. Shirmohammadi and A. Ferrero, "Camera as the instrument: the rising trend of vision based measurement," IEEE Instrum. Meas. Mag., vol. 17, no. 3, pp. 41-47, Jun. 2014.
3. P. Pouladzadeh et al., "Measuring calorie and nutrition from food image," IEEE Trans. Instrum. Meas., vol. 63, no. 8, pp. 1947-1956, Aug. 2014.
4. J. Zhou et al., "A new hand measurement method to simplify calibration in CyberGlove-based virtual rehabilitation," IEEE Trans. Instrum. Meas., vol. 59, no. 10, pp. 2496-2504, Oct. 2010.
5. S. A. Mohammed, S. Shirmohammadi, and S. Altamimi, "A multimodal deep learning based distributed network latency measurement system," IEEE Trans. Instrum. Meas., Jan. 2020.
6. J. Schmidhuber, "History of Computer Vision Contests Won by Deep CNNs on GPU," March 2017. [Online]. Available: http://people.idsia.ch/~juergen/computer-vision-contests-won-by-gpu-cnns.html
7. Брякин И.В. Прикладные аспекты малоглубинной магниторазведки // Проблемы автоматики и управления.– Бишкек: Илим, 2016, №1. – С.65–75.
8. Ханасова В.В., Верзунов С.Н. Двойной Т-образный фильтр для цифрового трёхкомпонентного индукционного магнитометра // Вестник Кыргызско-Российского Славянского университета. – 2017. – Т. 17. – № 5. – С. 101–104.
9. http://www.izmiran.ru/~kozlov/pdf/part_rus2.pdf (02.03.2022)
10. Ковязин, В.А. Анализ применения схем замещения катушки индуктивности в расчетах электрических цепей постоянного тока / В. А. Ковязин, В. П. Кобазев, Л. И. Иванова // Наукові праці Донецького національного технічного університету. – Донецьк, 2008. – Вип.8(140). – С.55–57.
11. Верзунов, С.Н. Способ измерения параметров катушек индуктивности магнитометров и его реализация на основе платы сбора данных Л Кард Е502/ Верзунов С.Н.// Проблемы автоматики и управления. – Бишкек, 2018. – № 2 (35) – С. 94 –102.
12. Чечет, П.Л. Программная реализация измерения индуктивности с учётом паразитной ёмкости/ Чечет П.Л. // Известия гомельского государственного университета им. Ф. Скорины. – 2012. – №6(75). – С. 152–155.
13. Дворяшин, Б. В. Основы метрологии и радиоизмерения / Дворяшин Б. В.// Учеб. пособие для вузов – М.: Радио и связь, 1993.

14. http://www.machinelearning.ru/wiki/index.php?title=Генетический_алгоритм (дата обращения: 27.10.2020)
15. Верзунов, С. Н. Разработка кроссплатформенного программного компонента трассоискателя / С. Н. Верзунов // Проблемы автоматики и управления. – 2020. – № 1(38). – С. 50–59. – DOI 10.5281/zenodo.3904110.
16. L502/E502 Руководство программиста Ревизия 1.1.7 Ноябрь 2016 Автор руководства: Борисов Алексей ООО "Л Кард" 117105, г. Москва, URL: http://lib.knigi-x.ru/23tehnicheskie/188243-1-l502-e502-rukovodstvo-programmista-reviziya-117-noyabr-2016-avtor-rukovodstva-borisov-aleksey-ooo-l-kard.php (02.03.2022)
17. Верзунов, С. Н. Сравнительный анализ возможностей мультивейвлетной нейросетевой модели для решения задач прогнозирования / С. Н. Верзунов, Н. М. Лыченко // Вестник Кыргызско-Российского Славянского университета. – 2019. – Т. 19. – № 4. – С. 39–45.
18. Верзунов, С. Н. Применение глубоких нейронных сетей для краткосрочного прогноза дальности видимости / С. Н. Верзунов // Проблемы автоматики и управления. – 2019. – № 1(36). – С. 118–130. – DOI 10.5281/zenodo.3253019.
19. https://hpiers.obspm.fr/eoppc/eop/eopc04/eopc04.62-now (дата обращения: 20.11.2016).
20. http://hpiers.obspm.fr/eop-pc/models/UT1/UT1LOD_marees.php (дата обращения: 11.12.2016).
21. Верзунов С.Н., Лыченко Н.М. Мультивейвлетная полиморфная сеть для прогнозирования геофизических временных рядов // Проблемы автоматики и управления. – 2017. – № 1 (32). – С. 78–87.
22. Ratnadip A., Agrawal R.K. Homogeneous Ensemble of Artificial Neural Networks for Time Series Forecasting // International Journal of Computer Applications. 2011 vol. 32 no. 7. pp. 1-8.
23. Zhang G. Time Series Forecasting Using a Hybrid ARIMA and Neural Net-work Model // Neurocomputing. 2003. vol. 50, pp. 159-175.
24. Верзунов, С.Н. Мультивейвлетная полиморфная сеть для прогнозирования геофизических временных рядов / Верзунов С.Н., Лыченко Н.М. // Проблемы автоматики и управления. 2017. – № 1 (32). – С. 78–87.
25. Верзунов, С.Н. Аппроксимация временных рядов полиморфной вейвлет-сетью с обратными связями / Верзунов С.Н., Лыченко Н.М. // Математические структуры и моделирование. – 2016. –№ 2 (38). – С. 16–26.
26. Шолле, Ф. Глубокое обучение на Python / Шолле Ф.// – СПб.: Питер, 2018.
27. Zhang G. Time Series Forecasting Using a Hybrid ARIMA and Neural Net-work Model // Neurocomputing. 2003. vol. 50, pp. 159-175

28. Miljanovic M. Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction // Indian Journal of Computer Science and Engineering (IJCSE). –2012 V. 3 N. 1 – P. 180–191.
29. Качество воздуха в Центральной Азии: пора бить тревогу. https://livingasia.online/2020/09/01/kachestvo-vozduha-v-czentralnoj-azii-pora-bit-trevogu/ (дата обращения: 30.10.2020)
30. Верзунов С.Н., Лыченко Н.М. Краткосрочное прогнозирование индекса качества воздуха на основе ARIMA-моделей // Математическое и компьютерное моделирование: Сборник материалов VII Международной научной конференции (22 ноября 2019г.). – Омск: Изд-во Омск. гос. ун-та, 2019.
31. Верзунов С.Н., Лыченко Н.М. Анализ и ARIMA-модели динамики изменения концентрации PM2.5 в атмосферном воздухе г. Бишкек // Проблемы автоматики и управления. – N1. – Бишкек: Илим, 2019. – С. 21–30.
32. Великанова Л.И., Лыченко Н.М. Мультирегрессионные и обобщенно-регрессионные нейросетевые модели краткосрочного прогноза загрязнения PM2.5 в г. Бишкек с учетом метеорологических параметров// Проблемы автоматики и управления. – N2. – Бишкек: Илим, 2019. – С. 42–51.
33. Лыченко Н.М., Сороковая А.В. Применение LSTM-нейронных сетей для классификации индекса качества воздуха г. Бишкек // Проблемы автоматики и управления. – 2020. – № 1 (38). – С. 70–80. DOI: 10.5281/zenodo.3904130
34. Бокс Д., Дженкинс Т. Анализ временных рядов: прогноз и управление. –М.: Мир, 1974. –242 с.
35. Современное прогнозирование. [Электронный ресурс]. URL: https://forecasting.svetunkov.ru/etextbook/ (дата обращения: 30.09.2020).
36. Филатова Т.В. Применение нейронных сетей для аппроксимации данных. Вестник Томского государственного университета. – 2004. – №284. – С. 121–125.
37. Голохвост К.С., Кику П.Ф., Христофорова Н.К. Атмосферные взвеси и экология человека // Экология человека. – №10. – 2012. – Стр. 5–10 .
38. Сайт «Расписание погоды rp5.ru» Архив погоды в Бишкеке https://rp5.ru/%D0%90%D1%80%D1%85%D0%B8%D0%B2_%D0%BF%D0%BE%D0%B3%D0%BE%D0%B4%D1%8B_%D0%B2_%D0%91%D0%B8%D1%88%D0%BA%D0%B5%D0%BA%D0%B5 (дата обращения: 30.04.2020)
39. Лыченко Н.М. Регрессионный анализ метеорологических факторов и концентраций частиц PM2.5 в атмосферном воздухе г. Бишкек//

Проблемы автоматики и управления. – N2. – Бишкек: Илим, 2019. – С. 5–15.
40. Лучевая диагностика коронавирусной болезни (COVID-19): организация, методология, интерпретация результатов: методические рекомендации / Сост. С. П. Морозов, Д. Н. Проценко, С.В. Сметанина [и др.] // Серия «Лучшие практики лучевой и инструментальной диагностики». – Вып. 65. – М.: ГБУЗ «НПКЦ ДиТ ДЗМ», 2020.
41. Акишева А.Б. Частота встречаемости пневмонии по материалам протоколов патологического бюро // Сборник статей XXXIX Международной научно-практической конференции : в 2 ч. Том. Часть 1. – 2020 г. – С. 182–184.
42. Wong HYF, Lam HYS, Fong AH-T, Leung ST, Chin TWY, et al. Frequency and distribution of chest radiographic findings in COVID-19 positive patients. Radiology. – 2020.
43. Zech JR, Badgeley MA, Liu M, Costa AB, Titano JJ, Oermann EK. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study. PLoS Med. 2018;15:e1002683.
44. Daniel S. Kermany et all, Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning, Cell, Volume 172, Issue 5, 2018, Pages 1122-1131.e9, ISSN 0092-8674, https://doi.org/10.1016/j.cell.2018.02.010.
45. Верзунов С.Н., Раимжанов Х.А. Сравнение глубоких нейронных сетей на основе различных предварительно обученных cnn для диагностики COVID-19 по рентгеновским снимкам // Проблемы автоматики и управления. – №1. – 2021. – С. 12–25.
46. Туберкулез в Европейском регионе ВОЗ // Информационный бюллетень. - Копенгаген, март 2017 г. https://www.euro.who.int/__data/assets/pdf_file/0006/397446/Factsheet_WHO_WTBD_2019_RUSS.pdf (дата обращения (01.10.2020))
47. Apostolopoulos ID, Mpesiana TA. Covid. 19: automatic detection from X-ray images utilizing transfer learning with convolutional neural networks. Phys Eng Sci Med. 2020;43:635–40.
48. Pham, T.D. Classification of COVID-19 chest X-rays with deep learning: new models or fine tuning?. Health Inf Sci Syst 9, 2 (2021). https://doi.org/10.1007/s13755-020-00135-3
49. Верзунов, С. Н. Сравнение глубоких нейронных сетей на основе различных предварительно обученных CNN для диагностики COVID-19 по рентгеновским снимкам / С. Н. Верзунов, Х. А. Раимжанов // Проблемы автоматики и управления. – 2021. – № 1(40). – С. 12–25.

50. Daniel S. Kermany et all, Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning, Cell, Volume 172, Issue 5, 2018, Pages 1122-1131.e9, ISSN 0092-8674, https://doi.org/10.1016/j.cell.2018.02.010.
51. COVID-19 Image Data Collection: Prospective Predictions Are the FutureJoseph Paul Cohen and Paul Morrison and Lan Dao and Karsten Roth and Tim Q Duong and Marzyeh GhassemiarXiv:2006.11988, https://github.com/ieee8023/covid-chestxray-dataset, 2020
52. Rahman, T., Khandakar, A., Qiblawey, Y., Tahir, A., Kiranyaz, S., Kashem, S.B.A., Islam, M.T., Maadeed, S.A., Zughaier, S.M., Khan, M.S. and Chowdhury, M.E., 2020. Exploring the Effect of Image Enhancement Techniques on COVID-19 Detection using Chest X-ray Images
53. Tawsifur Rahman, Amith Khandakar, Muhammad A. Kadir, Khandaker R. Islam, Khandaker F. Islam, Zaid B. Mahbub, Mohamed Arselene Ayari, Muhammad E. H. Chowdhury. (2020) "Reliable Tuberculosis Detection using Chest X-ray with Deep Learning, Segmentation and Visualization". IEEE Access, Vol. 8, pp 191586 - 191601. DOI. 10.1109/ACCESS.2020.3031384
54. https://www.tensorflow.org (дата обращения 17.10.2021)
55. https://keras.io/examples/vision/integrated_gradients/ (дата обращения 17.10.2021)
56. https://arxiv.org/abs/1704.04861 (дата обращения: 23.03.2021)
57. https://arxiv.org/abs/1801.04381 (дата обращения: 23.03.2021)
58. Метелев Б., Кочеров А. Поиск повреждений трасс: кабелеискатель изобретен заново // ПЕРВАЯ МИЛЯ. – 2013. – №6 (39). – С. 68-73.
59. http://www.ki7.org/o-pribore/ (дата обращения 26.09.2018)
60. https://www.radiodetection.com/sites/default/files/RDMANGER-OPMAN-ENG_05.pdf?_buster=5Uh_IqE9 (дата обращения 26.09.2018)
61. https://geospb.ru/trassoiskateli-geomax/2366-programmnoe-obespechenie-logicat.html (дата обращения 26.09.2018)
62. http://www.lcard.ru/download/x502_low_level.pdf (дата обращения 26.09.2018)
63. https://bitbucket.org/lcard/x502api (дата обращения 26.09.2018)
64. Измайлов Д.Ю. Виртуальная измерительная лаборатория PowerGraph // ПиКАД. – 2007. – № 3. – С. 42–47.
65. Верзунов С.Н. Вейвлет-преобразование как инструмент анализа магнитовариационных данных // Проблемы автоматики и управления. – 2014. – №2 (27). – С. 52–61.
66. Астафьева Н.М. Вейвлет-анализ: основы теории и примеры применения // Успехи физических наук. – 1996. – №11. – Т.166. – С. 1145–1170.

67. Верзунов С.Н. Разработка программных средств для вейвлет-анализа одномерных временных рядов // Проблемы автоматики и управления. – 2014. – №2 (27). – С. 62–71.
68. Маккинли, Уэс Python и анализ данных / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 482 с.
69. Верзунов, С. Н. Разработка программного компонента трассоискателя на базе устройства сбора данных Л КАРД Е502 / С. Н. Верзунов, И. В. Бочкарев // Электротехнические системы и комплексы. – 2018. – № 2(39). – С. 42–48. – DOI 10.18503/2311-8318-2018-2(39)-42-48.
70. Моуэт Э. Использование Docker –М.: ДМК-Пресс, 2017. –354 с.
71. https://github.com/libusb/libusb (дата обращения 07.06.2019)
72. https://bitbucket.org/lcard/lqmeasstudio/src/free/ (23.10.2019)
73. Верзунов, С. Н. Разработка кроссплатформенного программного компонента трассоискателя / С. Н. Верзунов // Проблемы автоматики и управления. – 2020. – № 1(38). – С. 50–59. – DOI 10.5281/zenodo.3904110.
74. Крылов В.В., Крылов С.В. Большие данные и их приложения в электроэнергетике: монография / В.В. Крылов; М.: Lennex Corp. – Из-во Нобель Пресс. – 2014. – 168 с.
75. Верзунов, С. Н. Система искусственного интеллекта для онлайн-мониторинга подземных силовых кабельных линий на основе технологий глубокого обучения / С. Н. Верзунов // Проблемы автоматики и управления. – 2021. – № 3(42). – С. 83–94.
76. Верзунов С. Н., Токсаитов Д.А. Способы оптимизации расчёта 3D-графики для тайловых графических процессоров на примере визуализации моделирования гравитационного взаимодействия N-тел. // Проблемы автоматики и управления.– 2018. – №1 (34). – С. 26–36.
77. Navik. Microbenchmark Based Performance Evaluation of GPU Rendering./ Navik, Ankit P.// Emerging Research in Computing, Information, Communication and Applications. Springer, New Delhi. - 2015.
78. S. N. Verzynov, I. V. Bochkarev and V. R. Khramshin, "Development of Line Locator Software Component for Mobile Operating Systems," 2020 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russia, 2020, pp. 1-5, doi: 10.1109/ICIEAM48468.2020.9112019
79. Ma, Xiaohan Characterizing the performance and power consumption of 3D mobile games // Computer 46.4. – 2013. – С.76 –82.
80. Patel B, Bera P. Detection of power swing and fault during power swing using Lissajous figure // IEEE trans power deliv. – 2018. – N 33(6). – P. 3019–3027

81. K. R. K, Dash PK, A new real-time fast discrete S-transform for cross-differential protection of shunt-compensated power systems // IEEE Trans Power Deliv. – 2013. – N 28(1). – P. 402–410
82. Mohd Amiruddin AAA, Zabiri H, Taqvi SAA, Tufa LD Neural network applications in fault diagnosis and detection: an overview of implementations in engineering-related systems // Neural Comput Appl. – 2020. – N 32(2). – P. 447–472
83. Zhang F, Liu Q, Liu Y, Tong N, Chen S, Zhang C. Novel fault location method for power systems based on attention mechanism and double structure GRU neural network // IEEE Access. – 2020. – N 8. – P. 75237–75248
84. Swaminathan, R., Mishra, S., Routray, A. et al. A CNN-LSTM-based fault classifier and locator for underground cables // Neural Comput & Applic. 2021.
85. https://arxiv.org/abs/1412.6980v9 (дата обращения: 07.06.21)
86. Chen K, Hu J, He J Detection and classification of transmission line faults based on unsupervised feature learning and convolutional sparse autoencoder // IEEE Trans Smart Grid. - 2016. - N9(3). – P. 1748–1758

Верзунов Сергей Николаевич

МОНИТОРИНГ И ИДЕНТИФИКАЦИЯ ГЕОФИЗИЧЕСКИХ ПРОЦЕССОВ НА БАЗЕ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

Редактор Комарова Е.В.
Компьютерная верстка Першакова Е.Ю.
Подписано в печать 2.04.22.
Формат 60х881/16. Печать офсетная.
Объем 18.25 печ.л.
Тираж 100 экз.

ИЦ «Илим» при Президиуме НАН КР
720071, г. Бишкек, пр. Чуй, 265а