Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

МОДЕЛИРОВАНИЕ И ОПТИМАЛЬНОЕ ПРОЕКТИРОВАНИЕ ТЕХНИЧЕСКИХ СИСТЕМ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники для специальности 1-39 80 03 «Электронные системы и технологии» в качестве учебно-методического пособия

Минск БГУИР 2024

УДК 621.396.6(076.5)
ББК 32.844-02я73
М74

Авторы:
В. Ф. Алексеев, Д. В. Лихачевский, Г. А. Пискун, В. В. Шаталова

Рецензенты:
кафедра программного обеспечения информационных систем и технологий
Белорусского национального технического университета
(протокол № 5 от 06.12.2022);

директор государственного центра «Белмикроанализ»
ОАО «ИНТЕГРАЛ» − управляющая компания холдинга ОАО «ИНТЕГРАЛ»
кандидат физико-математических наук А. Н. Петлицкий

М74 **Моделирование** и оптимальное проектирование технических систем : учеб.-метод. пособие / В. Ф. Алексеев [и др.]. − Минск : БГУИР, 2024. – 100 с. : ил.
ISBN 978-985-543-720-9.

Приводится описание общих принципов проектирования технических систем. Рассмотрены основные этапы и методология моделирования и проектирования технических систем. Приведены практические примеры моделирования технических систем.

Предназначено для студентов второй ступени высшего образования технических университетов, может быть использовано аспирантами и инженерами, занимающимися вопросами моделирования и оптимального проектирования технических систем.

УДК 621.396.6(076.5)
ББК 32.844-02я73

ISBN 978-985-543-720-9

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Моделирование (в широком смысле) является основным методом исследований во всех областях знаний и научно обоснованным методом оценки характеристик сложных систем, используемых в различных сферах инженерной деятельности. Данный процесс является важнейшим и неотъемлемым этапом процедуры проектирования современных технических устройств и систем. Применительно к техническим системам (ТС) моделирование – это процесс замещения объекта исследования некоторой его моделью и проведение исследований на модели с целью получения необходимой информации об объекте или системе.

Создание новых конкурентоспособных изделий требует сокращения сроков и повышения качества проектно-конструкторских работ. Эти требования можно обеспечить, только применяя новые технологии проектирования, основанные на использовании методов математического моделирования и ресурсах вычислительной техники. Современные технологии основываются как на опыте инженерной практики, так и на научных теоретических и экспериментальных исследованиях. Поэтому инженер должен уметь практически решать задачи, требующие применения современных математических методов.

В настоящее время сложно представить себе специалиста не способного с помощью моделирования проверить обоснованность принятых технических решений.

Данное учебно-методическое пособие предназначено для магистрантов, обучающихся по специальности 1-39 80 03 «Электронные системы и технологии». Изложенный материал ориентирован на формирование у них знаний о современном состоянии и перспективах развития средств и методов моделирования и оптимального проектирования технических систем; умения ставить задачу по моделированию и оптимизации; выбирать структуру, а также алгоритмическую и программную реализацию имитационной модели сложного динамического объекта управления; получать математические модели объектов с элементами различной физической природы и оценивать их адекватность; умения ориентироваться в средствах и методах моделирования, выбрать и настроить современную среду автоматизированного моделирования.

Учебно-методическое пособие состоит из трех разделов.

В первом разделе рассматриваются общие принципы проектирования технических систем с точки зрения их моделирования и оптимизации параметров, перечисляются задачи, решаемые средствами моделирования, показывается роль и место моделирования в общей процедуре проектирования технических систем. Рассматриваются системы управления и их классификация. Поясняется, как теория систем управления сыграла жизненно важную роль в развитии техники и науки, а автоматическое управление стало неотъемлемой частью современных производственных и промышленных процессов. Рассматривается применение закона управления на примере сложной технической системы – управления полетом самолета.

Рассматриваются особенности проектирования конструкций технических систем на базе законов управления.

На примере высокопроизводительного робота описывается схема взаимодействия систем управления и объектов управления. Рассматривается принцип работы робота на примере устройства *HapticMaster*.

При рассмотрении вопроса проектирования больших систем на основе топологии с новых позиций рассматривается электромагнитная топология как современный метод изучения электромагнитной совместимости, являющийся одним из ключевых при разработке сложных электронных технических систем.

Во втором разделе рассматриваются основные этапы моделирования и проектирования технических систем. Современные технические системы используются в самых различных областях: радиолокации, радионавигации, системах связи, вычислительной технике, машиностроении, на транспорте, в научных исследованиях и т. д. В связи с этим возникает потребность в расширении функциональных возможностей ТС, а это, в свою очередь, предполагает серьезное улучшение показателей надежности, уменьшение массогабаритных признаков, технико-экономических показателей. Эти задачи могут быть успешно решены только на основе рассмотрения целого комплекса вопросов системо- и схемотехники, конструирования и технологии, производства и эксплуатации.

Показывается, что проектирование должно быть подчинено тому или иному критерию оптимизации (например, наибольшей мощности передачи сигнала, максимальному быстродействию, минимальным массогабаритным параметрам и т. д.) при ограниченных затратах или критерию быстрой окупаемости спроектированного ТС. Процесс проектирования представляется в виде иерархии решений и осуществляется системой проектирования, т. е. совокупностью взаимодействующих друг с другом проектировщиков (конструкторов, технологов, системотехников, схемотехников и эксплуатационников) и необходимых для проектирования технических и программных средств. Задачи разработки систем, как и прочие технические задачи, всегда тесно взаимосвязаны с другими процессами подготовки производства. Методика решения проектных задач характеризуется признаками синтеза структуры.

На примере проектирования беспилотных летательных аппаратов (БПЛА) показывается, что одной из основных проблем идентификации подвижных наземных целей с борта БПЛА в автоматическом режиме является идентификация нескольких целей, движущихся по разной траектории по неоднородному ландшафту либо в условиях сложной погодной обстановки, а также идентификация малых движущихся объектов. Современные беспилотные летательные аппараты являются сложными техническими устройствами, в работе которых заложено множество принципов. БПЛА могут управляться человеком с земли или автоматически следовать по заданному пути, в некоторых случаях даже возможна ситуация, в которой дрону приходится самостоятельно по радиосигналу или модели местности ориентироваться и выполнять поставленную задачу. Для

решения этих вопросов применяются различные методы и алгоритмы. Рассматриваются методы обработки видеопотока, поступающего из различных источников.

В третьем разделе излагаются некоторые сведения о методологии моделирования и проектирования технических систем. Особое внимание уделяется техническим системам слежения в видеопотоке. Область применения компьютерного зрения расширяется параллельно с тем, как оптимизируются алгоритмы, на основе которых решаются прикладные задачи компьютерного зрения. В то же время по мере повышения производительности компьютерного оборудования становятся актуальными алгоритмы, применение которых до недавнего времени было проблематичным. Именно поэтому в последнее время наиболее бурно развиваются те алгоритмы, что работают на основе машинного обучения: отчасти из-за скачка производительности доступного оборудования, а отчасти за счет появления больших массивов данных (в том числе маркированных).

Рассматривается цифровое моделирование систем автоматического управления имитационным методом в задачах САПР.

Авторы выражают особую благодарность магистру технических наук Старовойтову Алексею Игоревичу (научный руководитель – Алексеев Виктор Федорович, кандидат технических наук, доцент) за исследования методологии и принципов построения, обучения и регуляризации математических моделей на основе искусственных нейронных сетей, что позволило использовать отдельные положения в данном учебно-методическом пособии.

1 ОБЩИЕ ПРИНЦИПЫ ПРОЕКТИРОВАНИЯ ТЕХНИЧЕСКИХ СИСТЕМ

1.1 Системы управления и их классификация

Теория систем управления сыграла важную роль в развитии техники и науки. Автоматическое управление стало неотъемлемой частью современных производственных и промышленных процессов. Например, с его помощью осуществляется числовое управление станками в обрабатывающей промышленности, контроль давления, температуры, влажности, вязкости и текучести в перерабатывающей промышленности.

Когда ряд элементов или компонентов соединен в последовательность для выполнения определенной функции, образованная таким образом группа называется системой. Выходная величина называется регулируемой переменной, или реакцией, а входная величина называется командным сигналом, или возбуждением.

Система управления – это тип системы, которая управляет выходом для обеспечения желаемого ответа. Можно также сказать, что это группа электронных или механических устройств, которые используют контуры управления для управления другими системами или устройствами. Системы управления автоматизированы с помощью компьютеров. На рисунке 1.1 представлена простая схема системы управления.

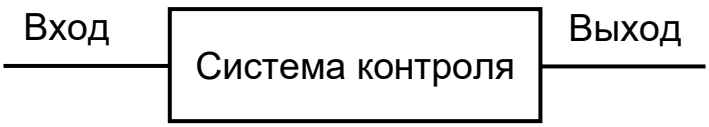

Рисунок 1.1 – Схема системы управления

На рисунке 1.1 система управления представлена одним блоком, поскольку выход управляется изменением входа.

Системы управления могут быть классифицированы несколькими способами. Выделим некоторые популярные классификации:

– в зависимости от методов анализа и проектирования система может быть линейной или нелинейной;

– в зависимости от типа сигналов система может быть изменяющейся во времени системой данных, неизменяющейся во времени непрерывной системой данных, дискретной системой данных, модулированной или немодулированной системой управления и т. д.;

– в зависимости от типа компонента системы система может быть электромеханической, биологической, гидравлической, тепловой или пневматической системой управления и т. д.;

– в зависимости от основного назначения система может управлять положением, скоростью и т. д.

Системы управления бывают двух типов: система разомкнутого цикла и замкнутая система.

Система управления без обратной связи

Любая физическая система, которая автоматически не корректирует изменение своего выходного сигнала, называется системой с разомкнутым контуром. Система управления, в которой выходная величина не влияет на входную величину, называется системой управления с разомкнутым (открытым) контуром (рисунок 1.2). Это означает, что выходной сигнал не является обратной связью.

Рисунок 1.2 – Система управления без обратной связи

В системе управления с разомкнутым контуром выходной сигнал может изменяться за счет изменения входного сигнала. Но из-за внешних возмущений выходной сигнал системы также может измениться. В системах с разомкнутым контуром изменения на выходе корректируются путем изменения входа вручную.

Замкнутая система управления

Системы управления для поддержания желаемого значения выхода, в которых выход влияет на входное количество, называются системами с замкнутым контуром.

Систему с разомкнутым контуром можно изменить как систему с замкнутым контуром, обеспечив обратную связь. Обеспечение обратной связи автоматически корректирует изменения на выходе из-за возмущений. Следовательно, система с замкнутым контуром также является системой автоматического управления. Общая схема автоматического управления показана на рисунке 1.3.

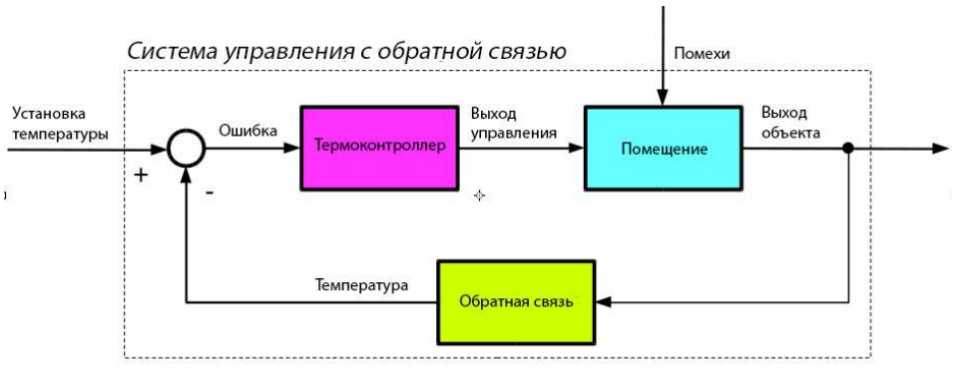

Рисунок 1.3 – Система управления с обратной связью

Из рисунка 1.3 видно, что система состоит из детектора ошибок, контроллера, объекта (система с разомкнутым контуром) и элементов цепи обратной связи.

Опорный сигнал (или входной сигнал) соответствует требуемому выходному сигналу. Элементы тракта обратной связи производят выборку выходного сигнала и преобразуют его в тот же тип, что и опорный сигнал. Сигнал обратной связи пропорционален выходному сигналу и подается на детектор ошибок. Сигнал ошибки, генерируемый детектором ошибок, представляет собой разницу между эталонным сигналом и сигналом обратной связи. Контроллер изменяет и усиливает сигнал ошибки, чтобы обеспечить лучшее управляющее действие. Измененный сигнал ошибки подается на установку для корректировки его выходных данных.

Преимущества системы управления с разомкнутым контуром: простота и экономичность, легкость построения, большая стабильность.

Недостатки открытых систем: неточность и ненадежность; изменения из-за внешних помех на выходе не корректируются автоматически.

Преимущества замкнутых систем: точность при наличии нелинейностей, чувствительность систем может быть уменьшена для большей стабильности, меньшая подверженность влиянию шума.

Недостатки замкнутых систем: сложность и дороговизна, обратная связь в замкнутой системе может привести к колебательному отклику, обратная связь снижает общий коэффициент усиления системы, стабильность является серьезной проблемой в системе с замкнутым контуром, требуется больше усилий для разработки стабильной системы с замкнутым контуром.

Примеры современных систем управления

Чтобы понять важное различие между принципами управления и их воплощением в реальной машине или системе, полезным будет рассмотреть следующие распространенные примеры управления.

Машины, которые не могут функционировать без управления (обратной связи)

Многие базовые устройства изготавливаются таким образом, чтобы их поведение можно было изменить с помощью некоторого внешнего управления. Как правило, такой же эффект не может быть достигнут (на практике, а иногда и в теории) путем какой-либо внутренней модификации характеристик устройства.

Например, транзисторные усилители вносят недопустимые искажения в звуковые системы, когда используются отдельно, но должным образом модифицированные системой управления с обратной связью они могут достичь любой желаемой степени точности воспроизведения.

Другой пример связан с полетом и двигателями. Первые испытатели потерпели неудачу не из-за незнания законов аэродинамики, а из-за того, что они не понимали важности управления и не знали основных принципов стабилизации неустойчивого по своей природе устройства посредством управления. Реактивные самолеты не могут эксплуатироваться без автоматического управления,

помогающего пилоту. Управление не менее важно и для вертолетов. Точность инерциального навигационного оборудования не может быть улучшена до бесконечности из-за основных механических ограничений, но эти ограничения могут быть уменьшены на несколько порядков с помощью управляемой компьютером статистической фильтрации, которая является вариантом управления с обратной связью.

Управление машинами

Машина в процессе выполнения задачи может управляться человеком (ручное управление) или подключаться непосредственно к измерительному прибору (автоматическое управление).

Например, термостат может использоваться для включения или выключения холодильника, духовки, кондиционера или системы отопления. Настройка диафрагмы камеры и правильная экспозиция цветных снимков могут регулироваться автоматически путем подключения фотоэлемента непосредственно к фотокамере. Родственными примерами являются дистанционное управление положением (сервомеханизмы) и регулирование скорости двигателей (регулятор). Можно отметить, что в таких случаях машина функционирует сама по себе.

Управление большими системами

Более продвинутые приложения управления находят применение в больших и сложных системах, само существование которых зависит от скоординированной работы с использованием множества отдельных устройств (обычно управляемых компьютером).

Примерами могут служить запуск космического корабля, круглосуточная работа электростанции, нефтеперерабатывающего или химического завода, управление воздушным движением вблизи крупного аэропорта. Существенным аспектом этих систем является то, что участие человека в задаче управления, хотя и возможно теоретически, является совершенно непрактичным – именно возможность применения автоматического управления и породила эти системы.

Биоконтроль

Развитие технологий (синтетическая биология) и более глубокое понимание процессов биологии (естественные технологии) открыли перспективы для их объединения. Устройства могут заменить некоторые естественные функции.

Примерами являются искусственное сердце или почки, протезы, управляемые нервами, и управление функциями мозга с помощью внешних электрических раздражителей. Хотя подобные устройства определенно уже перестали быть чем-то из области научной фантастики, прогресс в их использовании остается медленным не только из-за потребности в передовых технологиях, но и из-за недостатка фундаментальных знаний о деталях принципов управления, используемых в биологическом мире.

Роботы

Задачей науки управления на самом продвинутом уровне является создание роботов. Робот – это собирательный термин для устройств, демонстрирующих запрограммированное поведение под общим надзором (но без непосредственной помощи) людей.

Узкоспециализированные промышленные производственные роботы уже достаточно широко распространены, но для реальных прорывов потребуются фундаментальные научные достижения в отношении проблем, связанных с распознаванием образов и мыслительными процессами.

1.2 Законы управления

Законы управления – математическая форма преобразований воздействий, возмущений, эффектов обратных связей, определяющих управляющие воздействия.

Рассмотрим применение законов управления на примере сложной технической системы – управления полетом самолета.

В конструкции современных крупных коммерческих транспортных самолетов используются сложные бортовые компьютеры, которые помогают управлению и обеспечивают безопасность полета. Они регулируются вычислительными законами, которые назначают режимы управления во время полета.

Для самолетов с дистанционным управлением требуются компьютерные программы, которые способны самостоятельно определять режим работы (вычислительный закон) самолета. Потеря управления может быть вызвана отказом вычислительного устройства (к примеру, бортового компьютера), устройства предоставления информации (например, инерциального эталонного блока данных о воздухе (ADIRU)) или отказом нескольких систем (двойной гидравлический отказ, двойной отказ двигателя и т. д.). Электронные системы управления полетом (EFCS) обеспечивают повышенную защиту самолета от перегрузок или стабилизацию для снижения эффекта турбулентности и обеспечения демпфирования рыскания.

Производители самолетов производят коммерческие пассажирские суда с бортовыми компьютерами, которые могут работать в разных режимах управления полетом. В самолетах нового поколения используются более легкие электронные системы, обеспечивающие безопасность и производительность при одновременном снижении веса самолета.

Особенности проектирования самолетов на базе законов управления

В конструкциях самолетов более старых версий управление осуществлялось с помощью штурвала пилота, педалей руля направления, триммера или дроссельных заслонок, которые механически перемещали тросы, шкивы или гидравлические сервоклапаны (это гидравлическое устройство, которое преобразует малый входной электрический сигнал в большой входной, а именно – гидравлический), которые, в свою очередь, перемещали поверхности управления или изменяли настройки двигателя. Во многих новых самолетах механическое управление заменено дистанционными электросистемами. Такие самолеты оснащены компьютерами, которые с помощью электронных сигналов регулируют работу поверхностей управления и двигателей, информируют пилота о неисправностях и опасности и предоставляют информацию о деталях полета.

В старых моделях самолетов механическим органам управления противодействуют силы, воздействующие на поверхности управления и являющиеся причиной сваливания, превышения допустимой скорости или чрезмерного крена на высоких скоростях. Электронные системы управления регулируют движения поверхности управления, чтобы гарантировать, что пределы прочности самолета не будут превышены.

Конструкторы самолетов создали набор режимов управления полетом, которые включают резервную электронику для защиты от сбоев системы. Отказы могут возникать по отдельности или в сочетании, приводя к неработоспособности систем. Пилоты должны иметь возможность управлять летательным аппаратом, когда какая-либо часть или вся система выходит из строя. К примеру, логика закона управления *Airbus* допускает постепенное отключение автоматических средств защиты до тех пор, пока множественные отказы не приведут к переводу судна в незащищенный прямой режим работы. Также доступны ограниченные режимы механического управления, позволяющие контролировать летательный аппарат во время процесса сброса после временного отключения всех компьютеров управления полетом. Прямой режим *Boeing* также снимает многие вычислительные ограничения.

Другая функция законов управления полетом заключается в оценке характеристик самолета в различных условиях, таких как взлет, посадка, нормальный крейсерский полет, или в ситуациях, когда бортовые компьютеры частично или полностью выходят из строя. Разработчики предусмотрели возможность обхода компьютерных ограничений и работы резервных систем без участия компьютеров.

1.3 Схема взаимодействия систем управления и объектов управления

Рассмотрим технические принципы взаимодействия систем управления и объектов управления на примере высокопроизводительного робота с принудительным управлением. Такой робот, с одной стороны, существенно отличается от большинства промышленных роботов, а с другой – не похож на большинство тактильных интерфейсов. В рассуждениях будет использоваться парадигма контроля доступа, которая делает возможным сочетание высокой жесткости сустава с высокой чувствительностью к приложенной силе.

Возможность управления приложением силы является полезной функцией для робота, поскольку с ее помощью можно контролировать воздействие. Гибридное управление силой/положением также позволяет регулировать траекторию приложения сил. Эти типы управления позволяют выполнять задачи, требующие адаптации к окружающей среде. К сожалению, стабильное управление контактными силами роботов до сих пор остается нетривиальной задачей. Промышленные роботы с активным контролем силы воздействия все еще практически не используются.

Роботы с принудительным управлением чаще встречаются в экспериментальных установках. Преимущество обратной связи в задачах телеуправления заключается в том, что силы ведомого объекта могут ощущаться ведущим, что повышает точность и производительность оператора.

Парадигма управления доступом такова: пользователь прикладывает силу к тактильному устройству, и устройство реагирует соответствующим смещением. С точки зрения тактильного устройства парадигма такова: сила, направленная внутрь, инициирует смещение наружу. Значительная свобода в управлении механической конструкцией устройства, достигается за счет того, что прерывистое скольжение и инерция наконечника могут быть уменьшены с помощью внутреннего контура сервопривода. В результате механизм достаточно прочен и способен демонстрировать высокую стабильность и мощность.

Из-за наличия трения в суставах для управления силой промышленных роботов обычно требуется контроль доступа. Управление доступом используется в авиаиндустрии в течение многих лет, но из-за своей сложности и цены редко используется в тактильных интерфейсах. Устройства управления доступом способны обеспечивать очень высокую стабильность, близкое к нулю трение и практически нулевой вес концевого эффектора, что создает ощущение свободного движения. Данные устройства подходят для больших производств, приложений «ведущий – ведомый» и для работы со сложными концевыми эффекторами со многими степенями свободы. Кроме того, они регистрируют возникающие контактные силы, однако часто неспособны передавать очень малую массу, а это означает, что всегда будет ощущаться инерция.

Рассмотрим принципы работы робота на примере устройства *HapticMaster*.

Алгоритм управления

HapticMaster использует алгоритм контроля доступа. Измеряется сила взаимодействия, по которой виртуальная модель вычисляет положение, скорость и ускорение (*PVA*), которые будут приложены к объекту, которого касается робот, в результате действия этой силы. Виртуальная модель определяет свойства среды, в которой расположен объект (например, гравитацию, трение окружающей среды и т. д.), и свойства объекта (например, массу, жесткость, демпфирование, трение и др.). Виртуальная модель обычно учитывает лишь массу больше нуля, для избежания бесконечных ускорений и нестабильности системы. *PVA*-вектор служит опорным сигналом для робота, реализованным сервоконтуром сервоуправления (пропорционально-интегрально-дифференциальный регулятор – *PID*). При правильных настройках усиления обратной связи этот контур управления уменьшит реальную массу на конце манипулятора в шесть раз и снизит внутреннее трение. Так, если масса составляет 15 кг, то на концевом эффекторе оператор ощутит всего 2,5 кг. Алгоритм управления допуском показан на рисунке 1.4. Сила взаимодействия между человеком-оператором и роботом измеряется датчиком силы. Затем модель вычисляет *PVA*, которую виртуальный

объект получит в результате действия этой силы. *PID*-контур сервопривода с высокой пропускной способностью управляет роботом в соответствии с заданным вектором *PVA*.

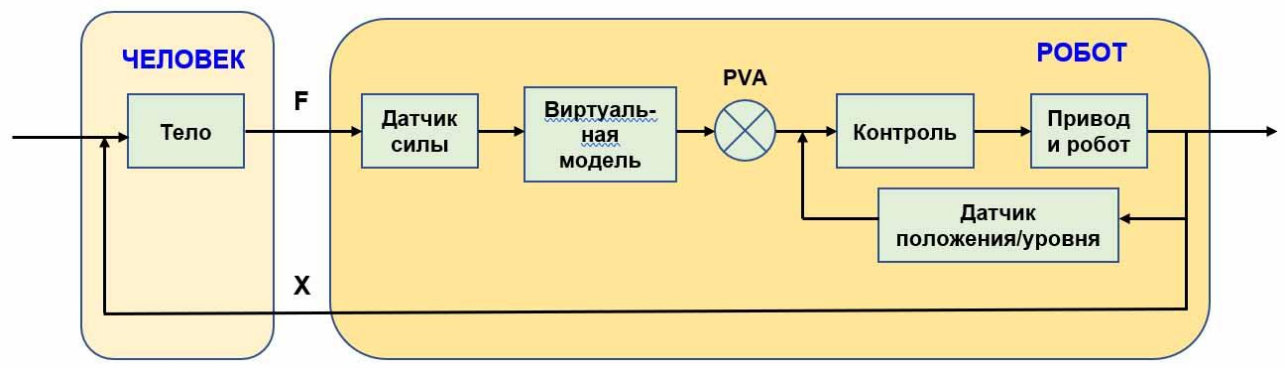

Рисунок 1.4 – Схема управления алгоритмом управления робота

Аппаратное обеспечение

Аппаратное обеспечение состоит из двух основных функциональных компонентов: манипулятора робота и блока управления (рисунок 1.5).

Рисунок 1.5 – Изображение системы *HapticMaster*

Манипулятор робота (слева) представляет собой дисплей силы, тогда как блок управления (справа) содержит необходимую электронику (усилители, реле

безопасности и тактильный сервер). Механизм манипулятора робота предполагает нулевой люфт и минимальное ощущение веса. Нулевой люфт является обязательным требованием, поскольку человеческие органы чувств чрезвычайно чувствительны к вибрационным эффектам.

Тактильные ощущения человека имеют пространственное разрешение до 10–100 мк для вибрации. Минимальный вес является требованием безопасности. Скорость и масса манипулятора робота определяют его энергоемкость при взаимодействии с человеком-оператором. Скорость установлена на значение нормального движения руки человека (1,6 м/с), а легкая конструкция из алюминиевых труб сводит к минимуму массу манипулятора робота. Кинематическая цепочка снизу вверх позволяет вращать основание, двигать плечо вверх/вниз и к входу/выходу, что дает три степени свободы на конце эффектора. Робот достаточно безопасен, хотя существует вероятность лобового, бокового и восходящего столкновения. Однако даже наиболее опасные движения содержат наименьшую энергию и имеют относительно малый момент инерции. Таким образом создается объемное рабочее пространство, достаточное для того, чтобы безопасно вместить большое количество людей.

Чувствительный тензометрический датчик силы (рисунок 1.6) расположен сразу после концевого зажима на конце манипулятора робота. Тензометрический датчик измеряет силу, прилагаемую оператором, как можно ближе к руке человека, чтобы избежать искажения сигнала и оптимизировать работу системы.

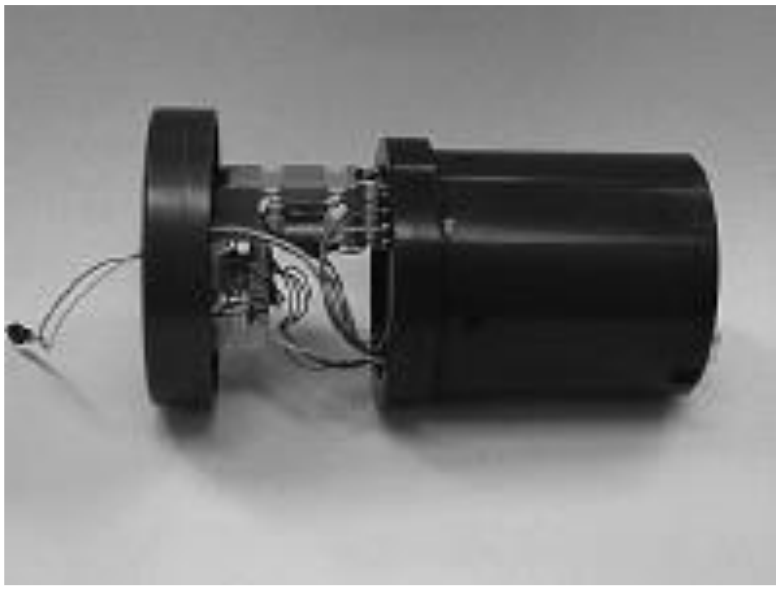

Рисунок 1.6 – Тензометрический датчик силы

Сменные концевые зажимы могут быть установлены на датчик силы в соответствии с целью применения (например, это могут быть медицинские инструменты или отвертки). Контактные силы инструмента можно моделировать в виртуальных настройках. На рисунке 1.7 к концевому эффектору прикреплена отвертка.

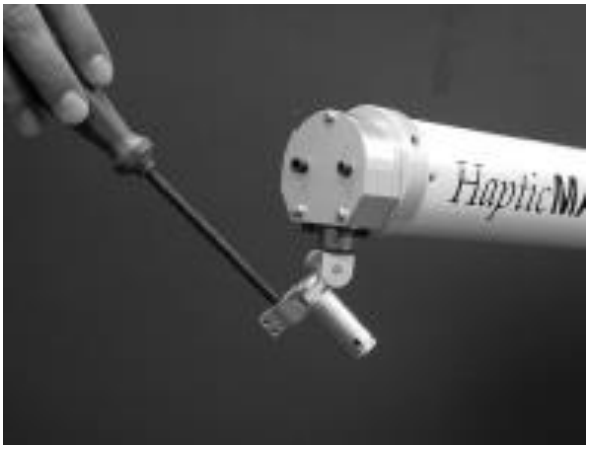

Рисунок 1.7 – Концевой эффектор, на котором закреплена отвертка

Другим примером является концевой эффектор, используемый для робототерапии, в который может быть помещена рука человека. Он показан на рисунке 1.8.

Рисунок 1.8 – Концевой эффектор с двумя пассивными и одной активной степенями свободы

Карданный концевой эффектор имеет одну активную и две пассивные степени свободы. При такой установке робототерапия проводится для пациентов, перенесших инсульт, в виртуальных условиях. Преимущество робототерапии в

том, что уровень помощи можно точно адаптировать к конкретным потребностям пациента.

1.4 Проектирование больших систем на основе топологии

Электромагнитная топология – современный метод изучения электромагнитной совместимости

Очевидно, что проблема электромагнитной совместимости (ЭМС) должна быть рассмотрена на начальном этапе проектирования. Это лучший способ оптимизации размещения экранов во всей системе. Иначе, принимая решение об экранировании всех стенок и систематическом размещении защитных устройств во входных цепях оборудования, получаем резкое увеличение веса системы. Кроме того, такое решение делает систему более сложной, что сказывается на времени жизни устройства.

В начале электромагнитная топология (ЭМТ) не рассматривалась в роли метода, направленного на изучение электромагнитной интерференции. Она появилась в 1980 году как часть разработки систем, защищенных от воздействия электромагнитного импульса (ЭМИ). Теория основывалась на геометрическом размещении блоков устройства в пределах друг друга (рисунок 1.9).

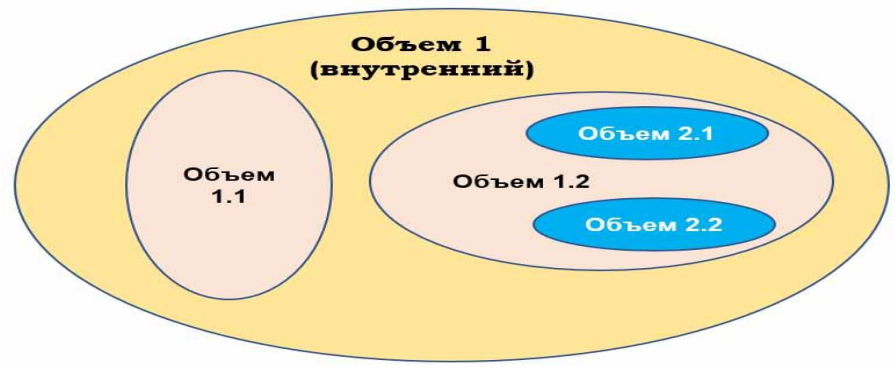

Рисунок 1.9 – Пример топологического разбиения (топологическая диаграмма)

Появление этой теории стало возможным благодаря принципу «хорошей аппроксимации экрана», который предполагает, что внутренний блок не может взаимодействовать с внешним блоком. Но на практике такое практически невозможно, так как проводники являются неизбежной частью системы, и делают блоки электромагнитно зависимыми.

Электромагнитная топология против классических методов ЭМС

ЭМТ представляет собой оригинальный метод для эффективной интеграции и контроля ЭМС в больших системах. Для этого она включает в себя множество разделов, предназначенных для прогнозирования внутренних помех. Дело в том, что разные методы (расчетный и экспериментальный) не могут дать существенных результатов в сложных системах. Поэтому можно сказать, что ЭМТ

является альтернативой двум техническим методам, которые предлагают взаимодействие с проблемой ЭМС: экспериментам на крупномасштабных моделях и трехмерному моделированию.

Еще совсем недавно при разработке электромагнитных систем эксперименты на физических моделях использовались повсеместно. Если модели создавались корректно, то они давали очень полезные результаты для решения проблемы обеспечения оптимальной защиты от ЭМС. Однако эти модели не могли помочь в решении проблем, появляющихся при дальнейшей разработке объекта. В связи с тем, что программы испытаний становились все более и более дорогими, большинство промышленных компаний вынуждены были уменьшать количество электромагнитных экспериментов и, в большинстве случаев, тесты на электромагнитную совместимость проводились уже на прототипах, много позже начальной фазы проектирования.

Применение компьютерного проектирования открыло новые перспективы трехмерного моделирования ЭМС, основанных на расчетах уравнений Максвелла. На низких частотах (до 500 МГц) этот метод хорошо адаптируется для использования в так называемых «внешних» задачах. Если предположить, что источник электромагнитного излучения находится вне системы, то можно с большой точностью рассчитать поверхностное распределение токов и интенсивность поля. Однако решение «внутренних» задач требует очень точного описания геометрии. Но несмотря на то, что многие трехмерные вычислительные методы в состоянии учитывать единственный проводник, они не подходят для многожильных кабелей, идущих внутри структуры. Для решения как «внешних», так и «внутренних» задач необходимо учитывать геометрию объекта в масштабе, соизмеримом с толщиной кабеля.

Специфика электромагнитной топологии

Одним из ограничений экспериментального и трехмерного математических методов является то, что они пытаются решить задачу только одним способом. ЭМТ раскладывает данную задачу на элементарные задачи, ограниченные меньшими по отношению к глобальной проблеме объемами. Далее к каждой отдельной задаче можно применить наиболее подходящий из методов.

Несмотря на конечную цель, которая состоит в решении всей задачи, целью ЭМТ является возможность модульной обработки. Очень важным является то, что все знания, приобретенные при использовании ЭМТ, можно применять в канонических задачах, принимая во внимание электромагнитную связь в сложных системах. Кроме того, модульность интересует нас не только с точки зрения совместимости в крупномасштабных задачах. Особенности модульности многочисленны и могут быть рассмотрены с трех сторон.

1 *Гибридизация методов.* Благодаря разделению задачи на элементарные модули каждый этап может быть рассмотрен с помощью специфического метода. Например, можно исследовать любой небольшой модуль при помощи трехмерного кода. В связи с тем, что размеры геометрической проблемы стали меньше исходной задачи, можно получить прирост производительности компьютеров (уменьшается время расчетов или появляется возможность более точно

описать внутреннюю геометрию). Появляется также возможность моделировать блок экспериментально (это дает возможность, например, полностью охарактеризовать цепь с помощью измерительной аппаратуры).

2 *Параллельный подход.* Каждый модуль может быть изучен отдельно от соседнего. Это дает возможность модифицировать характеристики в одном модуле, не рассчитывая при этом другие (могут изменяться как электрические, так и геометрические характеристики модулей). Кроме того решения можно поочередно подставлять в каждый модуль, что является эффективным способом определения, какое из решений является наиболее подходящим для этой системы. С другой стороны в глобальных задачах параметрический подход более сложен как в построении математической модели, так и в экспериментальном подходе. Вот почему оба технических метода применяются на наиболее подходящей конфигурации взаимодействия. Однако главная проблемой является ответ на вопрос какое решение будет наилучшим для правильной работы системы.

3 *Использование «топологической» базы данных.* Топологический метод позволяет характеризовать конструктивные компоненты модуля независимо от структуры. Поэтому для таких элементов, как кабели, электрические цепи, разъемы, которые можно охарактеризовать вне структуры (в лаборатории) и чьи характеристики могут быть использованы в различных задачах, создаются специальные базы данных.

Топологический подход к крупномасштабным структурам обладает тремя преимуществами:

– решает проблему электромагнитного взаимодействия;

– позволяет исследовать информацию в ее простейшем и наиболее доступном виде;

– предоставляет возможность использования полученных результаты в других конфигурациях.

Описание топологического подхода

1 *Разделение на частные задачи*

Эта операция носит интуитивный характер. Границы модулей очень часто ограничиваются физическими поверхностями (стенками, коробками, кабелями). Их можно разделить на два типа:

– поверхности, которые являются экранами и ослабляют сигнал, идущий с двух сторон. Это корпуса самолетов или обмотки кабелей. Они называются реальными поверхностями и содержат в себе один объем, называемый реальным объемом. Основной целью использования таких поверхностей является то, что они позволяют внешние воздействия и интерференцию, происходящую внутри, рассматривать отдельно. Это следует из принципа «хорошей аппроксимации экрана» – одного из базисов теории;

– поверхности, которые не создают какого-либо экранирующего эффекта, но которые разделяют два физических модуля. Это стенки с апертурными или кабельными пересечениями, где сигналы внутри отдельных блоков зависят друг от друга. Такие поверхности называются элементарными поверхностями, а

блоки внутри – элементарными блоками. Дело в том, что эти поверхности являются границей мысленного разделения и местом, где исследуемые данные легко доступны. На топологической диаграмме (рисунок 1.10) изображены оба типа поверхностей и объемов. Этот пример с малым числом реальных и множеством элементарных поверхностей является более соответствующим реальной топологии системы, чем изображенный на рисунке 1.9.

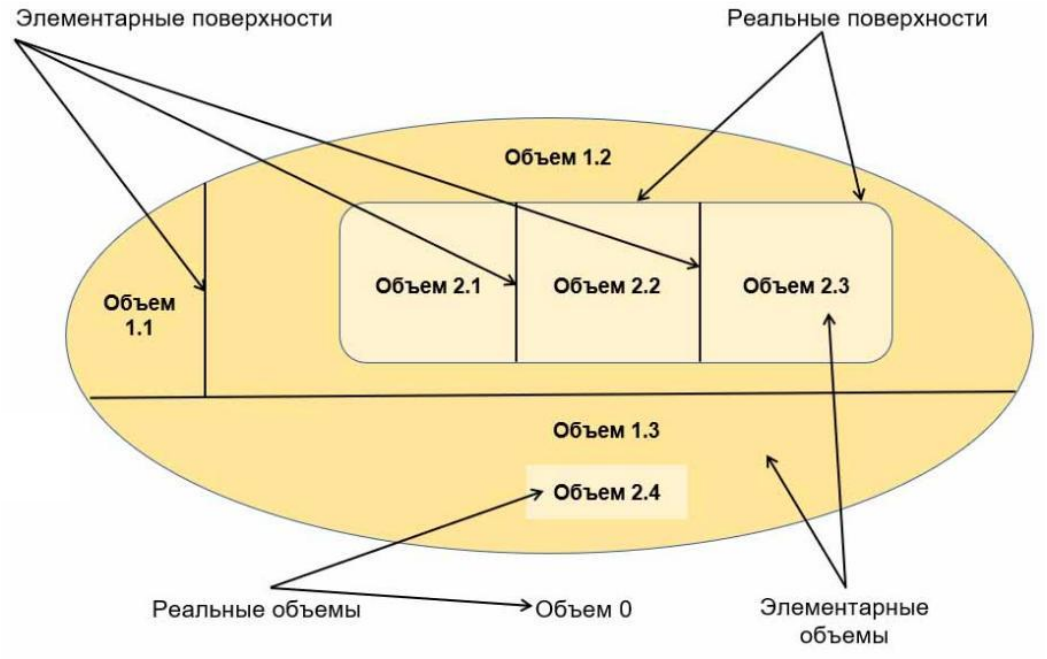

Рисунок 1.10 – Терминология поверхностей и объемов в электромагнитной топологии

2 *Описание глобальной задачи с помощью топологической сети*

После разделения на блоки электромагнитная топология предлагает сетевой метод для помощи в решении глобальной задачи благодаря определению характеристик рассеивания в каждой элементарной задаче.

Сеть представляет собой сумму связей, соединяющих различные блоки, из которых состоит исходное устройство. В сети каждый блок (элементарный или реальный) описывается в виде вершин, каждая вершина связана с другой при помощи ребер. Связь источников в сети выражается в виде эквивалентных генераторов, замененных ребрами. Вершины могут соответствовать элементарным элементам, таким как цепи, но в основном соответствуют целым блокам. Блок в свою очередь может быть представлен как сеть. Такая вершина называются эквивалентной. Со своих сторон ребра могут быть заменены эквивалентными генераторами. Частично они создают «межсоединение» между «локальными сетями». Очень важно то, что ребра подразумевают распространение сигнала между двумя вершинами. Поэтому они отлично приспособлены для описания электромагнитной связи по кабелям.

2 ОСНОВНЫЕ ЭТАПЫ МОДЕЛИРОВАНИЯ И ПРОЕКТИРОВАНИЯ ТЕХНИЧЕСКИХ СИСТЕМ

2.1 Сущность и этапы процесса проектирования

Современные технические системы (ТС) используются в самых различных областях: радиолокации, радионавигации, системах связи, вычислительной технике, в машиностроении, на транспорте, в научных исследованиях и т. д. В связи с этим возникает потребность в расширении функциональных возможностей ТС, а это, в свою очередь, предполагает серьезное улучшение показателей надежности, уменьшение массогабаритных признаков и увеличение технико-экономических показателей. Эти задачи могут быть успешно решены только на основе рассмотрения целого комплекса вопросов системо- и схемотехники, конструирования и технологии, производства и эксплуатации.

Многим исследователям свойственна одна общая черта: они обнаруживают стремление выявить сущность проектирования и изложить ее в виде некоего стандартного метода, дать своего рода рецепт, на который можно было бы положиться во всех ситуациях. Вот лишь некоторые определения и формулировки, описывающие процесс проектирования:

– отыскание существенных компонентов какой-либо физической структуры [1];

– целенаправленная деятельность по решению задач [2];

– принятие решений в условиях неопределенности с тяжелыми последствиями в случае ошибки [3];

– использование научных принципов, технической информации и воображения для определения механической структуры машины или системы, предназначенной для выполнения заранее заданных функций с наибольшей экономичностью и эффективностью [4];

– приведение изделия в соответствие с обстановкой при максимальном учете всех требований [5] и др.

Приведенные цитаты столь различны, что начинаешь задумываться над тем, что же на самом деле есть процесс проектирования и в чем его сущность. Создается впечатление, что имеется столько же различных процессов проектирования, сколько существует авторов, которые описывают эти процессы. Из этих

---

[1] Alexsander, C. The determination of components for an Indian village / C. Alexsander. – Oxford : Pergamon, 1963 ; New York : Macmmillan, 1963.

[2] Archer, L. B. Systematic method for designers / L. B. Archer. – London : Council of Industrial Design, 1965.

[3] Asimow, M. Introduction to design / M. Asimow. – New York : Prentice-Hall, 1962.

[4] Fielden, G. B. R. The Fielden Report. Engineering Design / G. B. R. Fielden. – London : H. M. Stat. Office, 1963.

[5] Page J. K. Contribution to building for people / J. K. Page. – London : Ministry of Public Building and Works, 1966.

цитат ясно, что в зависимости от обстоятельств характер процесса проектирования может меняться в очень широких пределах.

По мнению авторов данного пособия, ***процесс проектирования*** *– это осознанная мыслительная и творческая деятельность разработчика, базирующаяся на синтезе и анализе принимаемых решений и направленная на получение законченного изделия в виде конструкторской документации, макета или образца, изготовленных с помощью современных информационных, программных и технологических средств* [6].

Явное разнообразие определений, наблюдающееся в литературе о проектировании, может служить ключом к пониманию ситуации. Возможно, сознательно уклонившись от ссылок на чертежи и декларации привычных взглядов на процесс проектирования, теоретики нащупали то существенное, что позволит преодолеть недостатки традиционных методов проектирования. Это существенное как раз и состоит в *разнообразии,* причем в разнообразии столь широком, что оно выходит за пределы опыта и знаний любого отдельно взятого разработчика, любой конкретной проектной специальности и, по сути дела, любого отдельно взятого теоретика проектирования.

И все же всем приведенным выше определениям свойственна одна общая черта: они говорят не о результатах проектирования, а о его составных частях. Чтобы найти более надежную основу для рассуждения, попытаемся рассмотреть, в чем же заключается сущность процесса проектирования.

Проектирование предшествует созданию любого объекта. При этом под ***объектом проектирования*** *будем понимать любой объект, создаваемый человеком, но еще не существующий в действительности* (например, радиоэлектронное устройство, вычислительный комплекс или система), *и предназначенный для удовлетворения определенной потребности* [6].

Цель процесса проектирования состоит прежде всего в том, чтобы на основе априорной (исходной) информации и апостериорной (дополнительной) информации, получаемой в процессе проектирования, разработать техническую документацию, требуемую для изготовления технической системы.

Соответственно процесс проектирования содержит творческую, техническую, производственную и корректировочную части, соотношение между которыми на разных стадиях будет различным. Проектирование, по существу, представляет собой процесс с обратной связью (рисунок 2.1).

Техническое задание (ТЗ) формирует требования к проектируемому объекту. Результаты проектирования сравниваются с требованиями ТЗ и, если они не совпадают, цикл проектирования повторяется вновь до тех пор, пока ошибка (отклонение от заданных технических требований) не окажется в допустимых пределах.

Проектирование должно быть подчинено тому или иному критерию оптимизации, например, наибольшей мощности передачи сигнала, максимальному быстродействию, минимальным массогабаритным параметрам при ограниченных затратах, или критерию скорейшей окупаемости.

---

[6] Данное определение введено впервые в источнике [1]. – *Примеч. авт.*

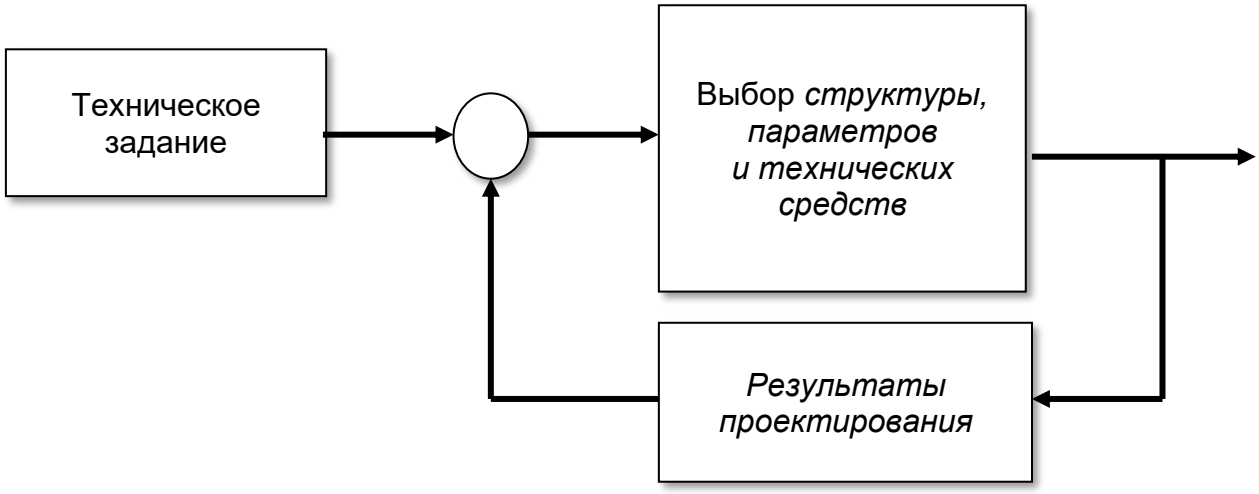

Рисунок 2.1 – Проектирование как процесс с обратной связью

Процесс проектирования можно представить в виде иерархии решений (рисунок 2.2).

Из рисунка видно, что при проектировании необходимо решить проблему *X*. Для ее решения возможны различные варианты: Y_1, Y_2, Y_3 (для упрощения будем рассматривать, что возможны только три варианта, а не *n* вариантов). Каждому из указанных вариантов соответствует также несколько подпроблем: Z_{11}, Z_{12}, Z_{21}, Z_{22}, Z_{23}, Z_{31}, Z_{32}, Z_{33} и т. д. Принятие варианта Y_1 требует решения подпроблем Z_{11}, Z_{12}, а принятие варианта Y_2 – подпроблем Z_{21}, Z_{22}, Z_{23} и т. д.

Иногда может оказаться возможным получение приемлемого решения для всех подпроблем, и в этом случае разработчик должен выбрать вариант, который наилучшим образом удовлетворяет целям проектирования и оптимизации.

Техническое задание является основным исходным документом для разработки продукции и технической документации на нее. ТЗ разрабатывают по ГОСТ 15.016–2016, ГОСТ Р 15.201–2000.

Основное требование при проектировании ТС состоит в том, чтобы создаваемое устройство было эффективнее своего аналога, т. е. превосходило его по функциональным свойствам, качеству функционирования, степени миниатюризации и технико-экономической целесообразности (рисунок 2.3).

Процесс разработки осуществляется системой проектирования, т. е. совокупностью взаимодействующих друг с другом проектировщиков (конструкторов, технологов, системотехников, схемотехников и эксплуатационников) и необходимых для проектирования технических и программных средств.

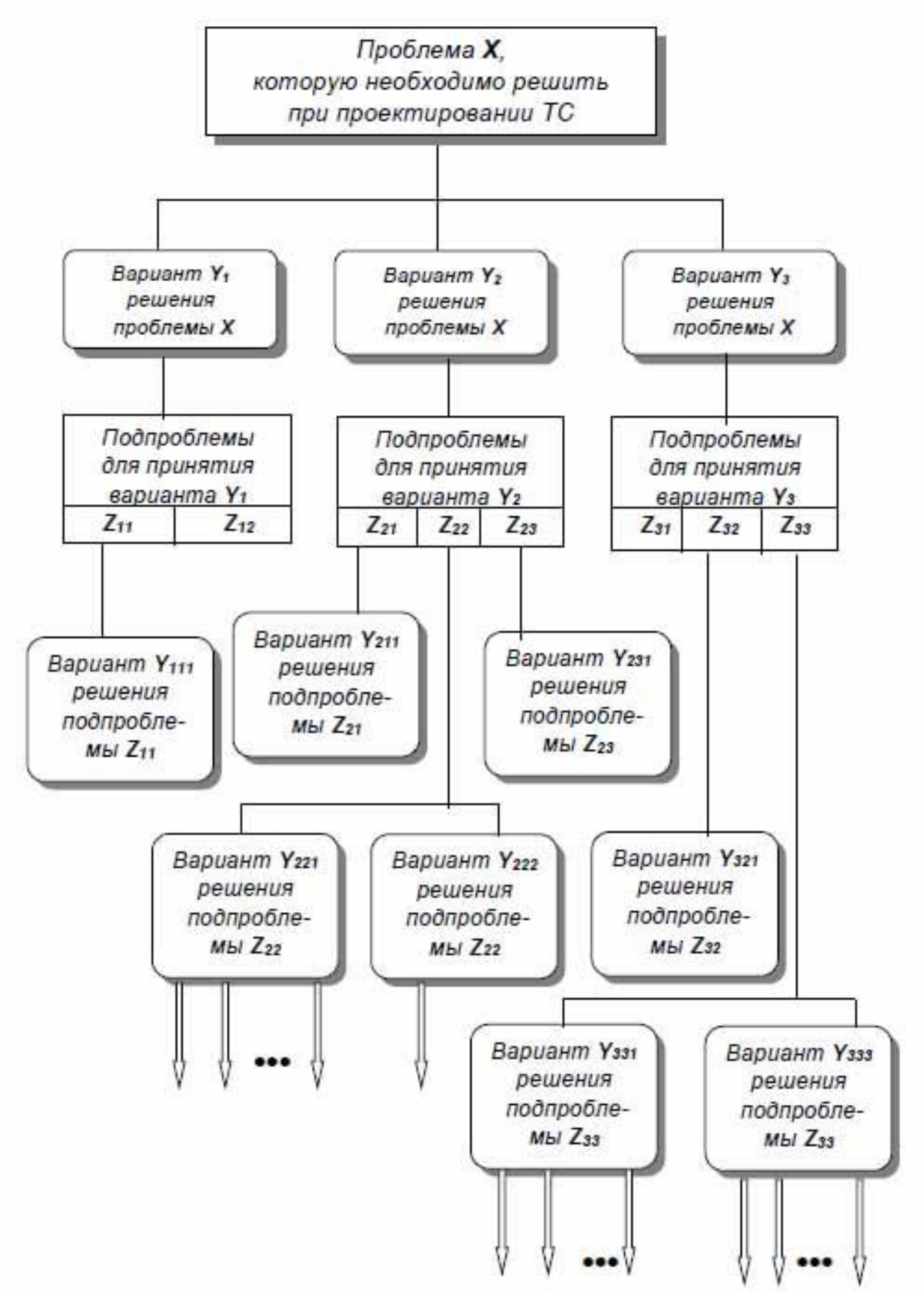

Рисунок 2.2 – Процесс проектирования как иерархия принятия решений (стрелками указаны возможные подпроблемы)

Рисунок 2.3 – Показатели эффективности технических систем

Основные этапы процесса проектирования можно представить как процесс от зарождения идеи до освоения серийного выпуска и начала эксплуатации ТС (рисунок 2.4).

Существует ряд трудностей, возникающих при осуществлении и описании процесса проектирования. Главная из них заключается в том, что проектировщик должен на основании современных данных прогнозировать некоторое

будущее состояние ТС, которое возникнет только в том случае, если его прогнозы верны. Предположения о конечном результате проектирования приходится делать еще до того, как исследованы средства для его достижения.

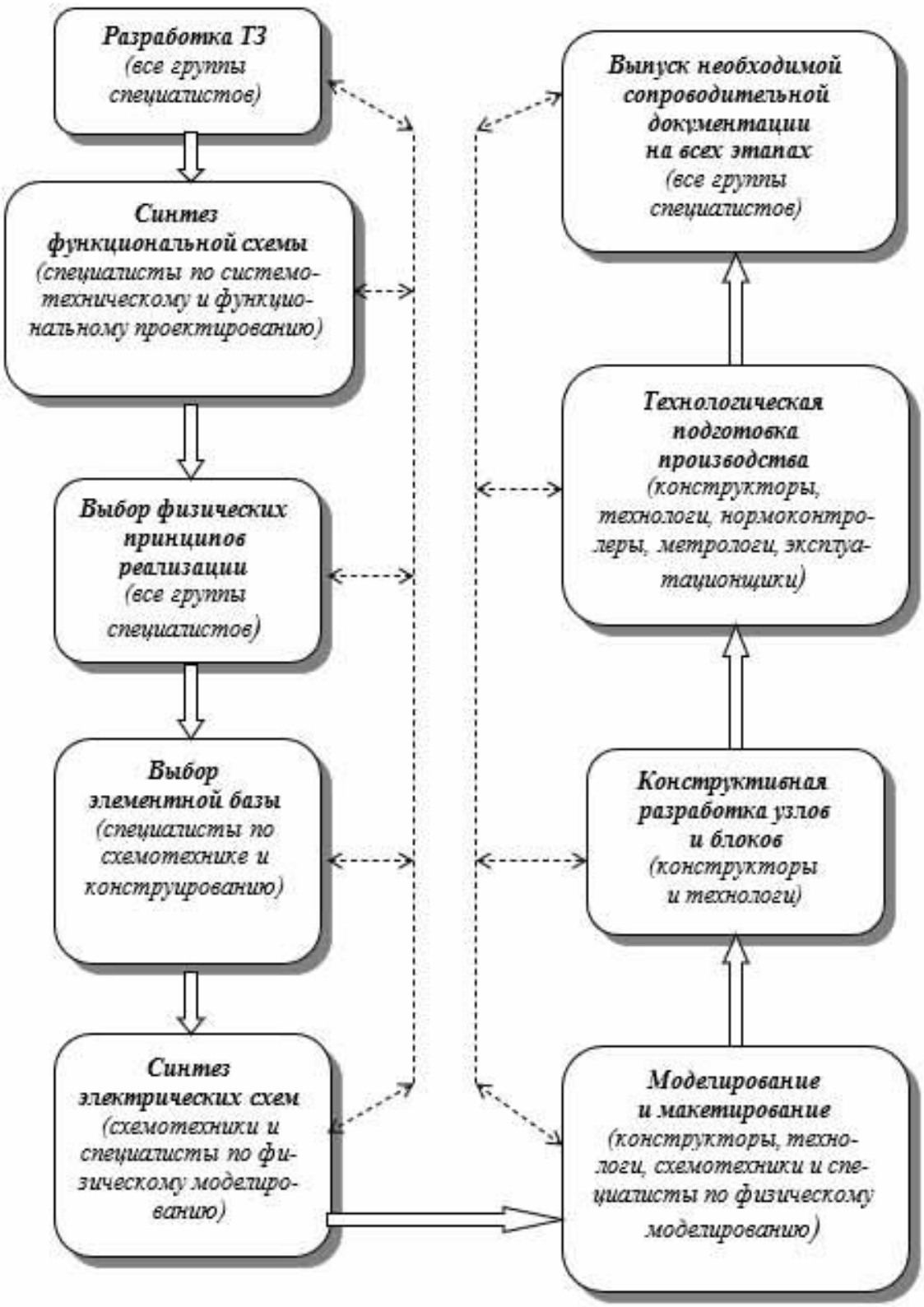

Рисунок 2.4 – Основные этапы проектирования технических систем

Проектировщик вынужден прослеживать события в обратном порядке: от следствий к причинам, от ожидаемого влияния данной разработки на мир – к началу той цепочки событий, в результате которой и возникнет это влияние. Часто случается, что в ходе такого прослеживания на одной из промежуточных ступеней обнаруживаются непредвиденные трудности или открываются новые, более благоприятные возможности. При этом характер исходной проблемы может коренным образом измениться, и разработчик будет отброшен на начальный этап проектирования. Именно эта нестабильность самой задачи и придает процессу проектирования гораздо более сложный и интересный характер, чем обычно думают те, кто никогда им не занимался.

Проектировщики должны добиться, чтобы каждый из многочисленных и разнообразных показателей, интересующих заказчика, обладал двумя свойствами:

– не выходил за пределы возможностей поставщиков, изготовителей, системы сбыта и т. д. ни на одном из этапов существования изделия;

– был связан с теми показателями, которые ему предшествуют, и с теми, что за ним следуют.

Тесные связи между далеко отстоящими друг от друга этапами существования изделия заставляют разработчика прибегать к прослеживанию зависимостей между следствиями и их отдаленными причинами. Чтобы избежать неувязок между отдельными этапами, проектировщик, используя свой главный козырь – воображение, заменяет свои исходные цели другими целями, которые легче увязываются друг с другом, оставаясь столь же приемлемыми с точки зрения поставленной задачи. Такая сильная зависимость целей от конкретных частных решений очень затрудняет процесс проектирования чисто логическими способами.

2.2 Задачи и характер моделирования и проектирования

Моделирование и проектирование занимает центральное место в процессе производства ТС.

Задачи разработки систем, как и прочие технические задачи, всегда тесно взаимосвязаны с другими процессами подготовки производства.

Подготовка производства является частью инновационной деятельности. Этот процесс охватывает все мыслительные, ручные и машинные операции, необходимые для предварительной проработки технического изделия, целью которых является получение описания технического изделия, достаточного для его производства и эксплуатации, на основе задачи моделирования и проектирования.

Процесс моделирования и проектирования оказывает решающее влияние на потребительскую стоимость самого изделия, а также экономичность его производства и эксплуатации. Так проектирование можно представить как длинную цепь взаимосвязанных предположений и уточнений (рисунок 2.5).

Процесс проектирования ТС представляется как ряд событий. Каждое из этих событий является особым этапом в жизненном цикле изделия и зависит от предшествующего события.

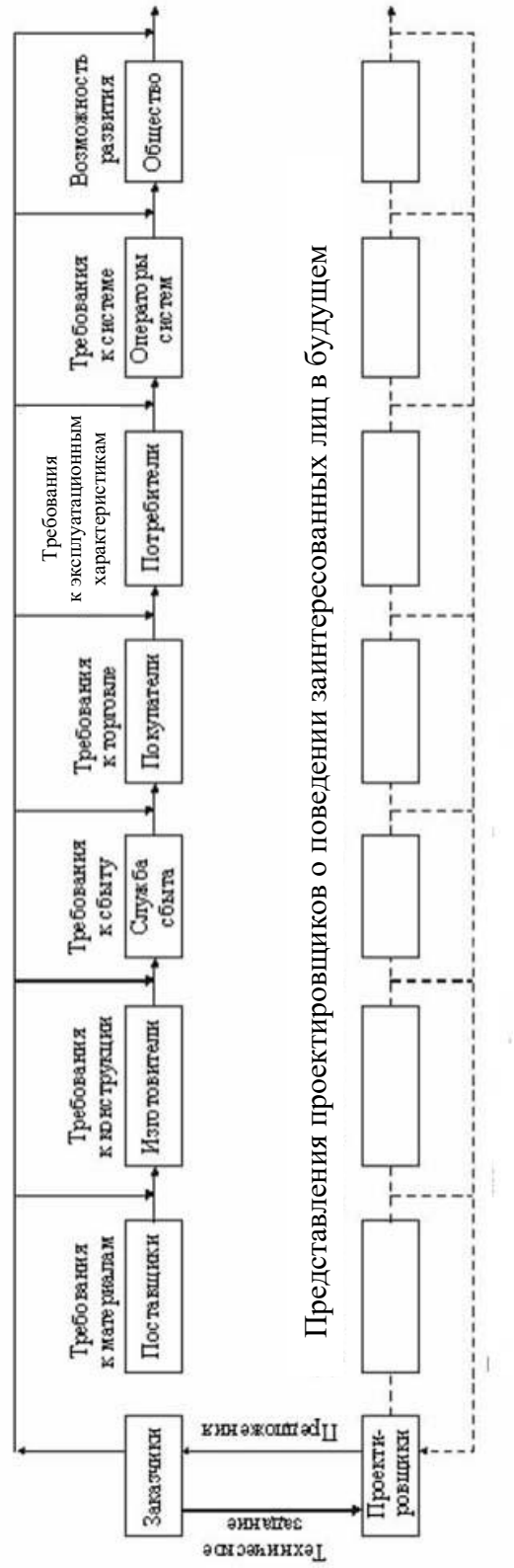

Рисунок 2.5 – Проектирование как цепь взаимосвязанных предположений и уточнений

В нижней половине рисунка 2.5 видно, что в соответствии с полученными заданиями разработчик должен подготовить свои предложения. От него требуется тем или иным способом предсказать свойства объекта и реакцию на них на каждом этапе его существования. Для этого он на моделях проводит экстраполяцию от известных характеристик аналогичных конструкций в прошлом к поведению объекта в будущем, в новой среде. При этом в отношении каждого этапа существования объекта проектировщик должен ответить на вопросы, указанные в таблице 2.1.

Таблица 2.1 – Вопросы, на которые должна ответить проектная группа

Вопросы относительно проектируемого объекта	**Кто дает на них ответ**
Понравится ли проект заказчику? В интересах ли заказчика вложить финансы в этот проект? Будет ли проект осуществлен?	Заказчик и финансирующие организации
Какие программные средства будут использоваться при разработке?	Разработчики
Оптимальным ли образом в проекте используются доступные материалы и комплектующие?	Поставщики
Можно ли достаточно экономично реализовать проект в рамках имеющихся ресурсов?	Изготовители
Можно ли распространить изделие по существующим каналам?	Менеджеры
Каковы требования к внешнему виду, эксплуатационным характеристикам, надежности и пр.?	Потребители и торговые организации
В какой мере объект будет связан с другими изделиями и сможет ли конкурировать с ними?	Другие заказчики
В какой мере изделие изменит существующую ситуацию, создаст ли новые потребности, новые возможности и новые трудности?	Операторы больших систем
В какой мере прямые и побочные эффекты использования изделия приемлемы для всех, кого они касаются?	Государственные учреждения и общественные группы

Не все заказчики осознают необходимость успешного функционирования изделия на каждом этапе его существования. Как правило, заказчик чаще всего не предполагает и того, что у изделия могут проявиться дополнительные достоинства и недостатки, скажем, уже после его продажи. Разработчик же в этих условиях может отчетливо осознавать, что какие-то варианты исполнения проектируемого объекта будут обладать теми или иными достоинствами или недостатками для потребителя, хотя заказчику они и безразличны. При этом у разра-

ботчика появляется искушение предлагать заказчику лишь такие решения, которые благоприятны для потребителя. Поступая таким образом, проектировщик выходит за рамки своей компетенции и ошибочно принимает решения от имени всего общества.

Методика решения проектных задач характеризуется признаками синтеза структуры. Проектировщик мысленно «проигрывает» все фазы эксплуатации будущего изделия.

Определение структуры S для заданной функции F представляет собой недетерминированный шаг с вероятностью перехода $p_{ii} < 1$, результат которого охватывает неограниченное число вариантов (см. рисунок 2.2):

$$F \longrightarrow \{S_i\} \text{ при } p_{ii} < 1, \qquad (2.1)$$

где F – требуемая функция, используемое для определенной цели свойство системы, благодаря которому необходимые для этого входные величины при некоторых условиях преобразуются в выходные величины;

p_{ii} – путь решения с вероятностью < 1;

$\{S_i\}$ – число структур, выполняющих функцию (S – это совокупность элементов M и отношений R между ними внутри системы, т. е. S = {M, R}).

С учетом объективно существующих ограничений можно выделить мероприятия, определяющие принципиальный порядок действий при проектировании (таблица 2.2). Многозначность при выборе решения является, во-первых, одной из возможностей оптимизации и, во-вторых, требует выполнения большого объема работ.

Таблица 2.2 – Систематизация рабочих операций при проектировании

Цель	**Средства**
Пополнение и концентрация исходной информации	– уточнение; – обобщение (абстрагирование); – ограничение
Ограничение неопределенности	– итерационный порядок действий; – использование уже существующих решений, хранящихся в базе данных; – циклический порядок действий (обратная связь)
Использование и ограничение множества решений	– расширение поля решений; – упорядочение (классификации систематизации); – выбор оптимальных вариантов

Неопределенность при синтезе может быть снижена итерационной (пошаговой) обработкой информации, объем которой в процессе проектирования по-

стоянно растет, а также использованием уже существующих решений, предварительным продумыванием решений или их элементов и сознательным возвращением к исходной ситуации для сравнения полученной структуры с требуемой.

Поиск решений обеспечивается при дополнении неполных данных, касающихся задачи проектирования (функции), и выделении в ней главного. Указанные в таблице 2.2 мероприятия являются методическими принципами решения любой проектной задачи. Их применение требует последовательности и систематичности в процессе проектирования.

2.3 Особенности проектирования беспилотного летательного аппарата

Беспилотные летательные аппараты (БПЛА) применяются в широком спектре задач от доставки грузов до выслеживания, сопровождения и поиска целей. Если работа предстоит в опасных условиях таких, например, как высокая задымленность, для сохранения жизни и здоровья человека также предпочтительно использование беспилотных летательных аппаратов. Не стоит забывать, что при поиске человека на большой площади могут потребоваться огромные группы, которые будут прочесывать местность сравнительно долго – в таком случае также может помочь беспилотное летательное средство, так как силами нескольких операторов возможно на порядок увеличить скорость поиска и площадь обследуемого пространства.

Помимо прочего, в последнее время развивается технология создания беспилотных летательных аппаратов, способных самостоятельно, в зависимости от внешних факторов, принимать решения: начинать слежку за целью, возвращаться на базу или начинать видеозапись при подлете к некоторой точке. Такая самостоятельность помогает облегчить труд человека-оператора и снизить возможную монотонность процесса, уменьшить усталость и повысить концентрацию в случае нештатных ситуаций.

Независимо от того, работает беспилотный летательный аппарат по собственному алгоритму либо под непосредственным управлением оператора, умение определять подвижные цели, находящиеся в области видимости, идентифицировать их и либо самостоятельно предпринимать какие-либо действия, либо передавать эту информацию оператору для принятия дальнейших решений является важной задачей, которая может в разы улучшить качество автономной работы беспилотного летательного аппарата или сократить время реакции оператора.

Одной из основных проблем идентификации подвижных наземных целей с борта беспилотного летательного аппарата в автоматическом режиме является идентификация нескольких целей, движущихся по разной траектории по неоднородному ландшафту либо в условиях сложной погодной обстановки. Также представляет сложность идентификация малых движущихся объектов.

На сегодняшний день существует достаточно большое число работ по изучению методов и алгоритмов идентификации подвижных наземных объектов с помощью беспилотных летательных аппаратов. Наиболее значимые результаты

были получены зарубежными учеными, которые проводили исследования, направленные на уточнение методов идентификации объектов, повышения быстродействия БПЛА и точности определения (Ю. Вэньшуай, Ю. Сючу, Ч. Пэнцян, С. Гаэль, А. М. Саид) а также в работах российских и белорусских ученых Е. В. Медведевой, К. А. Карлушина, Е. Е. Курбатовой, в которых описывались методы выделения контуров объектов в видеопотоке

Современные беспилотные летательные аппараты являются сложными техническими устройствами, в работе которых заложено множество принципов. БПЛА могут управляться человеком с земли или автоматически следовать по заданному пути. В некоторых случаях даже возможна ситуация, в которой дрону придется самостоятельно, по радиосигналу или плану местности, ориентироваться и выполнять поставленную задачу. Для решения этих вопросов применяются различные методы и алгоритмы.

Современные беспилотные летательные аппараты выполняют широкий ряд функций на различном расстоянии от человека-оператора. В состав современных программно-аппаратных комплексов входят:

– системы управления полетом (как наземные, так и воздушные);

– системы управления датчиками и камерами, установленными на летательном аппарате;

– полетные батареи и зарядные устройства, чехлы и кейсы для хранения беспилотника.

При моделировании полета БПЛА в состав комплекса также могут входить специальные устройства, необходимые для имитации внешних воздействий или динамики полета. В некоторых случаях также имеются диагностические устройства для оценки состояния летательного аппарата и др.

Технологии, применяемые для идентификации целей

Когда заходит речь об обнаружении или поиске объектов с помощью технических средств, первыми на ум приходят камеры оптического диапазона. Принцип их работы можно описать следующим образом: световой поток, проходя через объектив, фокусируется на фоточувствительной матрице, которая преобразует его в электрический сигнал. В цифровом исполнении сигнал со светочувствительной матрицы преобразуется в цифровой поток посредством встроенного кодера, в отличие от аналоговых камер, в которых сигнал с фоточувствительной матрицы подается непосредственно на выход камеры.

Как цифровые, так и аналоговые видеокамеры при работе в видимом диапазоне сталкиваются с проблемами плохой или чрезмерной освещенности при различных погодных условиях, таких как дождь, снег или туман. Дальность видимости в таких условиях будет крайне низкой и может потребовать дополнительного освещения. Проблему дальности видимости в условиях сложной погодной обстановки могут решить тепловизионные камеры.

Принцип работы таких камер заключается в следующем: тепловизионные камеры также улавливают световой поток, но уже другого, инфракрасного диапазона, излучаемого самим объектом, а не отраженного видимого диапазона,

как это делают обычные камеры. Так как такие камеры работают с инфракрасным (тепловым) диапазоном в некоторых из них применяются конструктивные решения охлаждения матрицы. Данное улучшение позволяет расширить чувствительность камеры и увеличить дистанцию обнаружения до двух – пяти километров. Потому важным параметром для тепловых камер является тепловое разрешение – минимальная улавливаемая температурная разность двух точек. Высокое температурное разрешение позволяет четче обрисовывать границы объектов. Минусами таких камер можно назвать «растворение» объекта наблюдения при одинаковой температуре окружения и объекта, а также сложность работы с материалами, обладающими высокими коэффициентами отражения, – камера в таком случае будет анализировать температуру отраженного света, а не излучение самого отражателя.

При использовании камер на борту беспилотного летательного аппарата остро встает вопрос о проблемах проведения аэрофотосъемки. Их решение требует: калибровки съемочного оборудования, применения правильных настроек в цифровых фотометрических системах, съемки с правильным перекрытием и в надлежащих погодных условиях, проведения съемки в правильное время суток.

Для быстрого сопоставления полученных кадров с местностью, определения координат цели или правильного формирования панорамного изображения также важно использовать системы высокоточного позиционирования, которые помогут БПЛА в правильном определении курса при полете в автоматическом режиме.

При построении систем с БПЛА следует помнить о необходимости наличия защищенных каналов связи между летальным аппаратом и станцией управления. При недостаточной защищенности данные могут быть украдены, но что еще хуже – подменены, что может вызвать потерю дорогостоящего оборудования. Кроме защиты связи необходимо правильно построить сам канал связи, от которого требуются высокие стабильность работы, скорость передачи, защищенность и пропускная способность.

Методы обработки видеопотока, поступающего из различных источников

В зависимости от видеофиксирующего оборудования, вида каналов связи и их загруженности, расстояния от комплекса управления, вычислительных возможностей бортовой аппаратуры и т. д. не всегда имеется возможность передавать стабильный видеопоток в высоком качестве и с отсутствием помех.

Для решения вопроса качества получаемого видеопотока и его объема применяются различные методы обработки, о которых пойдет речь далее.

Методы сжатия изображений и видео можно разделить на два основных типа:

– без потери информации;

– с потерей информации.

Методы сжатия без потери информации отличаются от второго типа тем, что после декомпрессии позволяют получить исходные данные без искажений, что может быть критически важно в некоторых ситуациях. Однако они обладают таким существенным недостатком, как малый коэффициент сжатия. При использовании второго типа сжатия данные, получаемые в результате декомпрессии, отличаются от исходных, но эти отличия возможно контролировать и удерживать на допустимом уровне. При использовании такого метода по сравнению с предыдущим существенно увеличивается коэффициент сжатия.

Основные идеи методов сжатия без потери информации можно определить следующим образом:

– одинаковые, повторяющиеся данные можно заменить на короткую последовательность, состоящую из одного общего элемента и счетчика повторений;

– часто используемые значения шифруются короткими кодами, а редко используемые – более длинными кодами.

Алгоритм сжатия *RLE*, алгоритм Хаффмана и арифметическое сжатие используют данный метод сжатия.

Основные идеи, используемые для методов сжатия с потерей информации, приведены ниже.

Метод отбрасывания реализуется путем простого отбрасывания части данных, из которых состоит цифровое изображение или видео. Существуют следующие виды отбрасывания:

– уменьшение формата изображения путем отбрасывания строк и столбцов. Например, исходное изображение формата 640 × 480 прореживается вдвое по каждой координате до размера 320 × 240;

– частным случаем предыдущего варианта является прореживание в цветовых плоскостях. Например, исходное изображение в формате *YUV* подвергается сокращению форматов цветоразностных составляющих. В результате происходит преобразование из формата 4 : 4 : 4 к форматам 4 : 2 : 2, 4 : 1 : 1 или 4 : 2 : 0;

– уменьшение разрядности данных путем отбрасывания младших разрядов. Например, исходное изображение имеет 10-битное представление, а после отбрасывания двух младших битов становится 8-битным. Соответственно изменяется и количество градаций яркости для черно-белого изображения или количество отображаемых цветов для цветного изображения.

В случае использования *метода палитризации* исходное изображение имеет полноцветное представление (скажем, для 8-битного представления общее количество возможных цветов составит около 16 млн цветов). Это количество цветов ограничивается до некоторого заданного значения, например, до 256 наиболее часто встречающихся цветов – это и называется палитрой. Остальные цвета изменяются в соответствии с ближайшими цветами из палитры.

Отдельно можно выделить *метод отбрасывания части кадров из видеопоследовательности* (как их прореживание, так и удаление повторяющихся кадров).

Общим свойством всех этих методов является сравнительная простота реализации, высокое быстродействие и отсутствие необходимости сохранения дополнительной информации (за исключением метода палитризации).

Вторым простейшим методом является *метод усреднения*. В этом случае производится вычисление из величин нескольких соседних элементов их среднего значения. При этом полученное среднее значение заменяет собой эти несколько элементов.

Применительно к цифровому изображению речь идет об объединении и усреднении соседних пикселей в выбранном окне (2 × 2, 4 × 4 и т. п.). Скажем, для исходного изображения формата 640 × 480 создается новое изображение формата 320 × 240 элементов. При этом усреднение проводится в окне 2 × 2 пикселя.

В отличие от методов отбрасывания метод усреднения использует не исключение данных, а их преобразование. Это значит, что при усреднении производится не просто отбрасывание части элементов с полной потерей информации, содержащейся в них, а частичное делегирование свойств этих элементов новому усредненному элементу. Таким образом, часть отбрасываемой информации удается сохранить.

Кроме описанных методов сжатия также можно применять *методы преобразования сигналов*.

Из теории обработки сигналов известно, что информационное содержание практически любого сигнала удобнее анализировать, рассматривая не изменение сигнала во времени, а его разложение на частотные составляющие. Анализ частотного состава позволяет отделить существенные информационные составляющие от менее значимых. Благодаря контролируемому удалению менее существенных составляющих можно уменьшить объем данных, требуемый для передачи и хранения информации об этом сигнале. При работе с изображениями и видеосигналом используется двумерное представление цифровой информации (в виде матриц). Тем самым, для реализации обработки таких двумерных сигналов используются двумерные преобразования.

Среди основных методов, реализующих пространственно-частотное преобразование, чаще всего используются дискретное косинусное преобразование и дискретное вейвлет-преобразование.

Также следует упомянуть и *методы межкадрового сжатия, алгоритмы сжатия, используемые форматом JPEG, MPEG,* и др.

Методы уточнения полученного изображения

Методы сжатия с потерей информации позволяют достигать более высокого коэффициента сжатия за счет потери информации. Однако в некоторых ситуациях нет необходимости в высоком качестве всего изображения –

достаточно конкретных объектов высокого качества, выделенных на заднем или переднем плане. Для отделения объектов от общего плана применяются следующие методы.

Методы вычитания фона являются самыми простыми и наиболее часто применяемыми для детектирования движущихся объектов. Суть их заключается в нахождении попиксельной разности между текущим кадром и некой моделью фона. В принципе, такая модель должна представлять собой сцену без движущихся объектов. При этом необходимо ее регулярное обновление, для того чтобы учитывать изменение условий освещенности и настроек камеры, таких как поворот, наклон и изменение фокусного расстояния. Главным недостатком методов вычитания фона является возможная классификация фоновых пикселей как переднеплановых. Это может происходить, например, с листьями деревьев, колышущимися на ветру, падающим снегом и дождем, тенями, отбрасываемыми движущимися объектами, и др. Кроме того, методам данного класса присуща латентность в обновлении модели фона: должно пройти некоторое время, прежде чем будут учтены изменения, связанные с началом движения или остановкой объекта. Наконец, методы вычитания фона в своей простейшей реализации предъявляют достаточно высокие требования к ресурсам вычислительной системы.

При использовании *вероятностных методов* задний план формируется в результате моделирования стохастического «пиксельного процесса», т. е. для каждого пикселя изменение его интенсивности от кадра к кадру рассматривается как временной ряд, состоящий из скалярных величин для полутоновых изображений и векторов для цветных. В результате фон представляет собой гауссову смесь, т. е. линейную комбинацию одномерных, нормально распределенных случайных величин. Более совершенными являются алгоритмы, создающие попиксельную модель всей сцены, в которой используются отдельно гауссовы смеси для фона, переднего плана и теней. Основываясь на времени существования и дисперсии каждого гауссиана в смеси, можно определить, какие из них относятся к фону. Пиксели, значения которых не укладываются в фоновые распределения, считаются переднеплановыми до тех пор, пока не появится гауссиан, позволяющий с достаточной точностью отнести их к фону. Такой подход позволяет учитывать медленные изменения освещенности путем подстройки параметров гауссианов. Кроме того, в рамках вероятностных методов возможен адекватный анализ распределений с несколькими максимумами, что является типичным для ситуаций с падающими тенями, отражениями, качающимися ветвями и др. Однако быстрые изменения фона и освещенности сцены данные алгоритмы описать не могут.

Методы временной разности позволяют отделить передний план от фона при помощи попиксельного вычитания двух или более последовательных кадров. С помощью методов временной разности хорошо определяются динамические изменения сцены, но часто не целиком выделяются все однородные пиксели одного объекта, что приводит к фрагментированности выделенных объектов (часто внутри них образуются пустоты).

Для выделения переднего плана из видеопоследовательности весьма перспективным является *метод оптического потока*. Понятие потока обычно используется для описания когерентного движения точек или характерных признаков (контрольных точек) между последовательными кадрами. Выделение фона, основанное на вычислении оптического потока, использует характеристики вектора потока движущихся объектов для нахождения тех областей видеопоследовательности, в которых происходят изменения. Также с помощью оптического потока можно получить информацию о расположении, размерах и некоторых других параметрах таких областей. В принципе, с помощью методов оптического потока может быть проведено наиболее аккуратное выделение движущихся объектов даже в случае перемещения камеры. Однако алгоритмы данного класса являются слишком ресурсоемкими и, кроме того, чрезвычайно чувствительными к шуму – вследствие этого на данный момент без дорогих специализированных процессов они не могут быть применены к видеопотокам в реальном времени.

В настоящее время развиваются методы построения заднего плана, основанные на применении искусственных нейронных сетей. Эти методы используют способность нейронной сети адаптироваться к входным данным за счет введения настраиваемых обратных связей. Каждый пиксель фона управляется своей нейронной сетью, в результате чего через некоторое время, требуемое для настройки (обучения) нейронной сети, формируется модель фона, способная заданным образом подстраиваться к изменениям входного изображения.

После выделения интересующего объекта или же без него уточнение изображения может проводиться по следующим методам:

– методы повышения контраста изображения позволяют улучшить различимость объектов на изображении, полученном в условиях трудной погодной обстановки, такой как туман или дождь;

– методы нормализации изображений позволяют улучшить качество и другие характеристики изображений с помехами или неподходящего качества;

– метод улучшения качества изображения с помощью эмпирической фильтрации Винера в вейвлет-области.

3 МЕТОДОЛОГИЯ МОДЕЛИРОВАНИЯ И ПРОЕКТИРОВАНИЯ ТЕХНИЧЕСКИХ СИСТЕМ

3.1 Слежение за ключевыми точками лица в видеопотоке

Область применения компьютерного зрения расширяется параллельно с тем, как оптимизируются алгоритмы, на основе которых решаются прикладные задачи компьютерного зрения. В то же время, по мере того как повышается производительность компьютерного оборудования, становятся актуальными некоторые алгоритмы, применение которых до недавнего времени было проблематичным. Именно поэтому в последнее время наиболее бурно развиваются алгоритмы, работающие на основе машинного обучения: отчасти из-за скачка производительности доступного оборудования, а отчасти за счет появления больших массивов данных (в том числе маркированных).

Вопрос производительности алгоритмов компьютерного зрения является краеугольным во многих прикладных задачах: моделировании, биометрической аутентификации, поиске, кодировании и даже помощи в сборе и автоматической маркировке массивов данных для моделей, не связанных напрямую с компьютерным зрением.

Существует целый ряд подходов к решению задач компьютерного зрения с использованием алгоритмов машинного обучения. Однако, несмотря на значительное количество исследований в этом направлении, отсутствует детальная информация о сравнении различных алгоритмов для решения задачи обнаружения опорных точек лица и оптимальной реализации подобных моделей.

Благодаря возможности слежения за движением можно определять движение объекта, а затем применять полученные данные к движению другого объекта, например, другого слоя или контрольной точки эффекта, что позволяет создавать композиции, в которых изображения и эффекты следуют движению. Также возможна стабилизация движения. В этом случае данные отслеживания используются в анимации слоя для компенсации движения объекта на этом слое. Можно привязать свойства к данным отслеживания, используя выражения, позволяющие задействовать различные сценарии применения.

3.1.1 Анализ существующих алгоритмов компьютерного зрения на основе методов машинного обучения

Типы алгоритмов и виды архитектур

Существует множество разновидностей алгоритмов на основе машинного обучения, которые используются для решения задач компьютерного зрения: нейронные сети, деревья решений, эволюционные алгоритмы и др. Рассмотрим несколько классов данных алгоритмов, а также проведем анализ с целью определения возможности их использования для слежения за опорными точками лица в видеопотоке.

В области компьютерного зрения, как правило, используются два класса алгоритмов машинного обучения: алгоритмы для решения задач классификации, которые чаще всего рассматривают дискретные переменные, и алгоритмы для решения задач регрессионного анализа, которые оперируют непрерывными переменными.

Задача классификации – это задача, в которой существует множество объектов или ситуаций, разделенных на классы. Задается конечное множество объектов, для которых известно соответствие искомым классам, – подобное множество назовем выборкой. Принадлежность к классам остальных рассматриваемых объектов неизвестна. В рамках задачи классификации требуется построить алгоритм, способный определить принадлежность произвольного объекта из исходного множества к какому-либо классу. В данном контексте классификацией объекта называют наименование класса, который алгоритм классификации выдает в результате его применения к конкретному объекту.

В математической статистике задачи классификации также называют задачами дискриминантного анализа. В машинном обучении задачи классификации часто решаются с помощью алгоритмов на основе искусственных нейронных сетей (при обучении с учителем), но могут быть также решены и другими алгоритмами, например, деревом принятия решений.

Также существуют другие способы постановки эксперимента (например, обучение без учителя) и комбинации разных видов обучения. Обучение без учителя, как правило, используется для решения другого типа задач – задач кластеризации, где разделение объектов обучающей выборки на классы не задается изначально, и разбиение объектов на классы происходит исключительно на основе их сходства друг с другом. В некоторых прикладных областях часто не различают задачи кластеризации и задачи классификации из-за их близости.

Задачи регрессионного анализа, напротив, работают с непрерывными переменными. Регрессионный анализ – это статистический метод исследования влияния одной или нескольких независимых переменных на зависимую переменную. Независимые переменные также называют регрессорами или предикторами, а зависимые переменные – критериальными переменными. В общем случае целями регрессионного анализа являются:

– определение степени детерминированности вариации зависимой переменной независимыми переменными;

– предсказание значения зависимой переменной с помощью независимой переменной или переменных;

– определение величины вклада отдельных независимых переменных в вариацию зависимой.

В контексте задач компьютерного зрения результатом регрессионного анализа чаще всего является именно предсказание значения зависимой переменной с помощью независимых. В частности, таковой является задача, рассматриваемая в данной работе: предсказание значений координат опорных точек лица на основе значений пикселей в исходном изображении, взятых в одном из цветовых

пространств. Рассмотрим алгоритмы машинного обучения, которые можно применить для решения задачи регрессионного анализа.

Как и в случае задач классификации, задачи регрессионного анализа чаще всего решаются с помощью алгоритмов на основе искусственных нейронных сетей при обучении с учителем. Но в определенных прикладных областях вполне обосновано применение других алгоритмов машинного обучения – например, деревьев решений, которые в этом случае называют регрессионными деревьями.

Деревья решений

Деревья решений осуществляют разбиение пространства объектов в соответствии с некоторым набором правил. Эти правила являются логическими утверждениями в отношении той или иной переменной и могут быть истинными или ложными.

Ключевые правила деревьев решений:

– правила разбиения позволяют реализовать последовательную дихотомическую сегментацию входного сигнала;

– два объекта считаются похожими, если они оказываются в одном и том же сегменте разбиения;

– на каждом шаге разбиения увеличивается количество информации об исследуемой переменной.

Деревья классификации и регрессии находят применение в решении множества практических задач благодаря их преимуществам в сравнении с другими алгоритмами:

– алгоритмические модели, построенные на основе деревьев решений, являются очень легко интерпретируемыми. Фактически они представляют собой набор правил вида «если …, то …». Интерпретация также облегчается за счет представления этих правил в виде наглядной древовидной структуры;

– деревья решений позволяют работать с переменными любого типа, не требующими какой-либо предварительной подготовки для ввода в модель (например, логарифмирования или преобразования категориальных переменных в индикаторные);

– при построении модели нет необходимости задавать форму взаимосвязи между откликом и предикторами в явном виде. Это оказывается особенно полезным при работе с данными, о свойствах которых мало что известно;

– деревья решений, по сути, автоматически выполняют отбор информативных предикторов, учитывая возможные взаимодействия между ними. Это, в частности, делает деревья решений полезным инструментом для анализа данных;

– деревья решений можно эффективно применять к данным с пропущенными значениями, что очень полезно при решении целого ряда практических задач.

К недостаткам этого класса моделей обычно относят нестабильность, невысокую точность предсказаний и слабую способность к обобщению. По своей сути деревья используют «наивный подход» в том смысле, что они исходят из

предположения о взаимной независимости признаков. Поэтому модели регрессионных деревьев статистически наиболее эффективны, когда анализируемые переменные имеют низкую степень корреляции.

К сожалению, это сложно гарантировать в задачах компьютерного зрения, и решение рассматриваемой прикладной задачи потребует применения значительного комплекса мероприятий по регуляризации модели на основе деревьев решений. Рассмотрим альтернативный подход в виде алгоритмов на основе искусственных нейронных сетей, которые, пожалуй, получили наибольшее распространение среди алгоритмов машинного обучения конкретно в области компьютерного зрения.

Искусственные нейронные сети

Под искусственной нейронной сетью понимается математическая модель, а также ее программная и аппаратная реализация, построенная по принципу биологических нейронных сетей – нервных клеток живого организма. Понятие «нейронные сети» возникло при попытке смоделировать процессы, протекающие в мозге человека.

Искусственная нейронная сеть представляет собой систему простых процессоров (искусственных нейронов), соединенных и взаимодействующих между собой. Каждый из процессоров сети работает с периодически поступающими или передающимися другим процессорам сигналами. Будучи соединенными в большую сеть, они способны решать сложнейшие задачи в самые короткие сроки, несмотря на простоту индивидуальных компонентов сети.

С математической точки зрения нейронные сети представляют собой способ решения многопараметрической задачи нелинейной оптимизации. Кибернетика использует теорию нейронных сетей в решении задач адаптивного управления, построении алгоритмов для робототехники. В программировании нейронная сеть – один из способов решения проблемы эффективного параллелизма.

Программирование нейронных сетей состоит из двух этапов: разработки структуры сети и процесса обучения. Именно благодаря обучению сеть способна выявлять сложные зависимости между входными и выходными данными и, главное, обобщать полученные знания.

Искусственные нейронные сети имитируют работу нервной системой живых организмов. Мозг человека и его нервная система состоят из нейронов, соединенных нервными волокнами. Между нейронами при помощи нервных волокон передаются электрические импульсы и фактически все действия, которые происходят с живым организмом, все раздражения кожи, глаз, боль, процессы мышления – есть передача электрических импульсов между нейронами.

Биологический нейрон устроен следующим образом: он принимает входные импульсы от произвольного количества предшествующих нейронов через дендриты. Полученные импульсы суммируются, и, если у суммарного импульса превышен некий порог, то нейрон переходит в возбужденное состояние, формирует собственный импульс и посылает его далее через аксон.

Рассмотрим математическую модель описанного процесса на примере, представленном на рисунке 3.1.

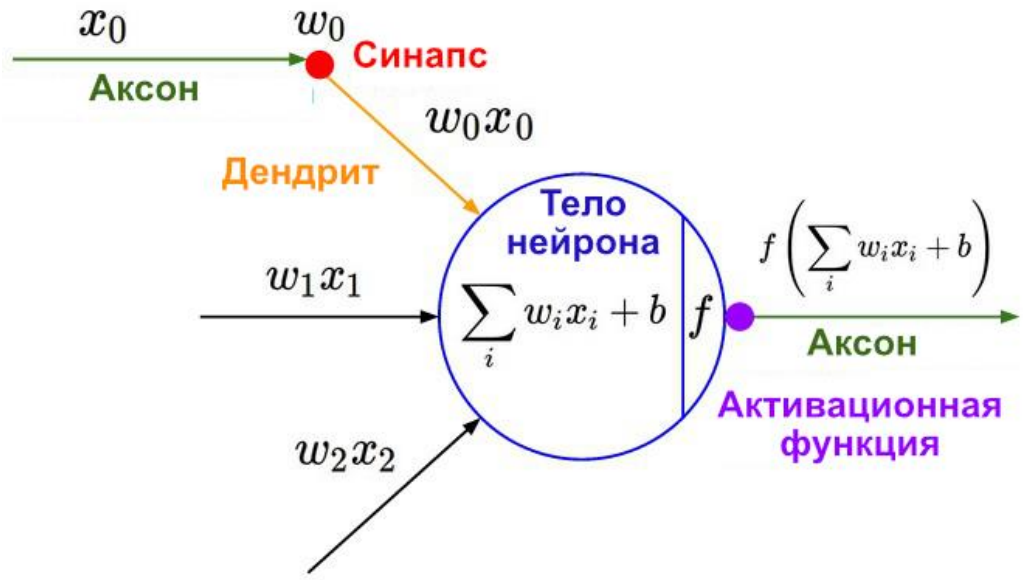

Рисунок 3.1 – Модель прохождения сигнала через нейрон

Данная модель описывает нейрон, к которому поступают входящие импульсы по трем дендритам. Синапсы имеют веса w_i. К синапсам поступают входные сигналы x_i. К нейрону после прохождения синапсов и дендритов поступают импульсы $x_i w_i$. Таким образом, в рассматриваемой модели нейрон получает суммарный импульс x:

$$x = x_0 w_0 + x_1 w_1 + x_2 w_2. \quad (3.1)$$

Далее нейрон преобразует этот импульс в соответствии с активационной функцией $f(x)$, получая силу выходного импульса y:

$$y = f(x) = f(x_0 w_0 + x_1 w_1 + x_2 w_2). \quad (3.2)$$

В более общем случае на вход искусственного нейрона поступает множество сигналов, каждый из которых одновременно является выходом другого нейрона. Такой вход умножается на соответствующий вес, затем произведения суммируются и к сумме применяется активационная функция, определяя уровень активации нейрона.

Стоит отметить, что математическая модель искусственного нейрона игнорирует большинство свойств биологического нейрона, например, задержки во времени, которые воздействуют на динамику системы. Входные сигналы сразу

же порождают выходной сигнал. Кроме того, искусственный нейрон не учитывает воздействий синхронизирующей функции биологического нейрона. Однако нельзя не принимать во внимание исключительное сходство живого нейрона и искусственного.

Индивидуальный нейрон – лишь простейший компонент сети в виде элементарного процессора, абстракция ее функциональной единицы. Пожалуй, наиболее важным этапом решения задачи при помощи алгоритма на основе искусственных нейронных сетей является разработка и построение архитектуры сети, т. е. того, каким образом нейроны сгруппированы и как они взаимодействуют друг с другом. Существует огромное количество типов архитектур, которые отражают основные шаблоны построения взаимосвязей между нейронами, предполагая, что архитектуры внутри одного типа в основном отличаются некоторыми параметрами, такими как количество слоев и их размер.

Остановимся подробнее на некоторых типах архитектур нейронных сетей, которые часто применяются в области компьютерного зрения и будут использованы в данной работе, например, сети прямого распространения (*Feed Forward Network*) и глубокой сверточной сети (*Deep Convolutional Network*).

Сверточные нейронные сети

Сверточная нейронная сеть (*CNN*) – особая архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году, нацеленная на эффективное распознавание образов. Архитектура сверточных нейронных сетей использует некоторые особенности зрительной коры головного мозга: основная идея заключается в чередовании сверточных слоев и субдискретизирующих слоев, или слоев подвыборки. Структура сети – однонаправленная, т. е. без обратных связей, принципиально многослойная, из-за чего сверточные нейронные сети включают в состав технологий глубокого обучения. Для обучения используются стандартные методы, чаще всего метод обратного распространения ошибки.

Работа сверточной нейронной сети обычно интерпретируется как переход от конкретных особенностей изображения к более абстрактным деталям, и далее к еще более абстрактным деталям вплоть до выделения понятий высокого уровня, как будет продемонстрировано далее. При этом в процессе обучения сеть самонастраивается и самостоятельно вырабатывает необходимую иерархию уровней абстракций и самих абстрактных признаков, фильтруя маловажные детали и выделяя существенные.

Подобная интерпретация носит скорее метафорический характер. Фактически «признаки», вырабатываемые сложной и разветвленной сетью, трудно интерпретировать или понять. Более того, если анализ этих признаков выявляет игнорирование моделью каких-либо существенных явлений в исходных данных, это может говорить о том, что либо не хватает данных для обучения, либо структура сети обладает недостатками и система не может обнаружить эффективных решений для данных явлений. В этом случае рекомендуется усовершенствовать саму структуру и архитектуру сети, вместо того чтобы корректировать признаки

напрямую. Тем не менее иллюстрация работы сверточной нейронной сети на простом примере может быть полезной для интуитивного понимания принципов ее работы. Рассмотрим визуализацию значений первого сверточного слоя в относительно неглубокой сети, используя их как интенсивность пикселей в изображении.

На рисунках 3.2 и 3.3 визуализирован первый слой сверточной нейронной сети размером 32 блока, с ядром свертки размерностью 5 × 5 – в результате обработки получены 32 изображения размером 5 × 5 пикселей.

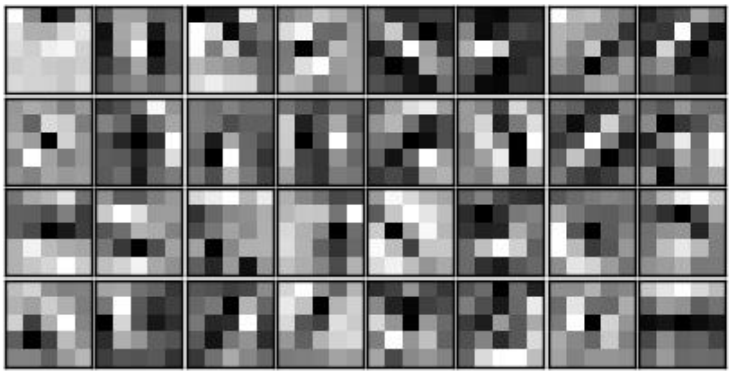

Рисунок 3.1 – Исходная визуализация

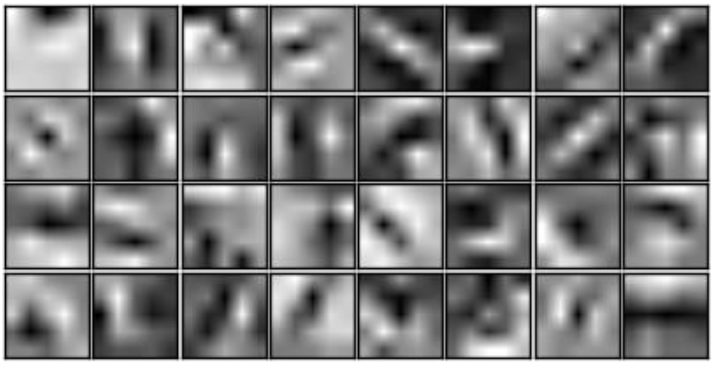

Рисунок 3.2 – Интерполированная визуализация

Интуитивное понимание работы сверточных нейронных сетей подсказывает, что первый слой должен содержать фильтры для обнаружения простейших шаблонов пикселей, таких как контуры и линии, что и подтверждает приведенная визуализация. Эти простейшие фильтры используются моделью на последующих слоях для обнаружения более сложных шаблонов, являющихся комбинациями приведенных простейших шаблонов. Таким образом, каждый слой сверточной нейронной сети опирается на предшествующие слои, обучаясь обнаруживать более замысловатые шаблоны на их основе – можно сказать, что каждый последующий слой сверточной нейронной сети является новым уровнем абстракции с переходом от простых шаблонов к более сложным.

Свое название архитектура сети получила из-за наличия операции свертки, которая умножает каждый фрагмент изображения на матрицу (ядро) свертки поэлементно, суммирует результат и записывается в аналогичную позицию выходного слоя.

В обычном перцептроне, представляющем из себя полносвязную нейронную сеть с одним скрытым слоем, каждый нейрон связан со всеми нейронами предыдущего слоя, а каждая связь имеет свой персональный весовой коэффициент. В сверточной нейронной сети в операции свертки используется лишь ограниченная матрица весов небольшого размера, которая последовательно «передвигается» по всему обрабатываемому слою (в самом начале – непосредственно по входному изображению), после каждого сдвига формирует сигнал активации для нейрона следующего слоя с аналогичной позицией. Таким образом, для всех нейронов последующего слоя используются одна и та же матрица весов, которую также называют ядром свертки. Матрица весов является графическим кодированием какого-либо признака, например, наличия линии или края объекта, который можно легко интерпретировать как резкую смену интенсивности пикселей. Затем следующий слой, полученный в результате операции свертки этим ядром, определяет наличие данного признака в обрабатываемом слое и его координаты, формируя так называемую карту признаков. Даже в простой сверточной нейронной сети используется целый набор подобных признаков, кодирующих отдельные элементы изображения (например линии, дуги и др.). Ключевая особенность сверточных сетей состоит в том, что набор этих признаков не закладывается при разработке архитектуры сети, а динамически формируется самой моделью в процессе обучения сети классическим методом обратного распространения ошибки. Проход каждым ядром свертки формирует свой собственный экземпляр карты признаков, и, соответственно, создается множество независимых карт признаков на каждом сверточном слое. Также следует отметить, что при переборе слоя матрицей весов ее чаще всего передвигают не на весь размер этой матрицы, а на небольшое расстояние, называемое шагом (*stride*). Так, например, при размерности матрицы весов 5×5 ее сдвигают на один или два нейрона (пикселя) вместо пяти, чтобы не «перешагнуть» искомый признак.

Так как операция свертки предполагает глубокую архитектуру самой сети, т. е. большое количество слоев, в сверточных сетях формируется огромное количество параметров, значительно большее, чем в случае неглубокой полносвязной искусственной нейронной сети. Большое число параметров приводит к риску переобучения или неспособности модели к обобщению полученных знаний. С этим явлением можно бороться еще на уровне проектирования архитектуры сети при помощи операции субдискретизации, или подвыборки, которая уменьшает размерность сформированных карт признаков. В случае сверточных нейронных сетей считается, что информация о факте наличия искомого признака важнее точного знания его координат. Поэтому если из нескольких соседних нейронов карты признаков выбрать максимальный и принять за один нейрон уплотненной карты признаков меньшей размерности, это не приведет к потере важной информации, но значительно снизит количество параметров сети. Помимо ускорения

дальнейших вычислений, за счет данной операции сеть становится более инвариантной к масштабу входного изображения.

Рассмотрим типовую структуру сверточной нейронной сети более подробно. Как правило, подобная сеть состоит из большого количества слоев, где после начального слоя (чаще всего входного изображения) сигнал проходит серию сверточных слоев, в которых чередуются собственно свертка и субдискретизация (подвыборка). Чередование слоев позволяет составлять карты признаков из карт признаков, формируя множество уровней абстракции; на каждом следующем слое карта уменьшается в размере, но увеличивается количество каналов. На практике это означает способность распознавания сложных иерархий признаков. Обычно после прохождения нескольких слоев карта признаков вырождается в вектор или даже скаляр, но в результате мы получаем огромное множество таких карт признаков. На выходе сверточных слоев сети дополнительно устанавливают несколько слоев полносвязной нейронной сети, на вход которых подаются финальные карты признаков. Рассмотрим описанные слои более подробно.

Слой свертки – это основной блок сверточной нейронной сети. Он включает в себя фильтр для каждого канала, ядро свертки, которое обрабатывает предыдущий слой по фрагментам, суммируя результаты матричного произведения для каждого фрагмента. Ядро свертки представляет из себя небольшую двумерную матрицу, весовые коэффициенты которой неизвестны и устанавливаются в процессе обучения.

Слой активации. Скалярный результат каждой свертки попадает на вход функции активации, которая представляет собой некую нелинейную функцию. Слой активации обычно логически объединяют со слоем свертки и часто считают, что функция активации встроена в слой свертки. Функция нелинейности может быть любой, но традиционно используются функции типа гиперболического тангенса или сигмоиды. Относительно недавно была предложена и исследована новая функция активации – линейный выпрямитель (*ReLU, rectified linear unit*), – которая позволила существенно ускорить процесс обучения и одновременно упростить вычисления за счет простоты самой функции: по сути, линейный выпрямитель – это операция отсечения отрицательной части скалярной величины [13]. Линейный выпрямитель в качестве функции активации будет использован во всех рассматриваемых архитектурах в данной работе. На данный момент эта функция и ее модификации являются наиболее часто используемыми в глубоких нейронных сетях, в частности, в сверточных.

Слой подвыборки, или *субдискретизации*, представляет собой нелинейное уплотнение карты признаков, где группа пикселей (обычно размером 2 × 2) уплотняется до одного пикселя, проходя нелинейное преобразование (в данном примере итоговое количество параметров снижается в четыре раза). Наиболее употребительна при этом функция максимума. Преобразования затрагивают непересекающиеся прямоугольники или квадраты, каждый из которых ужимается до одного пикселя с максимальным значением. Операция подвыборки позволяет существенно уменьшить пространственный объем изображения, что, как было отмечено выше, повышает способность нейронной сети к обобщению и не влечет

серьезных потерь в точности. Субдискретизация интерпретируется так: если на предыдущей операции свертки уже были выявлены некоторые признаки, то для дальнейшей обработки не требуется такое же подробное изображение, и оно уплотняется до менее подробного.

Слой подвыборки, как правило, добавляется после слоя свертки перед слоем следующей свертки. Кроме функции максимума можно использовать и другие функции подвыборки – например, функцию среднего значения или $L2$-нормирования. Однако практика показывает, что преимущество имеют именно подвыборки с функцией максимума, которые включаются в типовые системы и программные пакеты. Стоит отметить, что в целях более агрессивного уменьшения размерности моделей все чаще находят применение идеи использования меньших фильтров или полный отказ от слоев подвыборки.

Полносвязная нейронная сеть. После нескольких прохождений свертки входного сигнала и уплотнения с помощью подвыборки система перестраивается от конкретной сетки пикселей с высоким разрешением (в случае обработки изображения) к более абстрактным картам признаков. Как правило, на каждом последующем слое увеличивается число каналов («толщина» сверточного слоя) и уменьшается размерность изображения в каждом канале. В конце остается большой набор каналов, хранящих небольшое число данных (в некоторых случаях даже один параметр), которые интерпретируются как самые абстрактные понятия, выявленные из исходного сигнала. Эти данные объединяются и передаются на обычную полносвязную нейронную сеть, которая также может состоять из нескольких слоев. Полносвязные слои не имеют пространственной структуры пикселей и обладают сравнительно небольшой размерностью. Можно считать сверточную часть сети этапом выявления полезных признаков или абстракций, а полносвязную часть – непосредственно решающей исходную задачу классификации или регрессии.

3.1.2 Выбор целевой и оптимизирующей функций

Искусственные нейронные сети в целом и сверточные сети в частности не программируются в привычном понимании, а обучаются. Возможность обучения – одно из главных преимуществ нейронных сетей перед традиционными детерминированными алгоритмами. Технически обучение заключается в нахождении значений параметров сети, коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными и выходными данными, а также выполнять обобщение данных, не представленных в процессе обучения. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании отсутствовавших в обучающем наборе, а также неполных, зашумленных или частично искаженных данных.

Одним из самых распространенных методов обучения модели на основе искусственной нейронной сети является метод градиентного спуска, основанный на нахождении локального экстремума (минимума или максимума) функции с

помощью движения вдоль градиента. Для минимизации функции в направлении градиента используются методы одномерной оптимизации.

Классический метод градиентного спуска с некоторой модификацией широко применяется для обучения искусственных нейронных сетей, и также известен как метод обратного распространения ошибки: основная идея этого метода состоит в распространении сигналов ошибки от выходов сети к ее входам, т. е. в направлении, обратном прямому распространению сигнала в обычном режиме работы сети. При обучении искусственной нейронной сети требуется изменять параметры (как правило, весовые коэффициенты) таким образом, чтобы минимизировать среднюю ошибку на выходе при последовательной подаче на вход данных из обучающей выборки. Формально один шаг по методу градиентного спуска предполагает последовательную подачу на вход сети абсолютно всего набора обучающих данных, вычисление ошибки для каждого объекта обучающих данных и расчет необходимой коррекции коэффициентов сети без выполнения самой коррекции. Затем, уже после подачи всех данных, предполагается расчет суммы для корректировки каждого коэффициента сети (суммы градиентов) и собственно коррекция коэффициентов «на один шаг». При большом наборе обучающих данных алгоритм в данном режиме будет работать крайне медленно, поэтому на практике обычно производят корректировку коэффициентов сети после каждого элемента обучения или после так называемой партии входных данных (*batch*), где значение градиента аппроксимируются градиентом функции стоимости, вычисленном только на этом элементе или партии. Такой метод называют стохастическим градиентным спуском, или оперативным градиентным спуском, – он является одной из форм стохастического приближения.

Таким образом, одним из ключевых моментов обучения нейронной сети является выбор целевой функции, или функции оценки работы сети. В большинстве практических задач есть возможность оценить точность работы сети – в этом случае обучение нейронных сетей можно фактически представить как задачу оптимизации. В данном случае под оценкой работы сети понимается некая количественная характеристика, определяющая, насколько хорошо или плохо сеть решает поставленные задачи. Для этого выбирается функция оценки, которая, как правило, явно зависит от выходных сигналов сети и неявно (через функционирование) – от всех ее параметров и коэффициентов. Простейшим и самым распространенным примером функции оценки работы сети является сумма квадратов расстояний от выходных сигналов сети до их требуемых значений:

$$H = \frac{1}{2} \sum_{\tau \in v_{out}} (Z(\tau) - Z^*(\tau))^2, \tag{3.3}$$

где $Z^*(\tau)$ – требуемое значение выходного сигнала.

Следует отметить, что метод наименьших квадратов далеко не всегда является лучшим выбором для целевой функции. Тщательное конструирование

функции оценки позволяет на порядок повысить эффективность обучения сети и ее способность к обобщению, а также получать дополнительную информацию (так называемый «уровень уверенности» сети в даваемом ответе) в ряде задач. Тем не менее в данной работе в качестве функции оценки работы сети будет использован именно метод наименьших квадратов: он достаточно эффективен в контексте решаемой задачи, а благодаря его широкому применению подобная реализация присутствует в большинстве современных инструментов для работы с алгоритмами машинного обучения.

Метод градиентного спуска показывает себя очень медленным при движении «по оврагу» (овраг – эффект, возникающий, когда поверхности уровня целевой функции сильно вытянуты), причем при увеличении числа переменных целевой функции такое поведение метода становится типичным. Для борьбы с этим явлением используется множество приемов ускорения сходимости: метод оврагов, метод сопряженных градиентов, импульсный метод, метод Нестерова и пр.

Импульсный метод призван ускорить обучение (особенно в условиях высокой кривизны), небольших, но устойчивых или зашумленных градиентов, которые могут также сигнализировать о неправильной подготовке входных данных или не самой оптимальной архитектуре сети. В импульсном алгоритме вычисляется экспоненциально затухающее скользящее среднее предыдущих градиентов, которое оказывает влияние на дальнейшее направление движения спуска. Работа импульсного метода иллюстрируется на рисунке 3.3.

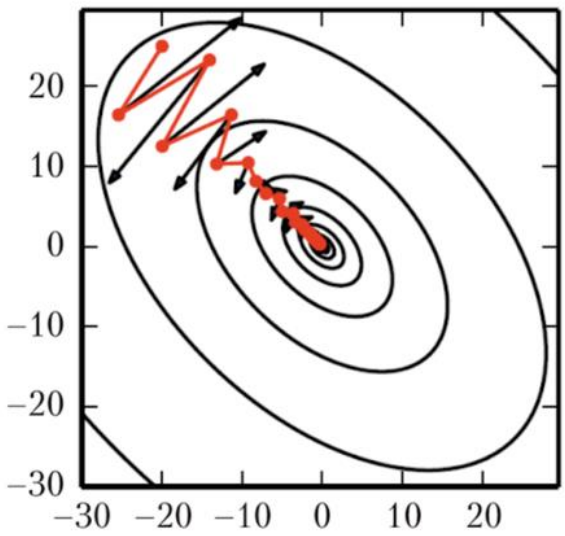

Рисунок 3.3 – Иллюстрация импульсного метода

Красная линия соответствует траектории, выбираемой в соответствии с правилом обучения импульсным методом в процессе минимизации целевой функции. Для каждого шага обучения стрелка показывает, какое направление выбрал бы в этот момент стандартный метод градиентного спуска. Как видно, плохо обусловленная квадратичная целевая функция выглядит как длинная узкая

долина, или овраг с крутыми склонами (предполагается, что локальный минимум находится в центре графика). Импульсный метод меняет направление спуска, заставляя его перемещаться вдоль оврага, тогда как обычный градиентный спуск впустую тратил бы время на цикличное перемещение поперек оврага.

Формально говоря, в импульсном алгоритме вводится переменная v, играющая роль скорости, – это направление и скорость перемещения градиентного спуска в пространстве параметров. Скорость устанавливается равной экспоненциально затухающему скользящему среднему градиента, но со знаком минус. Название алгоритма проистекает из физической аналогии импульса и скорости. Согласно этой аналогии отрицательный градиент – это ньютоновская сила, под действием которой физическая частица перемещается в пространстве параметров. В физике импульсом называется произведение массы на скорость; в импульсном алгоритме масса частицы предполагается единичной, поэтому вектор скорости v можно рассматривать как импульс частицы. Гиперпараметр a, определенный в промежутке $a \in [0, 1]$, определяет скорость экспоненциального затухания вкладов прошлых градиентов.

В классическом методе спуска размер шага равен норме градиента, умноженной на скорость обучения. Теперь же шаг зависит еще и от величины и сонаправленности предшествующих градиентов. Размер шага тем больше, чем большее количество последовательных градиентов указывает точно в одном и том же направлении. Это коррелирует с описанной выше физической аналогией: импульсный алгоритм можно рассматривать как имитацию движения физической частицы, подчиняющейся динамике Ньютона.

Также существует еще один вариант импульсного алгоритма, созданный под влиянием метода ускоренного градиента Нестерова. Ключевая разница между методом Нестерова и стандартным импульсным методом заключается в той точке, где вычисляется градиент. В методе Нестерова градиент вычисляется уже после применения текущей скорости спуска. Таким образом, метод Нестерова можно интерпретировать как попытку модифицировать стандартный импульсный метод, добавив к нему поправочный множитель. В случае выпуклой оптимизации градиентным спуском партиями метод Нестерова значительно повышает скорость сходимости с $O(\frac{1}{k})$ после k шагов до $O(\frac{1}{k^2})$.

3.2 Проектирование и обучение математических моделей для распознавания опорных точек лица в индивидуальных кадрах

3.2.1 Проектирование и обучение модели на основе неглубокой нейронной сети с одним скрытым слоем

В качестве первой модели будет спроектирована полносвязная нейронная сеть с одним скрытым слоем (перцептрон).

Перцептрон – математическая или компьютерная модель восприятия информации мозгом, которая стала одной из первых моделей искусственных

нейронных сетей. В общем случае перцептрон состоит из трех типов элементов: датчиков входных сигналов, ассоциативных элементов и реагирующих элементов. Поступающие входные сигналы передаются ассоциативным элементам, а затем – реагирующим элементам. Таким образом, перцептроны позволяют создать набор так называемых ассоциаций между входными сигналом и необходимой реакцией на выходе. В биологическом контексте это соответствует преобразованию, например, зрительной информации в физиологический ответ от двигательных нейронов.

Согласно современной терминологии перцептроны могут быть классифицированы как искусственные нейронные сети:

– с одним скрытым слоем;

– с пороговой передаточной функцией;

– с прямым распространением сигнала.

Спроектируем и разработаем первую модель решения практической задачи на основе перцептрона – фактически это будет довольно простая искусственная нейронная сеть с одним скрытым слоем. Можно предложить следующую архитектуру сети (рисунок 3.4).

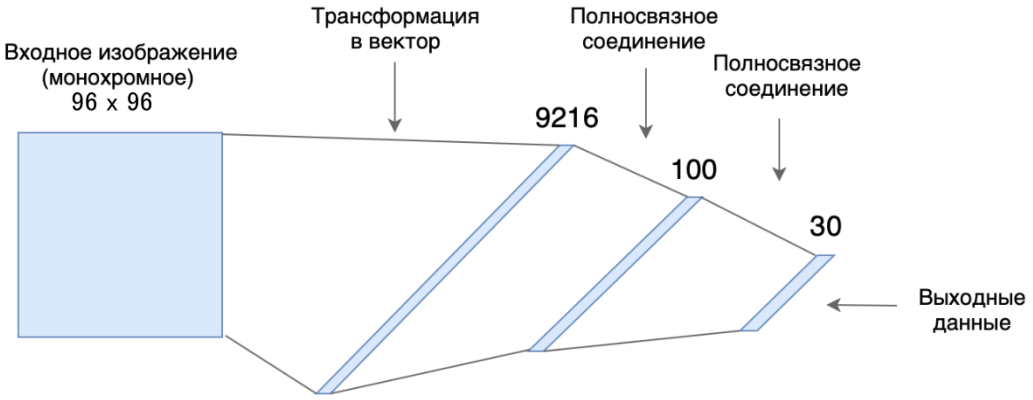

Рисунок 3.4 – Архитектура модели на основе неглубокой сети с одним скрытым слоем

Данная архитектура подразумевает прямое распространение сигнала, разделенное на следующие этапы:

1) трансформация матрицы входных данных в вектор;

2) полносвязное соединение с одним скрытым слоем размерностью 100 элементов;

3) применение линейного выпрямителя в качестве пороговой передаточной функции;

4) применение вектора выходных данных, предсказывающего координаты 15 опорных точек на исходном изображении.

Для реализации данной модели искусственной нейронной сети при помощи программного пакета *TensorFlow* будет введен набор подпрограмм, которые также будут использованы и в последующих моделях.

Для начала введем функцию распространения сигнала через слой с полносвязным соединением, используя код в листинге 3.1. В качестве параметров она принимает входные данные в виде вектора, а также размера слоя. Необходимые переменные вычислительного графа будут созданы автоматически в нужном пространстве имен, кроме того, размерность переменных будет рассчитана исходя из входных данных и правил умножения матриц и векторов.

Листинг 3.1 – Прохождение сигнала через слой с полносвязным соединением

```
def fully_connected(input, size):
    weights = tf.get_variable( 'weights',
        shape = [input.get_shape()[1], size],
        initializer = tf.contrib.layers.xavier_
initializer()
      )
    biases = tf.get_variable( 'biases',
        shape = [size],
        Initializer = tf.constant_initializer(0.0)
      )
    return tf.matmul(input, weights) + biases
```

Подпрограмма, использующая код из листинга 3.2, будет производить необходимые вычисления для сквозного прохода сигнала через всю модель. Так как данная архитектура содержит лишь один скрытый слой, проход будет выглядеть следующим образом.

Листинг 3.2 – Прохождение входного сигнала через всю модель

```
def model_pass(input):
    with tf.variable_scope('hidden'):
        hidden = fully_connected(input, size = 100)
    relu_hidden = tf.nn.relu(hidden)
    with tf.variable_scope('out'):
        prediction = fully_connected(relu_hidden,
size = 30)
    return prediction
```

Предполагается, что входные данные уже преобразованы в вектор. Далее они передаются на вход скрытому слою нейронной сети, после чего к результату применяется пороговая передаточная функция. Здесь используется линейный выпрямитель (*ReLU*) в качестве пороговой передаточной функции. Результатом является вектор с предсказаниями координат 15 опорных точек на входном изображении.

С использованием введенных утилитарных функций строится вычислительный граф для обучения модели. С помощью кода в листинге 3.3 можно выполнить проход сигнала на основе партии входных данных произвольного размера. В качестве целевой функции используется средний квадрат ошибки (*MSE*) предсказанных значений.

Листинг 3.3 – Построение графа вычислений обучения модели

```
with tf.Graph().as_default():
    # Входные данные
    tf_x_batch = tf.placeholder(tf.float32, shape =
(None, 96 * 96))
    tf_y_batch = tf.placeholder(tf.float32, shape =
(None, 30))

    # Вычисления процесса обучения модели
    with tf.variable_scope(model_variable_scope):
        predictions = model_pass(tf_x_batch)

    loss = tf.reduce_mean(tf.square(predictions -
tf_y_batch))
    # Оптимизация
    optimizer = tf.train.MomentumOptimizer(
        learning_rate = 0.01,
        momentum = 0.9,
        use_nesterov = True
    ).minimize(loss)
```

В качестве оптимизирующей функции используется импульсный метод Нестерова, который обеспечивает быструю сходимость на используемых входных данных.

Следующим этапом процесса обучения является итеративное прохождение полного массива входных данных через реализацию архитектуры модели рандомизированными партиями. Для этого используется так называемая сессия *TensorFlow* (код в листинге 3.4).

Листинг 3.4 – Процесс обучения модели

```
with tf.Session(graph = graph) as session:
    session.run(tf.global_variables_initializer())
    for epoch in range(1000):
        # Обучение на полном массиве данных рандомизи-
рованными партиями
```

```
        batch_iterator = BatchIterator(batch_size = 36,
shuffle = True)
        for x_batch, y_batch in batch_iterator(x_train,
y_train):
            session.run([optimizer], feed_dict = {
                    tf_x_batch : x_batch,
                    tf_y_batch : y_batch
                }
            )
```

Таким образом, все изображения из тренировочного массива данных будут пропущены через модель 1000 раз. Один проход всего массива принято называть эпохой, т. е. в данном случае обучение занимает 1000 эпох, или итераций.

Важной частью обучения модели является внимательное наблюдение за процессом и ключевыми параметрами самого обучения. Исходя из того, как изменяются характеристики процесса обучения, можно судить, насколько эффективен этот процесс, т. е. «учится» ли модель тому, чему ее пытаются научить.

Наиболее эффективным способом наблюдения за процессом обучения является периодическая оценка производительности модели на независимом валидационном наборе данных. Ключевым моментом является то, что этот набор данных не используется при обучении. Таким образом, точность работы модели на нем будет демонстрировать не то, насколько хорошо нейронная сеть запомнила данные, на которых она обучалась, а именно ее способность переносить эти знания на новые, ранее не представленные ей данные.

Введем несколько новых подпрограмм для более пристального наблюдения за процессом обучения. Для начала добавим подпрограмму вычисления целевой функции, высчитывающей средний квадрат ошибки предсказанных значений, используя код из листинга 3.5.

Листинг 3.5 – Вычисление значения целевой функции ошибки

```
def calc_loss(predictions, labels):
    return np.mean(np.square(predictions - labels))
```

Введем утилиту для вычисления предсказаний модели, основываясь на ее текущих параметрах, используя код из листинга 3.6. Для этого будет использоваться активная сессия *TensorFlow*, как и в листинге 3.4.

Листинг 3.6 – Вычисление предсказаний модели с текущими параметрами

```
def get_predictions_in_batches(X, session):
    p = []
    batch_iterator = BatchIterator(batch_size = 128)
    for x_batch, _ in batch_iterator(X):
```

```
        [p_batch] = session.run([predictions],
feed_dict = {
                tf_x_batch : x_batch
            }
        )
        p.extend(p_batch)
    return p
```

Модифицируем реализацию процесса обучения, добавив необходимые наблюдения в код из листинга 3.7. На каждой итерации будет сохраняться текущая производительность модели для каждой эпохи, формируя историю процесса обучения. Точность работы модели будет оцениваться как значение минимизируемой целевой функции, т. е. средний квадрат ошибки.

Листинг 3.7 – Обучение с сохранением данных для наблюдения

```
with tf.Session(graph = graph) as session:
    session.run(tf.global_variables_initializer())
    train_loss_history = np.zeros(1000)
    valid_loss_history = np.zeros(1000)
    for epoch in range(1000):
        # Обучение на полном массиве данных рандомизи-
рованными партиями
        batch_iterator = BatchIterator(batch_size = 36,
shuffle = True)
        for x_batch, y_batch in batch_iterator(x_train,
y_train):
            session.run([optimizer], feed_dict = {
                    tf_x_batch : x_batch,
                    tf_y_batch : y_batch
                }
            )
        # Сохраним значение ошибки на тренировочных
данных:
        train_loss = calc_loss(
            get_predictions_in_batches(x_train,
session),
            y_train
        )
        train_loss_history[epoch] = train_loss

        # Сохраним значение ошибки на валидационных
данных:
        valid_loss = calc_loss(
```

```
            get_predictions_in_batches(x_valid,
session),
            y_valid
        )
        valid_loss_history[epoch] = valid_loss
```

Так как процесс обучения состоит в минимизации целевой функции, значения ошибок для тренировочного и валидационного массива данных должны со временем падать, пусть и не на каждой итерации.

Итоговое значение функции ошибки модели, обученной на валидационном наборе данных, составило 0,00272522. Рассмотрим график изменения значений ошибки на обоих массивах данных в зависимости от эпохи (т. е. итерации) на рисунке 3.5.

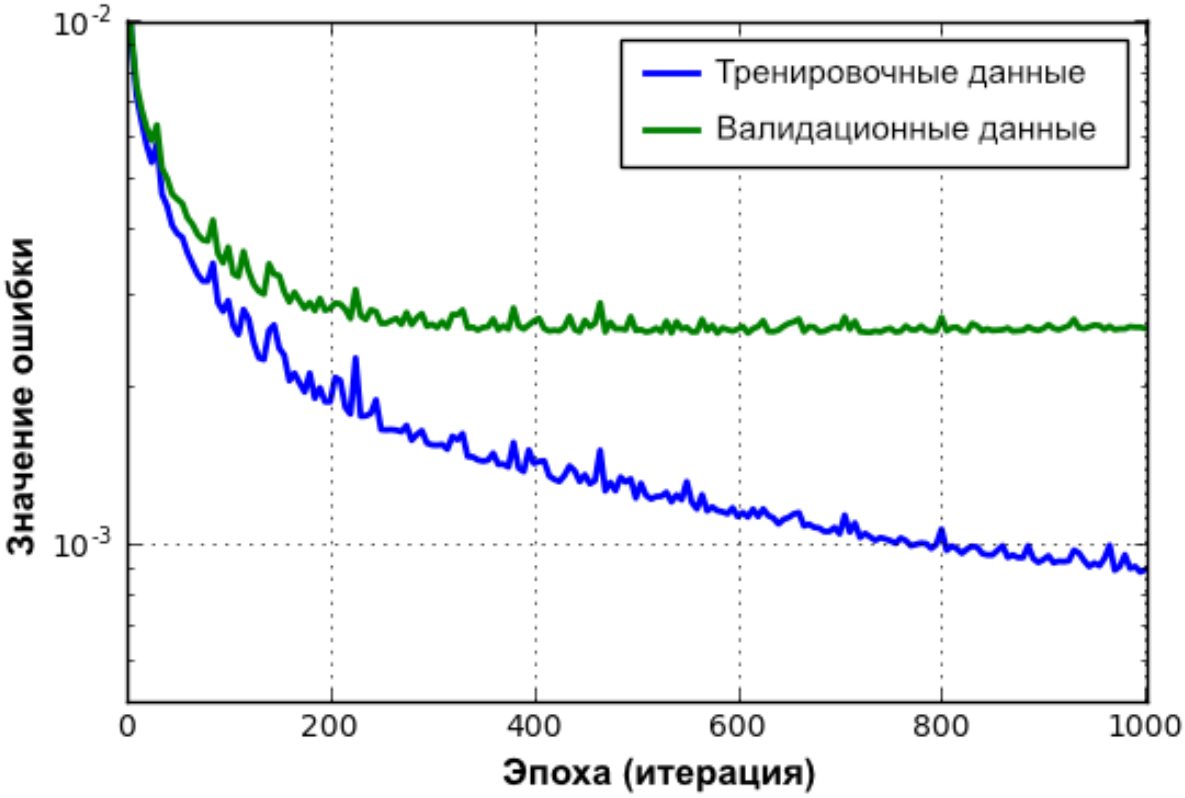

Рисунок 3.5 – График изменения значения ошибки в процессе обучения

Из графика на рисунке 3.5 можно сделать ряд выводов относительно процесса обучения:

– точность прогнозов выше на тренировочных данных;

– значения ошибок прогнозов модели на обоих массивах данных падает в процессе обучения, а точность, соответственно, возрастает.

Более высокая точность работы на тренировочных данных связана с тем, что это именно та выборка, на которой модель обучается. Данные из этого набора

участвуют в распространении градиентов в параметрах модели, соответственно, более высокая точность на этой выборе выглядит закономерно.

Также вполне закономерным выглядит снижение ошибки для обоих выборок исходных данных; но стоит отметить, что после порядка 300 итераций значение ошибки на валидационных данных перестает снижаться, в то время как ошибка на тренировочных данных продолжает падать. Это явный признак того, что после 300 итераций модель достигает своей максимальной производительности для текущей архитектуры и гиперпараметров. Дальнейшее обучение лишь позволяет модели подстраиваться под конкретные тренировочные данные, но не дает ей возможности узнавать что-либо новое и обобщать полученные знания, применяя их к новым, ранее не представлявшимся ей данным.

Такое явление называют переобучением, или переподгонкой – построенная модель хорошо понимает примеры из обучающей выборки, но относительно плохо работает на примерах, не участвовавших в обучении, т. е. на примерах из тестовой выборки или, как в рассматриваемом случае, на примерах из валидационной выборки. Это связано с тем, что в процессе обучения в обучающей выборке обнаруживаются некоторые случайные закономерности, которые отсутствуют в генеральной совокупности или в валидационном наборе данных. Модель не в состоянии отличить случайные закономерности от неслучайных и просто пытается как можно лучше предсказывать значения на тренировочном массиве данных.

Из рассмотренного случая можно сделать вывод, что обучения на 300 итерациях будет достаточно и дальнейшее обучение не принесет прироста точности модели на произвольных данных.

Далее будут рассмотрены способы и методы обучения моделей на основе алгоритмов машинного обучения, которые позволят не тратить лишнее время на обучение, если модель не получает новых знаний. На данном этапе это не столь важно, т. к. обучение неглубокой нейронной сети проходит достаточно быстро даже без использования специализированного оборудования. Обучение же моделей, рассматриваемых далее, потребует значительных временных затрат, поэтому любой способ ускорения процесса будет полезен. В целом анализ архитектуры и наблюдение за процессом обучения состоят в итеративном поиске оптимальных гиперпараметров модели, и чем меньше времени занимает ее обучение, тем больше времени остается на эксперименты с различными значениями этих гиперпараметров.

Рассмотрим несколько примеров прогнозов модели на исходных данных (рисунок 3.6).

Как можно судить по этим примерам, точность прогнозов модели неплоха в целом, но тем не менее оставляет желать лучшего на отдельных примерах, несмотря на то, что, исходя из наблюдений за процессом обучения, она достигает максимума своих возможностей усвоения информации.

Рисунок 3.6 – Примеры предсказаний неглубокой сети с одним скрытым слоем

3.2.2 Проектирование и обучение модели на основе сверточной нейронной сети

Попробуем улучшить точность модели, использовав архитектуру на основе сверточной нейронной сети. Сверточные нейронные сети являются одним из лучших алгоритмов в сфере компьютерного зрения и работы с изображениями и успешно применяются в прикладных задачах, схожих с рассматриваемой.

Основываясь на типовых подходах к построению сверточных нейронных сетей, реализуем модель на основе следующей нейронной сети, изображенной на рисунке 3.7.

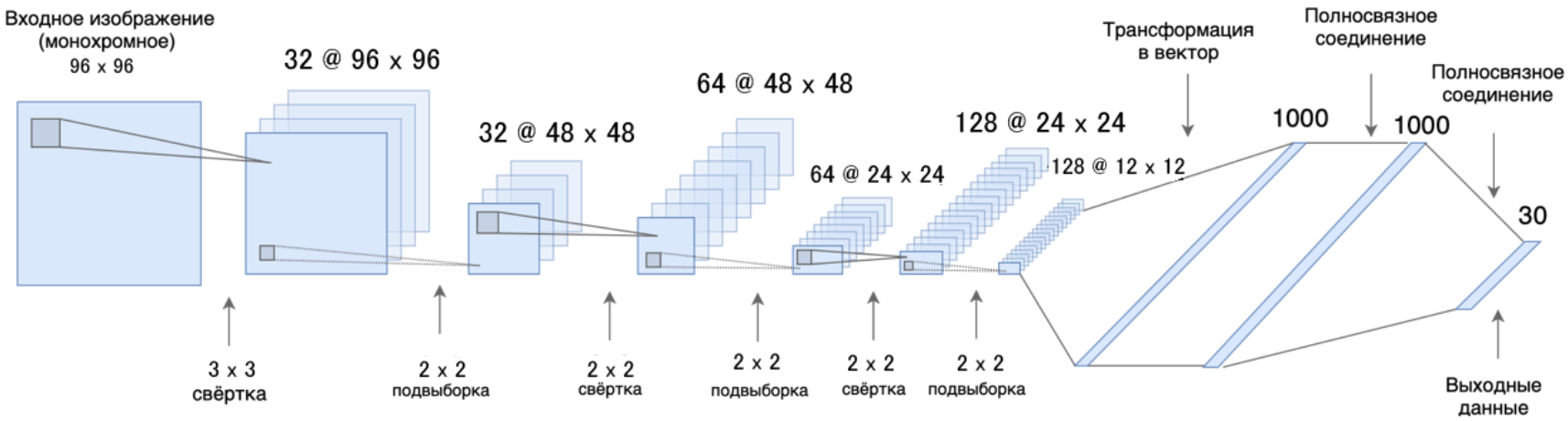

Рисунок 3.7 – Архитектура модели на основе сверточной нейронной сети

Данная архитектура подразумевает прохождение сигнала через следующие этапы:

1) свертка с ядром размерностью 3 × 3 в слой размерностью 32 с линейным выпрямителем в качестве функции активации;

2) операция подвыборки методом пулинга с функцией максимума, уплотнение группы пикселей размером 2 × 2 до одного пикселя;

3) свертка с ядром размерностью 2 × 2 в слой размерностью 64 с линейным выпрямителем в качестве функции активации;

4) операция подвыборки методом пулинга с функцией максимума, уплотнение группы пикселей размером 2 × 2 до одного пикселя;

5) свертка с ядром размерности 2 × 2 в слой размерностью 128 с линейным выпрямителем в качестве функции активации;

6) операция подвыборки методом пулинга с функцией максимума, уплотнение группы пикселей размером 2 × 2 до одного пикселя;

7) трансформация матричного сигнала в вектор;

8) полносвязное соединение со скрытым слоем размерностью 1000 элементов;

9) полносвязное соединение с еще одним скрытым слоем размерностью 1000 элементов;

10) полносвязное соединение с выходным вектором размерностью 30 элементов (количество координат предсказываемых опорных точек во входных данных).

Условно данную архитектуру можно разделить на сверточную часть, состоящую из трех сверточных слоев, и полносвязную часть, состоящую из двух полносвязных слоев.

Часто при построении архитектуры модели на основе искусственных нейронных сетей возникает вопрос выбора гиперпараметров: как определить оптимальное число слоев, оптимальное количество нейронов в слое, темп обучения и прочее. К сожалению, не существует универсального рецепта подбора гиперпараметров – в той или иной степени они уникальны для каждой решаемой задачи. Практически всегда окончательная архитектура модели является результатом долгого и кропотливого поиска оптимальной комбинации гиперпараметров. Как правило, этот поиск состоит в итеративных экспериментах с различными значениями каждого из них и внимательного наблюдения за успешностью обучения и изменениями в точности работы модели.

Для реализации данной модели расширим набор вспомогательных подпрограмм. Введем функцию распространения сигнала через слой с полносвязным соединением с дополнительным применением функции активации в виде линейного выпрямителя, используя код из листинга 3.8.

Листинг 3.8 – Прохождение сигнала через слой с полносвязным соединением

```
def fully_connected_relu(input, size):
    return tf.nn.relu(fully_connected(input, size))
```

Далее введем функцию свертки, которая пропустит входной сигнал через сверточный слой с указанным размером ядра и размерностью, а также применит функцию активации, с помощью кода из листинга 3.9. В данной модели использован шаг свертки размером 1 – это означает, что при переборе слоя матрицей весов она будет передвигаться на один нейрон (пиксель).

Листинг 3.9 – Прохождение сигнала через слой свертки

```
def conv_relu(input, kernel_size, depth):
    weights = tf.get_variable( 'weights',
        shape = [kernel_size, kernel_size,
input.get_shape()[3], depth],
        initializer = tf.contrib.layers.xavier_
initializer()
      )
    biases = tf.get_variable( 'biases',
        shape = [depth],
        Initializer = tf.constant_initializer(0.0)
      )
    conv = tf.nn.conv2d(input, weights,
        Strides = [1, 1, 1, 1], padding = 'SAME')
    return tf.nn.relu(conv + biases)
```

Также введем подпрограмму прохождения сигнала через слой подвыборки методом пулинга с функцией максимума, используя код из листинга 3.10. Данный слой будет применяться после каждого слоя свертки.

Листинг 3.10 – Прохождение сигнала через слой подвыборки

```
def pool(input, size):
    return tf.nn.max_pool(input, ksize = [1, size,
size, 1],
        strides = [1, size, size, 1], padding = 'SAME'
    )
```

По аналогии с моделью на основе перцептрона, используя введенные подпрограммы, реализуем функцию полного прохождения сигнала через всю модель, т. е. через всю сверточную нейронную сеть, используя код из листинга 3.11.

Листинг 3.11 – Прохождение сигнала через всю модель

```
def model_pass(input):
    # Сверточные слои
    with tf.variable_scope('conv1'):
        conv1 = conv_relu(input, kernel_size = 3,
depth = 32)
        pool1 = pool(conv1, size = 2)
    with tf.variable_scope('conv2'):
        conv2 = conv_relu(pool1, kernel_size = 2,
depth = 64)
        pool2 = pool(conv2, size = 2)
    with tf.variable_scope('conv3'):
        conv3 = conv_relu(pool2, kernel_size = 2,
depth = 128)
        pool3 = pool(conv3, size = 2)

    # Трансформация матричного сигнала в вектор
    shape = pool3.get_shape().as_list()
    flattened = tf.reshape(pool3, [-1, shape[1] *
shape[2] * shape[3]])

    # Полносвязные соединения со скрытыми слоями
    with tf.variable_scope('fc4'):
        fc4 = fully_connected_relu(flattened,
size = 1000)
    with tf.variable_scope('fc5'):
        fc5 = fully_connected_relu(fc4, size = 1000)
    with tf.variable_scope('out'):
        prediction = fully_connected(fc5, size = 30)
    return prediction
```

Хотя данная функция точно реализует прохождение сигнала через модели, в нее необходимо внести еще одно небольшое, но существенное изменение. Сверточные сети имеют значительно большее количество параметров, чем, скажем, полносвязные нейронные сети с прямым прохождением сигнала. Вследствие этого модели на основе сверточных нейронных сетей склонны к переобучению, поэтому для успешного обучения необходимо принять меры предотвращения переобучения еще на этапе реализации модели, а именно добавить возможность так называемого исключения, затухания скорости обучения и видоизменения входных данных.

Исключение (*dropout*) – метод регуляризации искусственных нейронных сетей, предназначенный для предотвращения переобучения сети.

В стандартной нейронной сети производная, полученная каждым параметром, сообщает ему, как он должен измениться, чтобы, учитывая деятельность остальных блоков, минимизировать функцию конечных потерь. Поэтому блоки могут меняться, исправляя при этом ошибки других блоков. Это может привести к чрезмерной совместной адаптации и чрезмерно тесной связи между индивидуальными нейронами, что, в свою очередь, приводит к переобучению, поскольку эти совместные адаптации невозможно обобщить в отношении данных, не участвовавших в обучении.

Суть метода исключения заключается в том, что в процессе обучения выбирается слой, из которого случайным образом выбрасывается определенное количество нейронов (например, 30 %), которые выключаются из дальнейших вычислений. Такой прием улучшает эффективность обучения и качество результата, т. к. более обученные нейроны получают в сети больший вес. Фактически метод исключения предотвращает совместную адаптацию для каждого скрытого блока нейронов, делая присутствие других скрытых блоков ненадежным. Поэтому скрытый блок не может полагаться на другие блоки в исправлении собственных ошибок.

Стоит отметить, что метод исключения предназначен не только для использования со сверточными нейронными сетями – он показывает отличные результаты регуляризации слоев сетей произвольного типа. К примеру, в данной модели исключение будет применено и к сверточным слоям, и к полносвязным. Для этого модифицируем подпрограмму, реализующую прохождение сигнала через модель, используя код из листинга 3.12.

Листинг 3.12 – Прохождение сигнала через модель с возможностью исключения

```
def model_pass(input, training):
    # Сверточные слои
    with tf.variable_scope('conv1'):
        conv1 = conv_relu(input, kernel_size = 3,
depth = 32)
        pool1 = pool(conv1, size = 2)
        # Применить исключение нейронов, но только в
процессе обучения
        pool1 = tf.cond(training,
            lambda: tf.nn.dropout(pool1,
keep_prob = 0.9),
            lambda: pool1
        )
    with tf.variable_scope('conv2'):
        conv2 = conv_relu(pool1, kernel_size = 2,
depth = 64)
        pool2 = pool(conv2, size = 2)
```

```
        # Применить исключение нейронов, но только в
процессе обучения
        pool2 = tf.cond(training,
            lambda: tf.nn.dropout(pool2,
keep_prob = 0.8)
            lambda: pool2
        )
    with tf.variable_scope('conv3'):
        conv3 = conv_relu(pool2, kernel_size = 2,
depth = 128)
        pool3 = pool(conv3, size = 2)
        # Применить исключение нейронов, но только в
процессе обучения
        pool3 = tf.cond(training,
            lambda: tf.nn.dropout(pool3, keep_prob = 0.7),
            lambda: pool3
        )

    # Трансформация матричного сигнала в вектор
    shape = pool3.get_shape().as_list()
    flattened = tf.reshape(pool3, [-1, shape[1] *
shape[2] * shape[3]])

    # Полносвязные соединения со скрытыми слоями
    with tf.variable_scope('fc4'):
        fc4 = fully_connected_relu(flattened,
size = 1000)
        # Применить исключение нейронов, но только в
процессе обучения
        fc4 = tf.cond(training,
            lambda: tf.nn.dropout(fc4, keep_prob = 0.5),
            lambda: fc4
        )
    with tf.variable_scope('fc5'):
        fc5 = fully_connected_relu(fc4, size = 1000)
    with tf.variable_scope('out'):
        prediction = fully_connected(fc5, size = 30)
    return prediction
```

Исключение будет применено следующим образом:

– 10 % нейронов будет исключено случайным образом на первом сверточном слое;

– 20 % нейронов будет исключено случайным образом на втором сверточном слое;

– 30 % нейронов будет исключено случайным образом на третьем сверточном слое;

– 50 % нейронов будет исключено случайным образом на первом полносвязном скрытом слое.

Второй полносвязный слой не будет участвовать в исключении. Кроме того, важно отметить, что исключение должно применяться только в процессе обучения. Так как приведенная подпрограмма прохождения сигнала будет использоваться и при обучении, и при оценке работы сети, вводится дополнительная переменная, которая включает или выключает операцию исключения нейронов на всех задействованных в исключении слоях.

Затухание – еще одна стратегия регуляризации искусственных нейронных сетей, предназначенная для предотвращения переобучения сети. Суть ее состоит в изменении значения скорости (или темпа) обучения со временем, т. е. от эпохи к эпохе. В данной модели будет использовано экспоненциальное затухание скорости обучения, реализованное в программном пакете *TensorFlow*. Вместо использования константы значение темпа обучения будет зависеть от текущей итерации (эпохи), как показано в листинге 3.13.

Листинг 3.13 – Затухание темпа обучения

```
learning_rate = tf.train.exponential_decay(0.03,
current_epoch, decay_steps = num_epochs,
decay_rate = 0.03)
```

Используя описанную методику, модифицируем реализацию процесса оптимизации целевой функции, используя код из листинга 3.14.

Листинг 3.14 – Построение графа вычислений для обучения модели

```
with tf.Graph().as_default():
    # Входные данные
    tf_x_batch = tf.placeholder(tf.float32, shape =
(None, 96, 96, 1))
    tf_y_batch = tf.placeholder(tf.float32, shape =
(None, 30))
    is_training = tf.placeholder(tf.bool)
    # Порядковый номер текущей итерации
    current_epoch = tf.Variable(0)

    # Затухание темпа обучения
    learning_rate = tf.train.exponential_decay(
        0.03,
        current_epoch,
        decay_steps = 1000,
```

```
        decay_rate = 0.03
    )

    # Вычисления процесса обучения модели
    with tf.variable_scope(model_variable_scope):
        predictions = model_pass(tf_x_batch,
is_training)

    loss = tf.reduce_mean(tf.square(predictions -
tf_y_batch))

    # Оптимизация целевой функции
    optimizer = tf.train.MomentumOptimizer(
        learning_rate = learning_rate,
        momentum = 0.9,
        use_nesterov = True
    ).minimize(loss)
```

Прежде чем перейти к реализации процесса обучения данной модели, применим еще один весьма эффективный способ регуляризации и борьбы с переобучением сети – приращение данных.

Используем описанный ранее способ приращения данных, основанный на зеркальном отражении по горизонтали. Можно было бы просто отразить все исходные данные, увеличив их объем в два раза, но это значительно замедлит процесс обучения, т. к. на каждой эпохе оптимизатору будет подан на вход в два раза больший объем данных. Чтобы оптимизировать процесс обучения, разработаем специальный компонент для итеративной обработки данных партиями, с дополнительным применением операции зеркального отражения с вероятностью 0,5. Таким образом, при обработке каждого изображения оно будет инвертировано с вероятностью 0,5, учитывая:

– что на каждой эпохе оптимизатор пропускает через модель весь объем данных;

– обучение проходит на сотнях эпох (до тысячи).

Каждое изображение будет использовано в процессе обучения и в исходном виде, и в инвертированном примерно одинаковое (и значительное) количество раз.

Реализуем компонент итеративной обработки партиями со случайным зеркальным отражением по горизонтали, используя код из листинга 3.15. Главной сложностью является то, что часть размеченных координат опорных точек необходимо просто отразить по горизонтали (например, кончик носа), а часть – перенести из одного класса в другой, а вернее, поменять местами (например, координаты правого уголка губ должны быть трансформированы в левый).

Листинг 3.15 – Компонент обработки данных партиями с аугментацией

```
class FlipBatchIterator(BatchIterator):
    # Правила инвертирования опорных точек
    flip_indices = [
        (0, 2), (1, 3),
        (4, 8), (5, 9), (6, 10), (7, 11),
        (12, 16), (13, 17), (14, 18), (15, 19),
        (22, 24), (23, 25),
    ]
    def transform(self, Xb, yb):
        Xb, yb = super(FlipBatchIterator, self).
transform(Xb, yb)
        # Случайное отражение половины изображений в
партии
        bs = Xb.shape[0]
        indices = np.random.choice(bs, int(bs / 2),
replace = False)
        Xb[indices] = Xb[indices, :, ::-1, :]
        if yb is not None:
            # Часть координат просто отражается горизон-
тально по оси X
            yb[indices, ::2] = yb[indices, ::2] * -1

            # Часть координат меняется местами, например,
координаты центра левого глаза трансформируются в коорди-
наты центра правого глаза
            for a, b in self.flip_indices:
                yb[indices, a], yb[indices, b] = (
                    yb[indices, b], yb[indices, a])
        return Xb, yb
```

Используя этот компонент, реализуем процесс обучения с помощью кода из листинга 3.16. Как и прежде, на каждой итерации будет сохраняться история значений функции ошибки или потери (целевой функции) для наблюдения за процессом обучения и сравнения точности с предыдущей моделью.

Листинг 3.16 – Процесс обучения модели

```
with tf.Session(graph = graph) as session:
    session.run(tf.global_variables_initializer())
    saver = tf.train.Saver()
    train_loss_history = np.zeros(1000)
    valid_loss_history = np.zeros(1000)
    for epoch in range(1000):
```

```
        current_epoch = epoch
        # Обучение на полном массиве данных рандомизиро-
ванными партиями
        batch_iterator = FlipBatchIterator(
            batch_size = 36,
            shuffle = True
        )
        for x_batch, y_batch in batch_iterator(x_train,
y_train):
            session.run([optimizer], feed_dict = {
                    tf_x_batch : x_batch,
                    tf_y_batch : y_batch,
                    is_training : True
                }
            )
        # Сохраним значение ошибки на тренировочных
данных
        train_loss = calc_loss(
            get_predictions_in_batches(x_train,
session),
            y_train
        )
        train_loss_history[epoch] = train_loss
        # Сохраним значение ошибки на валидационных
данных
        valid_loss = calc_loss(
            get_predictions_in_batches(x_valid,
session),
            y_valid
        )
        valid_loss_history[epoch] = valid_loss
```

Как и в случае предыдущей архитектуры сети, процесс обучения состоит в минимизации целевой функции, соответственно, значения ошибок для тренировочного и валидационного массива данных должны со временем падать, пусть и не на каждой итерации.

Итоговое значений функции ошибки обученной модели на валидационном наборе данных составило 0,00112052, что более чем в два раза ниже значения ошибки модели на основе перцептрона. Рассмотрим график изменения значений ошибки (рисунок 3.8) на обоих массивах данных в зависимости от эпохи (т. е. итерации), добавив их к графику с соответственными значениями предыдущей архитектуры нейронной сети с одним скрытым слоем.

Из графика, представленного на рисунке 3.8, можно сделать ряд выводов относительно процесса обучения:

– точность прогнозов на тренировочных данных примерно одинакова в процессе обучения для обеих моделей;

– точность прогнозов на валидационных данных значительно выше для модели на основе сверточной нейронной сети;

– обучение сверточной нейронной сети происходит значительно более равномерно.

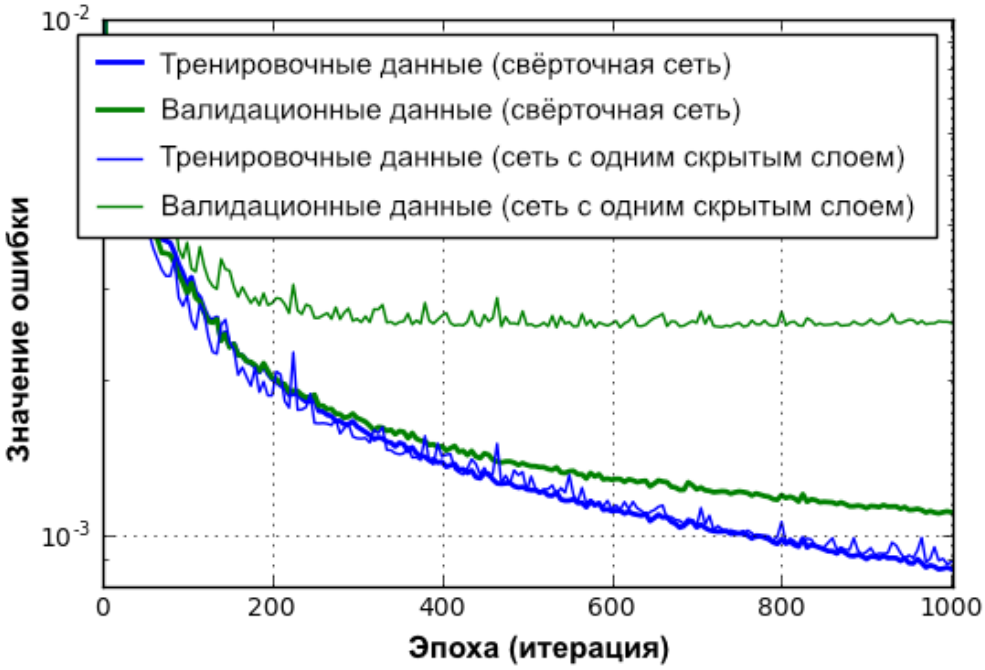

Рисунок 3.8 – График изменения значения ошибки в процессе обучения

Более высокая точность работы на тренировочных данных обоих моделей вполне закономерна. Интересна разница в точности на валидационных данных – это именно те данные, которые в обучении не участвуют, соответственно, при оценке точности модель видит эти данные впервые. Здесь с огромным отрывом лидирует модель на основе сверточной нейронной сети. Причем в то время как модель на основе сети с одним скрытым слоем прекращает обучаться в районе 300 итерации, по всей видимости, достигая максимальной вместимости знаний, модель на основе сверточной сети продолжает дообучаться, улучшая точность работы на валидационных данных вплоть до 1000 итераций.

Такой эффект, безусловно, достигается значительно большим количеством параметров, чем в модели на основе нейронной сети с одним скрытым слоем. Большее количество параметров означает более значительную общую вместимость знаний, т. е. объем знаний, которые модель может усвоить в принципе. Но эта дополнительная вместимость была бы бесполезной, и модель никогда не смогла бы достигнуть своего потенциала без оптимально подобранных гиперпараметров и, главное, техник регуляризации, препятствующих переобучению модели: исключения, затухания скорости обучения и видоизменения входных данных.

Также заметно увеличение скорости полезного обучения, т. е. модель обнаруживает необходимые закономерности уже после нескольких сотен итераций – и предсказывает искомые значения значительно более точно, чем предыдущая модель на основе неглубокой сети с одним скрытым слоем после такого же количества эпох обучения. В дальнейшем будут применены дополнительные методы для ускорения (а точнее, прекращения) процесса обучения, в тех ситуациях, когда он перестает быть полезным с точки зрения получения моделью новой информации.

Одним из эффектов данных мер также является более плавный процесс обучения без значительных скачков значения функции ошибки. Благодаря регуляризации (как явной, так и неявной за счет использования слоев свертки, которые сами по себе являются регуляризаторами) процесс поиска локального минимума функции проходит более равномерно и без заметных блужданий вокруг искомого значения.

Рассмотрим несколько примеров прогнозов модели на исходных данных (рисунок 3.9).

Как можно видеть, модель на основе сверточной нейронной сети показывает заметно более высокую точность по сравнению с моделью на основе нейронной сети с одним скрытым слоем.

Рисунок 3.9 – Примеры предсказаний сверточной сети

3.2.3 Проектирование и обучение группы специализированных моделей

Точность предсказаний модели можно увеличить, использовав особенности исходного массива данных. Рассмотрим распределение маркированных данных по классам, а вернее, сведения, насколько каждая искомая ключевая точка лица представлена в массиве данных (рисунок 3.10).

Для упрощения процесса обучения в первых двух моделях часть исходных данных отбрасывалась – оставлялись лишь те примеры, в которых присутствовали все опорные точки. Таким образом, для части точек отбрасывались почти две трети представленных данных, что, разумеется, довольно расточительно, учитывая ценность маркированных данных для производительности модели.

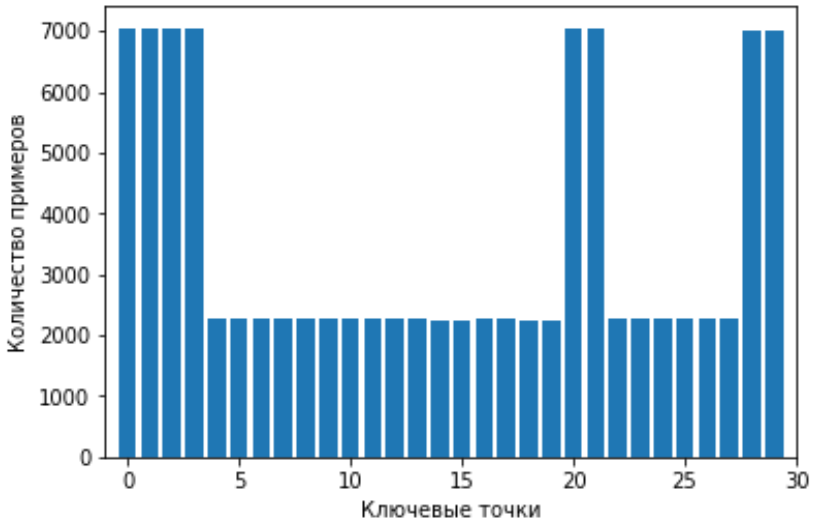

Рисунок 3.10 – Распределение исходных данных по опорным точкам

Рассмотрим более практичный подход к использованию исходных данных. Вместо обучения одной модели на всех исходных данных разобьем задачу на шесть подзадач, исходя из групп лицевых точек. Для каждой подзадачи обучим отдельную модель, используя только примеры, в которых представлены искомые лицевые точки. Общая модель будет представлять из себя группу сверточных нейронных сетей (ансамбль), работающую как комитет экспертов в каждой из индивидуальных подзадач. Стоит отметить, что данный подход является достаточно популярным и применяется не только в задачах компьютерного зрения.

Исходя из объема маркированных данных для каждого класса, а также схожести признаков, которые модель будет изучать, разобьем искомые лицевые точки на следующие группы:

– группа 1: центр правого глаза, центр левого глаза;
– группа 2: кончик носа;
– группа 3: левый уголок рта, правый уголок рта, центр верхней губы;
– группа 4: центр нижней губы;

– группа 5: внутренний уголок левого глаза, внутренний уголок правого глаза, внешний уголок левого глаза, внешний уголок правого глаза;

– группа 6: внутренний кончик левой брови, внутренний кончик правой брови, внешний кончик правой брови, внутренний кончик правой брови.

Для каждой подзадачи будет обучена отдельная модель – так называемый «специалист» (*specialist*). Для каждого специалиста будет использована архитектура модели на основе сверточной нейронной сети, представленная на рисунке 3.11, схожая с архитектурой предыдущей модели.

Стоит отметить, что данный подход не обязательно является оптимальным, и при дальнейшем изучении поставленной задачи возможна разработка индивидуальных архитектур для каждого специалиста, учитывающих особенности решаемых подзадач, которые покажут более высокую точность в целом [31].

Рассматриваемая модель становится все более сложной, и для того, чтобы более эффективно реализовать обучение, введем структуру, описывающую настройки индивидуальных моделей-специалистов, используя код из листинга 3.17.

Листинг 3.17 – Структура описания специалиста из ансамбля моделей

```
SPECIALIST_SETTINGS = [
    dict(
        name = 'Specialist Name',
        columns = ('column_1', 'column_2'),
        flip_indices = (),
    )
]
```

Данная структура будет содержать следующие настройки:

– название модели-специалиста;

– идентификаторы столбцов исходных данных, содержащих координаты опорных точек, которые предсказывает данный специалист;

– индексы координат, которые необходимо инвертировать при расширении (аугментации) исходных данных при обучении.

Используя код из листинга 3.18, ведем настройки для всех шести моделей.

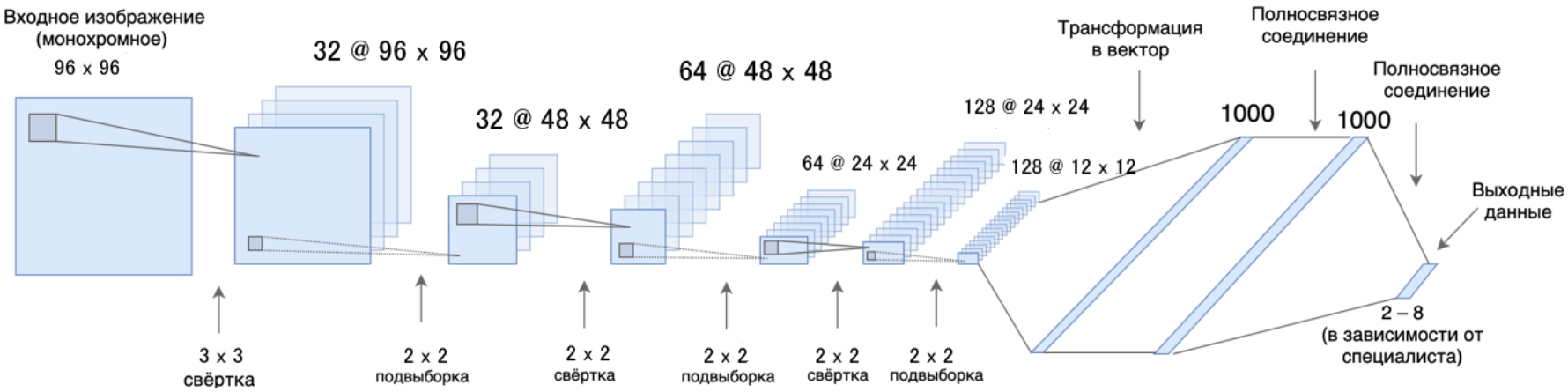

Рисунок 3.11 – Архитектура сверточной сети из ансамбля моделей

Листинг 3.18 – Настройки всех специалистов в ансамбле моделей

```
SPECIALIST_SETTINGS = [
    # Группа 1
    dict(
        name = "eye_center",
        columns = (
            'left_eye_center_x', 'left_eye_center_y',
            'right_eye_center_x', 'right_eye_center_y',
            ),
        flip_indices = ((0, 2), (1, 3)),
    ),
    # Группа 2
    dict(
        name = "nose_tip",
        columns = (
            'nose_tip_x', 'nose_tip_y',
            ),
        flip_indices = (),
    ),
    # Группа 3
    dict(
        name = "mouth_corner_top",
        columns = (
            'mouth_left_corner_x', 'mouth_left_
corner_y',
            'mouth_right_corner_x', 'mouth_right_
corner_y',
            'mouth_center_top_lip_x', 'mouth_
center_top_lip_y',
            ),
        flip_indices = ((0, 2), (1, 3)),
    ),
    # Группа 4
    dict(
        name = "mouth_bottom",
        columns = (
            'mouth_center_bottom_lip_x',
            'mouth_center_bottom_lip_y',
            ),
        flip_indices = (),
    ),
    # Группа 5
    dict(
```

```
        name = "eye_corner",
        columns = (
            'left_eye_inner_corner_x', 'left_eye_
inner_corner_y',
            'right_eye_inner_corner_x', 'right_eye_
inner_corner_y',
            'left_eye_outer_corner_x',
'left_eye_outer_corner_y',
            'right_eye_outer_corner_x',
'right_eye_outer_corner_y',
            ),
        flip_indices = ((0, 2), (1, 3), (4, 6), (5, 7)),
    ),
    # Группа 6
    dict(
        name = "eyebrow",
        columns = (
            'left_eyebrow_inner_end_x', 'left_
eyebrow_inner_end_y',
            'right_eyebrow_inner_end_x', 'right_
eyebrow_inner_end_y',
            'left_eyebrow_outer_end_x', 'left_
eyebrow_outer_end_y',
            'right_eyebrow_outer_end_x', 'right_
eyebrow_outer_end_y',
            ),
        flip_indices = ((0, 2), (1, 3), (4, 6), (5, 7)),
    ),
]
```

С помощью данных настроек можно более эффективно и сжато реализовать обучение всех специалистов, используя общий набор компонентов и подпрограмм обучения моделей.

Реализуем прохождение сигнала через модель, используя архитектуру, описанную выше. Как было указано ранее, для всех специалистов будет использована одинаковая архитектура, поэтому достаточно реализовать прохождение сигнала один раз в виде подпрограммы, используя код из листинга 3.19, а затем применить ее в процессе обучения для всех специалистов, но с разными настройками.

Листинг 3.19 – Прохождение сигнала через модель одного из специалистов

```
def model_pass(input, keypoints, training):
    # Сверточные слои
    with tf.variable_scope('conv1'):
        conv1 = conv_relu(input, kernel_size = 3,
depth = 32)
        pool1 = pool(conv1, size = 2)
        # Применить исключение нейронов, но только в про-
цессе обучения
        pool1 = tf.cond(training,
            lambda: tf.nn.dropout(pool1, keep_prob = 0.9),
            lambda: pool1
        )
    with tf.variable_scope('conv2'):
        conv2 = conv_relu(pool1, kernel_size = 2,
depth = 64)
        pool2 = pool(conv2, size = 2)
        # Применить исключение нейронов, но только в про-
цессе обучения
        pool2 = tf.cond(training,
            lambda: tf.nn.dropout(pool2, keep_prob = 0.8)
            lambda: pool2
        )
    with tf.variable_scope('conv3'):
        conv3 = conv_relu(pool2, kernel_size = 2,
depth = 128)
        pool3 = pool(conv3, size = 2)
        # Применить исключение нейронов, но только в про-
цессе обучения
        pool3 = tf.cond(training,
            lambda: tf.nn.dropout(pool3, keep_prob = 0.7),
            lambda: pool3
        )

    # Трансформация матричного сигнала в вектор
    shape = pool3.get_shape().as_list()
    flattened = tf.reshape(pool3, [-1, shape[1] *
shape[2] * shape[3]])

    # Полносвязные соединения со скрытыми слоями
    with tf.variable_scope('fc4'):
        fc4 = fully_connected_relu(flattened,
size = 1000)
```

```
        # Применить исключение нейронов, но только в про-
цессе обучения
        fc4 = tf.cond(training,
            lambda: tf.nn.dropout(fc4, keep_prob = 0.5),
            lambda: fc4
        )
    with tf.variable_scope('fc5'):
        fc5 = fully_connected_relu(fc4, size = 1000)
    with tf.variable_scope('out'):
        prediction = fully_connected(fc5, size =
keypoints)
    return prediction
```

Обучение сверточной нейронной сети может требовать значительное количество времени, и, как было продемонстрировано ранее, количество в 1000 эпох не обязательно является оптимальным для определенных задач. Велика вероятность, что для некоторых из указанных подзадач модель достигнет оптимальной точности гораздо раньше и после определенного количества итераций не будет обобщать полученные знания. Чтобы ускорить процесс обучения и не продолжать его впустую, а также предотвратить переобучение модели, используем метод ранней остановки.

В машинном обучении ранняя остановка – это форма регуляризации, используемая для избегания переобучения модели при обучении с учителем для итеративных методов обучения, таких как градиентный спуск. Подобные итеративные методы обновляют параметры модели на каждой итерации, стараясь минимизировать значение целевой функции (функции ошибки) на тренировочных данных. До определенной степени этот процесс улучшает производительность модели и при работе с данными за пределами учебного набора. Однако в какой-то момент повышение точности модели на тренировочных данных начинает происходить за счет ухудшения обобщения полученных знаний. Метод ранней остановки устанавливает ограничения на количество итераций при обучении, рекомендуя прекратить обучение при обнаружении признаков переобучения модели.

Воспользуемся методом ранней остановки на основе значения функции ошибки для валидационного набора данных. Для этого разработаем компонент (листинг 3.20), который будет наблюдать за историей точности модели от эпохи к эпохе и рекомендовать прекратить обучение, когда точность на валидационном наборе данных перестанет улучшаться на настраиваемом количестве последних итераций.

Листинг 3.20 – Компонент ранней остановки

```
class EarlyStopping(object):
    def __init__(self, saver, session, patience = 100,
minimize = True):
```

```
        """
        # Инициализирует обьект класса
        Parameters
        ----------
        saver   :
                 # Обьект TensorFlow Saver, используемый
для сохранения и восстановления параметров модели на
определенной итерации
        session :
                 # Обьект TensorFlow Session, содержащий
текущий граф вычислений модели
        patience:
                 # Предел ожидания улучшения точности мо-
дели. Количество итераций, которое данный компонент будет
ожидать для улучшения показателей производительности,
прежде чем он восстановит лучшие параметры модели на дан-
ный момент и рекомендует остановку обучения
        """
        self.minimize = minimize
        self.patience = patience
        self.saver = saver
        self.session = session
        self.best_monitored_value = np.inf if minimize
else 0.
        self.best_monitored_epoch = 0
        self.restore_path = None

    def __call__(self, value, epoch):
        """
        # Проверяет, нужно ли остановить обучение и вос-
станавливает параметры модели, на которых была зафикси-
рована лучшая точность к данному моменту
        Parameters
        ----------
        value  :
                # Значение, зафиксированное на последней
итерации
        epoch  :
                # Порядковый номер последней эпохи (ите-
рации)

        Returns
        -------
```

```
        # `True`; если рекомендуется прекратить обуче-
ние, и `False` в противном случае
        """
        if (self.minimize and value < self.best_
monitored_value)
        or (not self.minimize and value > self.best_
moni-tored_value):
            self.best_monitored_value = value
            self.best_monitored_epoch = epoch
            self.restore_path = self.saver.save(
                self.session,
                os.getcwd() + "/early_stopping_
checkpoint"
            )
        elif self.best_monitored_epoch + self.patience <
epoch:
            if self.restore_path != None:
                self.saver.restore(self.session,
self.restore_path)
            else:
                print("ERROR: Failed to restore
session")
            return True
        return False
```

Еще один метод оптимизации процесса обучения состоит в так называемом подходе переносимых знаний (*transfer learning*). Как было продемонстрировано ранее, сверточные нейронные сети самостоятельно структурируют полученные знания, исходя из уровней абстракции предметной области.

В случае компьютерного зрения эту структуру особенно легко представить интуитивно: первый слой содержит фильтры для обнаружения простейших шаблонов пикселей, таких как контуры и линии. Последующие слои нейронной сети используют эти фильтры для обнаружения более сложных шаблонов, являющихся комбинациями приведенных простейших шаблонов. Таким образом, каждый слой сверточной нейронной сети опирается на предшествующие слои, обучаясь обнаруживать более замысловатые шаблоны как комбинации более простых.

Это свойство можно использовать, инициализируя параметры сверточной нейронной сети не случайным образом, а используя параметры другой, уже обученной нейронной сети. Интуитивно этот процесс можно представить следующим образом: часть знаний, которые хранит уже обученная нейронная сеть, можно перенести в другую модель, которая решает задачу в схожей предметной области. Так как в значениях параметров первых слоев сверточной нейронной

сети хранятся знания обнаружения простейших визуальных шаблонов, новая модель вполне может найти их полезными. Ей будет достаточно научиться использовать композицию этих простых шаблонов для решения своей задачи и не тратить время на построение собственных простейших шаблонов.

В рассматриваемом случае параметры, перенесенные из другой, уже обученной модели, не будут зафиксированы. Новая нейронная сеть будет обладать полной свободой модифицировать их в процессе обучения, однако основной процесс обучения будет происходить на последующих слоях, которые будут инициализированы случайно.

В процессе предобучения (*pre-training*) будут использованы параметры сверточной нейронной сети с архитектурой модели, которая предсказывает все опорные лицевые точки сразу. Основными преимуществами подобной архитектуры являются более быстрая сходимость, а также улучшенная способность модели обобщать полученные знания для работы с данными, не представленными в процессе обучения.

Реализация метода переносимых знаний из листинга 3.21 довольно тривиальная, т. к. поддержка данной функциональности присутствует в большинстве программных пакетов, работающих с математическими моделями на основе нейронных сетей.

Листинг 3.21 – Реализация метода переносимых знаний

```
variables_to_restore = [
    v for v in tf.global_variables() if "/out/" not in
v.op.name
]
loader = tf.train.Saver({
    v.op.name.replace(spec_var_scope, initialising_
model):
        v for v in variables_to_restore
})
loader.restore(session, initialising_model_path)
```

Ключевыми моментами являются:

– исключение последнего полносвязного слоя из восстанавливаемых параметров;

– модификация пространства имен переменных графа вычислений.

Так как необходимо перенести только знания на низших уровнях абстракции, нет смысла затрагивать веса последнего слоя нейронной сети. Кроме того, модели индивидуальных специалистов будут предсказывать другое количество искомых опорных точек лица, поэтому размерность слоев не будет совпадать.

Модификация пространства имен переменных необходима, т. к. разделение областей видимости переменных – это ключевой способ управления пере-

менными в вычислительном графе программного пакета *TensorFlow*. Его использование упрощает ряд типичных операций, применяемых к математическим моделям на основе нейронных сетей, к примеру таким, как использование метода переносимости знаний.

Реализация процесса обучения будет немного более объемной, чем в предыдущих случаях (листинг 3.22). Для оптимальной реализации и использования машинного времени будет введена подпрограмма обучения индивидуальной модели-специалиста. Она будет предназначена для сквозного обучения, включающего загрузку исходных маркированных данных и разбиение их на тренировочный и валидационный наборы. В ней же будут реализованы все рассмотренные ранее механизмы регуляризации: затухание темпа обучения, аугментация данных, исключение нейронов, ранняя остановка, импульсный метод оптимизации и пр.

Как и ранее, в процессе обучения будет сохранена история значений целевой функции для каждого специалиста для дальнейшего анализа процесса обучения. Кроме того, в данном случае эти значения будут также использованы для ранней остановки.

Листинг 3.22 – Обучение ансамбля моделей

```
def train_specialist(spec_setting):
    # Загрузка данных и разбиение на массивы
    X, y = load2d(cols = spec_setting['columns'])
    x_train, x_valid, y_train, y_valid =
train_test_split(
        X, y, test_size = 0.3
    )
    spec_var_scope = model_name(spec_setting['name'])
    # Вычисление ряда динамических гиперпараметров
    max_epochs = int(1e7 / y.shape[0])
    num_keypoints = y.shape[1]
    # Построение графа вычислений
    with tf.Graph().as_default() as graph:
        # Входные данные
        tf_x_batch = tf.placeholder(
            tf.float32, shape = (None, 96, 96, 1)
        )
        tf_y_batch = tf.placeholder(
            tf.float32, shape = (None, num_keypoints)
        )
        is_training = tf.placeholder(tf.bool)
        current_epoch = tf.Variable(0)
```

```
        # Затухание темпа обучения
        learning_rate = tf.train.exponential_decay(
            0.03,
            current_epoch,
            decay_steps = max_epochs,
            decay_rate = 0.03
        )
        # Вычисления процесса обучения модели
        with tf.variable_scope(spec_var_scope):
            predictions = model_pass(tf_x_batch,
num_keypoints, True)
        loss = tf.reduce_mean(tf.square(predictions -
tf_y_batch))
        # Оптимизация целевой функции
        optimizer = tf.train.MomentumOptimizer(
            learning_rate = learning_rate,
            momentum = 0.9,
            use_nesterov = True
        ).minimize(loss)
    # Запуск процесса обучения
    with tf.Session(graph = graph) as session:
        session.run(tf.global_variables_initializer())
        saver = tf.train.Saver()
        early_stopping = EarlyStopping(saver, session)
        train_loss_history = np.zeros(max_epochs)
        valid_loss_history = np.zeros(max_epochs)
        # Исключим последний слой из восстанавливаемых
параметров
        variables_to_restore = [
            v for v in tf.global_variables() if "/out/"
not in v.op.name
        ]
        # Заменим пространство имен переменных на значе-
ния текущей модели
        loader = tf.train.Saver({
            v.op.name.replace(spec_var_scope,
initialising_model):
                v for v in variables_to_restore
        })
        # Загрузим веса обученной модели вместо случай-
ной инициализации
        loader.restore(session, initialising_model_path)
        for epoch in range(max_epochs):
            current_epoch = epoch
```

```
            # Обучение на полном массиве данных случай-
ными партиями
            batch_iterator = FlipBatchIterator(
                batch_size = 36, shuffle = True
            )
            batch_iterator.flip_indices = spec_
setting['flip_indices']
            for x_batch, y_batch in batch_
iterator(x_train, y_train):
                session.run([optimizer], feed_dict = {
                        tf_x_batch : x_batch,
                        tf_y_batch : y_batch,
                        is_training : True
                    }
                )
            # Сохраним значение ошибки на тренировочных
данных
            train_loss = calc_loss(
                get_predictions_in_batches(
                    x_train, session, predictions,
tf_x_batch, False
                ),
                y_train
            )
            train_loss_history[epoch] = train_loss
            # Сохраним значение ошибки на валидационных
данных
            valid_loss = calc_loss(
                get_predictions_in_batches(
                    x_valid, session, predictions,
tf_x_batch, False
                ),
                y_valid
            )
            valid_loss_history[epoch] = valid_loss
            # Проверка на раннюю остановку
            if early_stopping(valid_loss, epoch): break
```

Для упрощения восприятия в исходном коде опущена реализация дополнительного сохранения промежуточных данных, параметров для наблюдения, а также других вспомогательных средств.

Используя данную подпрограмму, а также настройки специалистов, рассмотренных ранее, запуск обучения для всех необходимых моделей-специалистов выглядит достаточно тривиально, как просто вызов функции тренировки специалиста для каждого набора настроек.

Как и в предыдущих случаях, процесс обучения состоит в минимизации целевой функции, соответственно, значения ошибок для тренировочного и валидационного массива данных должны со временем падать, пусть и не на каждой итерации.

Итоговое значений функции ошибки обученной модели на валидационном наборе данных составило 0,00079437 (в среднем для всех опорных точек, т. е. при условии всех специалистов, работающих вместе), что значительно лучше модели на основе одной сверточной нейронной сети. Рассмотрим график изменения значений ошибки на обоих массивах данных в зависимости от эпохи для всех шести моделей на рисунке 3.12. Прерывистая линия соответствует значению ошибки на тренировочном массиве данных, сплошная – значению ошибки на валидационном массиве данных, т. е. данных, которые не были задействованы в процессе обучения.

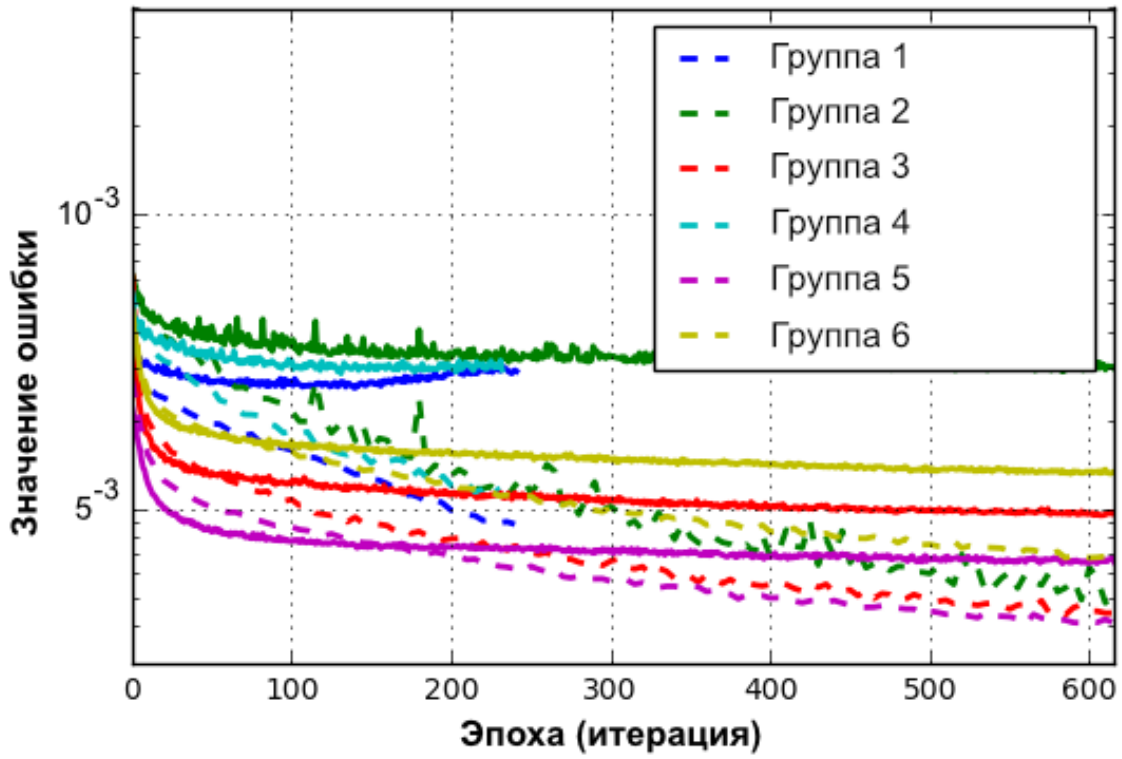

Рисунок 3.12 – График изменения значений ошибок в процессе обучения

В графике использованы следующие цветовые обозначения:

– группа 1: синий цвет, модель обнаружения центра правого глаза и центра левого глаза;

– группа 2: зеленый цвет, модель обнаружения кончика носа;

– группа 3: красный цвет, модель обнаружения левого уголка рта, правого уголка рта и центра верхней губы;

– группа 4: голубой цвет, модель обнаружения центра нижней губы;

– группа 5: фиолетовый цвет, модель обнаружения внутреннего уголка левого глаза, внутреннего уголка правого глаза, внешнего уголка левого глаза и внешнего уголка правого глаза;

– группа 6: желтый цвет, модель обнаружения внутреннего кончика левой брови, внутреннего кончика правой брови, внешнего кончика правой брови и внутреннего кончика правой брови.

Как и ранее, рассмотрим несколько примеров прогнозов модели на исходных данных (рисунок 3.13). Как и в предыдущих случаях, для демонстрации улучшения точности модели будут использованы те же изображения.

Рисунок 3.13 – Примеры предсказаний ансамбля сверточных сетей

Различия по сравнению с предыдущей, общей моделью на основе сверточной нейронной сети, незначительные, но определенно заметные в некоторых краевых случаях, на которых предыдущая модель делала заметные невооруженным глазом ошибки.

Анализ результатов и сравнительная оценка

Рассмотренные архитектуры модели возрастали по сложности, и последняя модель в виде ансамбля, группы специалистов на основе сверточных нейронных сетей, показала наилучшую точность как исходя из значений целевой функции, так и при визуальной оценке полученных результатов.

С повышением сложности архитектуры возрастало и количество параметров, которые модель наполняла смыслами в процессе обучения. Мы выяснили, что повышение вместимости, как правило, ведет к усложнению и замедлению процесса обучения, а также к снижению способности модели к обобщению полученных знаний на примерах, не представленных в процессе обучения.

Для предотвращения этих негативных эффектов был рассмотрен и применен ряд мероприятий по регуляризации и оптимизации процесса обучения. Эти мероприятия также усложнялись и принимали все более точечный и детальный характер от архитектуры к архитектуре. Их применение принесло ожидаемые результаты, и конечная информационная модель отлично справилась с точным обнаружением опорных лицевых точек даже на изображениях, которые не участвовали в процессе обучения (таблица 3.1).

Таблица 3.1 – Оценка работы моделей на основе различных архитектур нейронных сетей

Архитектура модели	Значение ошибки на валидационном наборе данных
Нейронная сеть с одним скрытым слоем	$2{,}72522 \times 10^{-3}$
Глубокая сверточная сеть	$1{,}12052 \times 10^{-3}$
Ансамбль сверточных сетей	$7{,}9437 \times 10^{-4}$

Как видно из таблицы 3.1, наименьшее значение ошибки (и, соответственно, наивысшую точность) показала модель на основе группы сверточных нейронных сетей, предсказывающих индивидуальные наборы опорных лицевых точек.

3.3 Цифровое моделирование систем автоматического управления имитационным методом в задачах САПР

В литературе уделяется недостаточно внимания формулировке требований к системам автоматизированного проектирования (САПР) динамических систем,

особенно на этапе доведения изделий до уровня, обеспечивающего конкурентоспособность. Главной особенностью этого этапа является необходимость учета многообразных технологических факторов, проявляющихся при реализации даже традиционных систем автоматического управления (САУ). Поэтому САПР, которые вытесняют малоэффективные натурные испытания, должны обеспечивать проектировщику, с одной стороны, возможность максимально подробно описывать объект исследования, а с другой − обеспечивать оперативный расчет указанного объекта с высокой точностью.

В настоящее время достаточно часто решение поставленной задачи осуществляется с помощью имитационного метода без формирования системы уравнений какого-либо стандартного вида, например, в форме Коши. При этом автономный анализ отдельного звена САУ выполняется вместе с обменом значениями сигналов с внешней средой. Имитационный метод позволяет эффективно решить основную проблему размерности при использовании традиционных подходов и проявляющуюся:

– в появлении слабозаполненных топологических матриц;

– необходимости разделения системы уравнений на аналоговую и дискретную, а также − расчета начальных условий при скачках дискретных функций;

– необходимости разделения возможных форм описания отдельных звеньев САУ в виде принципиальной схемы, передаточной функции, матричной формы, формул и т. д.;

– возможности получения плохо обусловленных систем уравнений;

– необходимости адаптации алгоритма расчета САУ с большим разбросом постоянных времени;

– трудности локализации ошибок заданных исходных данных.

Несмотря на преимущества имитационного метода, традицией стало рассматривать имитационный метод как численный эксперимент, когда достаточно осуществить «прогон» имитационных моделей, а не «решать» их. Невысокая точность расчетов при таком подходе для САУ связана с некомплексным применением имитационного метода, где естественное блочное расчленение сложного объекта используется без объединения частных решений в единую систему уравнений.

Следовательно, обоснованное определение стратегии повышения точности расчетов динамики САУ требует формализовать имитационный метод как оригинальный метод решения системы дифференциальных уравнений с учетом специфики замкнутых систем.

3.3.1 Математическая модель объекта анализа

Аналогично определению ребра графа электрической цепи введем определение ребра графа динамической системы.

Определение – Ребром графа динамической системы называется направленная линия, связывающая вход и выход отдельного его звена. При этом совокупность ребер не имеет никаких общих точек, кроме вершин.

Будем подразумевать, что звенья сложения либо вычитания сигналов обратных связей либо сигналов внешних возмущений вырождаются в вершины графа системы.

Введем для произвольной САУ структурную матрицу S_m^n, связывающую вершины (индекс m) и ребра (индекс n) топологического графа. В этом случае математическая модель для обобщенной линейной САУ имеет вид

$$Y^m(p) = S_n^m\left[W_n^n(p)X^n(p)\right] + F^m(p), \tag{3.4}$$

где p – оператор Лапласа;

$X^n(p)$ – вектор изображений функций входных сигналов звеньев, соответствующих ребрам графа;

$Y^m(p)$ – вектор изображений функций сигналов в вершинах графа;

$F^m(p)$ – вектор изображений функций внешних возмущений;

$W_n^n(p)$ – диагональная матрица передаточных функций.

Во временной области (с учетом наличия нелинейностей) в формуле (3.4) происходят следующие переходы:

а) $p \to \frac{d}{dt}$;

б) $w_n(p) \to \frac{\partial y_n}{\partial x_n}$ – для n-го нелинейного звена с характеристикой $y_n(x_n)$;

в) $w_n(p) \cdot x_n(p) \to \sum_{k=0}^{N} \alpha_k x[T(n-k)] + \sum_{k=1}^{N} \beta_k y[T(n-k)]$ – для n-го звена цифровой обработки сигнала типа «дискретный фильтр» (ДФ);

г) $w_n(p) \cdot x_n(p) \to u_+(x,t)$– для n-го звена типа «зависимый источник», например, вентильный преобразователь (ВП).

3.3.2 Алгебраизация дифференциальных уравнений

Наиболее рациональными разностными схемами для алгебраизации дифференциальных уравнений инерционных звеньев САУ являются разностные схемы, предлагаемые для расщепленных задач. При этом дифференциальные уравнения редуцируются к виду

$$y_{i+1}^n = a_{i+1}^n x_{i+1}^n + b_{i+1}^n, \tag{3.5}$$

где x_{i+1}^n– значение входного сигнала n-го звена в t_{i+1}момент времени;

y_{i+1}^n – значение выходного сигнала n-го звена в t_{i+1} момент времени;

a_{i+1}^n, b_{i+1}^n – коэффициент алгебраизации.

Для апериодического звена первого порядка, дифференциальное уравнение которого имеет вид $T_n y^n + y^n = x^n$, этап алгебраизации с помощью абсолютно устойчивой разностной схемы Эйлера первого порядка (погрешность

$\sim h$) выполняется по формуле

$$T_n \frac{y_{i+1}^n - y_i^n}{h} = x_{i+1}^n, \quad (3.6)$$

где T_n – параметр апериодического звена;

h – шаг интегрирования.

С помощью абсолютно устойчивой разностной схемы Кранка – Николсона второго порядка (погрешность $\sim h^2$) алгебраизация выполняется по формуле

$$T_n \frac{y_{i+1}^n - y_i^n}{h} + \frac{y_{i+1}^n + y_i^n}{2} = \frac{x_{i+1}^n + x_i^n}{2}. \quad (3.7)$$

Следующей ступенью повышения точности является экстраполяция по Ричардсону по формуле

$$y_{i+1}^n = \frac{4}{3} y_{\frac{h}{2}, i+1}^n - \frac{1}{3} y_{h, i+1}^n, \quad (3.8)$$

где $y_{\frac{h}{2}, i+1}^n$ – решение с шагом $\frac{h}{2}$,

$y_{h, i+1}^n$ – решение с шагом h.

При этом погрешность экстраполяции $\sim h^4$.

3.3.3 Решение алгебраизованной системы уравнений

Простые итерации

Воспользовавшись топологической связью звеньев САУ в выражении (3.4), получим алгебраизованную систему уравнений в матричном виде:

$$AY = F. \quad (3.9)$$

Так как структурная матрица S_m^n в выражении (3.4) близка к диагональной, то систему (3.9) можно представить в виде

$$A_1 Y^{(k+1)} = F - A_2 Y^{(k)}, \quad (3.10)$$

где $Y^{(k+1)}$, $Y^{(k)}$ – искомые решения на $(k+1)$-й и k-й итерациях соответственно;

A – ленточная матрица;

A_1 – матрица обратных связей, в которой количество ненулевых элементов равно количеству обратных связей в САУ (l). При этом $A_1 + A_2 = A$.

Отсюда выражение (3.9) может решаться в частном координатном базисе

$$\hat{O}(\hat{Y}) = 0, \quad (3.11)$$

где $\hat{Y}$ − вектор значений выходных сигналов звеньев обратных связей $(\hat{Y}^l \in Y^m)$;

$\hat{O}$ − вектор функции от $\hat{Y}$ (в общем случае − нелинейной).

Рассмотрим наиболее распространенный (одноциклический) вариант (k = 1) использования имитационного метода при расчете САУ как разомкнутой (l = 0). Алгоритм формирования начальных условий в i-й момент времени $\hat{Y}^{(0)}$ имеет вид

$$\hat{Y}_i^0 = \hat{Y}_{i-1}^1. \tag{3.12}$$

Возможности имитационного метода можно сформулировать в виде леммы и теоремы.

Лемма об имитационном методе

Решение систем управления с обратной связью с помощью одного имитационного «прогона» есть решение системы уравнений первого порядка аппроксимации.

Доказательство − Построим алгоритм решения САУ имитационным методом (рисунок 3.14, порядок расчета отдельных звеньев соответствует их нумерации) со встроенной в контур управления передаточной функцией вида

$$w(p) = \frac{a_n p^n + a_{n-1} p^{n-1} + \ldots + a_0}{b_m p^m + b_{m-1}^{m-1} + \ldots + b_0} = \frac{A_n(p)}{B_m(p)}. \tag{3.13}$$

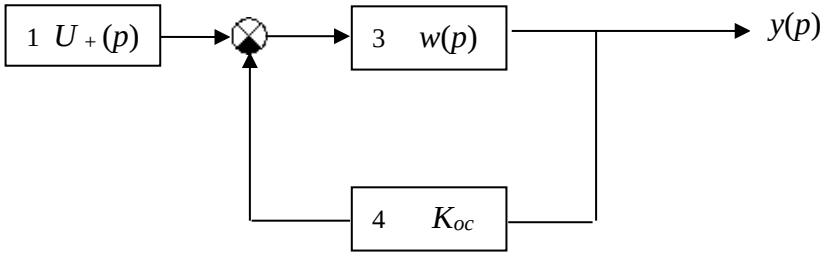

Рисунок 3.14 − Алгоритм решения САУ

В алгебраизованном виде будем иметь

$$y_i = \left(\widetilde{U}_{+i} - K_{oc} y_{i-1}\right) \frac{\triangle_A^n}{\triangle_B^m}, \tag{3.14}$$

где $\frac{\triangle_A^n}{\triangle_B^m}$ – коэффициент алгебраизации $w(p)$;

$\widetilde{U}_{+i}$ – значение приведенного сигнала входного воздействия в t_i момент времени.

Сравнивая формулу (3.14) с точным решением, получим погрешность решения *D(h)*:

$$D(h) = \left| K_{oc} h \frac{\triangle_A^n}{\triangle_B^m} \frac{dy_{i-1}}{dt} \right| \sim h. \tag{3.15}$$

Выбор h связан с условием сходимости ряда Тейлора:

$$\left| \frac{h \frac{dy_{i-1}}{dt}}{y_{i-1}} \right| \prec 1. \tag{3.16}$$

Замечание − Для нелинейных систем вследствие «малого» h выражения (3.16) вывод сохраняется.

Следствие 1 − Точность расчета отдельных звеньев, входящих в контур системы управления не превышает первый порядок аппроксимации при любом методе их расчета.

Следствие 2 − Локальная погрешность расчета при $m = n + 1$ составляет $\sim h^2$, т. к. $\frac{\triangle_A^n}{\triangle_B^m} \sim \frac{h}{T_{\min}}$ ($T_{\min}$ − минимальная постоянная времени фильтра).

Получим необходимые и достаточные условия сходимости и устойчивости простых итераций, применяемых для систем с подчиненным регулированием. Для этого можно рассмотреть одноконтурную систему.

Теорема 1

Для сходимости метода простых итераций при расчете систем с подчиненным регулированием необходимо и достаточно соблюдать условие устойчивости метода:

$$\frac{\triangle_A^n}{\triangle_B^m} K_{oc} \prec 1. \tag{3.17}$$

Доказательство − Обозначив $a = \frac{\triangle_A^n}{\triangle_B^m}$ и последовательно применяя формулу (3.11) для итерационного процесса на каждом шаге интегрирования, получим выражение

$$\begin{aligned} y_i = \lim \big[1 - aK_{oc} + \\ + (aK_{oc})^2 - \ldots + (-1)^{k-1} (aK_{oc})^{k-1} \big] a\widetilde{U}_{+i} + \big[(aK_{oc})^k y^{(0)} \big], \end{aligned} \tag{3.18}$$

где k − номер итерации.

Из выражения (3.15) следует, что сумма в скобках сходится как геометрическая прогрессия при $aK_{oc} < 1$. При этом получаем $y_i = \frac{a\tilde{U}_{+i}}{1 + aK_{oc}}$, что соответствует точной формуле решения для статических систем с обратной связью.

Следствие − Устойчивость и сходимость одноциклического имитационного метода доказывается аналогично, т. к. при $h \to 0$ имитационный метод переходит в итерационный.

Замечание − В случае невозможности допущения теоремы 1 о расчете системы с подчиненным регулированием полученные результаты являются лишь необходимым условием, т. к. в общем случае теорема 1 доказывается на основе расчета канонических норм матрицы параметров системы уравнений.

Теорема 2

Разностная схема Кранка – Николсона реализуется для расчета систем управления с обратной связью с помощью двухциклического варианта имитационного метода.

Доказательство − Представим выражение (3.4) в виде системы дифференциальных уравнений соответствующей форме покомпонентного расщепления:

$$\frac{dy}{dt} + CY = F(t, \hat{Y}), \tag{3.19}$$

где $C = \begin{Vmatrix} C_1 & 0 & 0 & \dots & 0 \\ 0 & C_2 & 0 & \dots & 0 \\ 0 & 0 & C_3 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & C_{\mathrm{n}} \end{Vmatrix}$ − диагональная матрица параметров системы (для n-го апериодического звена первого порядка $C_n = \frac{1}{T_n}$).

Покажем, что точность аппроксимации выражения (3.19) по схеме из формулы (3.7) второго порядка $\sim h^2$. Для этого выполним следующее разложение в ряд Тейлора:

$$F_{i+1} = F_{i+\frac{1}{2}} + \frac{h}{2}\frac{\partial F_{i+\frac{1}{2}}}{\partial t} + \frac{h^2}{8}\frac{\partial F_{i+\frac{1}{2}}}{\partial t} + \dots, \tag{3.20}$$

$$F_i = F_{i+\frac{1}{2}} - \frac{h}{2}\frac{\partial F_{i+\frac{1}{2}}}{\partial t} + \frac{h^2}{8}\frac{\partial F_{i+\frac{1}{2}}}{\partial t} - \dots. \tag{3.21}$$

При сложении (3.20) и (3.21) получаем

$$\frac{Y_{i+1} - Y_i}{h} + C\frac{Y_{i+1} + Y_i}{2} = F_{i+\frac{1}{2}} + O(h^2). \tag{3.22}$$

Так как согласно лемме одноциклический имитационный метод дает погрешность лишь $\backsim h$, то разностная схема (3.4) может реализовываться по формуле

$$\frac{Y_{i+1} - Y_i}{h} + C\frac{Y_{i+1} + Y_i}{2} = \frac{F^*_{i+1} + F_i}{2}, \tag{3.23}$$

где F^*_{i+1} – значение вектора входного сигнала, найденное на втором итерационном цикле.

При этом на первой итерации рациональней применение разностной схемы (3.7).

Замечание – Применение третье и последующих итераций лишь незначительно повышает точность расчета.

Метод Ньютона

Одним из наиболее эффективных методов расчета нелинейных систем алгебраических уравнений вида (3.8) является *метод Ньютона*:

$$\hat{Y}^{(k+1)} = \hat{Y}^{(k)} - J_k^{-1}\left[\hat{Y}^{(k)}\right] \cdot \hat{O}\left[\hat{Y}^{(k)}\right], \tag{3.24}$$

где $J_k^{-1}\left[\hat{Y}^{(k)}\right]$ – обратная матрица Якоби, в матричной форме имеющая вид $J_k\left[\hat{Y}^k\right] = \frac{\partial \hat{O}}{\partial \hat{Y}}$.

Замечание – Матрица Якоби $J\left[\hat{Y}\right]$ в принятом частном координатном базисе имеет порядок, равный количеству обратных связей САУ (l), и не имеет нулевых элементов.

Для одноконтурной САУ алгоритм расчета будет следующим:

$$\hat{O}(\hat{Y}) = \hat{Y} - \left(\tilde{\mathrm{U}}_+ - \hat{\mathrm{Y}}\right)aK_{oc}\,. \tag{3.25}$$

Выражение (3.25) показывает, что трудоемкость однократного расчета $\hat{O}\left[\hat{\mathrm{Y}}\right]$ равна трудоемкости имитационного метода, рассмотренного в лемме. Принимая за единицу трудоемкости указанный вычислительный режим и пренебрегая трудоемкостью обращений матрицы Якоби из-за того, что количество обратных связей САУ значительно меньше общего числа звеньев ($l \prec\prec n$), получим общую формулу трудоемкости метода Ньютона:

$$N = K_u(2l + 1) + 1, \tag{3.26}$$

где K_u – предельное количество итераций.

Условия сходимости метода Ньютона позволяют преодолеть «барьер» применимости метода простых итераций (теорема 1) и в некоторых случаях решить задачу автоматизации разделения движений.

3.3.4 Имитационный метод и диакоптика

Рассмотрим краткую сравнительную характеристику имитационного метода и метода диакоптики, т. к. оба метода используют структурность объекта исследования не только в представлении, но и непосредственно в его расчете. В основе диакоптики, излагаемой в электротехнической терминологии − определение структурной связи между токами (i), напряжениями (e), импедансами ($\dot{z}$) отдельных ветвей электрической схемы и контурными токами (I), контурными импедансами (Z) и источниками напряжения (E) анализируемой схемы в виде

$$E = A_R^j \dot{z} A_{t,R}^j I, \tag{3.27}$$

где A_R^j − матрица инциденций электрической схемы, определяющая связь между токами ветвей (i) и контурными токами (I) по уравнению

$$i^j = A_{t,R}^j I_R, \tag{3.28}$$

где R − индекс по независимым контурам схемы,

j − индекс по ветвям.

Сравнивая выражения (3.4) и (3.27), следует отметить, что введенная структурная матрица S_m^n является аналогом матрицы A_R^i. Однако формула (3.27) сложнее схемы (3.4) вследствие дуализма токов и напряжений в электрической цепи.

При топологическом расчленении электрической схемы на n подсхем «собирание» частных решений в общее выполняется за счет расчета (n + 1)-й схемы соединений небольшой размерности после чего вносятся поправки в частные решения. Подобная вычислительная технология используется в имитационном методе. Однако в различных вычислительных режимах этот процесс проявляется по-разному.

1. В традиционном варианте имитационного метода, изложенного в лемме, решение системы уравнений выполняется за счет выбора «малого» h, при этом в зоне устойчивости метода частные решения и поправки рассчитываются в ходе каждого «прогона».

2. При использовании простых итераций либо двухциклического метода (теорема 2) дополнительные затраты при решении приходятся на организацию дополнительных итерационных циклов (оптимально − еще одного).

3. При использовании метода Ньютона в частном координатном базисе схема решения практически полностью соответствует методу диакоптики, где роль математической модели (n + 1)-й схемы соединений отводится матрице Якоби в указанном базисе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Viola, P. Rapid Object Detection using a Boosted Cascade of Simple Features / P. Viola // Accepted Conference on Computer Vision and Pattern Recognition (CVPR 2001) : proc. of the Intern. Conf., Kauai, Dec. 8–14, 2001. – Kauai, 2001. – P. 511–518.

2 Система технического зрения для информационного обеспечения автоматической посадки и движения по ВПП летательных аппаратов / С. М. Соколов [и др.] // Изв. ЮФУ. Техн. науки. – 2015. – № 1 (162). – С. 96–109.

3 Филатов, В. И. Система обучения визуальным понятиям на основе соотнесения лексем и ключевых точек / В. И. Филатов, А. С. Потапов // Науч.-техн. вестн. инф. технологий, механики и оптики. – 2016. – № 4 (16). – С. 689–696.

4 Alpaydin, E. Introduction to Machine Learning / E. Alpaydin. – MIT Press, 2010. – 9 p.

5 Hastie, T. The Elements of Statistical Learning: Data Mining, Inference, and Prediction / T. Hastie, R. Tibshirani, J. Friedman. – 2nd ed. – New York : Springer-Verlag, 2009. – 746 p.

6 Дрейпер, Н. Прикладной регрессионный анализ. Множественная регрессия / Н. Дрейпер, Г. Смит. – 3-е изд. – М. : Диалектика, 2007. – 912 с.

7 Радченко, С. Г. Методология регрессионного анализа / С. Г. Радченко. – Киев : Корнийчук, 2011. – 376 с.

8 Quinlan, R. Simplifying decision trees / R. Quinlan // Intern. J. of Man-Machine Studies. – 1987. – Vol. 27, iss. 3. – P. 221–234.

9 Quinlan, R. Learning efficient classification procedures / R. Quinlan // Machine learning: an artificial intelligence approach. – 1983. – P. 463–482.

10 Мак-Каллок, У. С. Логическое исчисление идей, относящихся к нервной активности / У. С. Мак-Каллок, В. Питтс. – М. : Изд-во иностр. лит., 1956. – С. 363–384.

11 Backpropagation Applied to Handwritten Zip Code Recognition / Y. LeCun [et al.] // Neural Computation. – 1989. – Vol. 1, iss. 4. – P. 541–551.

12 Subject independent facial expression recognition with robust face detection using a convolutional neural network / M. Matusugu [et al.] // Neural Networks. – 2003. – Vol. 16, iss. 5. – P. 555–559.

13 Nair, V. Rectified Linear Units Improve Restricted Boltzmann Machines Vinod / V. Nair, G. E. Hinton // ICML-1 : proc. of the 27th Intern. Conf. on Machine Learning, Haifa, June 21–24, 2010. – Haifa, 2010. – P. 807–814.

14 Striving for Simplicity: the All Convolutional Net [Electronic resource] / J. T. Springenberg [et al.]. – 2014. – Mode of access: https://arxiv.org/abs/1412.6806.

15 Элементарное введение в технологию нейронных сетей с примерами программ / Р. Тадеусевич [и др.]. – М. : Горячая линия – Телеком, 2011. – 408 с.

16 Галушкин, А. И. Синтез многослойных систем распознавания образов / А. И. Галушкин. – М. : Энергия, 1974. – 367 с.

17 LeCun, Y. Deep learning / Y. LeCun, Y. Bengio, G. Hinton // Nature. – 2015. – Vol. 524, iss. 7553. – P. 436–444.

18 Rumelhart, D. E. Learning representations by back-propagating errors / D. E. Rumelhart, G. E. Hinton, R. J. Williams // Nature. – 1986. – Vol. 323, iss. 6088. – P. 533–536.

19 Городецкий, С. Ю. Нелинейное программирование и многоэкстремальная оптимизация / С. Ю. Городецкий, В. А. Гришагин. – Н. Новгород : Изд-во Нижегородского ун-та, 2007. – 489 с.

20 Курвилль, А. Глубокое обучение / А. Курвилль, И. Бенджио, Я. Гудфеллоу. – М. : ДМК Пресс, 2017. – 652 с.

21 On the importance of initialization and momentum in deep learning / I. Sutskever [et al.] // PMLR : proc. of the 30th Intern. Conf. on Machine Learning, Atlanta, June 16–21, 2013. – Atlanta, 2013. – P. 1139–1147.

22 Nesterov, Y. A method of solving a convex programming problem with convergence rate O(1/k2) / Y. Nesterov // Soviet Mathematics Doklady. – 1983. – Vol. 27, iss. 2. – P. 372–376.

23 Nesterov, Y. Gradient methods for minimizing composite functions / Y. Nesterov // Math. Programming. – 2013. – Vol. 140, iss. 1. – P. 125–161.

24 Яковлев, С. С. Система распознавания движущихся объектов на базе искусственных нейронных сетей / С. С. Яковлев. – Минск : ИТК НАНБ, 2004. – С. 230–234.

25 Leinweber, D. J. Stupid data miner tricks / D. J. Leinweber // The J. of Investing. – 2007. – Vol. 16. – P. 15–22.

26 Tetko, I. V. Neural network studies. Comparison of Overfitting and Overtraining / I. V. Tetko, D. J. Livingstone, A. L. Luik // J. of Chem. Inform. and Modeling. – 1995. – Vol. 35, iss. 5. – P. 826–833.

27 Christian, B. Algorithms to live by: the Computer Science of Human Decisions / B. Christian, T. Griffiths. – New York : William Collins, 2017. – 368 p.

28 Improving neural networks by preventing co-adaptation of feature detectors [Electronic resource] / G. E. Hinton [et al.]. – 2014. – Mode of access: https://arxiv.org/abs/1207.0580.

29 Nouri, D. Using convolutional neural nets to detect facial keypoints [Electronic resource] / D. Nouri. – 2014. – Mode of access: http://danielnouri.org/notes/2014/12/17/using-convolutional-neural-nets-to-detect-facial-keypoints-tutorial.

30 Sermanet, P. Traffic sign recognition with multi-scale convolutional networks / P. Sermanet, Y. LeCun // IJCNN'11 : proc. of Intern. Joint Conf. on Neural Networks, San Jose, July 31 – Aug. 5, 2011. – San Jose, 2011. – P. 1–4.

31 A committee of neural networks for traffic sign classification / D. C. Ciresan [et al.] // IJCNN'11 : proc. of the Intern. Joint Conf. on Neural Networks, San Jose, July 31 – Aug. 5, 2011. – San Jose, 2011.– P. 1918–1921.

32 Krizhevsky, A. Imagenet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G. E. Hinton // Advances in Neural Information Processing Systems. – 2012. – Vol. 25. – P. 1097–1105.

33 Milletari, F. V-net: fully convolutional neural networks for volumetric medical image segmentation / F. Milletari, N. Navab, S. A. Ahmadi // 3DV : proc. of the 4th Intern. Conf. on 3D Vision, Stanford, Oct. 25–28, 2016. – Stanford, 2016. – P. 565–571.

34 Liu, Y. Ensemble Classification Based on ICA for Face Recognition / Y. Liu, Y. Lin, Y. Chen // CISP : proc. of the 1st Intern. Congr. on Image and Signal Processing, Cherbourg, July 2–4, 2018. – Cherbourg, 2008. – P. 144–148.

35 Raj Kumar, P. Distributed denial of service attack detection using an ensemble of neural classifier / P. Raj Kumar, S. Selvakumar // Computer Communications. – 2011. – Vol. 34, iss. 11. – P. 1328–1341.

36 Ван дер Линде, Р. К. HapticMaster – универсальный робот с силовым управлением для взаимодействия с человеком / Р. К. Ван дер Линде, П. Ламертсе // Industrial Robot. – 2003. – Т. 30, № 6. – С. 515–524.

37 Bonab, R. H. A Theoretical Framework on the Ideal Number of Classifiers for Online Ensembles in Data Streams / H. R. Bonab, F. Can. – USA : ACM, 2016. – 2053 p.

38 Кашницкий, Ю. С. Ансамблевый метод машинного обучения, основанный на рекомендации классификаторов / Ю. С. Кашницкий, Д. И. Игнатов // Интеллектуал. системы. Теория и приложения. – 2015. – Т. 19, № 4. – С. 37–55.

39 Кузнецов, И. А. Разработка ансамбля алгоритмов классификации с использованием энтропийного показателя качества для решения задачи поведенческого скоринга / И. А. Кузнецов, В. С. Киреев // Аналитика и управление данными в областях с интенсивным использованием данных : XVIII Междунар. конф. DAMDID/RSDL'2016, Москва, 11–14 окт. 2016 г. / ФИЦ ИУ РАН. – Москва, 2016. – С. 79–85.

40 Татьянкин, В. М. Подход к формированию архитектуры нейронной сети для распознавания образов / В. М. Татьянкин // Вестн. Югор. гос. ун-та. – 2016. – Т. 2, № 41. – С. 61–64.

41 Дрокин, И. С. Об одном алгоритме последовательной инициализации весов глубоких нейронных сетей и обучении ансамбля нейронных сетей // Вестн. С.-Петербург. ун-та. Прикладная математика. Информатика. Процессы управления. – 2016. – № 4. – С. 66–74.

42 Борисов, В. В. Искусственные нейронные сети. Теория и практика / В. В. Борисов, В. В. Круглов. – М. : Горячая линия – Телеком, 2001. – 382 с.

43 Каллан, Р. Основные концепции нейронных сетей / Р. Каллан. – М. : Вильяме, 2001. – 328 с.

44 Hinton, G. Distilling the knowledge in a neural network [Electronic resource] / G. Hinton, O. Vinyals, J. Dean. – 2014. – Mode of access: https://arxiv.org/abs/1503.02531.

45 An ensemble of fine-tuned convolutional neural networks for medical image classification / A. Kumar [et al.] // IEEE J. of Biomed. and Health Informatics. – 2016. – Vol. 24, iss. 1. – P. 31–40.

46 Han, B. Branchout: regularization for online ensemble tracking with convolutional neural networks / B. Han, S. Jack, A. Hartwig // IEEE International Conference on Computer Vision and Pattern Recognition : proc. of the Intern. Conf., Honolulu, July 21–26, 2017. – Honolulu, 2017. – P. 3356–3365.

47 Fast face-swap using convolutional neural networks / I. Korshunova [et al.] // IEEE International Conference on Computer Vision and Pattern Recognition : proc. of the Intern. Conf., Honolulu, July 21–26, 2017. – Honolulu, 2017. – P. 3677–3685.

48 A deep residual convolutional neural network for facial keypoint detection with missing labels / S. Wu [et al.] // Signal Processing. – 2018. – Vol. 144 – P. 384–391.

49 Duvenaud, D. Early stopping as nonparametric variational inference / D. Duvenaud, D. Maclaurin, R. Adams // Artificial Intelligence and Statistics. – 2016. – P. 1070–1077.

50 Deep neural network with weight sparsity control and pre-training extracts hierarchical features and enhances classification performance: evidence from whole-brain resting-state functional connectivity patterns of schizophrenia / J. Kim [et al.] // Neuroimage. – 2016. – Vol. 124. – P. 127–146.

51 Levi, G. Emotion recognition in the wild via convolutional neural networks and mapped binary patterns / G. Levi, T. Hassner // ACM-2015 : proc. of Intern. Conf. on multimodal interaction, Seattle, Nov. 9–13, 2015. – Seattle, 2015 – P. 503–510.

52 Staravoitau, A. Traffic Sign Classification with a Convolutional Network / A. Staravoitau // Pattern Recognition and Image Analysis. – 2018. – Vol. 28. – P. 155–162.

53 End to End Learning for a Driving Simulator / V. F. Alexeev [et al.] // Doklady BSUIR. – 2018. – Vol. 2, iss. 112. – P. 85–91.

54 Старовойтов, А. Visualising computer vision datasets / А. Старовойтов // Новые информационные технологии в научных исследованиях и в образовании : материалы XXIII Всерос. науч.-техн. конф. студентов, молодых ученых и специалистов, Рязань, 13 дек. 2018 г. – Рязань, 2018. – С. 198–200.

55 Старовойтов, А. Computation graphs in TensorFlow // Новые информационные технологии в научных исследованиях и в образовании : материалы XXIII Всерос. науч.-техн. конф. студентов, молодых ученых и специалистов, Рязань, 13 дек. 2018 г. – Рязань, 2018. – С. 278–280.

56 Старовойтов, А. Calibrating camera data // Компьютерное проектирование и технология производства электронных систем : сб. тез. 55-й Науч. конф. аспирантов, магистрантов и студентов, Минск, 22–26 апр. 2019 г. / Бел. гос. ун-т информатики и радиоэлектроники ; отв. ред. Раднёнок А. Л. – Минск, 2019.

57 Старовойтов, А. Extracting features for edge detection // Компьютерное проектирование и технология производства электронных систем : сб. тез. 55-й Науч. конф. аспирантов, магистрантов и студентов, Минск, 22–26 апр. 2019 г. / Бел. гос. ун-т информатики и радиоэлектроники ; отв. ред. А. Л. Раднёнок. – Минск, 2019.

58 Егоров, И. Н. Системы управления электроприводов технологических роботов и манипуляторов : учеб. пособие / И. Н. Егоров, В. П. Умнов. – Владимир : Изд-во ВлГУ, 2022. – 314 с.

59 Запорожцев А. В. Моделирование технических систем // Фундам. исслед. – 2014. – № 8 (6). – С. 1288–1294.

60 Балашов Е. П. Проектирование информационно-управляющих систем / Е. П. Балашов, Д. В. Пузанов. − М. : Радио и связь, 1987. − 256 с.

61 Губич Л. В. Моделирование процессов и объектов проектирования в машиностроении / Л. В. Губич, В. А. Сипайло. − Минск : ИТК АН БССР, 1990. − 158 с.

62 Алексеев В. Ф. Блочное представление вентильных преобразователей при цифровом моделировании радиоэлектронных средств / В. Ф. Алексеев, В. Ш. Пасик, С. А. Харитонов // Изв. Белорус. инженер. Акад. − 2002. − № 1 (13). – С. 211–219.

63 Алексеев В. Ф. Применение имитационного метода для численного анализа САУ / В. Ф. Алексеев, В. Ш. Пасик // Изв. Белорус. инженер. Акад. − 2002. − № 1 (13). – С. 219–225.

64 Норенков И. П. Основы автоматизированного проектирования : учебник для вузов / И. П. Норенков. − М. : Изд-во МГТУ им. Н. Э. Баумана, 2000. – 360 с.

65 Мироненко, И. Г. Автоматизированное проектирование узлов и блоков РЭС средствами современных САПР : учеб. пособие для вузов / И. Г. Мироненко, В. Ю. Суходольский, К. К. Холуянов ; под ред. И. Г. Мироненко. − М. : Высш. шк., 2002. − 391 с.

66 Автоматизация проектирования радиоэлектронных средств : учеб. пособие для вузов / О. В. Алексеев [и др.] ; под ред. О. В. Алексеева. − М. : Высш. шк., 2000. − 479 с.

67 Алексеев В. Ф. Принципы конструирования и автоматизации проектирования РЭУ : учеб. пособие / В. Ф. Алексеев. – Минск. : БГУИР, 2001. – 197 с.

68 Потемкин В. Г. Система инженерных и научных расчетов MATLAB 5.x. : в 2 т. / В. Г. Потемкин. – М. : ДИАЛОГ-МИФИ, 1999. – Т. 1 – 366 с. ; Т. 2 – 304 с.

69 Марчук Г. И. Повышение точности решения разностных схем / Г. И. Марчук, В. В. Шайдуров. − М. : Наука, 1970. – 324 с.

70 Бусленко В. Н. Автоматизация имитационного моделирования сложных систем / В. Н. Бусленко. − М. : Наука, 1970. − 302 с.

71 Шеннон Р. Ю. Имитационное моделирование систем – искусство и наука / Р. Ю. Шеннон. − М. : Мир, 1978. – 279 с.

Св. план 2022, поз. 11

Учебное издание

Алексеев Виктор Федорович
Лихачевский Дмитрий Викторович
Пискун Геннадий Адамович
Шаталова Виктория Викторовна

МОДЕЛИРОВАНИЕ И ОПТИМАЛЬНОЕ ПРОЕКТИРОВАНИЕ ТЕХНИЧЕСКИХ СИСТЕМ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *С. Г. Девдера*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *В. М. Задоля*

Подписано в печать 06.12.2024. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 5,93. Уч.-изд. л. 6,0. Тираж 30 экз. Заказ 40.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск