

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
«МИСИС» (СТИ НИТУ «МИСИС»)

На правах рукописи

ЗОРИН ИВАН СЕРГЕЕВИЧ

**НЕЙРОСЕТЕВЫЕ АЛГОРИТМЫ РАСПОЗНАВАНИЯ КОДОВ ЛИТЫХ
ЗАГОТОВОК ДЛЯ ПОДДЕРЖКИ ОПЕРАТОРА АСУ ТП МЕТАЛЛУРГИЧЕСКОГО
ПРЕДПРИЯТИЯ**

Специальность 2.3.3. «Автоматизация и управление технологическими процессами и
производствами»

Диссертация на соискание ученой степени кандидата технических наук

НАУЧНЫЙ РУКОВОДИТЕЛЬ
кандидат технических наук, доцент,
Полещенко Дмитрий Александрович

Старый Оскол - 2026

СОДЕРЖАНИЕ

Введение.....	5
Глава 1. Анализ процесса распознавания клейма литой заготовки.....	13
1.1 Описание участка контроля литых заготовок перед посадом в печи нагрева.....	13
1.2 Проблематика распознавания клейма литой заготовки	16
1.3 Анализ методов позволяющих реализовать алгоритм компьютерного зрения для идентификации элементов на изображении	17
1.3.1 Сегментация.....	19
1.3.2 Локализация и классификация.....	21
1.3.3 Искусственные нейронные сети	22
1.4 Глубокое обучение.....	24
1.4.1 Современные архитектуры CNN	25
1.4.1.1 INCEPTION.....	25
1.4.1.2 INCEPTION V1	25
1.4.1.3 INCEPTION V2.....	26
1.4.1.4 INCEPTION V3.....	26
1.4.1.5 RESNET	30
1.4.1.6 INCEPTION-RESNET V2	32
1.5 Набор данных	36
1.5.1 Существующие наборы оценочных данных	37
1.6 Аугментация	41
1.6.1 Существующие инструменты аугментации	42
1.6.1.1 Геометрические преобразования.....	42
1.6.1.2 Преобразования яркости и цвета.....	43
1.6.1.3 Добавление шума	45
1.6.1.4 Локальные фильтры.....	45
2. Выводы по главе 1.....	46
Глава 2. Разработка оптического детектора распознавания кода литой заготовки	48

2.1 Создание собственного уникального набора данных	48
2.2 Разработка алгоритма компьютерного зрения	53
2.2.1 Сегментация методом пороговой бинаризации	53
2.2.2 Локализация	60
2.2.2.1 Локализация объектов с помощью скользящих окон	60
2.2.2.2 Пирамида изображений	61
2.2.2.3 Классификатор наличия цифр клейма	62
2.2.2.4 Алгоритм локализации объекта	63
2.2.3 Классификация	64
2.2.4 Алгоритм автоматического распознавания кода заготовки	75
2.2.5 Результаты работы алгоритма	77
2.3 Детектор на основе глубоких нейронных сетей	78
2.3.1 Обучение модели глубокого обучения	78
2.3.2 Алгоритм работы системы на основе модели глубокого обучения	81
2.3.3 Результаты работы	83
3. Выводы по главе 2	85
Глава 3. Разработка способов повышения эффективности нейросетевого детектора	86
3.1 Аугментация	86
3.1.1 Контрольная выборка	87
3.1.2 Метод поворота	89
3.1.3 Метод сдвига	90
3.1.4 Метод изменения яркости	92
3.1.5 Методы размытия и изменения резкости	93
3.1.6 Способ оптимизации времени обучения	95
3.2 Результаты аугментации	97
3.3 Каскад малых нейронных сетей	99
3.3.1 Тестирование нейронных сетей прямого распространения	99
3.3.2 Тестирование свёрточных нейронных сетей	103
3.3.3 Тестирование нейронных сетей на двух классах	105

3.4 Алгоритм повышенной эффективности идентификации кодов литых заготовок	107
3.5 Результаты работы	110
4. Выводы по главе 3.....	110
Глава 4. Аprobация разработанной системы распознавания кода клейма литой заготовки	112
4.1. Техническое обеспечение системы	112
4.2. Системное программное обеспечение	115
4.2.1 Специальное программное обеспечение	116
4.2.2 АСУТП «Система слежения за металлом на участке печей СПЦ-1 PR-OS».....	121
4.3 Итоговый алгоритм работы системы	123
4.4 Испытания системы распознавания клейма, интегрированного в АСУ СПЦ-1 АО «ОЭМК».....	128
5. Выводы по главе 4.....	130
Заключение	131
Список использованной литературы.....	133
ПРИЛОЖЕНИЕ 1	142
ПРИЛОЖЕНИЕ 2	147
ПРИЛОЖЕНИЕ 3	149

Введение

Актуальность и степень разработанности темы исследования.

В настоящий момент времени металлургическая промышленность является одной из самых ресурсоёмких отраслей в Российской Федерации. Высокая сложность технологического процесса изготовления металлургических изделий, значительная стоимость необходимого оборудования и невозможность быстрой замены элементов в технологической цепочке предприятия без длительных простоев делает самым предпочтительным выходом, для повышения эффективности производства, интеллектуализацию производства с внесением изменений в программное и алгоритмическое обеспечение автоматизированных систем управления технологических процессов. А учитывая непрерывный рост потребности и стоимости металлургических продуктов, повышает актуальность увеличения эффективности производства.

Одной из первых технологических операций при изготовлении проката в сортопрокатном цехе является подача непрерывно литых заготовок со склада на печи нагрева. Перед посадом заготовки в печь, необходимо произвести распознавание её клейма. Для этого на посту контроля оператор посредством камеры сверяет клеймо, нанесённое на заготовку, с планом, и принимает решение о дальнейших действиях с ней. Ввиду того, что код заготовки считывает человек, который действует в жестких временных ограничениях в купе с наличием иных производственных функций, требующих его внимания, то в ситуациях, когда клеймо перевернуто, цифры клейма сильно повреждены окалиной, клеймовочная машина с искажениями выбила цифры или идёт быстрый темп подачи заготовок, риск ошибки при распознавании оператором кода клейма сильно возрастает. Ошибочное распознавание приводит к негативным последствиям, таким, например, как отгрузка заказчику металла с заниженными характеристиками из-за некорректной марки стали или даже полной остановки технологической цепочки сортопрокатного цеха. Все ранее

сказанное позволяет утверждать, что усовершенствование системы контроля за движением заготовок на посту управления посадом металла в нагревательную печь, которое позволяет увеличить точность распознавания кодов литых заготовок за счет разработки алгоритмов контроля на основе современных автоматизированных систем, является актуальной научной задачей.

Наиболее актуальным подходом для решения поставленной задачи будет применение технологий искусственного интеллекта, включая алгоритмы машинного обучения совместно с методами компьютерного зрения. В ходе исследования на основе собранной, аннотированной базы данных цифр клейма предполагается обучить ряд нейронных сетей, составляющих основу системы оптического распознавания. Внедрение функций автоматизированного контроля клейма литой заготовки в действующую АСУТП позволит снизить участие человека в задаче распознавания и будет задействовать его только когда обнаружено несоответствие между фактическим кодом заготовки и планом.

Общие вопросы использования методов машинного обучения и в частности глубоких нейронных сетей для поиска объектов на изображении исследуются в работах таких авторов как М. Нильсен, Д. Кризель, Г. Джеймс, Д. Уиттен, Т. Хасти, Р. Тибширани, Т. Сегаран, Р. Рохас, Л. Дэн, Я. Гудфеллоу, И. Бенджио, А. Курвилль, К.М. Бишоп, С. Хайкин. Использование нейронных сетей в металлургической промышленности представлено в работах Полещенко Д.А, Ю.И. Еременко, А.С. Платоновой, Л.Б. Аксёнова, Й. Петерс, Д. Х. Диас Ромеро, Т. Феррейра, Р. Нашсументу, Ж. Вейга, Ц. Тан, Ё. Ли, Х. Джун, В. Новак, П. Гуртик, Х. Хабибалла, Р. Али, К. Франсис, З. Нун, М. Ян, М. Кастелан, М. Мюллер, Д. Бриц, Ф. Мюклих.

Объектом исследования является процесс контроля кода клейма, нанесенного машинописным способом, в АСУТП движением литых заготовок на посту управления металлургического предприятия.

Предметом исследования являются математические методы машинного зрения и модели архитектур искусственных нейронных сетей.

Цель и задачи исследования.

Целью исследования является повышение эффективности контроля за движением заготовок на посту управления металлургического предприятия за счет внедрения в АСУТП системы поддержки принятия решений, способной автоматически распознавать код клейма, нанесенного машинописным способом на литую заготовку.

Для реализации поставленной цели необходимо решить следующие задачи:

1. Выполнить анализ и исследовать эффективность существующих наборов данных изображений цифр и методов распознавания образов на изображении.

2. Собрать в промышленных условиях, аннотировать и расширить с применением методов аугментации собственный уникальный набор изображений цифр клейма литой заготовки.

3. Выполнить анализ и исследовать эффективность существующих архитектур искусственных нейронных сетей глубокого обучения для применения в реальных условиях для задач распознавания цифр кодов литых заготовок.

4. Разработать алгоритм автоматического распознавания машинописных цифровых кодов на литых заготовках для действующей АСУТП, использующий основной нейросетевой детектор и ансамбль малых нейронных сетей, обеспечивающий адаптивную поддержку его работы.

5. Разработать автоматизированную систему поддержки принятия решений оператором, включающую интеллектуальное распознавание номеров заготовок, верификацию в базе данных цеховой АСУТП и выдачу рекомендаций в режиме реального времени.

6. Создать программное обеспечение для АСУТП, объединяющее контроль клейма литых заготовок, работу с контроллером, SCADA-системой

и модулем интеллектуального распознавания с поддержкой промышленных протоколов.

7. Провести экспериментальную оценку эффективности контроля за движением литых заготовок на посту управления металлургического предприятия разработанной и внедренной системы поддержки принятия решений, способной автоматически распознавать код клейма, нанесенного машинописным способом.

Научная новизна исследования.

- Разработана процедура искусственного увеличения обучающей выборки (аугментация), отличающаяся добавлением к ней изображений с шумами, изменениями их геометрии и ориентации на плоскости в определенных пропорциях, обеспечивающая повышение на 1,1 % точность распознавания цифр клейма литой заготовки. (соотв. п. 7 паспорта научной специальности 2.3.3)

- Разработана процедура распознавания кодов литых заготовок, отличающаяся ансамблированием нейросетевого детектора с каскадом малых нейронных сетей, которые адаптивно подключаются для выполнения уточняющего анализа, позволившая на 2,1 % увеличить точность распознавания относительно применения отдельных детекторов. (соотв. п. 7 паспорта научной специальности 2.3.3)

- Разработан алгоритм, обеспечивающий поддержку оператора в режиме советчика при решении прикладной задачи распознавания цифрового кода, нанесённого машинописным способом на литую заготовку в АСУТП металлургического предприятия, отличающийся интеллектуализацией обработки информации за счет применения предложенной процедуры распознавания кодов литых заготовок. (соотв. п. 14 паспорта научной специальности 2.3.3)

- В рамках АСУТП разработана система (программный комплекс), обеспечивающая поддержку оператора в режиме советчика, отличающаяся оригинальной процедурой обработки данных, основанной на

интеллектуальном распознавании номера литой заготовки, его верификацией в БД АСУТП металлургического комбината с выдачей рекомендаций оператору в SCADA систему в режиме реального времени. (соотв. п. 9 паспорта научной специальности 2.3.3)

Теоретическая значимость работы заключается в разработке алгоритмов для построения интеллектуальной системы для задач распознавания клейма, набиваемого машинописным способом на литых заготовках металлургических предприятий. Разработанная процедура аугментации промышленных данных позволяет формировать качественный набор для обучения глубоких нейронных сетей и учитывать специфику работы клеймовочной машины. Разработанные алгоритмы позволяют нивелировать такие факторы, мешающие корректному распознаванию, как недостаточная освещённость, наличие окалины или других физических повреждений, схожесть признаков для разных классов.

Также в ходе исследования был сформирован один из самых больших наборов данных, который пригоден для решения широкого класса задач машинного обучения, состоящий из изображений цифр клейма.

Практическая значимость заключается в программной реализации системы автоматического распознавания кода клейма литой заготовки. Разработанная система реализована в виде программы, захватывающей видеопоток с камеры, установленной на посту контроля и выдающей числовой код заготовки. Разработанная система реализована как система советчик оператору АСУТП и позволяет автоматизировать его работу на посту контроля, за счёт связи с промышленным контроллером Simatic S7-400 и непосредственной интеграцией в существующую SCADA систему WinCC. Данная система обеспечивает поддержку оператора в реальном времени при считывании клейма литой заготовки перед отправкой заготовок в печи нагрева. Разработанный оригинальный программный модуль интеллектуального детектирования клейма позволяет исключить ошибочный посад заготовки в печь нагрева, а соответственно избавить производственный

процесс от простоя технологической линии, поломок оборудования и связанных с этим финансовых издержек.

Методология и методы исследования. В ходе данного диссертационного исследования применялись классические методы теории компьютерного зрения, используемые для задач предобработки изображения для выделения признаков искомым элементов. Также в работе использовались методы машинного обучения, основным из которых выступает использование глубоких искусственных нейронных сетей для решения задач локализации и распознавания объектов на изображении.

Методологическую и теоретическую основу диссертационной работы составили научные труды отечественных и зарубежных авторов в области компьютерного зрения, машинного обучения и нейроструктурного моделирования.

Положения, выносимые на защиту.

- Оригинальная процедура искусственного увеличения обучающей выборки обеспечивает на 1,1 % повышение точности результатов распознавания цифр клейма литой заготовки за счёт добавления к обучающей выборке изображений с шумами, изменениями их геометрии и ориентации на плоскости.

- Применение процедуры ансамблирования нейросетевого детектора каскадом малых нейронных сетей, адаптивно подключающихся для выполнения уточняющего анализа, позволяет на 2,1 % увеличить точность распознавания относительно применения отдельных детекторов.

- Применение алгоритма поддержки оператора в режиме советчика при распознавании цифрового кода, нанесённого машинописным способом на литую заготовку в АСУТП металлургического предприятия, отличающегося применением нейронных сетей, позволяет повысить уровень автоматизации поста оператора и снизить вероятность ошибки за счет увеличения точности распознавания до 97,6 %.

- Применение системы (программный комплекс), обеспечивающей поддержку оператора в режиме советчика, основывающейся на оригинальной процедуре обработки данных, использующей интеллектуальное распознавание номера литой заготовки с последующей его верификацией в БД АСУТП металлургического комбината с выдачей рекомендаций оператору в SCADA систему, позволяет решать поставленную задачу в режиме реального времени.

Степень достоверности и апробации результатов. Достоверность результатов подтверждается тем, что они частично совпадают с результатами исследований нейросетевых структур зарубежных авторов, чьи работы представлены в научных изданиях, в корректном использовании методов аугментации, распознавания образов, сегментации, обучения нейросетевых архитектур. Также достоверность результатов подтверждается работой нейросетевого детектора клейма в условиях реального производства.

Результаты исследования были внедрены на ПУ7 СПЦ-1 АО «ОЭМК им. А.А. Угарова».

Публикации. По теме диссертации опубликовано 14 работ, в том числе 3 статьи, индексируемых в МБД Scopus; 2 статьи в рецензируемых журналах из перечня ВАК при министерстве науки и высшего образования РФ, 5 публикаций в материалах международных и всероссийских конференций, 4 свидетельства о регистрации программы для ЭВМ

Выносимые на защиту результаты получены лично соискателем. В соавторских публикациях вклад автора заключался:

- 1) построении и обучении моделей машинного обучения.
- 2) синтезе и настройке подсистем, корректирующих основную модель машинного обучения.
- 3) сборе, обработке и преобразовании данных, используемых для обучения нейросетевой модели.

Внедрение результатов исследования. Полученные в ходе исследования практические и теоретические результаты были внедрены в

производство в качестве системы распознавания клейма на литых заготовках в сортопрокатном цехе №1 АО «ОЭМК им. А.А. Угарова».

Также теоретические результаты были использованы в учебном процессе Старооскольского технологического института им. А.А. Угарова (филиала) ФГАОУ ВО «Национальный исследовательский технологический университет «МИСИС» в дисциплинах «Нейросетевые системы», «Нейросетевые технологии в прикладных задачах»

Объём и структура работы. Диссертация состоит из оглавления, введения, четырёх глав и списка использованной литературы, который содержит 79 наименований, и 3 приложение. Полный объём диссертации составляет 158 страниц. Основной текст работы включает 89 рисунка и 12 таблиц.

Диссертация соответствует пунктам паспорта специальности 2.3.3. «Автоматизация и управление технологическими процессами и производствами»:

10. Формализованные методы анализа, синтеза, исследования и оптимизация модульных структур систем сбора, хранения, обработки и передачи данных в АСУТП, АСУП, АСТПП и др.

12. Методы создания специального математического и программного обеспечения, пакетов прикладных программ и типовых модулей функциональных и обеспечивающих подсистем АСУТП, АСУП, АСТПП и др., включая управление исполнительными механизмами в реальном времени.

15. Теоретические основы, методы и алгоритмы диагностирования (определения работоспособности, поиск неисправностей и прогнозирования) АСУТП, АСУП, АСТПП и др.

Глава 1. Анализ процесса распознавания клейма литой заготовки

В современном мире рыночные условия диктуют крайне жёсткие требования для всех отраслей промышленности и влекут за собой требования к сокращению издержек и повышению эффективности технологического процесса. Одним из методов решения выступает замена рабочего персонала в низкоквалифицированных видах деятельности на автоматизированные системы, частично или полностью замещающие человека.

Несмотря на то, что каждое металлургическое производство в Российской Федерации имеет крайне высокий уровень внедрения средств автоматизации, новые возможности, принесённые в индустрию нейронными сетями и методами машинного обучения позволяют охватить тот спектр задач, который ранее требовал значительных экономических затрат и в сравнении со стоимостью рабочей силы был не рентабелен.

АО «ОЭМК им. А.А. Угарова» представляет собой современное металлургическое производство. В технологическую цепочку предприятия входят: сортопрокатный цех №1, сортопрокатный цех №2, электросталеплавильный цех, цех металлизации, цех окомкования, объекты складского и транспортного хозяйства и вспомогательные цеха.

1.1 Описание участка контроля литых заготовок перед посадом в печи нагрева

Продукция, изготавливаемая АО «ОЭМК им. А.А. Угарова», изготавливается по специальным техническим условиям, и по техническим характеристикам существенно лучше обычного металла.

В 1987 году был введён в эксплуатацию крупносортно-заготовочный стан 700. В качестве поставщика технологического оборудования выступила фирма «Шлеман-Зимаг», термические печи поставлялись фирмой «Стейн-

Ортей”, производителем средств автоматизации и прочего электрооборудования выступила фирма “Сименс”.

В состав цеха входят 9 участков.

В настоящее время на участок №2 «Нагревательные печи и печи гомогенизации», находящиеся в СПЦ-1, приходят заготовки с участка №1 «Склад для непрерывнолитых заготовок». На заготовке клеймовочной машинной выбит девятизначный код, содержащий в себе информацию о ней. Первый пять символов несут в себе информацию о номере плавки, шестой о номере машины литья, в последних трёх цифрах содержится информация о номере заготовки.

Для оператора наиболее информативными являются первые пять символов, так как в них содержится информация о металле, и ошибка повлечёт за собой материальный ущерб.

Оператор считывает код заготовки с монитора, на который по средствам аналоговой камеры поступает изображение, и в зависимости от значения кода происходит принятие решения.

Схема автоматизации при проектировании систем управления является основным документом, по которому устанавливают степень автоматизации всего технического процесса некоего объекта. Так же по этой схеме можно установить тех. оснащённость средствами и приборами автоматизации.

Схема автоматизации рассматриваемой системы управления изображена на рисунке 1. В данной схеме автоматизации применяются следующие обозначения:

М – привод;

IS – реле тока;

NS – электромагнитный пускатель;

GS – индуктивные датчики положения;

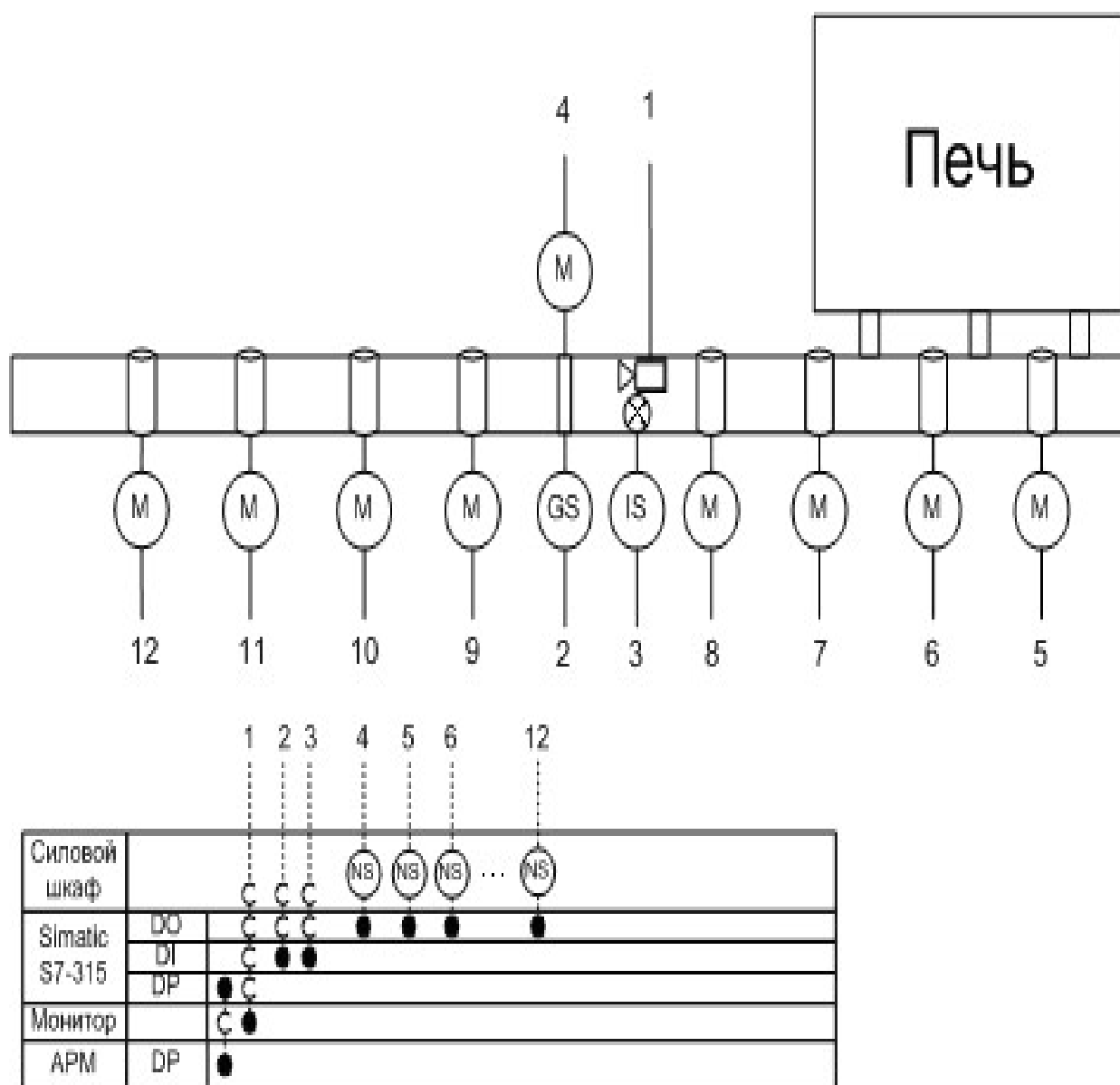


Рис. 1 – Схема автоматизации

На рисунке 2 представлена схема комплекса технических средств АСУ ТП.

Представленная схема содержит в себе три уровня. На нижнем (полевом) уровне располагаются исполнительные механизмы и датчики, которые выполняют функции сбора информации и выполнения технологических задач объекта управления. На среднем уровне расположен контроллер фирмы «Siemens» Simatic S7-315, который управляет и контролирует технологический процесс. На верхнем уровне располагаются АРМ (автоматизированное рабочее место) с монитором, АРМ связан контроллером по интерфейсу Ethernet.

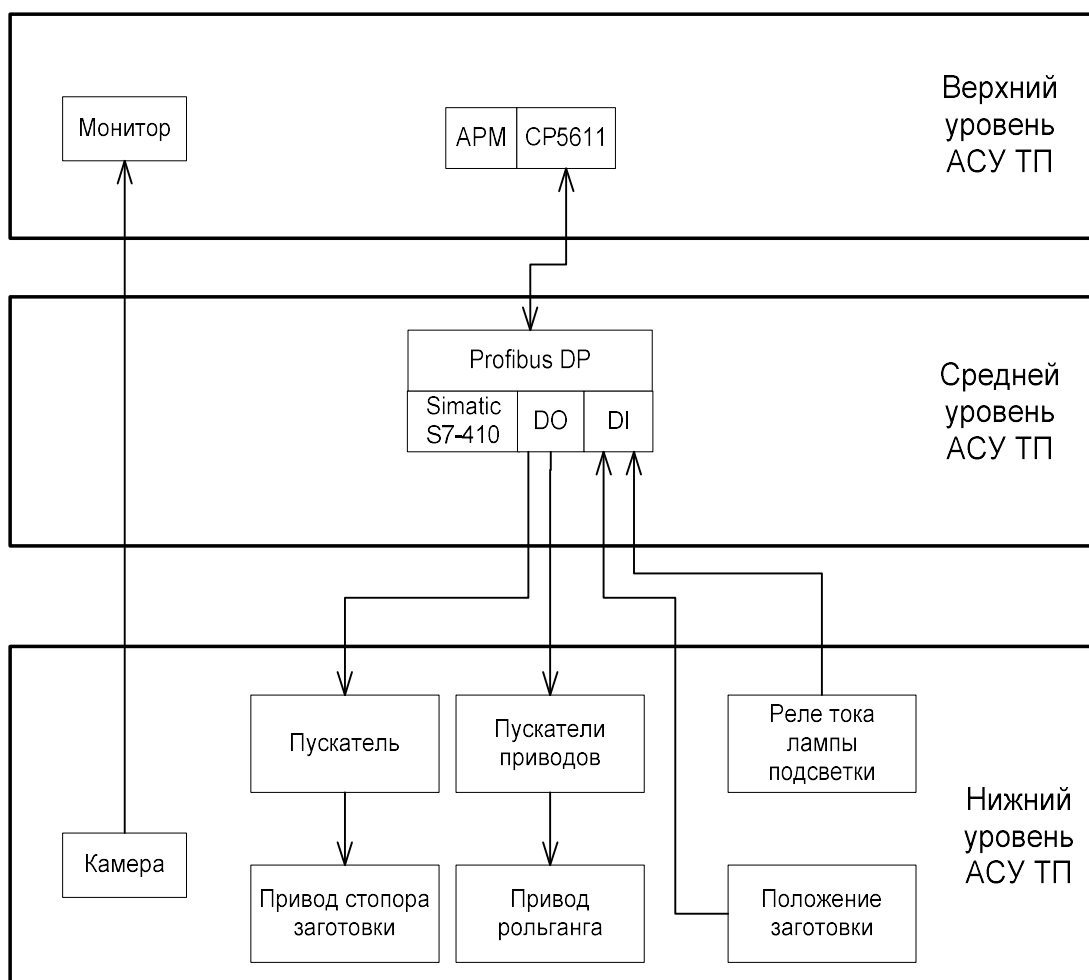


Рис. 2 – Схема КТС

1.2 Проблематика распознавания клейма литой заготовки

Существующий уровень автоматизации не способен в должной степени минимизировать возможные риски. Из-за того, что оператор сам считывает код заготовки, возможно неправильное определение цифр клейма, что может повлечь за собой простой и финансовые потери.

Из-за того, что цифры кода заготовки порой трудно читаемы (это вызвано погрешностями клеймовочной машины и окалиной, которая образуется на заготовке) в случаях, когда идёт непрерывная подача заготовок, у оператора есть порядка 3-х секунд чтобы считать информацию и принять решение по поводу управления движением заготовки. Ошибка оператора приводит к неблагоприятным последствиям.

Обнаружение ошибки оператора до проката приведёт к вынужденному извлечению заготовки из печи, что приведёт к срыву работы всего цеха из-за простоя.

В случае если ошибка не будет обнаружена на этапе её нахождения в печи, есть риск отправки потребителю продукции с другими свойствами, что повлечёт за собой финансовые взыскания.

Исходя из этого в работе разрабатывается алгоритм автоматического распознавания кода заготовки для верификации его с данными базы АО «ОЭМК» на план проката и снижения риска ошибки оператора.

1.3 Анализ методов позволяющих реализовать алгоритм компьютерного зрения для идентификации элементов на изображении

Алгоритм компьютерного зрения — это математическая модель, которая помогает компьютеру интерпретировать изображение. Люди интерпретируют изображения многими сложными способами, в то время как реальная способность компьютеров интерпретировать изображения чрезвычайно ограничена, даже с учётом новейших технологий. Алгоритмы компьютерного зрения способны получать определенные виды данных с изображения.

Задача инженера, использующего алгоритмы компьютерного зрения, заключается в том, что видение основывается на серии выводов, связанных с неизвестными элементами изображения. Ученые называют это «обратной проблемой» и говорят, что только сложные модели вероятностей на основе физики могут помочь технологии интерпретировать изображение по-разному [2]. Ученые работают над общей проблемой компьютерного зрения под разными углами, включая трехмерное моделирование, распознавание частей изображения и интеллектуальное моделирование изображений [1].

Широкий спектр алгоритмов компьютерного зрения работает по-разному. Некоторые из них работают над определением определенных частей фотографии или изображения [3]. Некоторые из таких технологий приводят к

появлению новых функций распознавания лиц на камерах или в оборудовании для обеспечения безопасности [5].

Другие виды алгоритмов компьютерного зрения пытаются захватить определенные виды данных из изображения. Многие из них теперь используются в розничной сфере, где идентификация шаблонов из документов может быть полезна, например, при анализе печатных штрих - кодов и аналогичных идентификаторов [4].

Задачи компьютерного зрения включают в себя методы получения, обработки, анализа и понимания цифровых изображений и извлечения данных из реального мира для получения числовой или символической информации. Понимание в этом контексте означает преобразование визуальных образов в формат понятный для машины, который может взаимодействовать с другими процессами и вызывать соответствующие действия.

Классической проблемой компьютерного зрения, обработки изображений и машинного зрения является определение того, содержит ли данное изображение определенный объект, функцию или активность. Существуют следующие разновидности проблем распознавания:

Классификация объектов – это возможность распознавания одного или нескольких предварительно определенных или изученных объектов, или классов объектов на изображении.

Идентификация объекта — это распознавание конкретного экземпляра объекта.

Обнаружение объекта — это способность идентифицировать, что в изображении есть объект.

В нашем случае необходимо создать алгоритм классификации объектов на изображении.

Данный алгоритм имеет три этапа:

Сегментация – на этом этапе происходит выделение символов на изображении. Это необходимо для приведения изображения к виду, удобному для анализа и дальнейшей локализации.

Локализация – на этом этапе происходит определения места расположения искоемых объектов на изображении для их дальнейшей классификации.

Классификация – на этом этапе система принимает решение о том, какому из заранее известных классов отнести выделенный на этапе локализации элемент.

1.3.1 Сегментация

Основной целью сегментации изображения является независимое от домена разбиение изображения на множество непересекающихся областей, которые визуальны различны, однородны и значимы в отношении некоторых характеристик или вычисленного свойства, такого как уровень серого, текстура или цвет, чтобы обеспечить легкий анализ изображения [6].

Сегментация изображений - полезный инструмент во многих сферах, включая промышленность, здравоохранение, астрономию и другие области [7].

Просто глядя на изображение, можно определить, какие объекты содержатся в изображении. Это здание, человек, ячейка или просто фон. Визуально очень легко определить, что представляет собой область интересов, а что нет. С другой стороны, сделать это с помощью компьютерного алгоритма не так просто. Как определить, что определяет регион. Какие особенности отличает один регион от другого [6].

Что определяет, сколько регионов есть в данном изображении. Разрыв и сходство или однородность являются двумя основными свойствами пикселей относительно их локальной окрестности, используемой во многих методах сегментации. Методы сегментации, основанные на свойстве разности пикселей, рассматриваются как граничные или основанные на краях методы и основанные на сходстве или однородности, основаны на региональных методах [8].

Сегментацию изображения осложняют следующие факторы:

- Шум изображения
- Слабые границы объектов
- Неоднородная область объекта
- Слабый контраст

Благодаря появлению различных технологий, обработка изображений была привлекательной задачей среди исследователей. Но сложна, поскольку изображение представляет собой различные формы, цвета, тональные градации, интенсивность, и эта информация должна сохраняться при обработке.

Это во многом зависит от особенностей извлечения, уникальности и правильности. Хотя функции могут меняться от изображения к изображению, существует несколько общих функций, таких как край или граница между объектом и фоном. Сегментация - один из полезных методов обработки таких изображений.

Сегментация изображения относится к процессу разбиения изображения на группы пикселей, которые являются однородными по отношению к некоторому критерию [6].

Она в основном обнаруживает край объектов путем нахождения переходов в яркости изображения. Хотя сегментирование с помощью обнаружения края является одним из основных инструментов, помогающих в анализе диапазона изображений, оно по-прежнему считается нерешенной проблемой.

Алгоритм работы с некоторыми типами изображений может не работать для некоторых других. Следовательно, он должен быть адаптивным по своей природе.

Алгоритм сегментации должен обладать следующими свойствами для достижения наилучшей производительности:

1. Универсальность: алгоритм должен работать для всех типов изображений. Он должен быть адаптивным.
2. Надежность: алгоритм должен быть устойчивым во всех аспектах.

3. Производительность: для практического применения метод должен быть высокоэффективным и обеспечивать лучшие результаты. Он должен обеспечивать точность и обеспечивать желаемую производительность.

4. Эффективное время: алгоритм должен обеспечить высокую скорость, чтобы обеспечить наилучшие результаты [7].

Для локализации объектов изображение необходимо сегментировать. Для сегментации обработанного изображения был выбран алгоритм бинарной сегментации изображений.

1.3.2 Локализация и классификация

Задача обнаружения объектов на изображении представляет собой одно из ключевых направлений в компьютерном зрении. Её суть заключается не только в том, чтобы ответить на вопрос, присутствует ли на картинке искомый объект (например, машина, человек или животное), но и в точном определении его местоположения. Для этого система должна выделить объект в пиксельной системе координат исходного изображения. В зависимости от используемой модели и алгоритма обучения, позиция объекта может быть задана различными способами: это может быть прямоугольная ограничивающая рамка (bounding box), набор ключевых точек, описывающих его характерные особенности, или же точный пиксельный контур (сегментация), повторяющий его форму [9].

Задача классификации изображений – это задача суть, которой в присвоении входному изображению одной метки из фиксированного набора категорий. Это одна из основных проблем в компьютерном зрении, которая, несмотря на свою простоту, имеет большое разнообразие практических ограничений.

Поскольку задача распознавания визуального образа является относительно тривиальной проблемой для человека, стоит рассмотреть проблемы, связанные с перспективой алгоритма компьютерного зрения. При

классификации изображения следует учитывать следующие проблемы влияющие на работу алгоритма:

- Изменение точки обзора. Один экземпляр объекта может быть по-разному ориентирован относительно камеры.
- Изменение шкалы. Визуальные классы часто демонстрируют вариации в своем размере (размер в реальном мире не только по степени их изображения).
- Окклюзия. Объекты, представляющие интерес, могут быть закрыты. Иногда может быть видна лишь небольшая часть объекта (всего несколько пикселей).
- Условия освещения. Эффект освещения сильно зависит от уровня пикселей.
- Фоновый беспорядок. Объекты, представляющие интерес, могут смешиваться с окружающей средой, что затрудняет их определение.
- Внутрикласовая вариация. Классы интересов часто могут быть относительно широкими. Существует много разных типов этих объектов, каждый со своим внешним видом.

Хорошая модель классификации изображений должна быть инвариантной к перекрестному произведению всех этих вариаций, одновременно сохраняя чувствительность к межклассовым вариациям [10].

В качестве классификатора предлагается использовать искусственные нейронные сети.

1.3.3 Искусственные нейронные сети

Искусственные нейронные сети — это вычислительные системы, предназначенные для преобразования входных значений в желаемый результат, функционирование которых основано на принципах биологической нервной системы.

Нейронная сеть — это алгоритм машинного обучения, который соединяет большое количество простых вычислительных блоков для формирования мощной модели [11]. Данную модель трудно обучить из-за требуемой вычислительной мощности и объема данных. Однако, поскольку в последнее время вычислительная мощность компьютеров и количество доступных наборов данных машинного обучения с каждым годом увеличиваются, нейронные сети становятся актуальными [12]. Особенно хорошо нейронные сети показывают себя в задачах классификации изображений. В последние годы произошел значительный сдвиг в решении задач компьютерной обработки изображений и в компьютерном зрении, в основном за счет использования сверточных нейронных сетей [13].

Нейронные сети были разработаны в начале 1940-х годов Маккаллохом и Питтсом [14], они представили первые вычисления с использованием нейронных сетей. Несколько лет спустя, а именно в 1985 году, Розенблатт разработал и сконструировал персептрон [15]. Персептрон состоит из трех слоев: первый слой для входных блоков, средний слой в качестве ассоциативного и последний слой в качестве выходного. Конструкция персептрона была способна распознавать простые цифры. Однако эта конструкция была неспособна решать такие проблемы, как проблема XOR. Затем, в 1974 году, Пол Вербос разработал метод обучения с обратным распространением, который в настоящее время является одним из наиболее важных методов обучения нейронных сетей [16]. Наконец, исследователи продолжали работать над разработкой нейронных сетей с различными задачами, в том числе с такими как распознавание образов.

Нейронные сети применяются для решения сложных задач. Эти проблемы могут быть легко решены человеком, но они слишком сложны для компьютера, например, распознавание образов. Нейронные сети учатся на собственном опыте вместо того, чтобы быть на прямую запрограммированными [17-20].

1.4 Глубокое обучение

Также в настоящее время во всех областях применения компьютерного зрения широкое применение получил такой метод машинного обучения как глубокое обучение искусственных нейронных сетей.

Глубокое обучение — это метод машинного обучения, который становится все более популярным в исследованиях компьютерного зрения. Основное различие между классическим машинным обучением (ML) и глубоким обучением заключается в том, как извлекаются функции. Для классических методов ML, таких как support vector machines (SVM) [21], извлечение функций выполняется заранее с использованием методов, созданных исследователями. Затем процедура обучения разрабатывает веса или правила, которые сопоставляют любой вектор объекта с меткой выходного класса. Обычная процедура глубокого обучения заключается в применении значений сигналов в качестве входных данных непосредственно к сети ML без каких-либо предварительных усилий по извлечению функций. Сеть принимает входной сигнал (значения пикселей, в нашем случае) и назначает метку класса на основе этих значений сигнала напрямую. Поскольку подход к глубокому обучению должен неявно выводить свои собственные особенности, он требует гораздо больше учебных образцов, чем традиционные системы ML.

Для исследователей доступно несколько пакетов глубокого обучения. Пакет, который мы использовали для реализации различных частей предлагаемой системы, - TensorFlow [22], который стал очень популярным в последнее время. TensorFlow - это гибкий и мощный пакет машинного обучения, созданный Google.

Другой пакет, который мы использовали для создания частей системы, - Keras [23], который является широко используемым пакетом глубокого обучения высокого уровня, который работает поверх нескольких пакетов

нижнего уровня, включая TensorFlow. Keras прост в использовании, гибок и может использовать мощные функции, предлагаемые TensorFlow.

1.4.1 Современные архитектуры CNN

В данной разделе проанализированы наиболее известные архитектуры CNN используемые для классификации изображений: Inception (начальная сеть), Resnet (остаточная сеть) и Inception-ResNet. На основе этих архитектур было проведено много исследований, с целью улучшения показателей точности классификации изображений. В следствии чего на протяжении многих лет основы этих архитектур (с теми или иными модернизациями) чаще остальных применяются на соревнованиях по распознаванию изображений ImageNet. [24]

1.4.1.1 INCEPTION

Развитие Inception можно условно разделить на четыре этапа, начиная с введения (Inception V1), добавления метода нормализации партии (Inception V2), оптимизации создания (Inception V3) и объединения с ResNet (Inception ResNet V2).

1.4.1.2 INCEPTION V1

Появление Inception стало заметной вехой в эволюции CNN. Раньше, когда этой архитектуры ещё не существовало, разработчики чаще всего шли самым очевидным путём, а именно наращивали глубину сети, один за другим добавляя свёрточные слои. Смысл подобных операций был просто, чем больше слоёв, тем выше точность и производительность. Но у подобного способа углубления есть обратная сторона, и она проявляется в ряде трудностей:

- Большое количество параметров, которые могут вызвать явление переобучения [25];

-Проблема затухающего градиента - значение ошибки обратного распространения станет близким к нулю при приближении к входному сигналу, с увеличением глубины CNN;

-Увеличение требуемых вычислительных ресурсов, необходимых для обучения CNN;

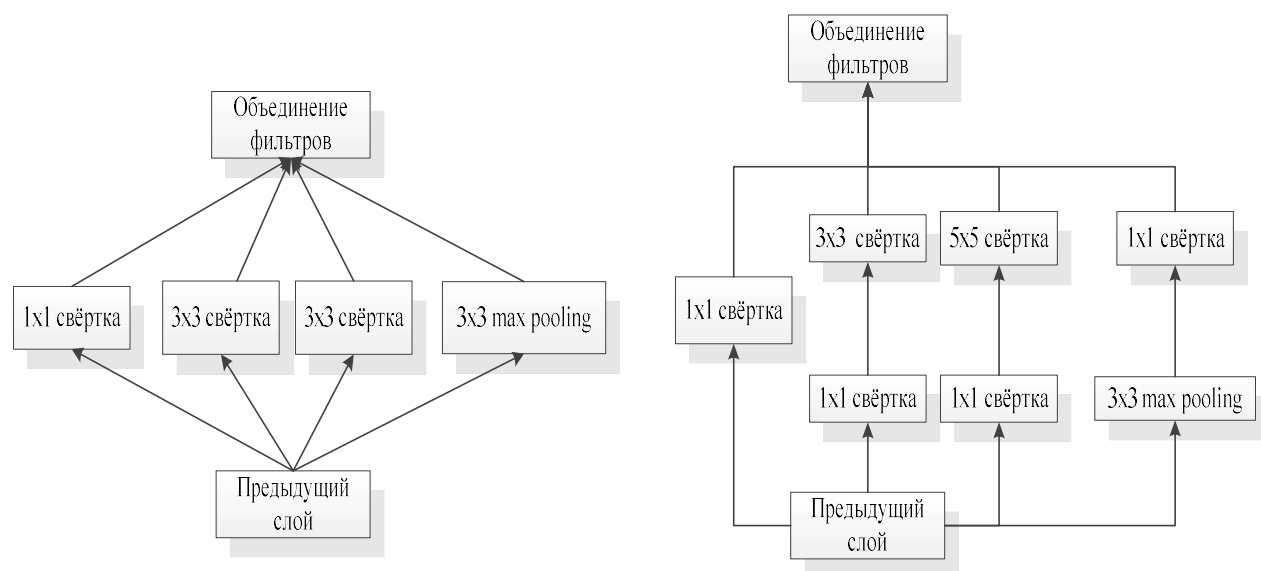


Рис. 3 – Inception блок

На рисунке 3 показано, как выглядит Inception блок. Причина, по которой Inception блок позволяет достичь более высокой производительности, заключается в том, что такой блок собирается захватить несколько различных абстрактных объектов, имея разный размер рецептивного поля на одном и том же уровне. Применение ядра размером 5 на 5 даёт начальному блоку возможность улавливать объекты, занимающие довольно крупные участки. Исследователи обратили внимание: число нейронов на каждом шаге растёт ощутимо, однако вычислительные затраты удаётся удерживать в разумных пределах, а именно за счёт того, что перед свёрткой более крупные блоки подвергаются понижению размерности ядрами 1×1 [26]. В тестах на ImageNet сеть Inception V1 показала точность 89,9% [27].

1.4.1.3 INCEPTION V2

Главное новшество, которое принесло это поколение Inception, особенностью которого является встраивание в сеть пакетной нормализации.

Данная особенность нужна, чтобы справиться с явлением, известным как внутренний ковариантный сдвиг [28]. Суть проблемы в том, что во время обучения параметры пересчитываются на каждом слое, а для CNN входом слоя n служит выход слоя $n-1$. Так как веса предыдущих уровней постоянно обновляются, картина распределения входных сигналов любого слоя всё время смещается, даже если на вход сети подаётся один и тот же образец. Последствия этого как правило негативны, а именно обучение идёт медленнее, приходится занижать скорость обучения и очень аккуратно подбирать начальные параметры. Вдобавок страдают модели с насыщающейся нелинейностью их тренировать становится тяжелее [28]. Для решения этой проблемы необходимо, чтобы для каждой порции данных выходы любого слоя подчинялись одному и тому же распределению. Именно это и делает пакетная нормализация: она приводит выходные значения каждого слоя к гауссову виду с параметрами $N(0, 1)$ [29].

1.4.1.4 INCEPTION V3

Третье поколение Inception выполнило дальнейшую оптимизацию в блоке Inception. Улучшения можно описать в трех аспектах: дальнейшая факторизация размера ядра свертки, использование вспомогательного классификатора и изменение способа изменения размера карт характеристик [30].

Отталкиваясь от идеи разложения свёрточного ядра по размеру, в Inception V3 предложили два подхода к такой факторизации: симметричный и асимметричный.

Первый работает так. Цельное ядро заменяется парой одинаковых ядер, но меньшего размера. К примеру, ядро 5×5 разбивается на два ядра 3×3 (рисунок 4). Выигрыш в вычислениях ощутимый нужно всего $(3 + 3) \times 3 / (5 \times 5) = 18/25 = 72\%$ от того, что требовал исходный начальный блок. При этом зона перекрытия двух ядер 3×3 формирует воспринимающее поле, которое справляется с той же задачей, что и одно ядро 5×5 .

Второй подход. Вместо одного ядра ставятся два разных по размеру. Так, ядро 5×5 превращается в пару из ядра 1×5 и ядра 5×1 (рисунок 5). Здесь затраты падают ещё сильнее до $(5 + 5) / (5 \times 5) = 10/25 = 40\%$. Правда, на первых слоях асимметричный вариант показывает себя слабо. Зато на промежуточных слоях, где-то в диапазоне 12–20, его эффект становится явным.

Следовательно, если начальных блоков много, факторизация свёрточного ядра позволяет серьёзно урезать число параметров. Ещё один из плюсов данного подхода заключается в том, что когда одно ядро делится на два, глубина CNN прибавляет единицу. Это даёт дополнительный уровень нелинейности, а значит, модель становится выразительнее, она способна охватывать больше пространственных объектов, а карты признаков разнообразнее [31].

Второе улучшение - добавление вспомогательного классификатора. Вспомогательный классификатор - это дополнительный классификатор, который выдает результат из середины модели. Было продемонстрировано, что вспомогательный классификатор может помочь улучшить показатели сходимости, а также стабилизировать процесс обучения [32].

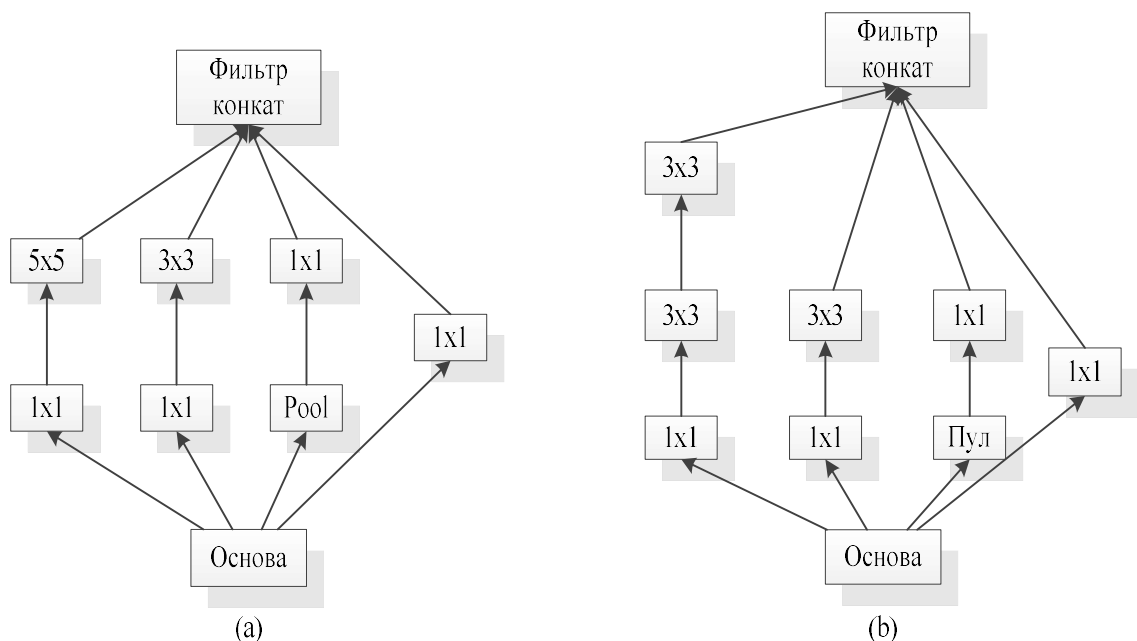


Рис. 4 – (а) Начальный блок без симметричной факторизации; (б) начальный блок, который заменяет ядро 5×5 на два ядра 3×3

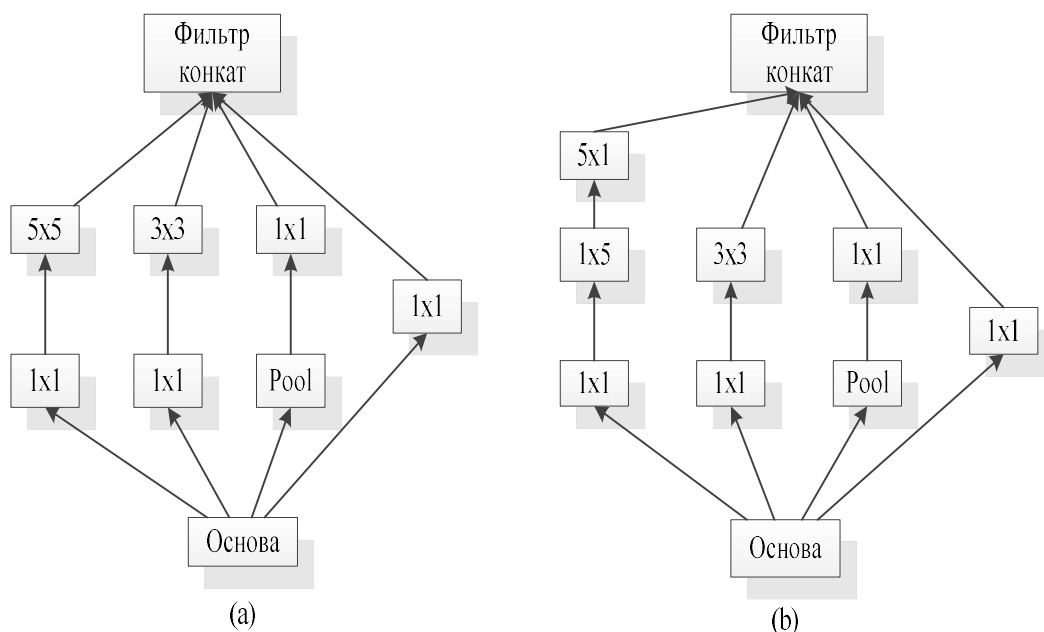


Рис. 5 – (a) Начальный блок без симметричной факторизации; (b) начальный блок, который заменяет ядро 5×5 одним ядром 1×5 и одним ядром 5×1

Третье обновление Inception V3 - эффективный способ уменьшения размера сетки. Обычный CNN обычно использует pooling нелинейных операций, максимальный pooling или средний pooling, для определения выходного размера на каждом уровне. Однако всегда существует критический недостаток, где бы ни находился pooling, до или после захвата объекта с помощью операции свертки (такой как inception блок): если pooling ставится перед слоем свертки, большая часть информации будет отброшена до того, как слои свертки попытаются уловить объект из-за того, что pooling играет роль понижающей дискретизации. Провал в захвате признаков обернётся для модели неспособностью выдавать верные прогнозы. Но есть и другая сторона, когда pooling ставят после свёрточных слоёв, этим слоям приходится вдвое увеличивать размер фильтра, иначе выразительная способность признаков после объединения не сохранится. Иллюстрация этого рисунок 5 (b). Расплатой за такой подход служит заметный рост вычислительного времени. В начале V3 предложили иной вариант, построенный на двух каналах, в одном работает свёрточный слой, в другом уровень pooling. Оба параллельных канала порождают карты объектов одинакового размера. А в финале

результаты этих двух каналов объединяются, что и даёт уменьшение размерности сетки [31,33].

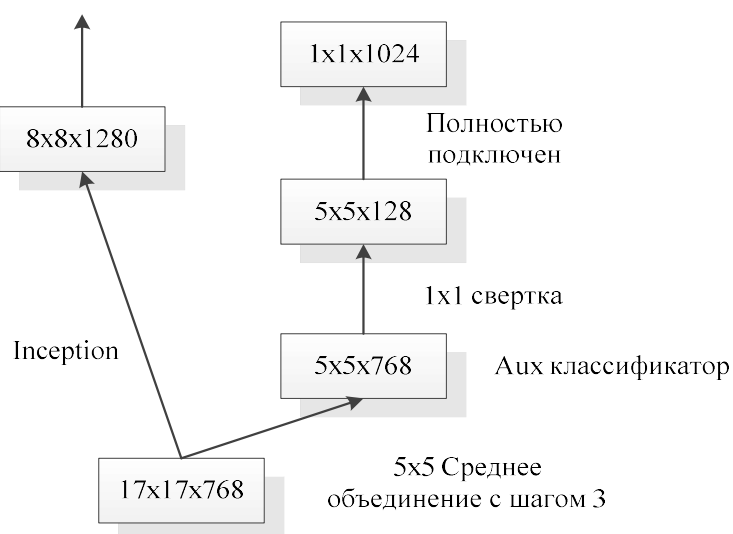


Рис. 6 – Вспомогательный классификатор, добавленный в Inception V3

Если взять замороженную модель Inception V3 из TensorFlow и прогнать через неё валидационную выборку ImageNet, получаем следующие цифры: 77,59% по Top-1 и 93,83% по Top-5 [31]

1.4.1.5 RESNET

ResNet, как и Inception, появился ради преодоления трудностей, которые возникают при чрезмерном углублении CNN [34]. Допустим, мы берём обычную CNN на 20 слоёв и ставим рядом такую же, но на 56 слоёв. Здравый смысл подсказывает, что версия с 56 слоями должна обойти младшую. На практике всё наоборот, что следует из рисунка 9. Сеть из 56 слоёв, как правило, проигрывает 20-слойной, причём и на обучающей выборке, и на тестовой. Эксперименты показывают любопытную картину, поначалу, пока уровень сети растёт, точность модели стабильно улучшается. Но стоит числу слоёв перевалить определённый порог и точность на обучении вместе с точностью на тестах резко падает. Вывод напрашивается сам, чем глубже становится сеть, тем сложнее её тренировать. Причина та же, что и в случае Inception V1, затухание градиента. Из-за него веса в ранних слоях не получают должной настройки.

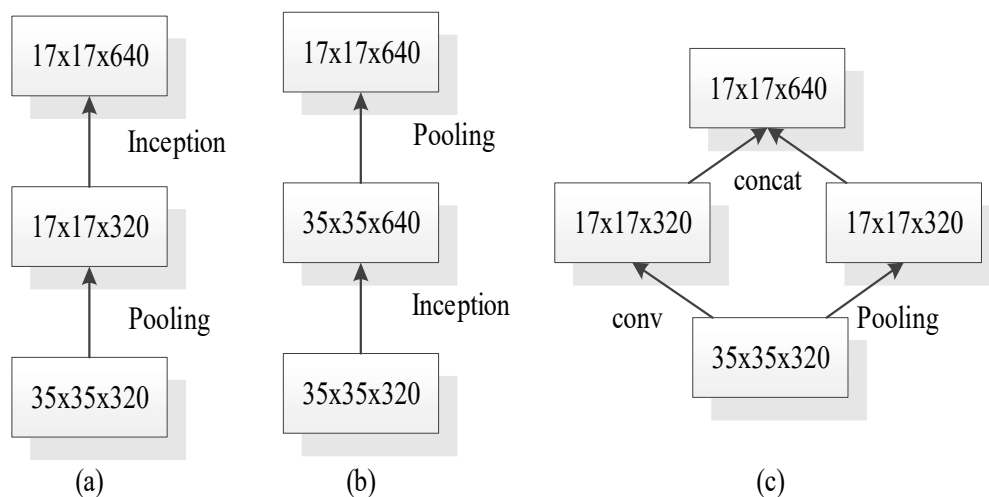


Рис. 8 – (a) Уменьшение размера сети путем объединения перед сверткой, что приведет к потере большого количества информации; (b) уменьшение размера сети путем объединения после свертки, что значительно увеличит объем вычислений; (c)

ResNet на самом деле является комбинацией нескольких мелких сетей. Он не пытается решить проблему затухающего градиента, но пытается ее избежать. Это может ускорить конвергенцию сети и иметь лучшие результаты для более глубоких моделей, поскольку это комбинация нескольких неглубоких сетей, в то время как мелкие сети не будут иметь явного градиентного исчезновения во время обучения [35].

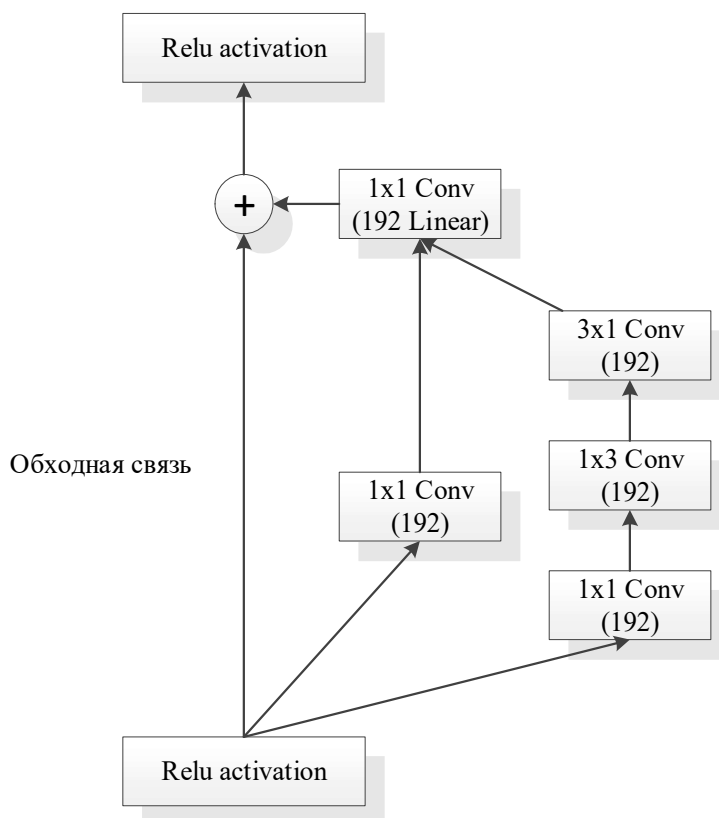


Рис. 22 – Остаточный блок в Inception

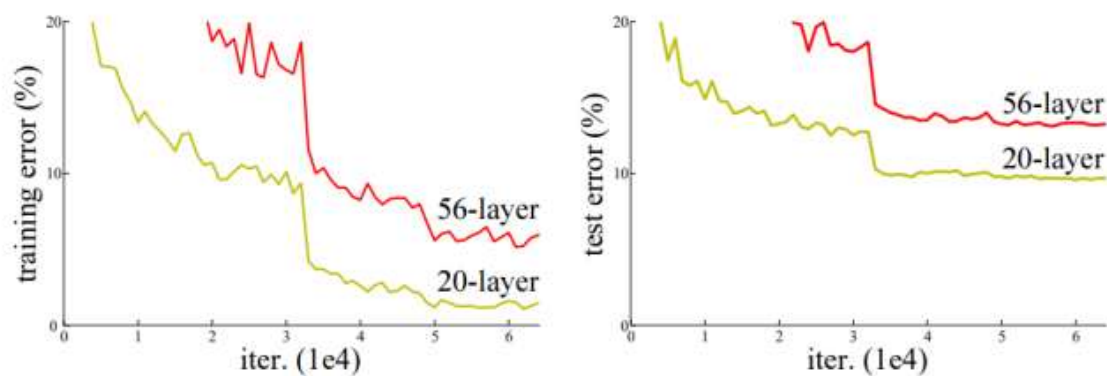


Рис. 9 – Обучение и тестирование трендов ошибок во времени с набором данных CIFAR-10 для обычного 20-слойного и 56-слойного CNN

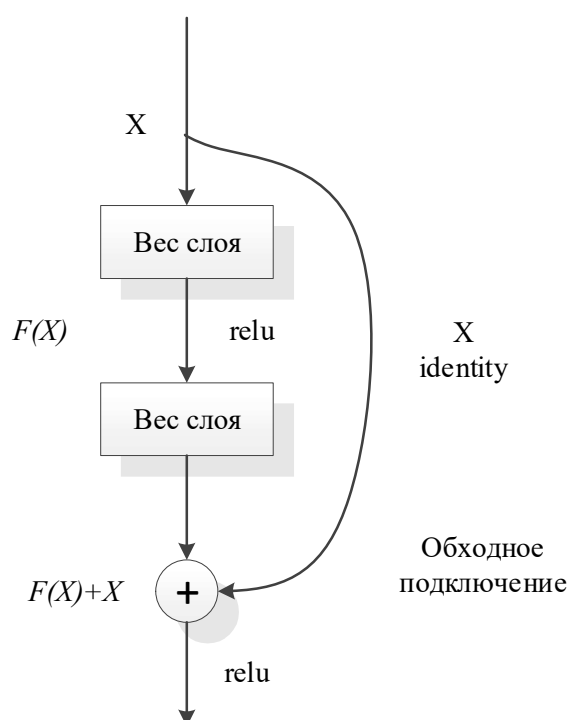


Рис. 10 – Остаточный блок в ResNet

1.4.1.6 INCEPTION-RESNET V2

Как упоминалось ранее, основным принципом развития традиционных архитектур сверточных нейронных сетей является, прежде всего, добавление дополнительных уровней и углубление сети. Однако, чем глубже сеть, тем сложнее ее тренировать. Более того, с определенного момента эффективность начнет снижаться. Microsoft, которая использовала сети с более чем 100

слоями для работы с набором данных ImageNet, решила эту проблему с помощью остаточного обучения [36].

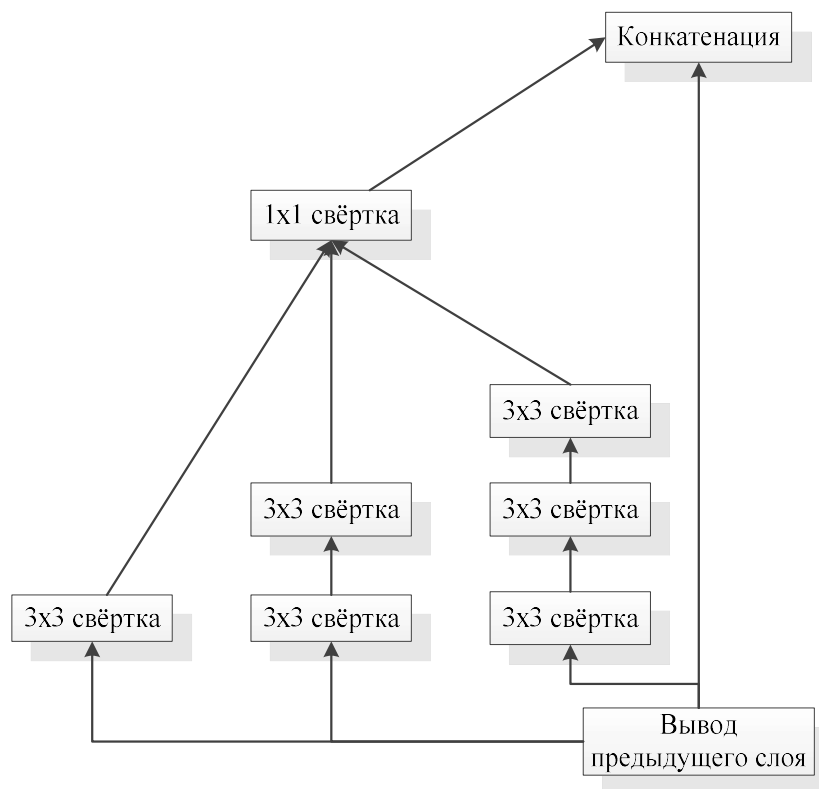


Рис. 11 – Модуль Inception-ResNet A

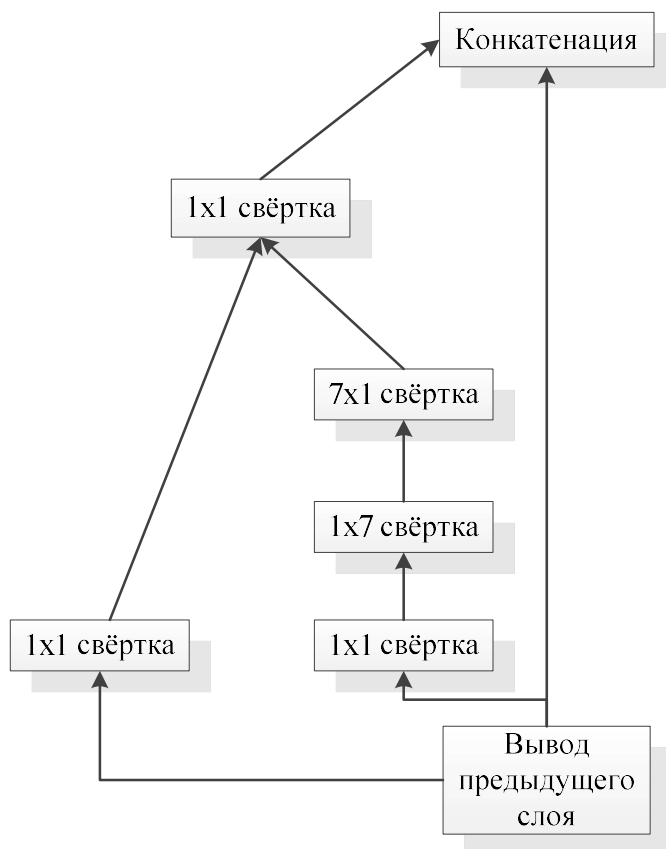


Рис. 12– Inception-ResNet Module B

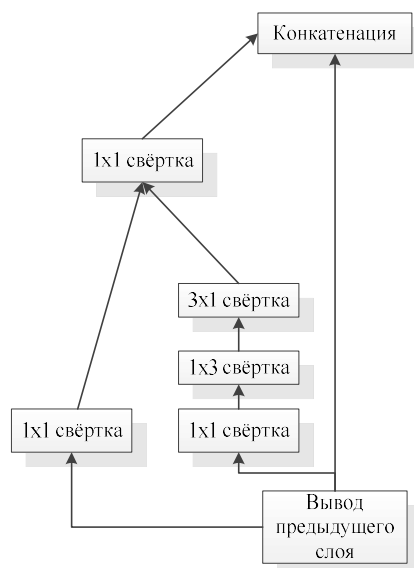


Рис. 13 – Модуль Inception-ResNet C

Основная идея этого заключается в том, чтобы научиться вставлять обходные связи между слоями, поэтому вход нижнего уровня также предоставляется верхнему слою. Это решает проблему снижения успеха сети и, кроме того, облегчает обучение сети.

Используя этот подход, Microsoft сумела выиграть конкурс ILSVRC в 2015 году с сетью под названием ResNet [37], и общее остаточное обучение показало большую полезность в сетях с глубокими свертками. По этой причине Google попытался включить его в новую версию своей Inception сети [36, 38].

Inception-ResNet v2 похож на Inception v4 и основан на подобной архитектуре. Они отличаются только модулями Inception, которые заменяются на новые модули Inception-ResNet. Модули в сети Inception v4 остаются в основном такими же, как и в предыдущей версии Inception v3. Модули Inception-ResNet не включают в себя операцию объединения, и каждый блок Inception сопровождается одной сверткой 1x1 без функции активации, чтобы выровнять выход блока по размерам с входом, который к нему добавлен, см. рисунки 11, 12 и 13 [36].

Исключая различия в модулях Inception / Inception-ResNet, сетевая архитектура InceptionResnet v2 такая же, как и сетевая архитектура Inception v4. В отличие от сетевой архитектуры Inception v3, она отличается в основном

входной частью (самые нижние уровни) сети, показанной на рисунке 14. Общая сетевая архитектура показана в таблице 1. Выходная часть входного сигнала составляет 35×35 , первое уменьшение уменьшает его до 17×17 , второе уменьшение - 8×8 .

В наборе данных ImageNet сети Inception v4 и Inception-ResNet v2 достигли примерно одинакового успеха. Однако использование остаточных соединений в Inception-Resnet v2 значительно увеличило скорость обучения в сети. [36]

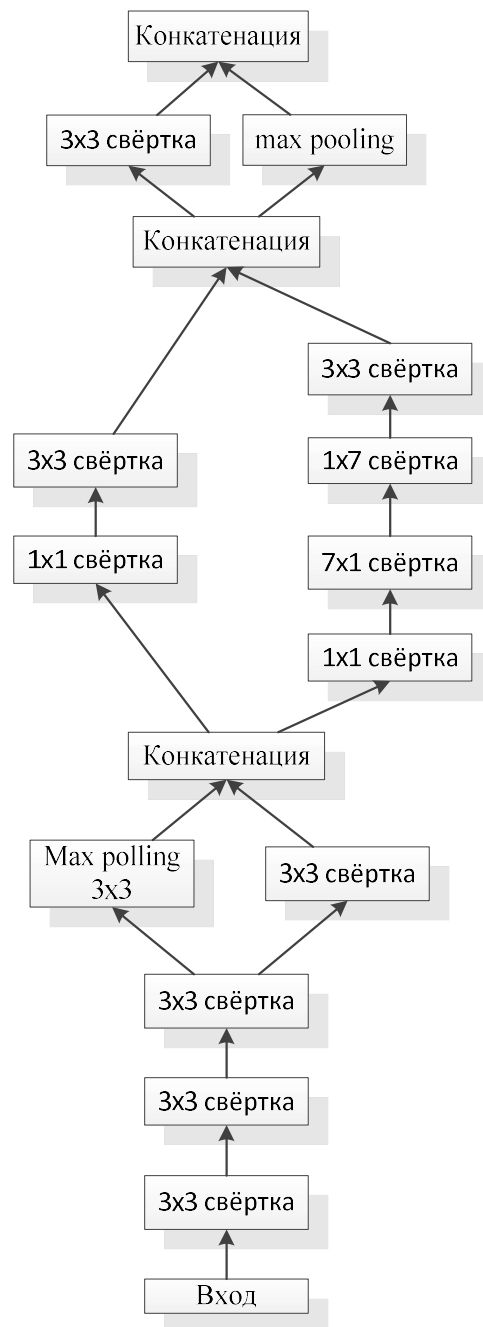


Рис. 14 – Вход Inception-ResNet v2 (нижний уровень)

Таблица 1 – Сетевая архитектура Inception-ResNet v2

Тип слоя	Размер ввода
входные слои	299x299x3
5 x Inception-Resnet A	35x35x256
Сокращение A	35x35x256
10 x Inception-Resnet B	17x17x896
Сокращение B	17x17x896
5 x Inception-Resnet C	8x8x1792
8x8 pooling	8x8x1792
dropout	1x1x1792
Полносвязный слой	1x1x1792

1.5 Набор данных

Процесс обнаружения и распознавания текста в естественных условиях остается сложной задачей. Это связано с разнообразием и сложностью размещения экземпляров, представленных на реальных изображениях, таких как изменение внешнего вида, искажения за счёт разных ракурсов камеры, низкого разрешения, освещения, коллизий, смешивания на сложном фоне, а также множество различных стилей, размеров и т.д.

Отсутствие достаточного количества обучающих наборов данных стало основной проблемой, которая влияет на продвижение исследований в области обнаружения и распознавания текста в реальных условиях. Это побуждает исследователей создавать новые наборы данных, а также разрабатывать новые методы для создания вариаций выборок из существующих наборов данных, используя, например, метод увеличения данных [39]. Увеличение данных - это подход к увеличению обучающих данных из исходного набора данных путем применения преобразования при сохранении исходного класса данных и маркировки. Увеличение данных было признано эффективным для улучшения результатов обучения во многих задачах компьютерного зрения [40, 41]. Доступно несколько библиотек расширения данных для создания дополненных изображений из исходных изображений, [42-44]. Кроме того, фреймворки глубокого обучения, такие как Keras [45] и Tensorflow [46], предоставляют функции расширения. Однако многие из этих библиотек расширения предназначены для решения задач классификации, в то время как

эти библиотеки не предоставляют эффективных инструментов аугментации для задач обнаружения и локализации объектов. Визуальное обнаружение и локализация могли бы извлечь значительные преимущества из расширения данных.

Эта глава способствует созданию нового набора данных. Данное действие мотивировано отсутствием достаточного количества аннотированных обучающих изображений, которые должны включать в себя полный спектр вариаций и разнообразия цифр клейма литых заготовок. Там, где методы глубокого обучения требуют больших объемов данных и различных диапазонов, обучающих данных. Более того, большинство существующих наборов данных аннотируются на уровне строк или слов, однако не хватает наборов данных аннотаций на уровне символов. Кроме того, вторым вкладом в эту главу является инструмент расширения, который создан для поддержки предлагаемого набора данных. Предлагается инструмент для обеспечения увеличения ограничивающих рамок без необходимости аннотирования новых изображений, где положение ограничивающих рамок и класс могут быть получены автоматически из исходного изображения.

1.5.1 Существующие наборы оценочных данных

В этом разделе рассматривается несколько популярных наборов данных, которые используются в исследованиях обнаружения текста.

Многие наборы данных были созданы для решения различных задач, связанных с анализом текста. Примеры задач включают обнаружение текста, числовое распознавание цифр, распознавание символов и распознавание текста на уровне слов [47].

Набор данных MSRA-TD500 представляет собой набор данных для обнаружения текста, который содержит 300 обучающих изображений и 200 тестовых изображений. Текстовые области произвольно ориентированы и аннотированы на уровне предложения. В отличие от других наборов данных, он содержит как английский, так и китайский текст [47].



Рис. 15 – Экземпляры набора данных MSRA-TD500

Так же популярные наборы данных представляются каждые два года на международной конференции по анализу и распознаванию документов (ICDAR). В рамках данной конференции речь идет о распознавании печатного и рукописного текста, анализе и распознавании графики, анализе документов, понимании документов, исторических документах и цифровых библиотеках, криминалистике на основе документов, анализе текста сцены на основе камеры и видео. Особое внимание стоит обратить на следующие наборы данных, созданных в рамках данной конференции:

ICDAR 2003 [48] — это набор данных для распознавания текста сцены. Всего он содержит 507 изображений естественных сцен (включая 258 обучающих изображений и 249 тестовых изображений). Изображения снабжены комментариями на уровне символов. Символы и слова могут быть обрезаны с изображений.



Рис. 16 – Экземпляры набора данных ICDAR 2003

ICDAR 2013 [49] состоит из 229 обучающих изображений и 233 тестовых изображений с предоставленными аннотациями на уровне слов. Это стандартный набор эталонных данных для оценки обнаружения текста вблизи горизонтали. [47]



Рис. 17 – Экземпляры набора данных ICDAR 2013

ICDAR 2015 был представлен как первый набор данных, посвященный случайному тексту [50]. Этот набор данных содержит 1000 цветных изображений для обучения и 500 изображений для тестирования.



Рис. 18 – Экземпляры набора данных ICDAR 2015

MNIST [51], пожалуй, один из самых известных наборов данных с изображением рукописных цифр, который содержит в своём составе десятки тысяч изображений рукописных цифр с метками, который был выпущен в 1998 году. Данный набор был создан в Национальном институте стандартов и технологии в США, из подмножества двух наборов данных [52] над которыми трудились порядка 250 человек. Набор данных MNIST содержит 70 000 изображений рукописных цифр (от 0 до 9). Эти цифры были разделены на две части. Первая часть содержит 60 000 изображений для обучающих наборов.

Эти изображения были нормализованы с помощью некоторого алгоритма нормализации и центрированы в $(28 \times 28) = 784$ пикселях шкалы серого [53]. Эти пиксели представляют собой размерный вектор, представляющий объекты в наборе данных MNIST. Более того, каждый пиксель в наборе данных MNIST представлен значением от 0 до 255, где 0 - это черных, а 255 - белый, а все, что находится между ними, представлено другим оттенком серого. Обучающая часть в самом наборе данных MNIST разделена на две части. Первая часть содержит 50 000 изображений в качестве обучающего набора и 10 000 изображений в качестве проверочного набора. Вторая часть набора данных MNIST содержит 10 000 изображений в качестве тестового набора. MNIST широко используется в учебниках из-за своей простоты и небольшого размера [53].

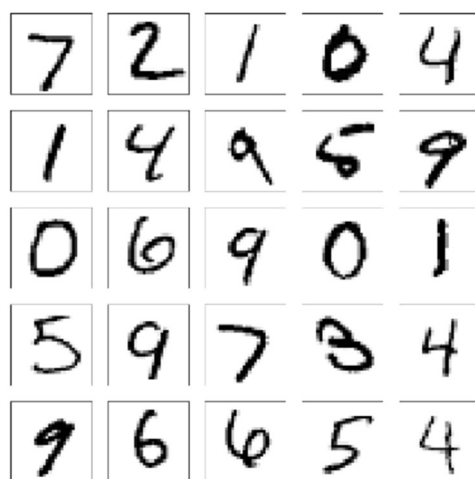


Рис. 19 – Экземпляры набора данных MNIST

Набор данных SVHN [54], как и набор данных MNIST, используется для классификации цифр. Этот набор данных содержит фотографии номеров зданий, полученные из Google Street View. Общий размер набора оставляет порядка 600000 изображений с метками, взятых из реальной среды и за счёт этого их значительно сложнее классифицировать, чем вышеописанный MNIST. Набор данных предоставляется в двух возможных форматах: целые изображения таблиц с цифрами или обрезанные отдельные цифры. Изображения разделены на 73257 обучающих и 26032 тестовых примеров.

Другие изображения были признаны менее сложными и доступны дополнительно. [54]

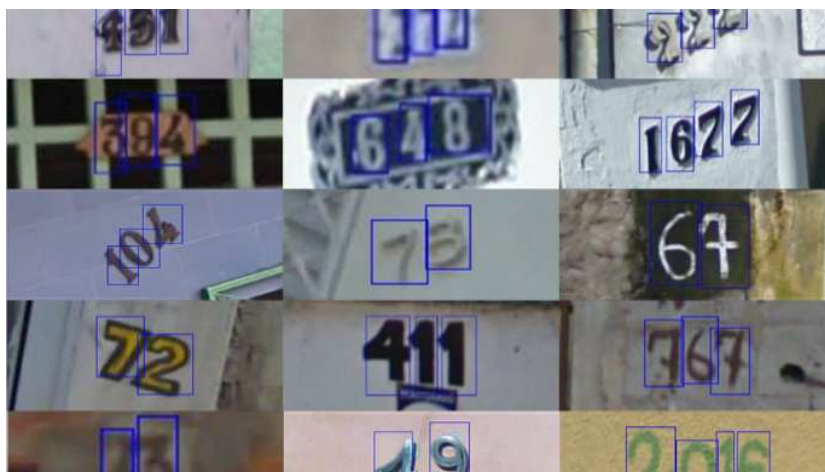


Рис. 20 –Экземпляры набора данных SVHN

1.6 Аугментация

Проводя параллели между работой искусственных нейросетей и когнитивными процессами человека, можно заметить следующее: люди познают мир через сенсорный опыт, постепенно накапливая знания. Столкнувшись с незнакомым занятием, будь то обучение ходьбе, верховой езде или запоминанию информации, человек поначалу действует неуверенно и допускает ошибки. Однако систематическая практика со временем превращает неумение в устойчивый навык.

Аналогичным образом вычислительные модели демонстрируют рост качества предсказаний по мере увеличения объёма релевантных примеров, описывающих исследуемую предметную область. Но на практике получение достаточного количества образцов часто сопряжено с серьёзными трудностями. Их сбор может быть дорогостоящим, трудоёмким или попросту невозможным в нужном объёме. Именно поэтому для восполнения дефицита исходных данных разрабатываются специальные приёмы их искусственного умножения. Данный подход не только решает проблему нехватки информации, но и служит эффективным инструментом регуляризации, снижая риск переобучения модели [55].

1.6.1 Существующие инструменты аугментации

Увеличение количества изображений включает в себя методы, которые манипулируют исходным изображением для создания нового, которое по-прежнему изображает объект интереса узнаваемым образом.

Существует множество инструментов для увеличения данных, особенно данных изображений. Большинство из них созданы как программы командной строки или непосредственно как библиотеки, которые могут быть добавлены в программу (чаще всего они сделаны на Python). Новые методы расширения баз данных изображений все еще разрабатываются и тестируются, что позволило бы повысить квалификацию объектов или скорость обучения нейронных сетей. Чаще всего эти инструменты и подходы создаются для улучшения результатов работы нейронных сетей.

1.6.1.1 Геометрические преобразования

Геометрические преобразования уменьшают зависимость модели от положения. Благодаря своей простоте и быстрой реализации геометрические преобразования широко используются для расширения баз данных изображений. Недостатком может быть память и более высокая вычислительная мощность. Необходимо проверять некоторые методы, чтобы убедиться, что они не слишком сильно повлияли на изображение, например, обрезка [56].

Некоторые геометрические преобразования создают пустые пространства, пиксели, которые не имеют однозначного значения и были добавлены к изображению для сохранения формата. Эти пустые области могут быть заполнены однородным цветом или цветом ближайшего пикселя или исходным значением.

Извлечение патчей — это метод, который возвращает меньшие фрагменты или участки исходного изображения. Таким образом, вместо одного исходного изображения можно использовать большое количество

вложенных изображений, и, поскольку они меньше, вычислительное время на изображение сокращается во время обучения [57].

Зеркальное отображение — это очень простой и в то же время эффективный метод. Это один из первых и наиболее широко используемых методов. Вновь сгенерированное изображение имеет те же размеры, что и исходное. Это осевая симметрия, относительно одной стороны рисунка. Он может быть применен в одной или обеих осях одновременно. Данный метод применим не во всех случаях. Например, если объект съёмки содержит элементы, пространственное положение которых не фиксировано, их использование в определённой зоне изображения может быть затруднено вследствие возможного смещения [58,59].

Перемещение - этот метод сдвигает изображение на определенное количество пикселей по одной или обеим осям, затем вводит пустое пространство с неопределёнными пикселями. Использование метода смещения подходит, если мы хотим добиться меньшей зависимости модели от положения объекта на изображении.

Вращение — это метод увеличения изображения, который поворачивает каждый пиксель изображения в другое положение, используя определенный угол [60, 61].

Масштабирование - этот метод включает в себя уменьшение или увеличение изображения на коэффициент, который умножает длину заданной оси, что означает изменение разрешения изображения.

1.6.1.2 Преобразования яркости и цвета

Преобразования яркости работают с яркостью, либо присваивая ей новое значение, либо регулируя его. Они работают либо только с заданным значением пикселя, основанным на общем правиле и характеристике. Преобразования яркости также могут быть выполнены на цветовых каналах. Цветовые преобразования и методы работают с несколькими каналами яркости, предоставляя им больше возможностей для настройки изображения.

Операции с яркостными характеристиками. Большое количество методов преобразования яркости и цвета может быть проиллюстрировано характеристикой преобразования яркости между входным и выходным изображениями.

Пороговые значения. Пороговое значение преобразует яркость в ограниченное число уровней, обычно в два. Разрыв между двумя уровнями называется пороговым значением *threshold*. Использование для расширения базы данных изображений, вероятно, будет иметь большее значение при многоуровневых пороговых значениях. Двухуровневый порог может убрать слишком много информации из картины для большинства примеров.

Уменьшение цвета. Уменьшение цвета работает по тому же принципу, что и пороговое значение, заданное число делится на количество цветов до результирующего, для цветного изображения минимальное количество цветов равно 6.

Настройка контрастности и яркости. Регулировка яркости добавляет или вычитает заданное число из всех значений яркости. При изменении контрастности все значения яркости умножаются на заданное изменение контрастности.

Инверсия. Инверсия изменяет все значения яркости, если значения находятся в диапазоне от 0 до 255, а значение равно нулю, поэтому после инверсии оно равно 255.

Методы работы с цветами. Другим средством редактирования изображений для увеличения их числа является редактирование цвета. Эти методы делают модель независимой от цвета. При его использовании необходимо подумать, можно ли использовать методы, изменяющие цвета, и насколько необходимо, чтобы база данных зависела от цветовой палитры. [62] Методы могут быть использованы в разных цветовых пространствах с разными результатами.

Перетасовка каналов. Перетасовка цветовых каналов изменяет их цветовое отображение, например, с красного на зеленый, и наоборот, порядок может быть, как определенным, так и случайным.

Преобразование в оттенки серого. Изображение может быть преобразовано в изображение в оттенках серого либо путем усреднения всех значений, либо путем выбора соотношения, которое определяет соотношение в соответствии с чувствительностью человеческого глаза.

Добавление значения к цветовому каналу. В этом методе константа добавляется к цветовому каналу. Различные константы могут быть добавлены или вычтены для всех каналов, так что значение остается на максимальном или минимальном значении при превышении значения. Его можно использовать в цветовом пространстве HSV и RGB.

1.6.1.3 Добавление шума

Добавление шума может изменить только некоторые пиксели или все изображение в целом, в зависимости от выбранного типа шума.

Импульсный шум изменяет случайные пиксели либо на черные, либо на белые. Количество изменённых пикселей можно определить по проценту от общего количества пикселей.

Гауссов шум добавляется к изображению аддитивно. Таким образом, это влияет на все пиксели. Величина шума регулируется параметрами нормального распределения, т.е. средним значением и дисперсией.

1.6.1.4 Локальные фильтры

Использование локальных фильтров для сглаживания или размытия изображения улучшает распознавание размытого объекта в движении. Напротив, фокусировка изображения может привести к тому, что объект останется более детализированным. Недостатком этих методов является аналогичный принцип работы, как и во внутренних механизмах сверточных

нейронных сетей. Следовательно, аналогичные результаты могут быть достигнуты путем настройки параметров стандартного слоя свертки.

Размытие выполняется средним фильтром с возможностью определения размера данного фильтра.

Фильтр Гаусса размывает с помощью фильтра, созданного на основе кривой Гаусса, можно определить размер и дисперсию.

Используя медиану, фильтр выбирает медианное значение из заданной квадратной области, размер которой произволен.

2. Выводы по главе 1

Анализ технологического процесса идентификации клейма литой заготовки на металлургическом предприятии показал, что данный процесс существенно осложняется за счёт шумовых загрязнений элементов кода, которые могут представлять собой как окалину, остающуюся на заготовках после схода её с МНЛЗ, несовершенство клеймовочных машин, не способных осуществлять унифицированное нанесение кода и ряда других причин.

Также в главе были разобраны основные теоретические методы позволяющие реализовать систему оптического распознавания кодов литой заготовки. Так был разобран классический алгоритм компьютерного зрения, включающий в себя предобработку изображения для выделения необходимых для распознавания свойств, методы локализации искомых объектов на изображении, а также способы из идентификации.

Были проанализированы подходы машинного обучения и разобраны самые популярные архитектуры глубоких нейронных сетей. Также в главе рассматривается проблема обучающего набора данных, а именно были проанализированы существующие наборы, которые можно использовать для решения задачи, и рассмотрены методы по созданию собственного набора данных.

Следовательно, для решения актуальной задачи повышения эффективности производственного процесса распознавания клейм перед их

посадом в печь нагрева предлагается разработать систему компьютерного зрения. Данная система должна в режиме реального времени идентифицировать каждую подходящую к посту контроля заготовку и сравнивать её код, с кодом заготовки, идущей по плану. Данная система обеспечит защиту производства от проблемы смешивания заготовок на складе и не позволит остановить всю прокатную линию из-за ошибки оператора по посту контроля.

Глава 2. Разработка оптического детектора распознавания кода литой заготовки

2.1 Создание собственного уникального набора данных

Как было сказано в предыдущей главе, для обучения нейронных сетей требуется большой набор данных, а так как рассматриваемая задача имеет ряд специфичных признаков существующие наборы данных не способны отразить в полной мере все особенности изображения, следовательно, было принято решение создать собственный набор данных состоящий из изображений цифр клейма литых заготовок СПЦ-1 АО «ОЭМК».

Для сбора изображений клейма использовалась видео камера Optimus IP-P082 1(10X). Технические характеристики данного устройства описаны в таблице 2.

Таблица 2 – Технические данные камеры Optimus IP-P082

Характеристика	Значение
Макс. скорость вызова предустановки	12 гр./сек.
Мощность	12Вт
Автоматический контроль скорости фокусировки	Автоматически настраивается в соответствии с фокусным расстоянием
Вертикальный угол вращения	от -5 до 60 гр.
Напряжение питания	12В пост. тока или PoE IEEE 802.3 af
Допустимая температура воздуха	от -40 С до +60 С
Условия эксплуатации	от -40 до +65 гр.
Допустимая влажность воздуха	<95% (без образования конденсата)
Декодер	встроенный
Допустимое давление	86 – 106кПа
Видео синхронизация	внутренняя
Класс защиты	IP66
Ручная скорость вращения	0,01-12гр./сек.
Стандарт защиты	TVS 3000V
Горизонтальный угол вращения	260 гр. непрерывного вращения

Для качественного обучения нейронной сети глубокого обучения требуется чтобы изображение содержало максимальное количество отличительных признаков распознаваемого объекта, что значительно затрудняет малый размер выбитых клемовочной машиной символов и разного рода дефекты (такие как окалина, огрехи работы клемовочной машины и прочее). Исходя из этого при съёмке клейма использовалось максимально

возможное разрешение, что позволило нам достичь необходимого качества начального материала для создания набора данных.

Установка нашей камеры осуществлялась рядом с камерой, которая передаёт изображение клейма на пост. Начальное изображение с камеры, установленной над рольгангом, имело плохое качество из-за слабой освещённости, поэтому был установлен фонарь, подсвечивающий торец заготовки, мощность светового потока создаваемого фонарём порядка 3000 люменов. Так же в процессе установки стало очевидно, что из-за близости участка контроля к печам нагрева на аппаратуру будет оказываться сильное термическое воздействие, что может повлечь преждевременный выход оборудования из строя. Дабы избежать подобного исхода устанавливаемое оборудование было защищено теплоизоляционным материалом.



Рис. 21 – Участок контроля НЛЗ

Для формирования обширной выборки необходимо цифр клейма, был разработан алгоритм автоматической съёмки, который сохраняет изображение в момент квитиования оператором заготовки. Процесс формирования базы изображений длился несколько дней и в результате было получено порядка 51000 фотографий с разрешением 1080x1960 пикселей. Каждое полученное

изображение сохранялось в формате «png» и занимало около 3Мб памяти. Данный размер изображения чрезмерно большой, это значительно затруднит процесс обучения и повысит требования к машине, на которой это обучение будет реализовываться. Следовательно, было решено изменить формат с «png» в «jpg» для снижения размера файла изображения до 600 Кб.



Рис. 22 – Экран оператора

Ввиду значительного числа бракованных изображений, что вызвано некорректной работой оператора, закрашиванием цифр клейма, сильной окалиной, из полученный выборки было удалено около 10000 изображений. Удаление такого большого объёма данных было необходимо, так как в случае их использования это могло негативно сказаться на результате классификации. Оставшиеся изображения были разделены на выборку для

обучения (36882 изображений) и выборку для тестирования (4236 изображений).

После формирования изображений для обучающей и тестовой выборки была произведена разметка символов, находящихся на изображении. В качестве инструмента для разметки было использовано приложение LabelImg [63]. Данный инструмент, некоммерческого использования, позволяет выделять объекты на изображении прямоугольной областью и присваивать этой области любой необходимый нам класс, к тому же он обладает открытым исходным кодом. Процесс разметки представлен на рисунке 23.



Рис. 23 – Интерфейс программы «LabelImg»

В результате разметки мы получили на каждом изображении до 9 аннотированных объектов с присвоением одного из десяти классов (от 0 до 9). Данные о разметке хранятся в отдельном файле с расширением xml. Пример результата разметки представлен на рисунке 24.

Так же был произведён дополнительный анализ результатов разметки с целью повышения качества соответствия для алгоритма обучения распознавая образов.

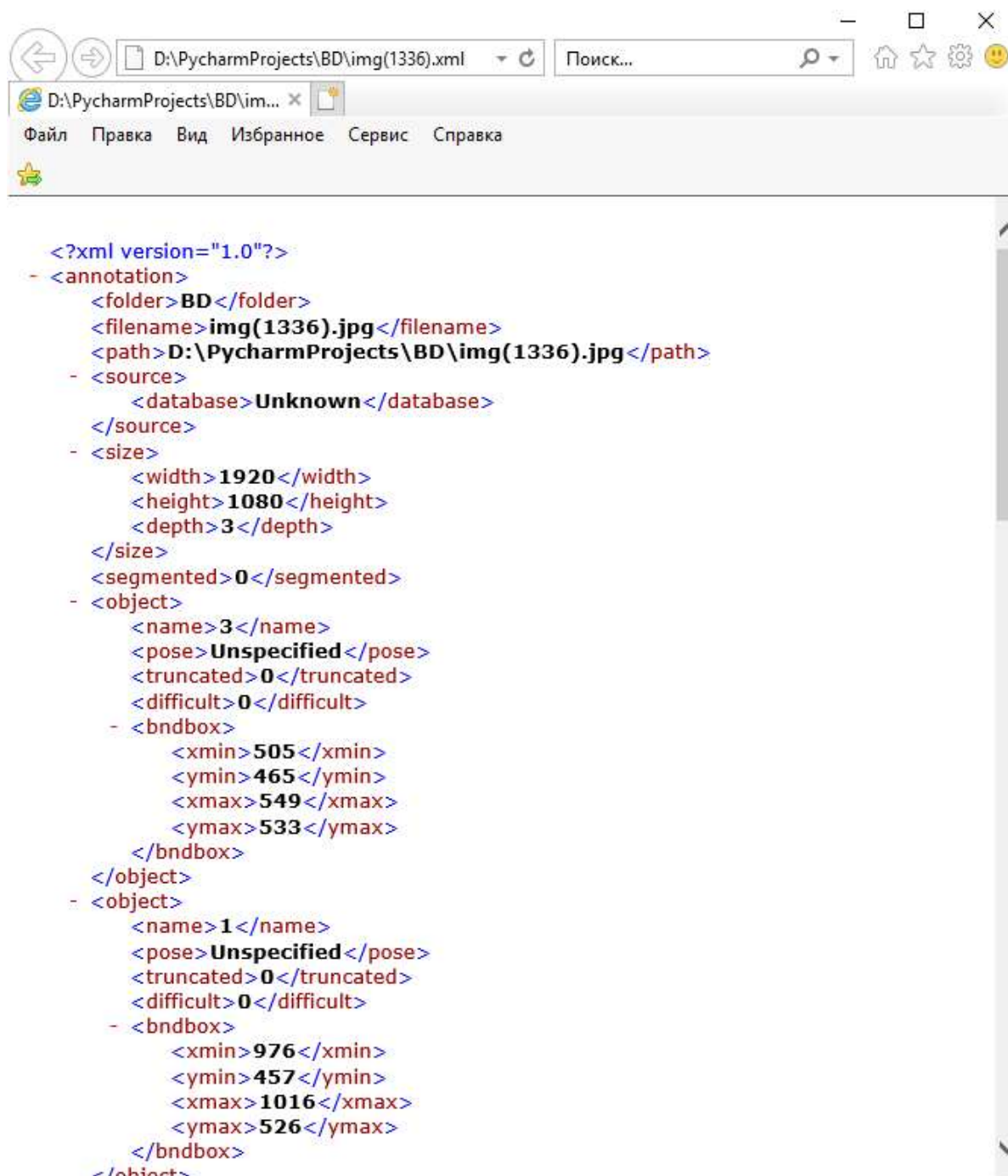


Рис. 24 – Файл разметки

По результатам обработки предоставленных с производства изображений клейма было сохранено 41118 изображений, которые содержат в себе порядка 370000 цифр, из которых 4236 изображений были использованы для оценки, это порядка 38000 символов.

В таблице 3 отражено количество элементов для каждого класса. Как видно из таблицы, некоторые цифры встречаются чаще, а некоторые реже. Но данный факт не сильно влияет на качество обучения за счёт огромного объёма элементов каждого класса.

Таблица 3 – Количество образцов каждого класса в существующем наборе данных.

Класс	Количество примеров
0	54072
1	58918
2	53503
3	55425
4	52634
5	29209
6	24359
7	14412
8	12973
9	14128

2.2 Разработка алгоритма компьютерного зрения

Сформировав обучающую выборку, приступили к реализации алгоритма компьютерного зрения. Как описывалось в первой главе, данный алгоритм включает в себя три основных этапа:

- Сегментация;
- Локализация;
- Классификация;

Рассмотрим первый этап.

2.2.1 Сегментация методом пороговой бинаризации

Бинаризация — это процедура обработки изображения, которая преобразует изображение в градациях серого в бинарное изображение, обычно она используется для выделения объектов на фоне. В частности, бинаризация широко применяется в качестве этапа предварительной обработки перед оптическим распознаванием символов (OCR), когда символьные компоненты сначала извлекаются из изображений. Качество бинаризации имеет решающее значение для точности распознавания, особенно для изображений с ухудшенными характеристиками [64].

Один из самых простых методов бинаризации называется пороговой бинаризацией, который представляет собой процесс, который изучает статистику значений пикселей в изображении и выбирает конкретное значение, которое разделяет или классифицирует передний план и фон. Например, в изображениях в градациях серого более темные значения можно классифицировать как символы, а более светлые - как фоновые. Пороговая бинаризация является одной из многих попыток отделить текст от фона в изображениях клейма от стальной заготовки в оттенках серого. Тем не менее, этот подход показывает себя не лучшим образом, когда обводки по символу слабы и очень похожи на фон.

Низкое качество изображения, которое препятствует правильной бинаризации изображения, связано с тем, что мы называем шумом. В контексте этой работы мы использовали термин шум, чтобы широко описать любую аномалию в изображении, который делает символы менее читаемыми, шумом являются артефакты с камеры, куски окарины, ухудшение качества клейма из-за освещенности и т. д. Шум на изображениях такого типа более проблематичен в тех частях изображения, где фон темнее. Этот шум часто искажает символы, из-за него человеку или компьютеру трудно их прочитать [65].

Существует много алгоритмов определения пороговых значений, которые пытаются отделить передний план (символы) от фона в изображениях, но большинство из них хорошо работают только для изображений, где текст и фон четко определены.

Алгоритмы определения порогов для изображений подразделяются на два класса: глобальные и локальные. Алгоритмы глобального определения порога вычисляют одно общее пороговое значение или значение уровня пикселя для всего изображения, а затем преобразуют изображение в двухуровневое изображение на основе этого порога.

Алгоритмы глобальной пороговой бинаризации обычно быстрее вычисляются, потому что они используют один порог, основанный на

глобальной гистограмме пикселей серого цвета изображения. Однако выбор правильного порогового значения для всего изображения обычно является сложной задачей. К глобальным методам пороговой обработки относятся методы Оцу, Киттлера и Иллингворта, Капура и др., Саху и Арора и Йена и др.

Алгоритмы локальной пороговой бинаризации используют тот факт, что локальная интенсивности меняются на всём изображении. Они используют небольшое окно и запускают его по всему изображению. Для каждого пикселя в изображении вычисляется новое значение в соответствии с соседними пикселями в локальном окне. Таким образом, они используют разные пороговые значения для разных частей изображения. Хотя этот подход может работать лучше, чем глобальное определение порога, он также может быть очень чувствительным к шуму и, следовательно, может не подходить для изображений с шумным фоном. К локальным методам пороговой обработки относятся методы Niblack, Sauvola и Pietikinen, White and Rohrer, Yasuda et al.

Задача нахождения порога для изображения до сих пор является важной проблемой при предобработке изображения, грамотное нахождение порога позволяет не только снизить влияние лишних факторов, но и закладывает серьёзную основу для понимания изображения и успешного распознавания целей.

Порог применяется во многих алгоритмах сегментации изображения. Он позволяет выделить на изображении интересующие нас символы относительно фона. Подобрав определённое значение порога T , исходное черно-белое изображение преобразовывается в двоичное (бинарное).

Двоичный вид изображения должен включать в себя всю информацию, которая может быть полезна при распознавании, например, как форма объекта, его размер, местоположение на плоскости изображения и др. Достоинство бинарного изображения состоит в упрощении данных, что в последствии позитивно сказывается на их распознавании.

Самым простым и в тоже время самым распространённым методом преобразования в двоичный вид изображения в градациях серого служит

выбор фиксированного порогового значения T . В этом методе все пиксели чьё значение интенсивности ниже этого порога будут рассматриваться как черные, а пиксели чьё значение интенсивности выше – белыми. Основной проблемой сегментации является выбор такого значения порога, который максимально упростил изображение, но при этом не удалил важные признаки объекта. Для правильного подбора порогового значения часто используют алгоритмы, основанные на анализе гистограмм типа изображения.

Предпочтительный вариант, когда гистограмма представляет два доминирующих режима и четко выраженный, глубокий, не зашумленный минимум между двумя пиками гистограммы. В данной ситуации выбор значения T будет равен точке между двумя режимами. Но в реальных ситуациях такие ситуации встречаются редко и не всегда подбор значения T является лёгкой задачей.

Метод пороговой бинаризации можно выразить как:

$$T = T[x, y, f(x, y)] \quad (13)$$

где: T – значение порога;

x, y - координаты точки порогового значения;

$f(x, y)$ – координаты пикселя;

Бинарное изображение можно определить по следующей формуле:

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases} \quad (14)$$

Результатом применения алгоритма порога исходное изображение сегментируется на два значения, 0 и 1.

В том случае, когда изображение включает в себя контрастные по отношению к фону объекты можно применять пороговое значение. Так как на большинстве изображений значения интенсивности пикселей объекты значительно отличаются от значений интенсивности пикселей фона, методы, основанные на бинарном пороге, хорошо подходят для сегментации на фон и объект.

За годы разработок в области сегментации изображений было выработано множество алгоритмов для подбора оптимального порогового значения. Наиболее популярным способом является метод назначения порога в реальном времени.

Пороговая схема – это автоматический подбор значения порога для любого изображения без какого-либо участия человека. Пороговая схема требует серьёзных знаний о характерных особенностях изображения, о количестве объектов, их размера и прочее.

Автоматическое пороговое значение хороший метод для извлечения признаков изображения, которые содержатся в пикселях, при уменьшении величины фонового шума. Данный эффект получается за счёт применения обратной связи для оптимизации порога перед преобразованием изначального изображения из черно-белого в двоичный.

Данный метод можно описать в следующих пунктах:

1. Определение начального значения порога;
2. Сегментация с помощью порога;
 - 2.1. Значения интенсивности пикселей, которые ниже порога – фон;
 - 2.2. Значения интенсивности пикселей, которые выше порога – объект;
3. Вычисление среднего значения на основе двух картинок.;
4. Расчёт нового порога путём усреднения двух предыдущих;
5. Если разница между предыдущим пороговым значением и новым пороговым значением ниже указанного предела, конец. В противном случае применить новый порог к исходному изображению.

Предел, упомянутый выше, определяется пользователем. Более высокий предел позволит увеличить разницу между последовательными пороговыми значениями. Преимущества этого могут быть более быстрыми, но с менее четкой границей между фоном и передним планом. Выбор начальных порогов часто выполняется путем вычисления среднего значения изображения в

градациях серого. Однако также можно выбрать начальные пороговые значения на основе двух хорошо разделенных пиков гистограммы изображения и найти среднее значение пикселей этих точек. Это может позволить алгоритму сходиться быстрее, позволяя выбрать гораздо меньший предел.

Автоматическое пороговое значение будет работать лучше всего, если существует хороший фон для контрастности переднего плана. Значение изображения должно быть выполнено в хороших условиях освещения с минимальным блеском.

Реализация метода бинарной(пороговой) сегментации производилась на языке программирования «Python».

На вход алгоритма было подано изображение, представленное на рисунке 25.

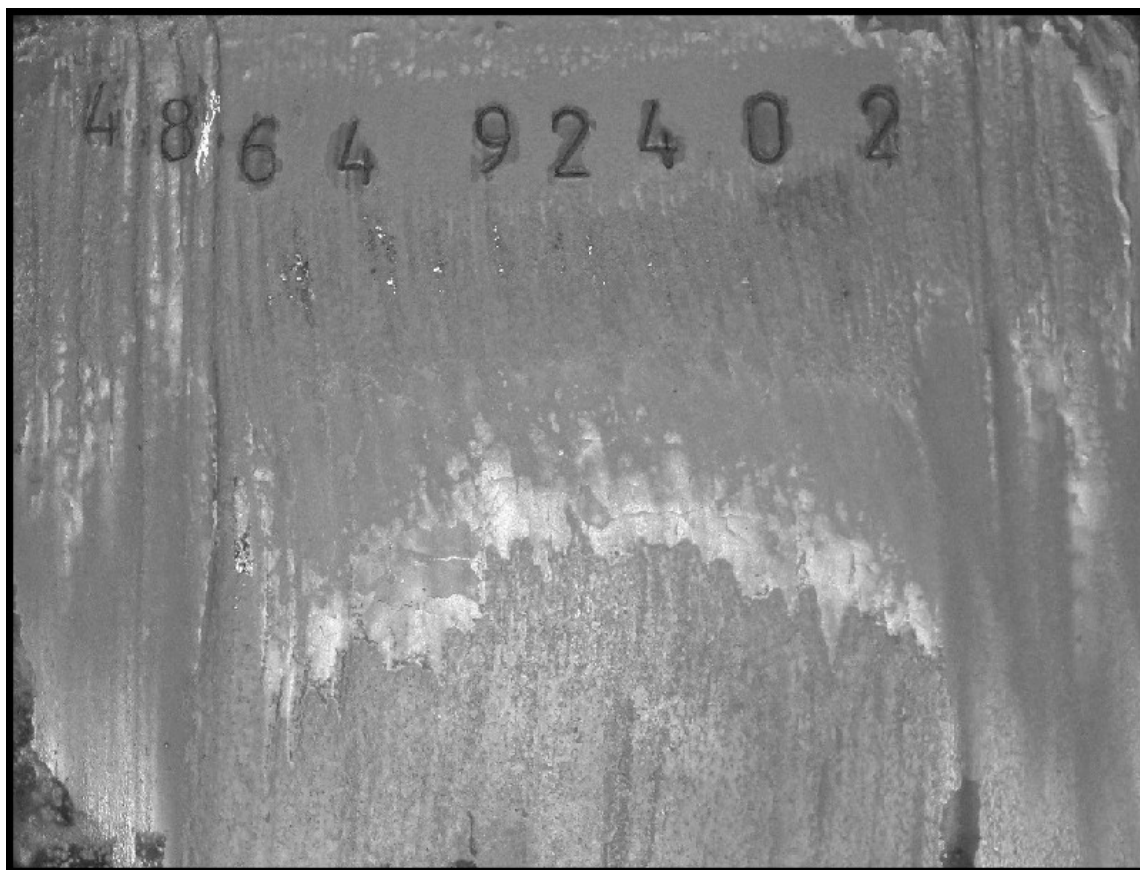


Рис. 25 – Исходное изображение клейма

Результатом работы метода бинаризации с порогом 80 получилось изображение, представленное на рисунке 26.

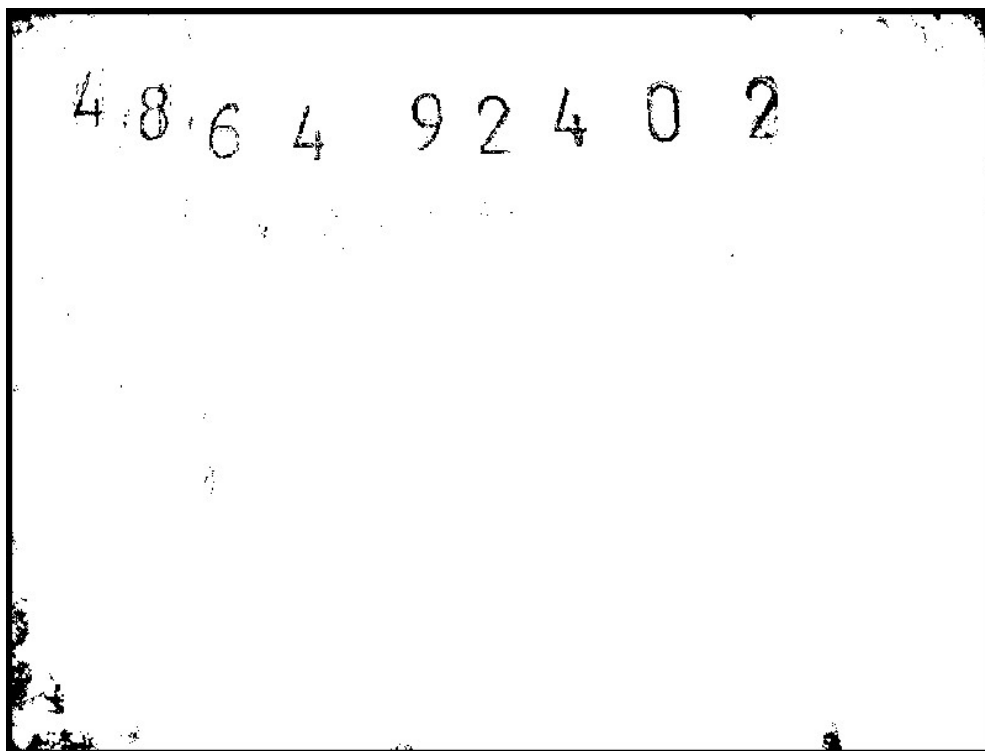


Рис. 26 – Результат пороговой бинаризации

Алгоритм достаточно хорошо выделил требуемые объекты. Так же алгоритм имеет высокую скорость обработки изображения относительно двух рассмотренных выше методов.

Также в ходе работы рассматривалась и другие методы сегментации, но все они значительно хуже справлялись с задачей выделения цифр на изображении, при этом тратя на обработку изображения на много больше времен. Результаты сегментации методом водораздела и методом выращивания регионов представлены на рисунке 27.

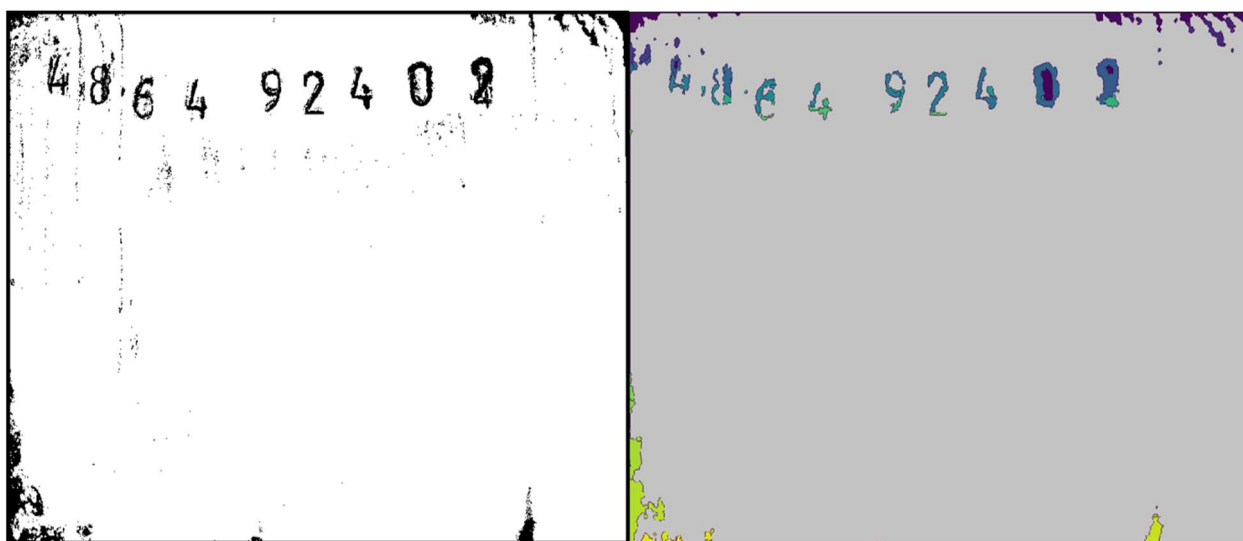


Рис. 27 – Результаты методов выращивания регионов и водораздела

2.2.2 Локализация

2.2.2.1 Локализация объектов с помощью скользящих окон

Обнаружение объекта с помощью скользящего окна является популярным методом идентификации и локализации объектов в изображении. Вырезка изображений может быть выполнена путем перемещения окна по изображению и прогнозирования вероятности нахождения искомого объекта для каждого окна.

Метод перемещения окна с фиксированным размером (меньшим чем изображение), и вырезка изображения в соответствии с размером окна называется скользящим окном (15). Прогноз может быть сделан для каждого вырезанного окна изображения, которое называется обнаружением объекта скользящего окна [67].

$$\begin{cases} P_{i,j} = I[j \cdot s_y : j \cdot s_y + h, i \cdot s_x : i \cdot s_x + w] \\ i = 0, 1, \dots, \frac{W-w}{s_x} \\ j = 0, 1, \dots, \frac{H-h}{s_y} \end{cases} \quad (15)$$

P – множество всех фрагментов,

W, w – ширина изображения и окна,

H, h – высота изображения и окна,

s_x, s_y – шаг скользящего окна по горизонтали и вертикали.

Решение о наличии искомой цифры может быть выполнено с помощью классификатора. Движение окна должно быть равномерным по всей плоскости изображения.

Каждая вырезанная часть изображения торца заготовки должна пройти через классификатор, чтобы определить, находится ли искомая цифра в данном сегменте. У такого подхода есть две проблемы.

1) Алгоритм может находить цифры, которые имеют тот же размер, что и окно. Скользящее окно пропускает цифру, если размер объекта больше размера окна.

2) Другая проблема заключается в том, что перемещение окна с большим шагом может привести к отсутствию нескольких цифр. Перемещение окна по каждому пикселю приведет к большому количеству дополнительных вычислений, следовательно, это замедлит работу системы.

2.2.2.2 Пирамида изображений

Мы не знаем на каком расстоянии от камеры остановится заготовка с клеймом, что влечёт неопределённость относительно размеров символов. Для решения этой проблемы был использован метод пирамиды изображений.

Пирамида изображения представляет собой многомасштабное представление изображения (16).

$$I_s(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I_0(i, j) \cdot g(i - s \cdot x, v - s \cdot y) \quad (16)$$

где: I_s – уровень пирамиды, I_0 – исходное изображение, s – коэффициент масштабирования, g – интерполяционная функция, x, y – координаты уменьшенного изображения, u, v – координаты исходного изображения.

Использование пирамиды изображения позволяет находить цифры в изображениях в разных масштабах изображения. И в сочетании со скользящим окном мы можем находить цифры в изображениях в разных местах.

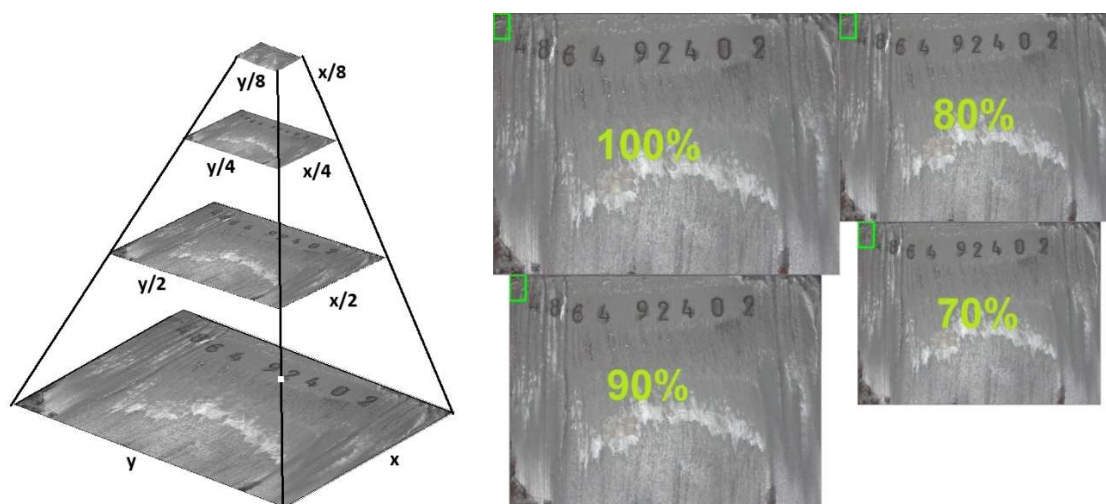


Рис. 28 – Пример пирамиды изображения. На каждом слое пирамиды изображение уменьшается

В нижней части пирамиды (рисунок 28) мы имеем исходное изображение торца заготовки с первоначальным размером (по ширине и

высоте). А на каждом последующем слое изображение изменяется и, возможно, сглаживается, обычно с помощью гауссовского размытия.

Изображение постепенно уменьшается до тех пор, пока не будет выполнен некоторый критерий остановки, который обычно является минимальным, и дальнейшее масштабирование не должно выполняться.

2.2.2.3 Классификатор наличия цифр клейма

Задача классификатора заключается в принятии решения, является ли фрагмент изображения, выделенный скользящим окном, изображением нужной нам цифры.

Для машинного анализа требуется вычислить некоторые числовые характеристики (признаки) областей:

- геометрические признаки
- фотометрические признаки

На основе этих характеристик можно классифицировать получаемые области.

В качестве классификации сегмента был разработан алгоритм на основе фотометрических признаков.

Для каждой области можно подсчитать некий набор простейших числовых характеристик:

- Средняя интенсивность изображения
- Средний цвет (если изображение цветное)
- Распределения концентрации пикселей

Экспериментальным путём были определены критерии, которые способны с высокой долей вероятности классифицировать объект:

- Концентрация пикселей в окне
- Концентрация пикселей в верхней части окна
- Концентрация пикселей в нижней части окна
- Концентрация пикселей в левой части окна

- Концентрация пикселей в правой части окна
- Концентрация пикселей в рамке окна

Для вычисления допусков выше представленных критериев проводился анализ вырезанных вручную символов. Результаты допусков представлены в таблице 4.

Таблица 4 – Допуски по наличию объекта в окне.

Критерий	Расчёт	Допуск
Концентрация пикселей	$Z = \frac{\sum_{i=1}^{80} \sum_{j=1}^{50} (I_{i,j})}{4000}$	0.04 – 0.4
Концентрация пикселей в верхней части изображения	$Z = \frac{\sum_{i=40}^{80} \sum_{j=1}^{50} (I_{i,j})}{2000}$	0.02 – 0.4
Концентрация пикселей в нижней части изображения	$Z = \frac{\sum_{i=1}^{40} \sum_{j=1}^{50} (I_{i,j})}{2000}$	0.02 – 0.4
Концентрация пикселей в левой части изображения	$Z = \frac{\sum_{i=1}^{80} \sum_{j=1}^{25} (I_{i,j})}{2000}$	0.02 – 0.4
Концентрация пикселей в правой части изображения	$Z = \frac{\sum_{i=1}^{80} \sum_{j=25}^{50} (I_{i,j})}{2000}$	0.02 – 0.4
Концентрация пикселей в рамке изображения	$Z = \frac{\sum_{i=1}^{80} \sum_{j=1}^{50} (I_{i,j}) - \sum_{i=5}^{75} \sum_{j=5}^{45} (I_{i,j})}{1200}$	0 – 0.02

2.2.2.4 Алгоритм локализации объекта

Были рассмотрены и проанализированы все этапы алгоритма локализации объекта на изображении. Итоговый алгоритм локализации выглядит следующим образом:

Сегментированное изображение торца заготовки подаётся на алгоритм скользящего окна, где оно разбивается на окна и происходит анализ каждого выделенного сегмента на наличие цифр клейма с помощью классификатора

наличия цифры. Если алгоритм не смог выделить все символы, то применяется метод пирамиды изображений, происходит уменьшение сегментированного изображения. [67]

Это выполняется для того, чтобы алгоритм мог находить объекты на изображении вне зависимости от их размера (размер может меняться из-за расстояния между камерой и заготовкой). Метод пирамиды изображений применяется до того момента пока не будут найдены все символы или не будет выполнен критерий остановки. В случае если алгоритм смог локализовать 9 цифр, то на выход алгоритма поступают координаты каждого выделенного символа. В противном случае выводится сообщение об ошибке.

2.2.3 Классификация

Задача классификации изображений – это задача суть, которой в присвоении входному изображению одной метки из фиксированного набора категорий. Это одна из основных проблем в компьютерном зрении, которая, несмотря на свою простоту, имеет большое разнообразие практических ограничений.

Поскольку задача распознавания визуального образа является относительно тривиальной проблемой для человека, стоит рассмотреть проблемы, связанные с перспективой алгоритма компьютерного зрения. При классификации изображения следует учитывать следующие проблемы влияющие на работу алгоритма:

- Изменение точки обзора. Один экземпляр объекта может быть по-разному ориентирован относительно камеры.
- Изменение шкалы. Визуальные классы часто демонстрируют вариации в своем размере (размер в реальном мире не только по степени их изображения).
- Деформация. Многие объекты, представляющие интерес, не являются твердыми телами и могут деформироваться экстремально.

- Окклюзия. Объекты, представляющие интерес, могут быть закрыты. Иногда может быть видна лишь небольшая часть объекта (всего несколько пикселей). В решаемой задаче окклюзия связана с закрытием символов клейма окалиной от окисления железной заготовки.

- Условия освещения. Эффект освещения сильно зависит от уровня пикселей.

- Фоновый беспорядок. Объекты, представляющие интерес, могут смешиваться с окружающей средой, что затрудняет их определение.

- Внутриклассовая вариация. Классы интересов часто могут быть относительно широкими. Существует много разных типов этих объектов, каждый со своим внешним видом.

Хорошая модель классификации изображений должна быть инвариантной к перекрестному произведению всех этих вариаций, одновременно сохраняя чувствительность к межклассовым вариациям.

В качестве классификатора были протестированы два типа нейронных сетей: обычная нейронная сеть прямого распространения и свёрточная нейронная сеть глубокого обучения.

Была использована полносвязная нейронная сеть, где каждый нейрон слоя был связан со всеми нейронами соседних слоёв. На выходе нейронная сеть имеет 10 нейронов и соответственно на выходе имеет 10 сигналов каждый из которых равен вероятности того какая цифра представлена на изображении.

Входным сигналом выступает значение интенсивности каждого пикселя изображения, всего таких сигналов 3500, по их количеству в изображении.

При выборе архитектуры нейронной сети были проанализированы и исследованы две нейронные сети с одним скрытым слоем и одна нейронная сеть с двумя скрытыми слоями.

Для процедуры обучения нейронных сетей прямого распространения был применён метод стохастического градиентного спуска с размером мини выборки 200 элементов.

На каждой эпохе обучения из обучающей выборки случайным образом выбирается один объект, исходя из этого вектор настраивается на каждый вновь выбираемый объект.

Обучение длилось в течении 50 эпох. Это обусловлено тем, что дальнейшее обучение повышает точность на незначительное значение, которым можно пренебречь. [68]

Архитектура нейронной сети прямого распространения с одним скрытым слоем, в котором находится 2800 нейронов, изображена на рисунке 29.

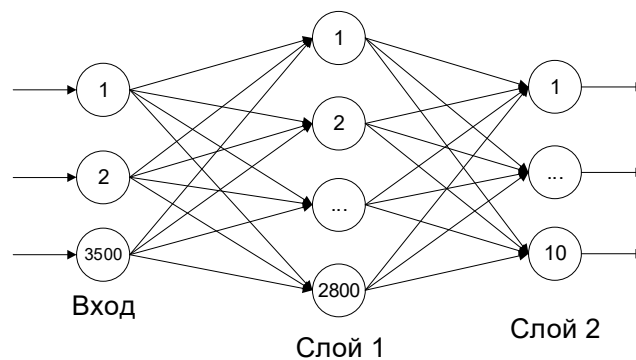


Рис. 29 – Архитектура нейронной сети прямого распространения №1

Математическая модель архитектуры нейронной сети прямого распространения №1 представлена следующими выражениями (17):

$$\begin{cases} X_j^{(1)} = f_{ReLU}((\sum_{i=1}^{3500} (X_i^0 \cdot W_{j,i}^{(1)}) + b_j^{(1)}), & j = \overline{1:2800} \\ X_k^{(2)} = f_{SoftMax}((\sum_{j=1}^{2800} (X_j^{(1)} \cdot W_{k,j}^{(2)}) + b_k^{(2)}), & k = \overline{1:10} \end{cases} \quad (17)$$

График процесса обучения представлен на рисунке 30.

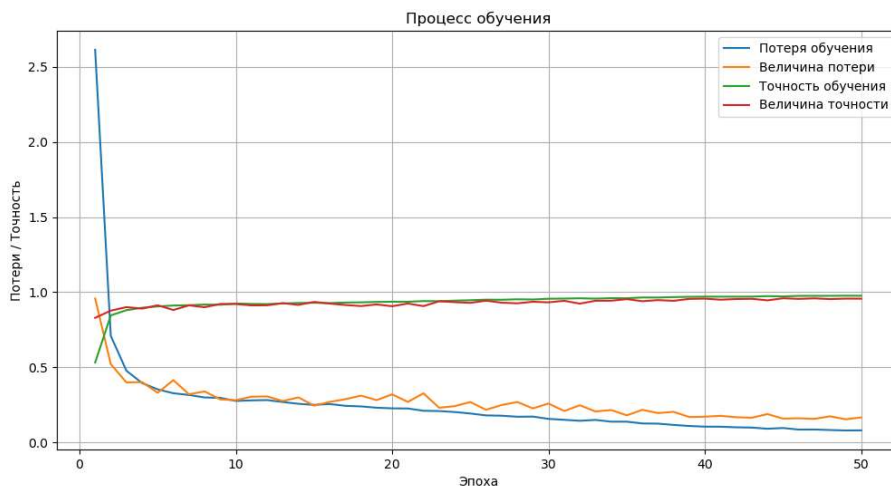


Рис. 30 – График процесса обучения нейронной сети прямого распространения №1

Время обучения: 150 сек.

Точность обучения: 95,7%

Результат тестирования: 96,02%

Скорость распознавания: 0,11 сек

Архитектура нейронной сети прямого распространения с одним скрытым слоем, в котором находится 6300 нейронов, изображена на рисунке 31.

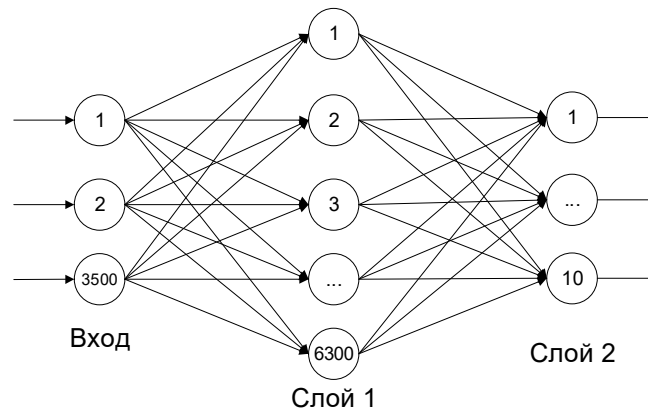


Рис. 31 – Архитектура нейронной сети прямого распространения №2

Математическая модель архитектуры нейронной сети прямого распространения №2 представлена следующими выражениями (18):

$$\begin{cases} X_j^{(1)} = f_{ReLU}((\sum_{i=1}^{3500} (X_i^0 \cdot W_{j,i}^{(1)}) + b_j^{(1)}), & j = \overline{1:6300} \\ X_k^{(2)} = f_{SoftMax}((\sum_{j=1}^{6300} (X_j^{(1)} \cdot W_{k,j}^{(2)}) + b_k^{(2)}), & k = \overline{1:10} \end{cases} \quad (18)$$

График процесса обучения представлен на рисунке 32.

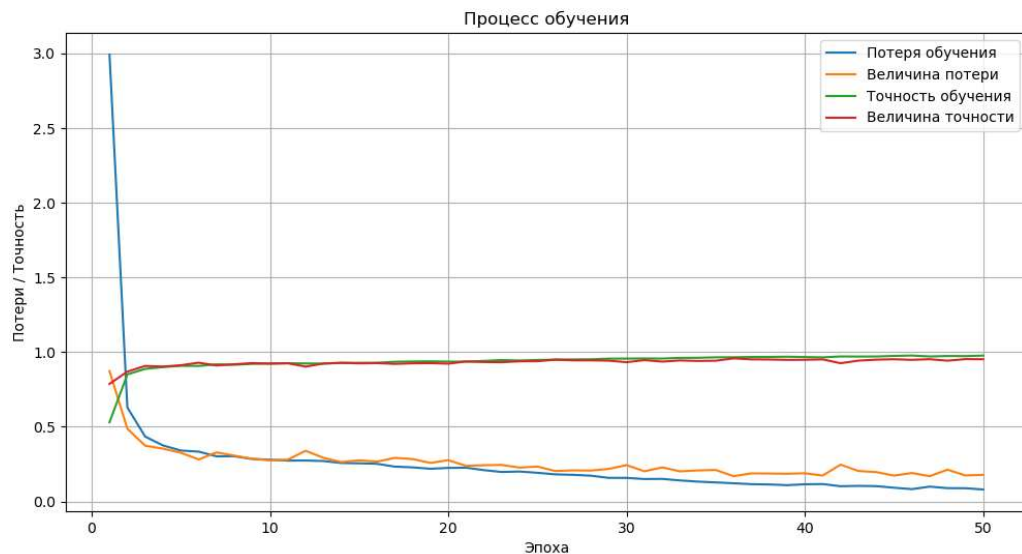


Рис. 32 – График процесса обучения нейронной сети прямого распространения №2

Время обучения: 250 сек.

Точность обучения: 95,22%

Результат тестирования: 96,75%

Скорость распознавания: 0,11 сек

Архитектура нейронной сети прямого распространения с двумя скрытыми слоями, в которых находится 2800 и 2100 нейронов соответственно, изображена на рисунке 33.

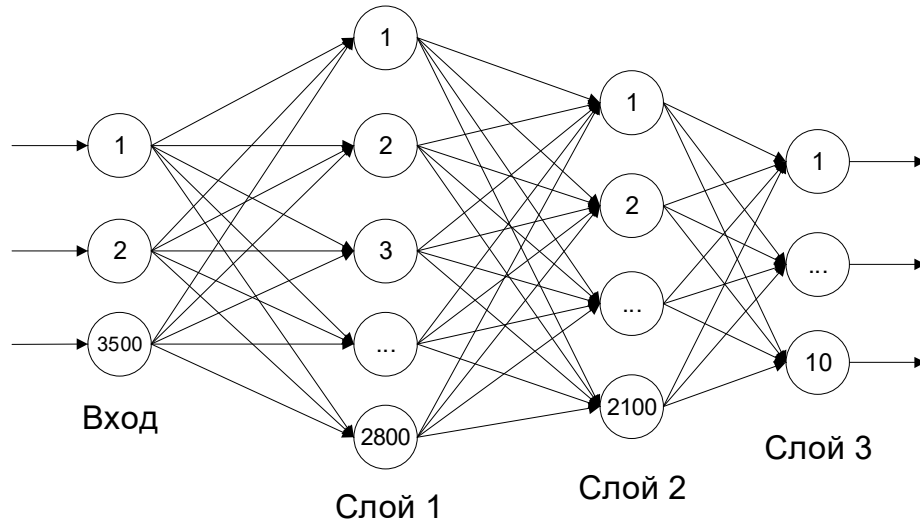


Рис. 33 – Архитектура нейронной сети прямого распространения №3

Математическая модель архитектуры нейронной сети прямого распространения №3 представлена следующими выражениями (19):

$$\begin{cases} X_j^{(1)} = f_{ReLU}((\sum_{i=1}^{3500} (X_i^0 \cdot W_{j,i}^{(1)}) + b_j^{(1)}), & j = \overline{1:2800} \\ X_k^{(2)} = f_{ReLU}((\sum_{j=1}^{2800} (X_j^{(1)} \cdot W_{k,j}^{(2)}) + b_k^{(2)}), & k = \overline{1:2100} \\ X_e^{(3)} = f_{SoftMax}((\sum_{l=1}^{2100} (X_l^{(2)} \cdot W_{e,l}^{(3)}) + b_e^{(3)}), & e = \overline{1:10} \end{cases} \quad (19)$$

График процесса обучения представлен на рисунке 34.

Время обучения: 200 сек.

Точность обучения: 95,67%

Результат тестирования: 96,89%

Скорость распознавания: 0,11 сек.

Архитектура свёрточной нейронной сети с одним каскадом, состоящим из двух свёрточных слоёв и слоя подвыборки, за которым следуют два

скрытых полносвязных слоя с 1000 и 600 нейронами соответственно, представлена на рисунке 35.

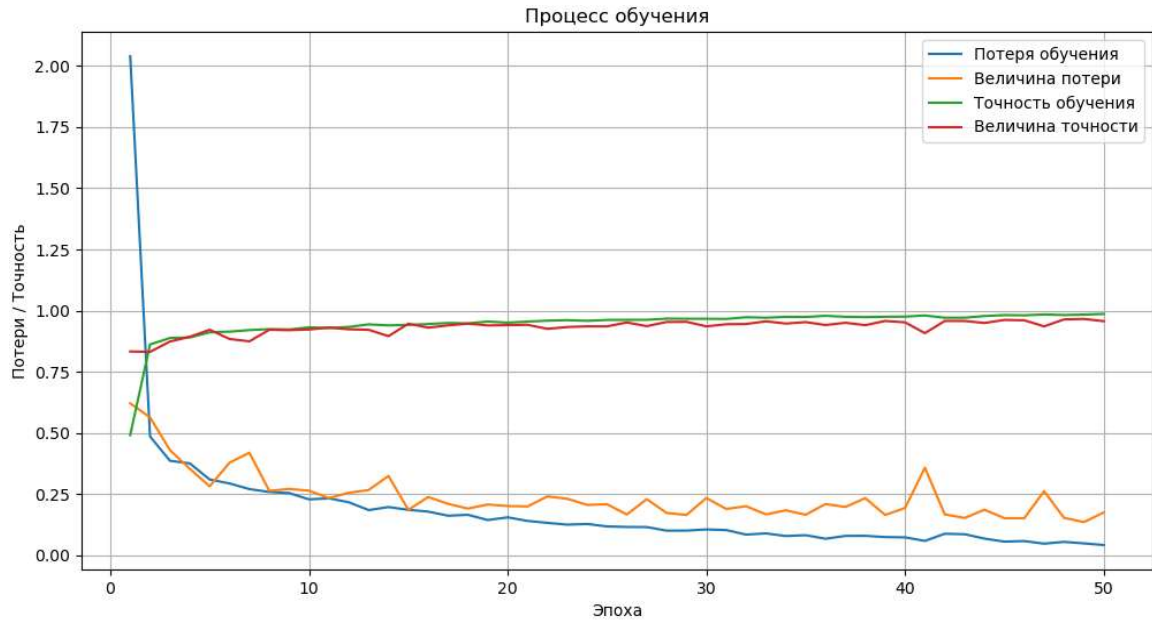


Рис. 34 – График процесса обучения нейронной сети прямого распространения №3

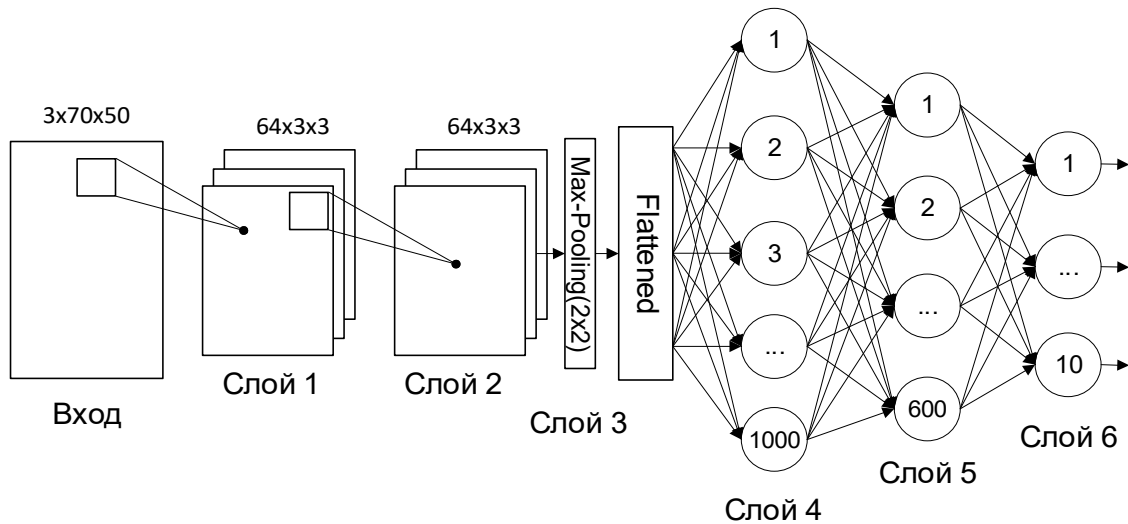


Рис. 35 – Архитектура свёрточной нейронной сети №1

Математическая модель архитектуры свёрточной нейронной сети №1 представлена следующими выражениями (20):

$$\begin{cases} X^{(1)} = f_{ReLU}(X^{(0)} \otimes k^{(1)} + b^{(1)}) \\ X^{(2)} = f_{ReLU}(X^{(1)} \otimes k^{(2)} + b^{(2)}) \\ X^{(3)} = a^{(3)} \cdot \text{subsample}(X^{(2)}) + b^{(3)} \\ X^{(4)} = f_{ReLU}(X^{(3)} \cdot W^{(4)} + b^{(4)}) \\ X^{(5)} = f_{ReLU}(X^{(4)} \cdot W^{(5)} + b^{(5)}) \\ X^{(6)} = f_{SoftMax}(X^{(5)} \cdot W^{(6)} + b^{(6)}) \end{cases} \quad (20)$$

График процесса обучения представлен на рисунке 36.

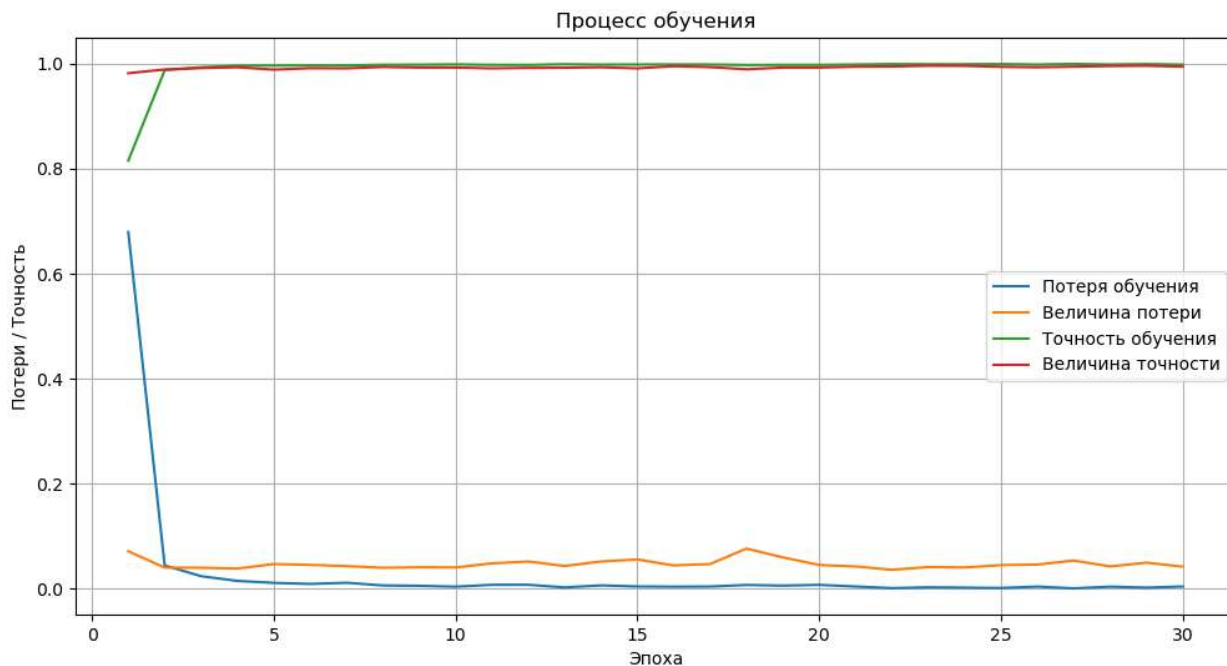


Рис. 36 – График процесса обучения свёрточной нейронной сети №1

Время обучения: 1320 сек.

Точность обучения: 99,46 %

Результат тестирования: 99,73%

Скорость распознавания: 0,14 сек

Архитектура свёрточной нейронной сети с двумя каскадами, состоящими из двух свёрточных слоёв и слоя подвыборки, за которыми следуют два скрытых полносвязных слоя с 2000 и 1500 нейронами соответственно, представлена на рисунке 37:

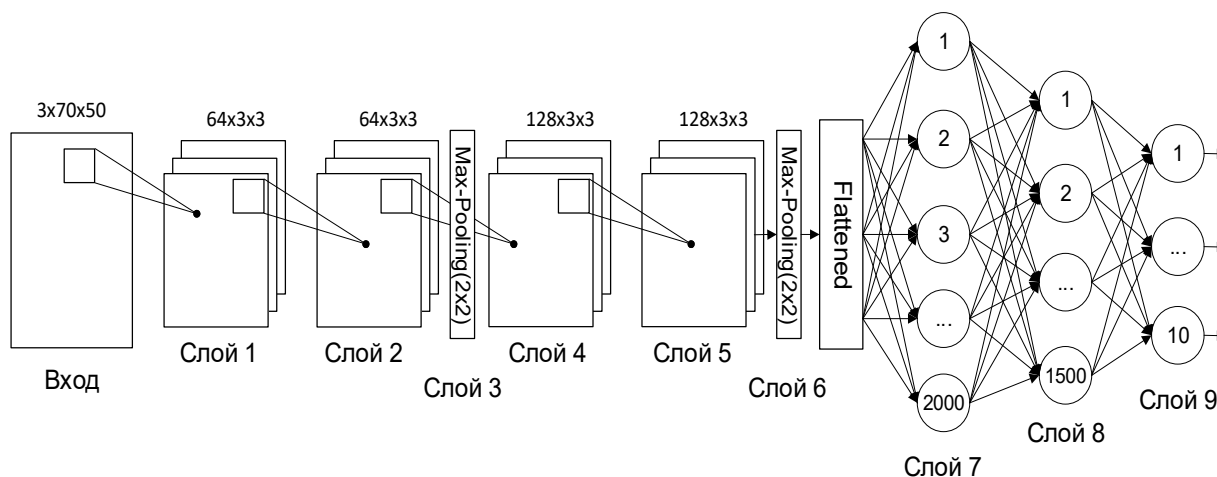


Рис. 37 – Архитектура свёрточной нейронной сети №2

Математическая модель архитектуры свёрточной нейронной сети №2 представлена следующими выражениями (21):

$$\left\{ \begin{array}{l} X^{(1)} = f_{ReLU}(X^{(0)} \otimes k^{(1)} + b^{(1)}) \\ X^{(2)} = f_{ReLU}(X^{(1)} \otimes k^{(2)} + b^{(2)}) \\ X^{(3)} = a^{(3)} \cdot \text{subsample}(X^{(2)}) + b^{(3)} \\ X^{(4)} = f_{ReLU}(X^{(3)} \otimes k^{(4)} + b^{(4)}) \\ X^{(5)} = f_{ReLU}(X^{(4)} \otimes k^{(5)} + b^{(5)}) \\ X^{(6)} = a^{(2)} \cdot \text{subsample}(X^{(5)}) + b^{(6)} \\ X^{(7)} = f_{ReLU}(X^{(6)} \cdot W^{(7)} + b^{(7)}) \\ X^{(8)} = f_{ReLU}(X^{(7)} \cdot W^{(8)} + b^{(8)}) \\ X^{(9)} = f_{SoftMax}(X^{(8)} \cdot W^{(9)} + b^{(9)}) \end{array} \right. \quad (21)$$

График процесса обучения представлен на рисунке 38.

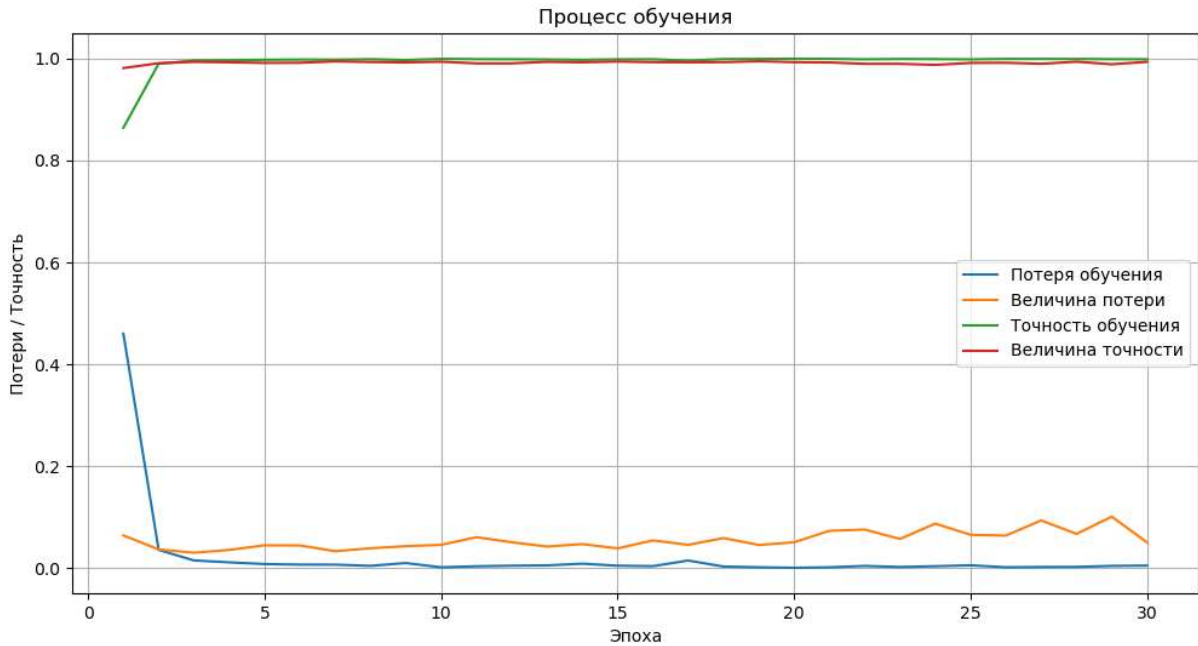


Рис. 38 – График процесса обучения свёрточной нейронной сети №2

Время обучения: 1950 сек.

Точность обучения: 99,37%

Результат тестирования: 96,71%

Скорость распознавания: 0,15 сек

Архитектура свёрточной нейронной сети с тремя каскадами, состоящими из двух свёрточных слоёв и слоя подвыборки, за которыми следуют два скрытых полносвязных слоя с 3000 и 2000 нейронами соответственно, представлена на рисунке 39:

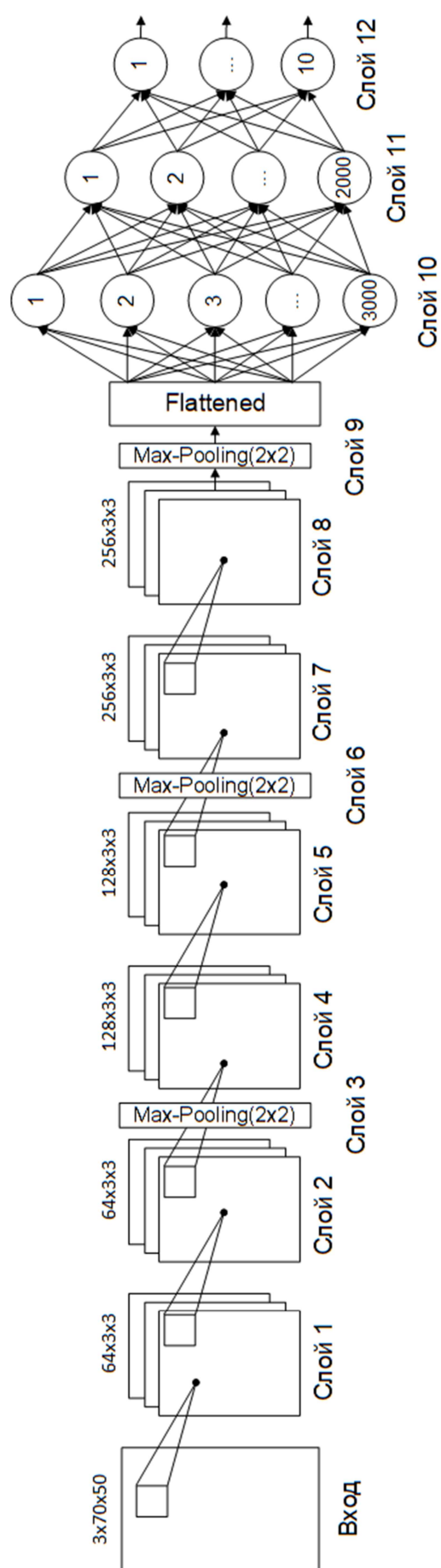


Рис. 39 – Архитектура свёрточной нейронной сети №3

Математическая модель архитектуры свёрточной нейронной сети №3 представлена следующими выражениями (22):

$$\left\{ \begin{array}{l} X^{(1)} = f_{ReLU}(X^{(0)} \otimes k^{(1)} + b^{(1)}) \\ X^{(2)} = f_{ReLU}(X^{(1)} \otimes k^{(2)} + b^{(2)}) \\ X^{(3)} = a^{(3)} \cdot \text{subsample}(X^{(2)}) + b^{(3)} \\ X^{(4)} = f_{ReLU}(X^{(3)} \otimes k^{(4)} + b^{(4)}) \\ X^{(5)} = f_{ReLU}(X^{(4)} \otimes k^{(5)} + b^{(5)}) \\ X^{(6)} = a^{(2)} \cdot \text{subsample}(X^{(5)}) + b^{(6)} \\ X^{(7)} = f_{ReLU}(X^{(6)} \otimes k^{(7)} + b^{(7)}) \\ X^{(8)} = f_{ReLU}(X^{(7)} \otimes k^{(8)} + b^{(8)}) \\ X^{(9)} = a^{(2)} \cdot \text{subsample}(X^{(8)}) + b^{(9)} \\ X^{(10)} = f_{ReLU}(X^{(9)} \cdot W^{(10)} + b^{(10)}) \\ X^{(11)} = f_{ReLU}(X^{(10)} \cdot W^{(11)} + b^{(11)}) \\ X^{(12)} = f_{SoftMax}(X^{(11)} \cdot W^{(12)} + b^{(12)}) \end{array} \right. \quad (22)$$

График процесса обучения представлен на рисунке 40.

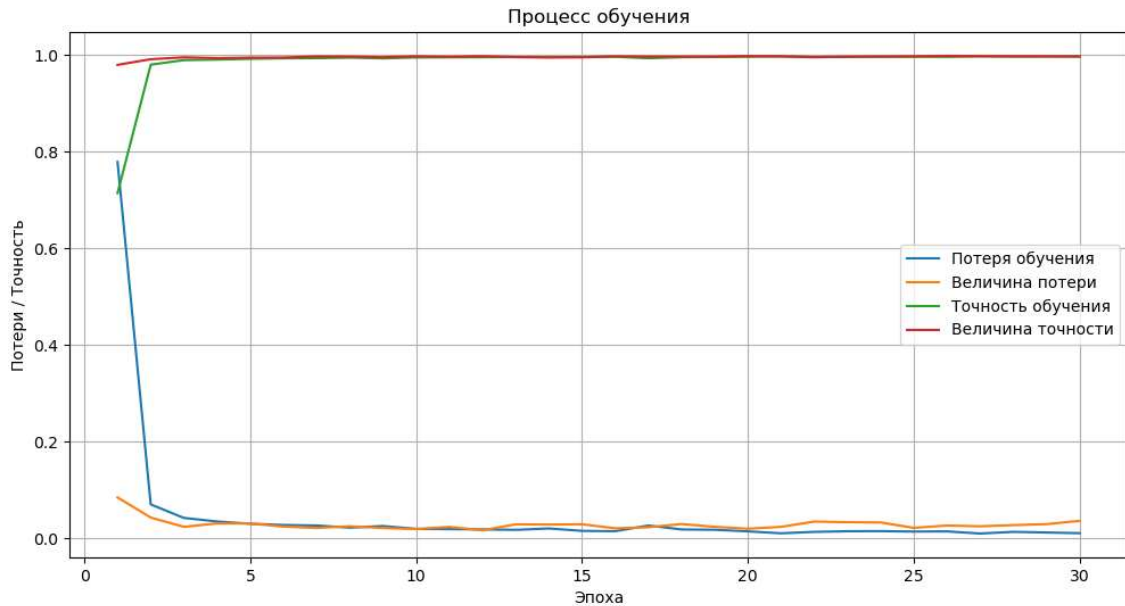


Рис. 40 – График процесса обучения свёрточной нейронной сети №3

Время обучения: 2280 сек.

Точность обучения: 99,64%

Результат тестирования: 99,8%

Скорость распознавания: 0,17 сек

Анализируя результаты обучения и тестирования шести выше представленных нейронных сетей можно заключить, что нейронные сети

прямого распространения несмотря на то, что в среднем на четверть быстрее свёрточных, показывают значительно меньшую результативность в распознавании и при изменении архитектуры незначительно изменяют своё быстродействие. Свёрточные сети при увеличении архитектуры значительно теряют в быстродействии при незначительном приросте распознавания. Исходя из полученных результатов в качестве дополнительного детектора будет использоваться свёрточная нейронная сеть №3 так как она показала наилучшие показатели в распознавании, а дальнейшее увеличение объёмов архитектуры не даст ощутимого результата.

В ходе разработки системы идентификации клейма литой заготовки были мнения, что нет критического значения в создании своего обучающего набора, для оценки этого мнения, по мимо собственного набора данных, возьмём ез1 и наиболее популярные наборы обучающих данных для цифр MNIST, SVHN, MSRA-TD500, ICDAR 2003/2013/2015. Для обучения также была взята нейронная сеть, описанная зависимостью 22, а тестирование проводилось на данных из собственного набора данных [68].

На рисунке 41 представлены результаты обучения нейронной сети на всех наборах данных.

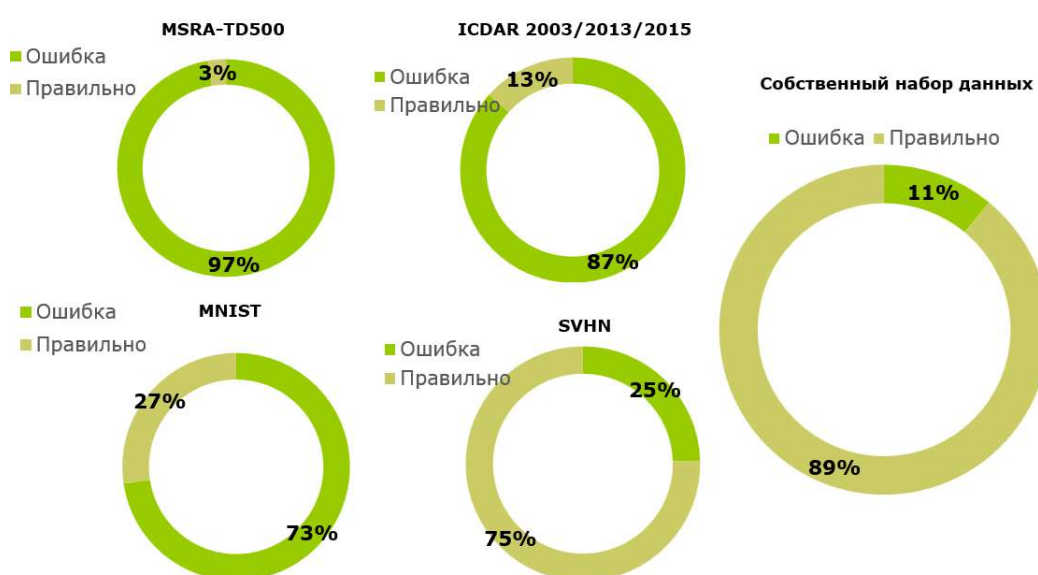


Рис. 41 – Результаты тестирования наборов данных

Крайне низкие результаты, показанные нейронными сетями обусловлены тем, что наборы данных MNIST, SVHN, MSRA-TD500, ICDAR 2003/2013/2015 не отображают в полной мере тех признаков, с которыми обладают цифры клейма литой заготовки. Так набор данных MNIST представляет собой рукописное начертание символа на абсолютно чистом фоне, в то время, когда наши символы имеют печатный вид и сильное шумовое загрязнение. Набор данных SVHN справился с задачей несколько лучше, чем MNIST, но всё равно из-за несоответствия признаков не может быть применим для решения данной задачи.

2.2.4 Алгоритм автоматического распознавания кода заготовки

Проанализировав все этапы алгоритма компьютерного зрения, был разработан алгоритм, представленный на рисунке 52. [72]

В начале изображение клейма заготовки проходит сегментацию методом пороговой бинаризации. Значение порога бинаризации подбирается автоматически. При начальном пороговом значении равным 80 изображение было отсегментировано. После сегментации производится расчёт критерия для сегментированного изображения, который укажет в какую сторону необходимо сдвинуть значение порога для достижения наилучшего результата сегментации.

В случае если значение критерия выше, чем допустимый допуск, происходит уменьшение порога бинаризации, если ниже, то порог увеличивается. После изменения порога изображение подвергается повторной сегментации, но уже с новым значением порога. Корректировка порога

После сегментации следует этап локализации. Отсегментированное изображение с помощью скользящего окна разбивается на ряд окон, каждое из которых подаётся на классификатор наличия объекта, который принимает решение, содержится ли объект в окне или нет.

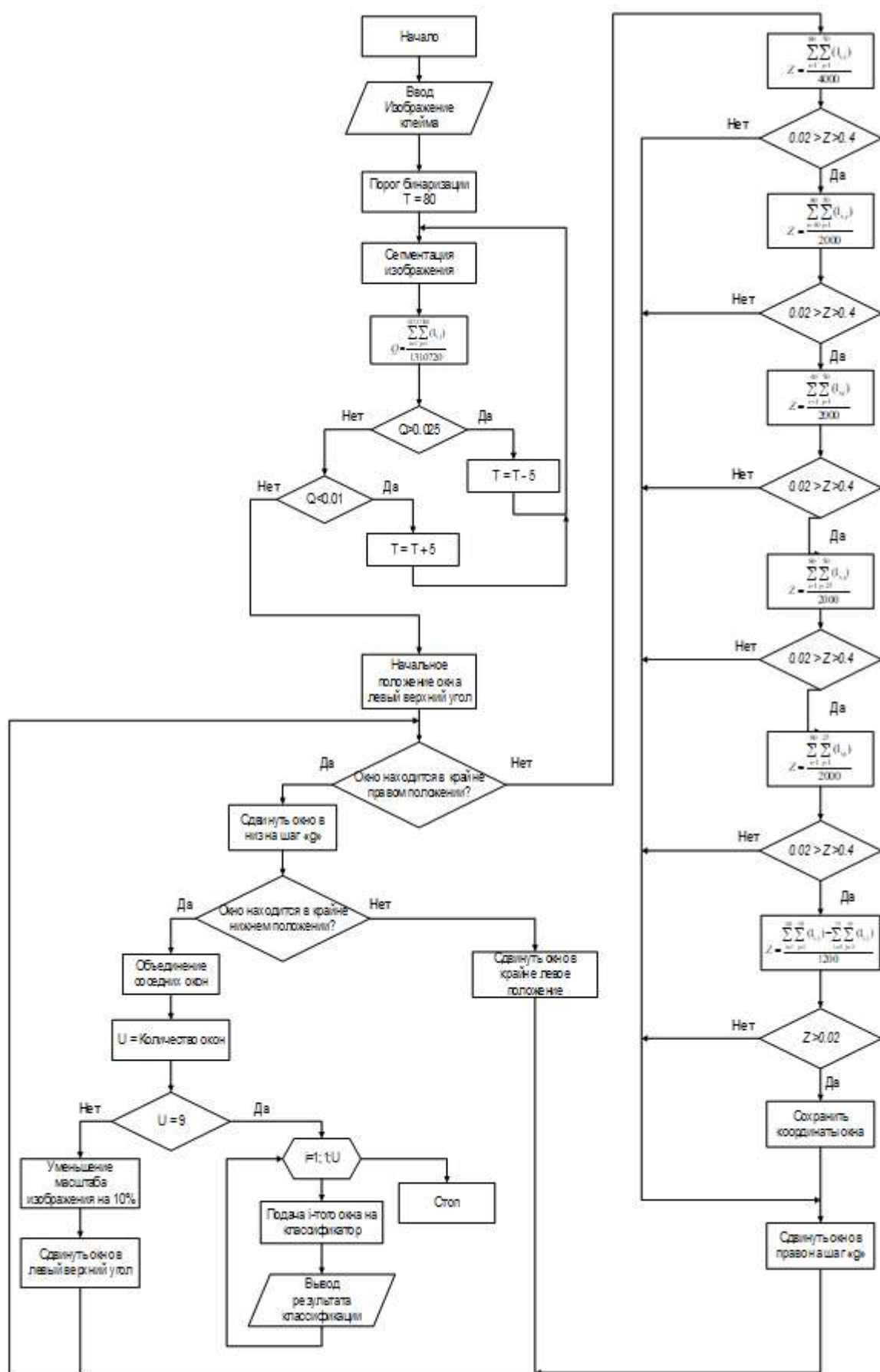


Рис. 52 – Алгоритм автоматического распознавания кода клейма

Объект может быть выделен более чем одним окном, для решения этой неопределённости производится усреднение соседних окон, чтобы каждый объект выделяло только одно окно.

Далее следует подсчёт выделенных объектов, если было выделено девять объектов (по счёту символов в клейма), вырезанные сегменты подаются на классификатор, где производится идентификация символов и верификация с информацией из СУБД. В роли классификатора выступает свёрточная нейронная сеть глубокого обучения.

Если же количество выделенных объектов отличается от девяти, то применяется принцип пирамиды изображения, на котором производится уменьшение изображения на 10%. Принцип пирамиды изображений применяется до того момента пока не будут выделены девять символов или не будет выполнен критерий остановки.

В случае если алгоритму не удалось выделить все символы или результат идентификации противоречит информации из СУБД, то система сигнализирует об ошибке.

2.2.5 Результаты работы алгоритма

Для тестирования разработанного алгоритма использовалась тестовая выборка, состоящая из 1000 изображений клейма, не использовавшихся при обучении и валидации нейронных сетей. Для оценки качества распознавания клейм использовался следующий критерий:

$$a_i = \begin{cases} 1, & \text{if } N_{\text{НС}i} = N_{\text{СУБД}i} \\ 0, & \text{if } N_{\text{НС}i} \neq N_{\text{СУБД}i} \end{cases} \quad (23)$$
$$E = \frac{\sum_{i=1}^I a_i}{I} \cdot 100\%$$

где $N_{\text{НС}i}, N_{\text{СУБД}i}$ – номер клейма, распознанного нейронной сетью и взятого из СУБД комбината соответственно.

Разработанная система показала неудовлетворительный результат и смогла корректно распознать только 57,8% поданных на неё клейм.

2.3 Детектор на основе глубоких нейронных сетей

Альтернативой стандартным методам компьютерного зрения являются нейросетевые детекторы на основе сетей, обученных с помощью глубокого обучения. Теоретический базис и популярные архитектуры подобных детекторов были разобраны в первой главе, тут же будет проведено их обучение и тестирование.

2.3.1 Обучение модели глубокого обучения

Вышеперечисленные модели нейросетевых детекторов показывают хорошие результаты в области компьютерного зрения, что позволяет использовать их для решаемой задачи.

Было рассмотрены 4 детектора на основе свёрточных нейронных сетей глубокого обучения:

1. Faster-RCNN Inception v2
2. Faster-RCNN Resnet-50
3. Faster-RCNN Resnet-101
4. Faster-RCNN Inception-Resnet v2

На рисунке 43 представлены графики обучения сети Faster-RCNN Inception v2. Обучение протекало 500000 тактов, время одного такта 0,26с.

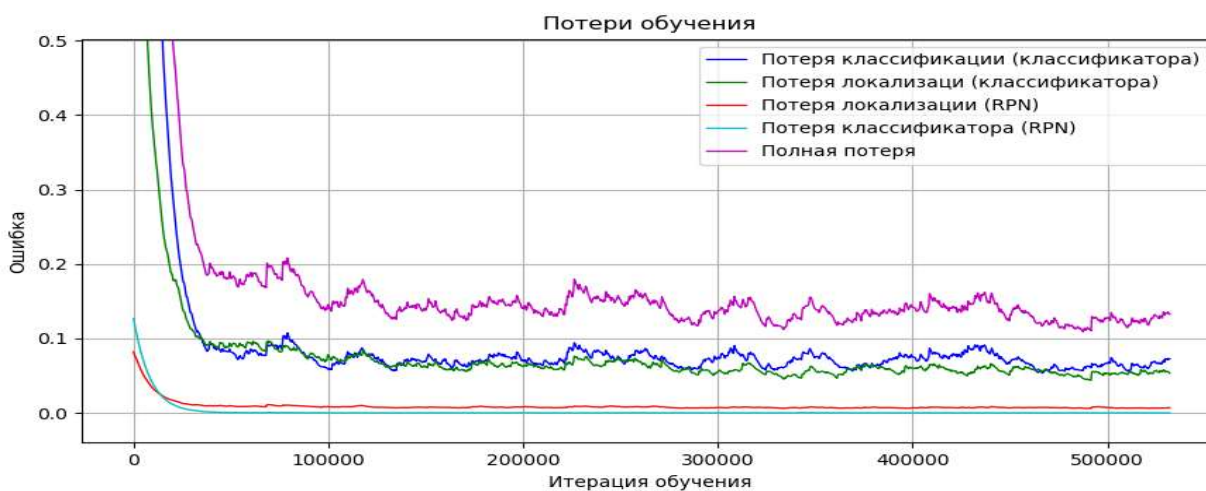


Рис. 43– Графики обучения сети Faster-RCNN Inception v2

На рисунке 44 представлены графики обучения сети Faster-RCNN Resnet-50. Обучение протекало 500000 тактов, время одного такта 0,16с.

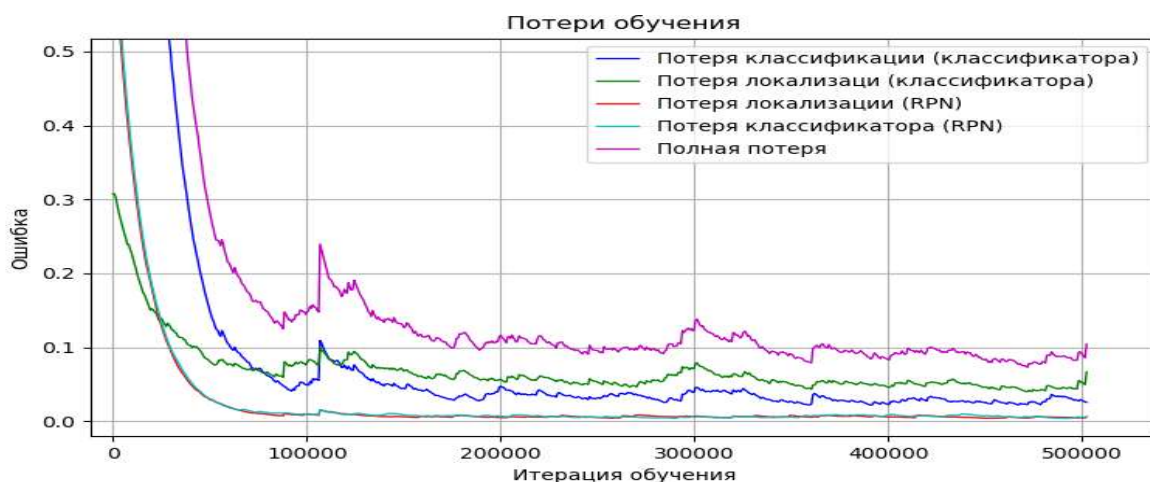


Рис. 44 – Графики обучения сети Faster-RCNN Resnet-50

На рисунке 45 представлены графики обучения сети Faster-RCNN Resnet-101. Обучение протекало 500000 тактов, время одного такта 0,4с.

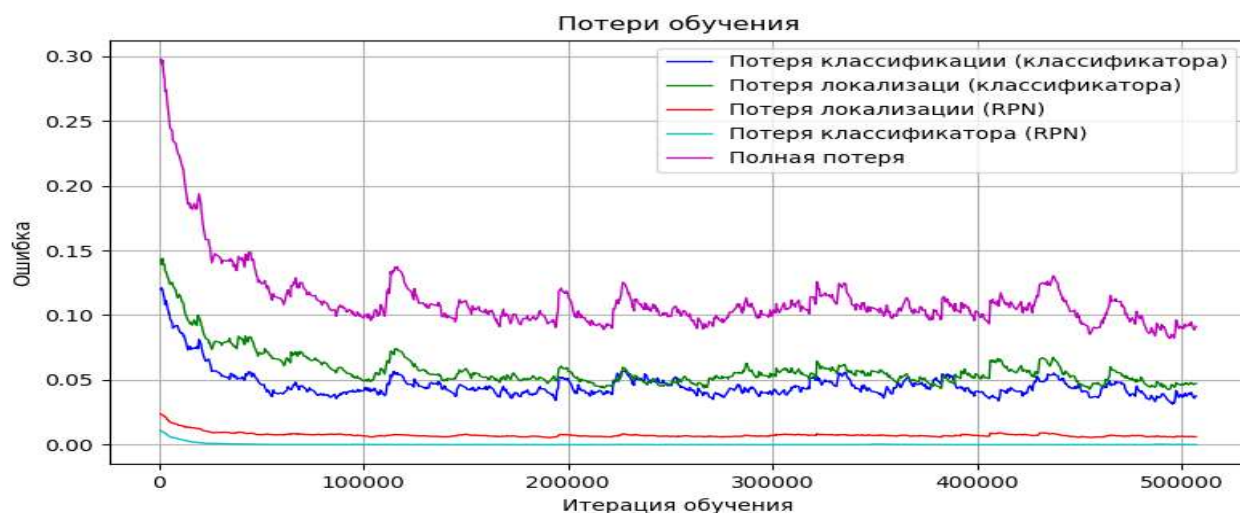


Рис. 45– Графики обучения сети Faster-RCNN Resnet-101

На рисунке 46 представлены графики обучения сети Faster-RCNN Resnet-101. Обучение протекало 500000 тактов, время одного такта 0,6с.

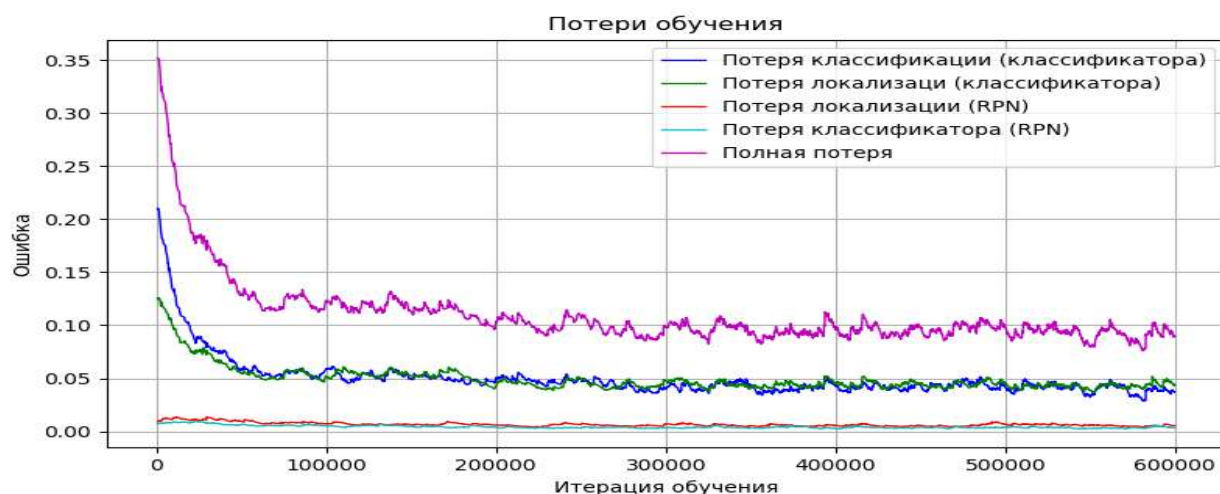


Рис. 46 – Графики обучения сети Faster-RCNN Inception-Resnet v2

Все модели были обучены в идентичных условиях, на протяжении 500 000 тактов, что обеспечило корректность последующего сравнения по временным и качественным характеристикам.

Анализ полученных результатов позволяет сделать следующие ключевые выводы:

1. Наименьшее время одного такта обучения зафиксировано у модели Faster-RCNN Resnet-50 (0,16 с), что делает её наиболее быстрой среди рассмотренных. Модель Faster-RCNN Inception v2 также показала высокое быстродействие (0,26 с). Увеличение глубины сети до 101 слоя (Resnet-101) привело к росту времени такта до 0,4 с, а наиболее сложная архитектура Inception-Resnet v2 оказалась самой медленной (0,6 с), что объясняется совмещением двух типов блоков (Inception и Residual) и значительным числом обучаемых параметров.

2. Представленные графики обучения (рис. 43–46) демонстрируют, что все четыре модели успешно проходят этап обучения и достигают устойчивой сходимости. Однако визуальный анализ кривых потерь и метрик точности позволяет предположить, что более глубокие архитектуры (Resnet-101 и Inception-Resnet v2) обеспечивают более низкие значения финальной функции потерь, что указывает на их потенциально более высокую точность детектирования. В то же время модели с меньшей глубиной (Resnet-50 и Inception v2) показывают более быструю начальную сходимость, что важно при ограниченных вычислительных ресурсах.

Исходя из совокупности полученных данных, выбор оптимального детектора должен определяться приоритетами конкретной прикладной задачи. Если ключевым критерием является скорость обработки и обучение планируется проводить на ограниченном оборудовании, предпочтительной является модель Faster-RCNN Resnet-50, если же на первый план выходит точность детектирования, а временные затраты допустимы, целесообразно использовать Faster-RCNN Inception-Resnet v2, несмотря на её самую низкую производительность.

2.3.2 Алгоритм работы системы на основе модели глубокого обучения

Проанализировав все этапы алгоритма компьютерного зрения, был разработан алгоритм, представленный на рисунках 47-49. [71]

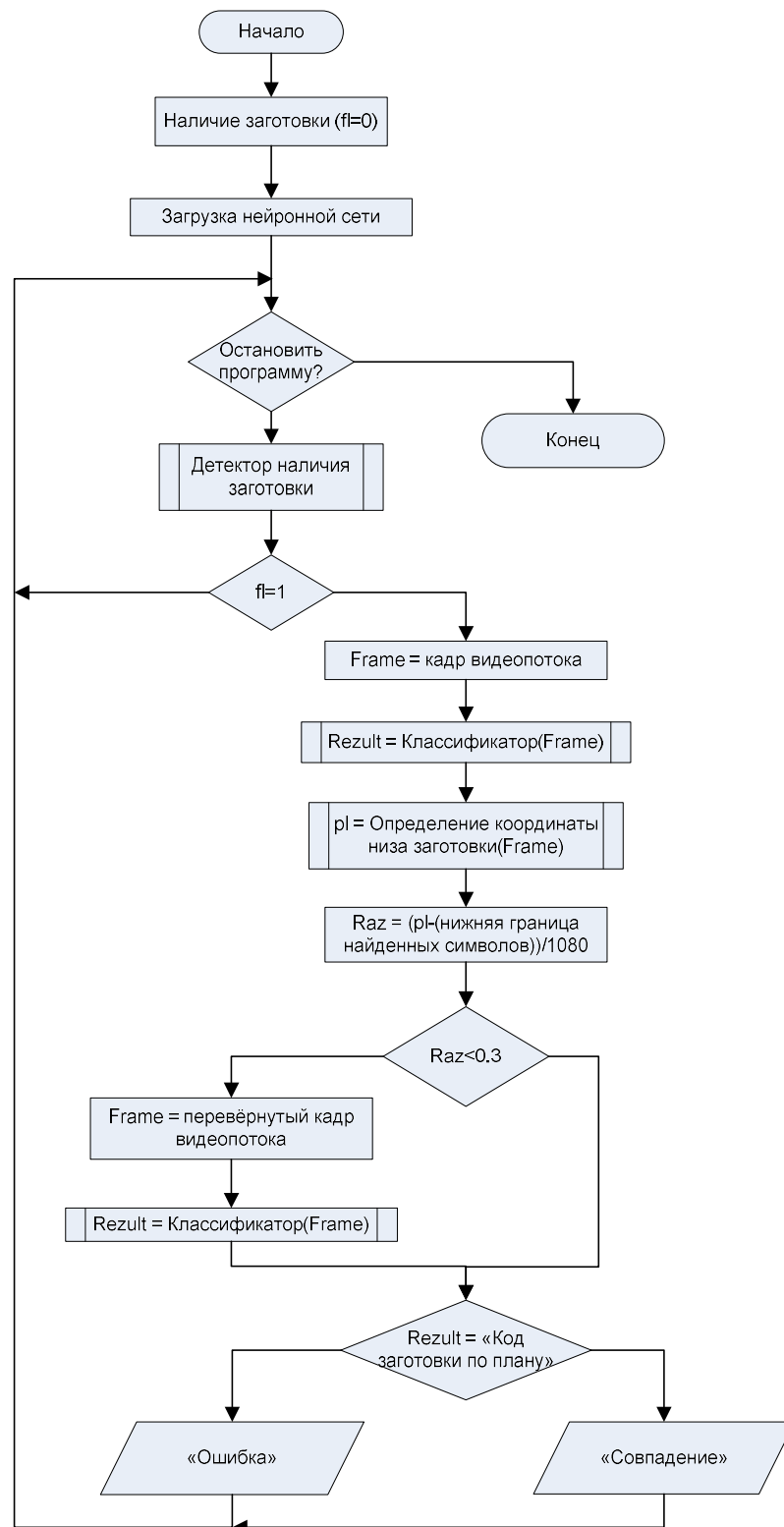


Рис. 47 – Алгоритм автоматического распознавания кода клейма на основе нейросетевого детектора

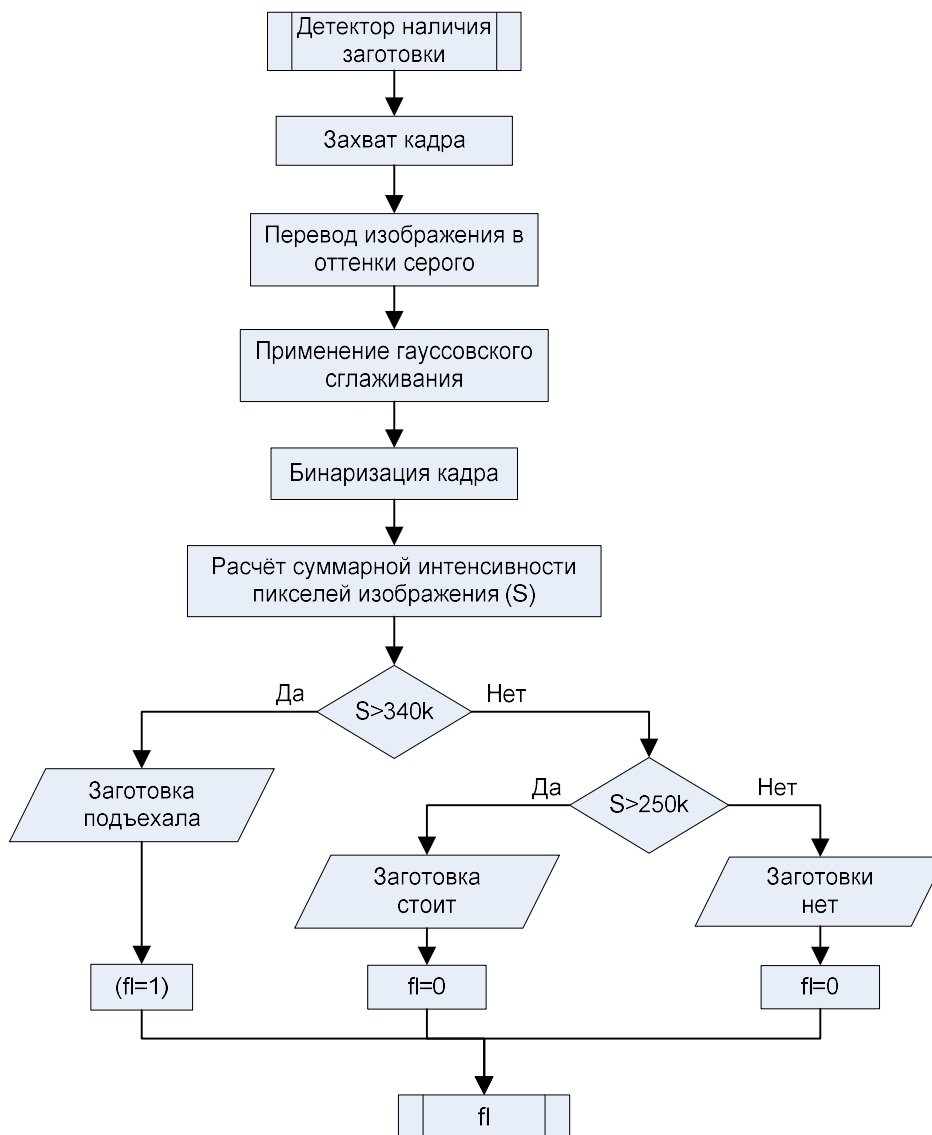


Рис. 48 – Алгоритм подпрограммы детектора наличия заготовки

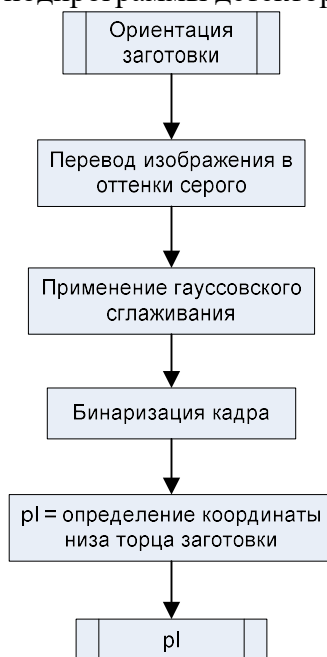


Рис. 49 – Алгоритм подпрограммы ориентации заготовки

2.3.3 Результаты работы

С целью определения какой из детекторов покажет лучшие результаты было проведено тестирование на изображениях, которые не участвовали в обучении. В ходе обучения нейросетевого детектора создавались контрольные точки, для анализа хода повышения результативности детектора и определения момента, когда дальнейшее обучение не приводит к улучшению результата.

Графики результатов тестирования приведены на рисунках 50-53. [72,73]

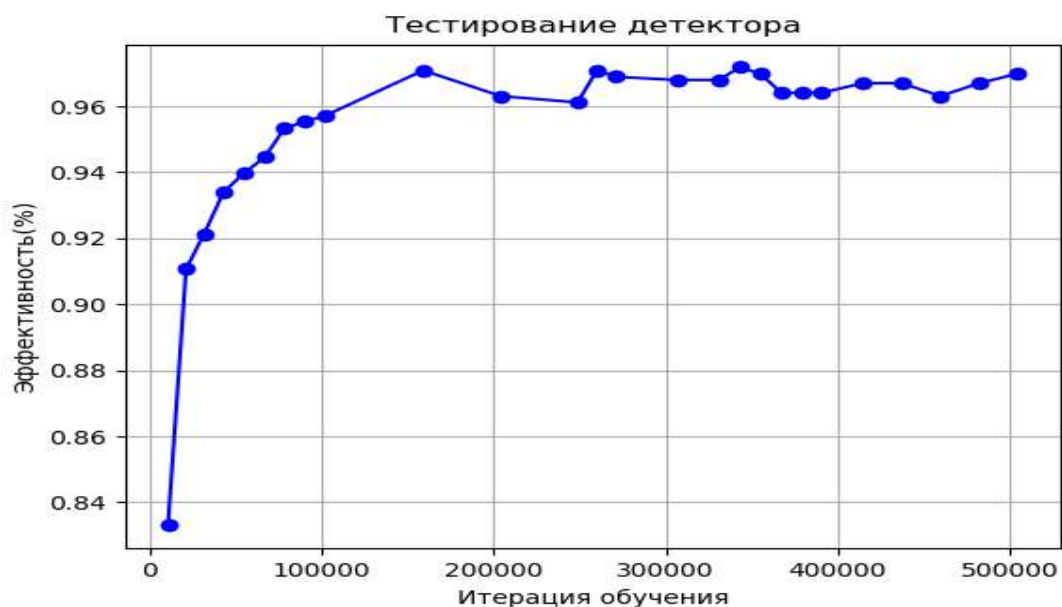


Рис. 50 – График результатов тестирования сети Faster-RCNN Inception v2



Рис. 51 – График результатов тестирования сети Faster-RCNN Resnet-50

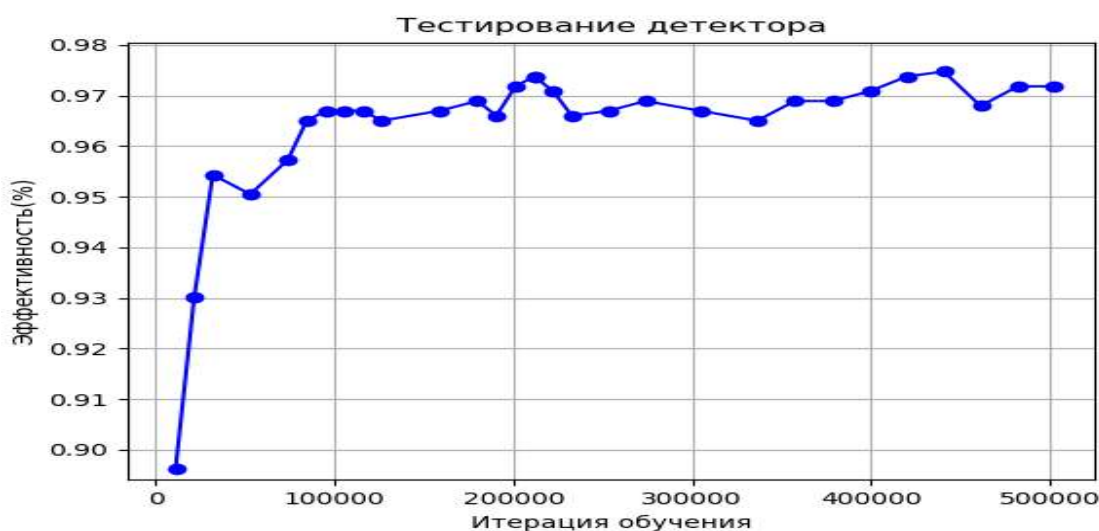


Рис. 52 – График результатов тестирования сети Faster-RCNN Resnet-101

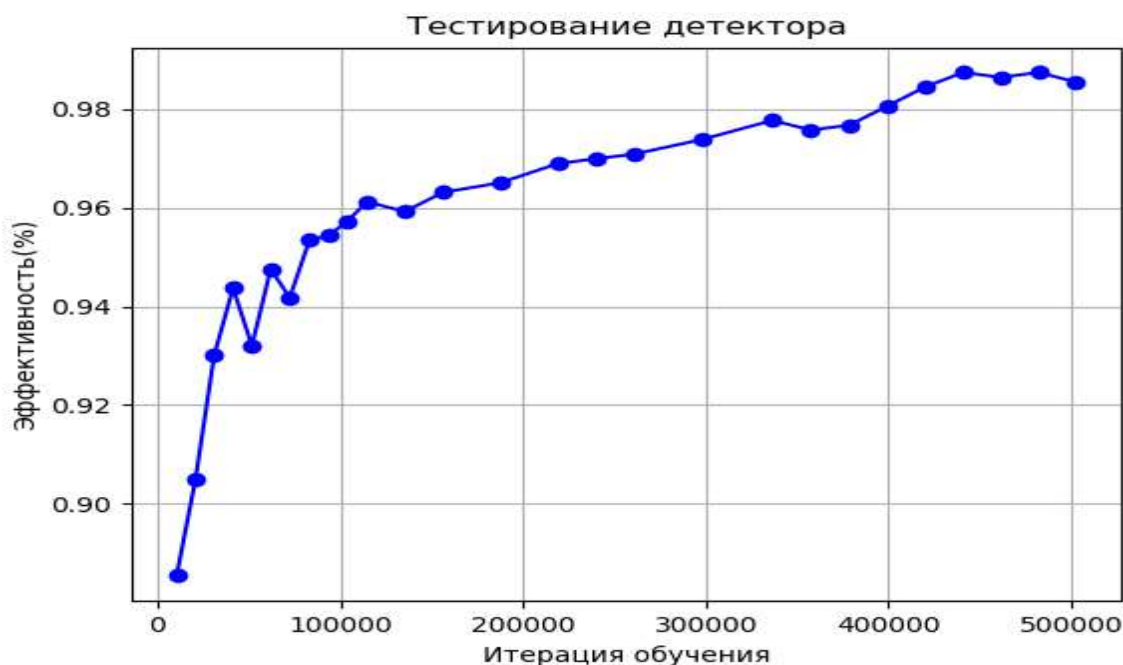


Рис. 53 – График результатов тестирования сети Faster-RCNN Inception-Resnet v2

По результатам тестирования были выбраны лучшие результаты для каждой сети:

1. Faster-RCNN Inception v2 – 97,1%
2. Faster-RCNN Resnet-50 – 97,4%
3. Faster-RCNN Resnet-101 – 97,5%
4. Faster-RCNN Inception-Resnet v2 – 98,7%

По результатам тестирования детектор Faster-RCNN Inception-Resnet v2 показал лучший результат, следовательно, в алгоритме будет использоваться именно он. Код алгоритма с выбранным детектором указан в приложении А.

3. Выводы по главе 2

В ходе осуществления исследований, представленных во второй главе, была выполнена разработка двух подходов к реализации системы автоматической идентификации цифр на изображении клейма.

Первый подход, основанный на классическом методе, показал себя крайне плохо. Это обосновывается тем, что подаваемые при тестировании данные имеют крайне высокую степень мешающих идентификации дефектов. А ввиду того, что повлиять на чистоту торца заготовки в условиях комбината не представляется возможным, данный подход не может быть реализован.

Второй же подход показал большую эффективность в распознавании цифр и обеспечил верную идентификацию 98,7% клейм на тестовой выборке. Следовательно, можно сделать вывод что глубокие нейронные сети способны извлекать большее количество признаков нежели простые и для поставленной задачи целесообразно использовать именно их.

Глава 3. Разработка способов повышения эффективности нейросетевого детектора

Несмотря на то, что во второй главе были получены приемлемые результаты, процент ошибочных решений остаётся достаточно высоким. Исходя из этого было принято решение о разработке подхода для повышения эффективности идентификации кодов литых заготовок путем совершенствования алгоритма распознавания.

3.1 Аугментация

Серьёзной проблемой при обучении нейронных сетей является сбор обучающей выборки. Сформированный во второй главе набор данных несмотря на то, что является одним из самых больших наборов данных из тех, что находятся в свободном доступе, имеет ряд недостатков, которые могут негативно влиять на результативность обучаемой модели.

Первым недостатком является объем. Хотя, как и говорилось выше, данным набор один из самых крупных, современные вычислительные мощности и архитектуры глубоких нейронных систем способны потреблять значительно большие объёмы информации.

Вторым недостатком является неравномерное распределение классов в выборке. Данный недостаток возник из-за того, что часть цифр реже используется при маркировке заготовок. Следовательно, можно предположить, что малое количество некоторых элементов приведёт к неустойчивости результата системы при распознавании по отдельным классам.

Третий недостаток заключается в однообразности данных. По опытным оценкам порядка 80-85% всех клейм имеют относительно хорошее качество нанесения, 10-15% имеют серьёзное шумовое загрязнение для одного или двух элементов кода, а малая часть клейм, менее 1%, имеет серьёзное шумовое загрязнение на большинстве элементах кода заготовки. Это приводит к тому,

что система отлично распознает хорошие клейма и испытывает трудности с кодами, имеющими какие-либо дефекты. [74]

Исходя из описанных выше проблем было принято решение о увеличении нашего набора данных по средствам методов аугментации.

3.1.1 Контрольная выборка

Для определения эффективности рассматриваемых методов аугментации необходимо провести контрольный эксперимент с базовой обучающей выборкой, которая в последствии будет размножена с помощью методов сегментации. Пример изображений обучающей выборки представлен на рисунке 54.

На изображении продемонстрировано, что искомые нами цифры клейма литой заготовки, могут находиться в разных местах изображения, на разном расстоянии от камеры, имеют разную освещенность, не имеют строго расположения относительно торца заготовки и друг друга, а также могут быть сильно повреждены окалиной или другими механическими воздействиями.



Рис. 54 – Пример обучающей выборки

На рисунке 55 продемонстрирован процесс обучения нейронной сети. Для обучения использовались 10000 изображений клейма, 20% которой использовались для тестирования, остальные 80% для обучения.

Здесь и во всех последующих экспериментах на графиках будут представлены такие характеристики как:

- Потеря классификации
- Потеря локализации
- Потеря классификации RPN
- Потеря локализации RPN

Так же для лучшего анализа полученных результатов их графическое представление было сглажено с помощью метода экспоненциально среднего скользящего (24) с коэффициентом альфа равным 0,95.

$$S_t = a \cdot x_t + (1 - a) \cdot S_{t-1} \quad (24)$$

где: S_t — значение ЕМА в момент t , x_t — текущее наблюдаемое значение, a — коэффициент сглаживания ($0 < a < 1$).

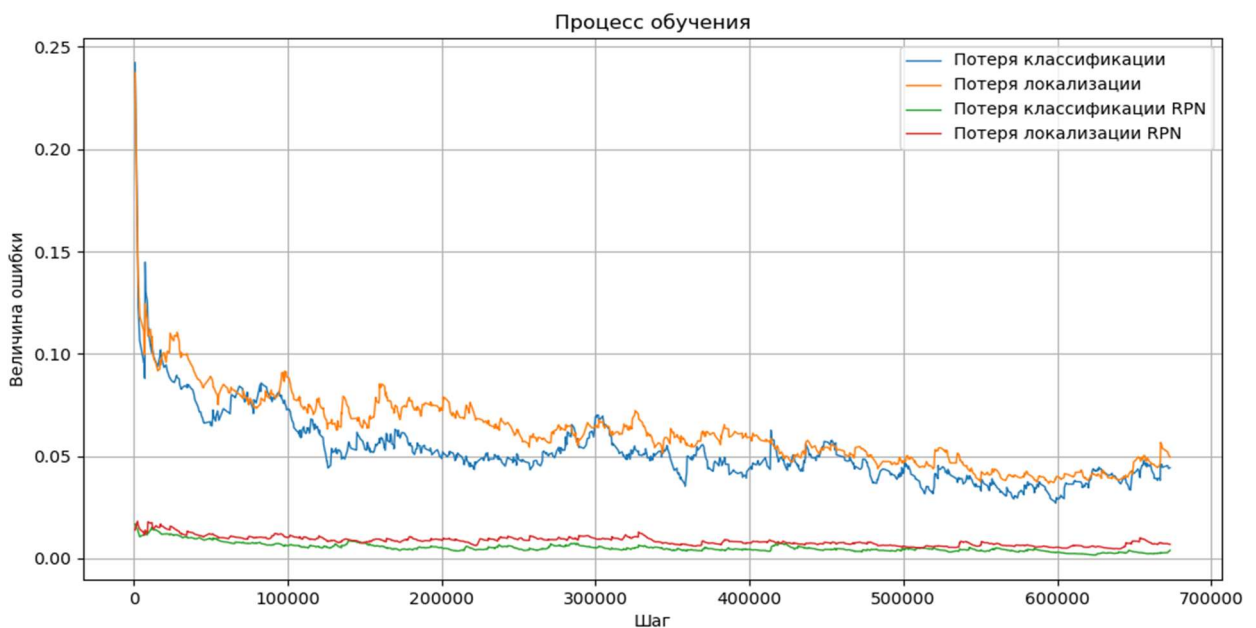


Рис. 55 – График потерь при обучении

После получения обученной модели она была протестирована на изображениях клейма, не участвующих в обучении. Результат корректного распознавания равняется 94,37%. При этом за всё время обучения значение ошибок классификации и локализации снизились до значения около 0,05, в то же время значения ошибок классификации и локализации для сети региональных предложений близко к нулю.

3.1.2 Метод поворота

Первым методом аугментации был выбран метод поворота изображения (25). Каждое из 10000 изображений было повернуто на 2,5 градусов вправо и влево, а также на 5 градусов вправо и влево. Всего вместе с начальной базой изображений было получено 50000 изображений клейма. Поворот изображения производился относительно одной точки, выбранной для всех изображений, относительно этой же точки производилось и корректирование ограничивающих рамок, локализирующих и классифицирующих искомые элементы.

$$\begin{cases} x' = (x - c_x)\cos\theta - (y - c_y)\sin\theta + c_x \\ y' = (x - c_x)\sin\theta - (y - c_y)\cos\theta + c_y \end{cases} \quad (25)$$

где: (x, y) – исходные координаты пикселя, θ – угол поворота, (x', y') – новые координаты пикселя, (c_x, c_y) – центр вращения.

Так же ограничивающие рамки были несколько расширены для того, чтобы корректно выделить элемент.

Результаты аугментации представлены на рисунке 56.



Рис. 56 – Полученные изображения

После получения обученной модели она была протестирована на изображениях клейма, не участвующих в обучении. Результат корректного распознавания равняется 93,88%.

Результаты тестирования показали, что сеть уменьшила свою результативность на 0,49%. Так же это отражается в графиках, иллюстрирующих процесс обучения, на которых видно, что к моменту завершения обучения ошибка классификации и локализации выше, чем у сети, которая обучалась на оригинальных данных.

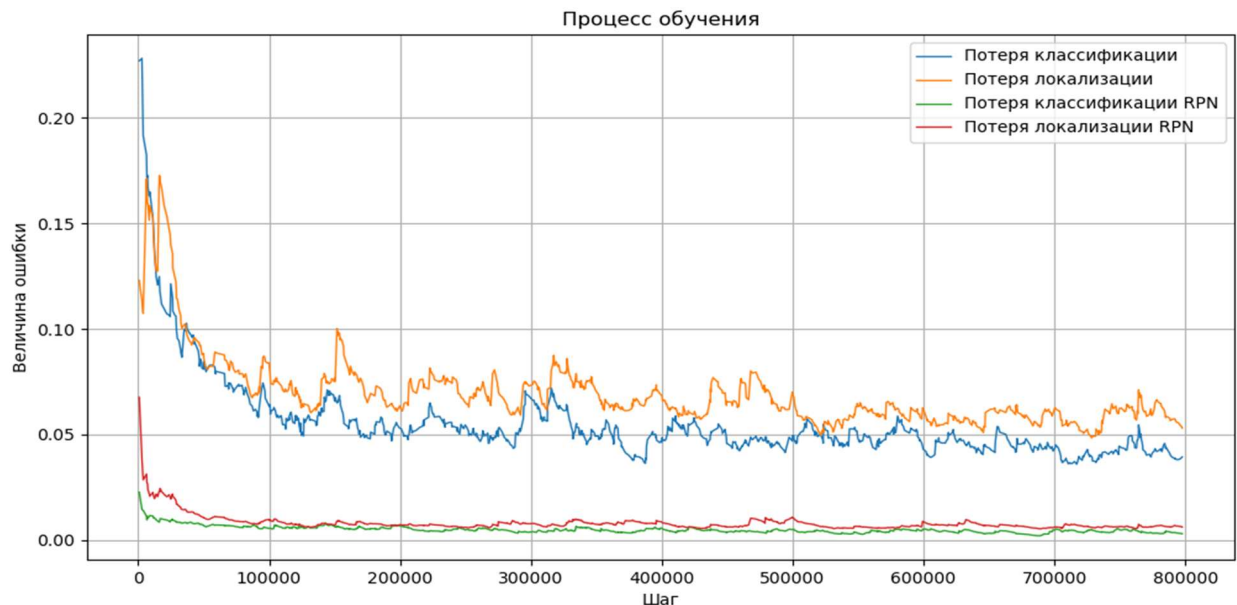


Рис. 57 – График потерь при обучении

Подобный результат можно обосновать тем, что как правило цифры клейма, имеют точно вертикальное расположение и поворот цифры является крайне редким огрехом клеймовочной машины. И из-за этого увеличение обучающей выборки не качественными данными снизило качество работы сети.

3.1.3 Метод сдвига

Следующим методом аугментации был выбран сдвиг изображения (26). По сути, изображение не изменилось, но все ограничивающие рамки были сдвинуты вверх и влево, вниз и вправо, вверх и вправо, вниз и влево. Также, как и в предыдущем эксперименте всего вместе с начальной базой изображений было получено 50000 изображений клейма.

$$\begin{cases} x' = x + \lambda_x(y - c_y) \\ y' = y + \lambda_y(x - c_x) \end{cases} \quad (26)$$

где: (x, y) – исходные координаты пикселя, (x', y') – новые координаты пикселя, $(\lambda_x \lambda_y)$ – коэффициенты интенсивности сдвига, (c_x, c_y) – координаты центра преобразования

Пример обучающей выборки представлен на рисунке 58.

После получения обученной модели она была протестирована на изображениях клейма, не участвующих в обучении. Результат корректного распознавания равняется 94,47%.



Рис. 58 – Полученные изображения

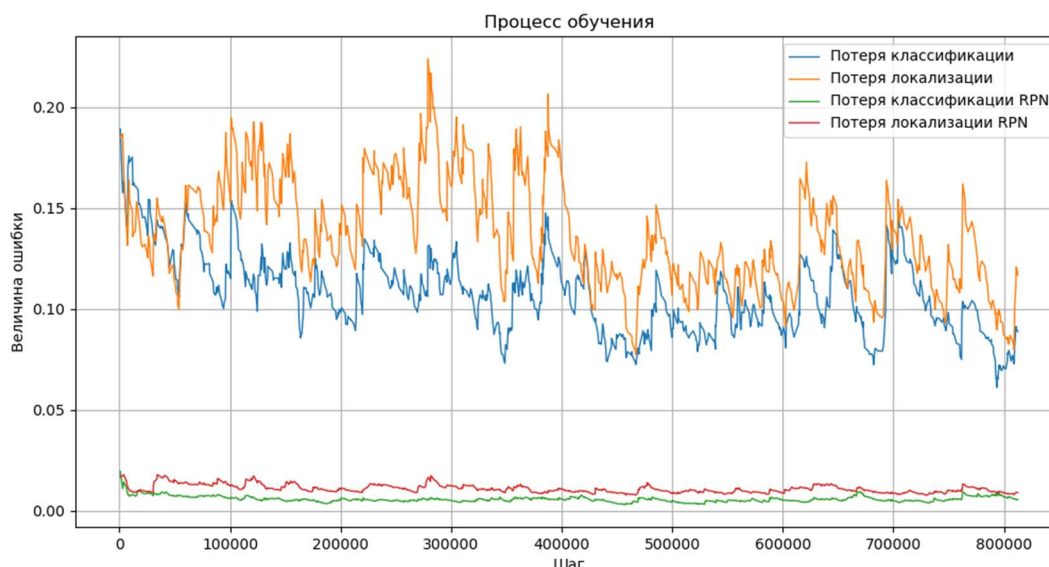


Рис. 59 – График потерь при обучении

Анализируя графики процесса обучения видно, что ошибки классификации и локализации значительно выше, чем в предыдущих случаях,

а также присутствуют значительные всплески, несмотря на высокий коэффициент сглаживания. Это можно объяснить тем, что одно и то же изображение обладает несколькими ограничивающими рамками лишь, которые частично не пересекаются. Хотя при этом результативность работы сети выросла на 0,1%.

3.1.4 Метод изменения яркости

Следующим методом стал метод изменения яркости изображения (27). Каждое оригинальное изображение было продублировано с изменением яркости в 0,3 и 0,5 – затемнение и в 1,4 и 1,8 – осветление. Как и в предыдущих случаях всего получилось 50000 изображений.

$$\begin{cases} I'(x, y) = I(x, y) + \beta \\ I'(x, y) = \text{clip}(I(x, y) + \beta, 0, 255) \end{cases} \quad (27)$$

где: $I'(x, y)$ – результирующее значение пикселя, $I(x, y)$ – результирующая интенсивность пикселя после аугментации, β – коэффициент сдвига яркости ($\beta > 0$ — изображение становится светлее, $\beta < 0$ — изображение становится темнее)

Пример обучающей выборки представлен на рисунке 0.



Рис. 60 – Полученные изображения

После получения обученной модели она была протестирована на изображениях клейма, не участвующих в обучении. Результат корректного распознавания равняется 94,66 %.

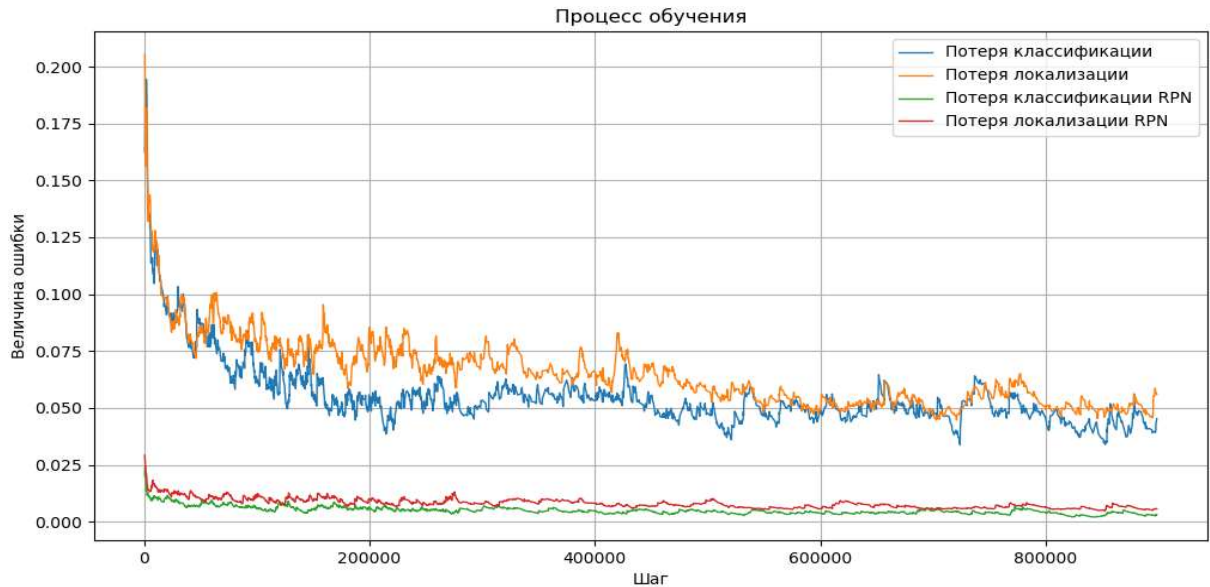


Рис. 61 – График потерь при обучении

3.1.5 Методы размытия и изменения резкости

В следующем эксперименте использовалось два метода аугментации, размытие изображения (28) и изменение резкости (30). Оригинальная база была дважды подвергнута размытию с коэффициентами 3 и 4, также была увеличена контрастность с коэффициентом 2,5 и уменьшена контрастность с коэффициентом 0,5.

$$I'(x, y) = \sum_{i=-k/2}^{k/2} \sum_{j=-k/2}^{k/2} I(x + i, y + j) \cdot G(i, j) \quad (28)$$

где: $I'(x, y)$ – результирующее значение пикселя, $I(x + i, y + j)$ – значения исходных пикселей из окрестности точки (x, y) , k — размер ядра, $G(i, j)$ — значения ядра (29).

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (29)$$

где: (x, y) – координаты элемента ядра относительно его центра, σ стандартное отклонение (параметр размытия).

$$I'(x, y) = I_c + a \cdot (I(x, y) - I_c) \quad (30)$$

где: I_c – среднее значение яркости по всему изображению; a – коэффициент контрастности, $I'(x, y)$ – Новое значение пикселя, $I(x, y)$ – исходное значение пикселя.

Пример обучающей выборки представлен на рисунке 62.

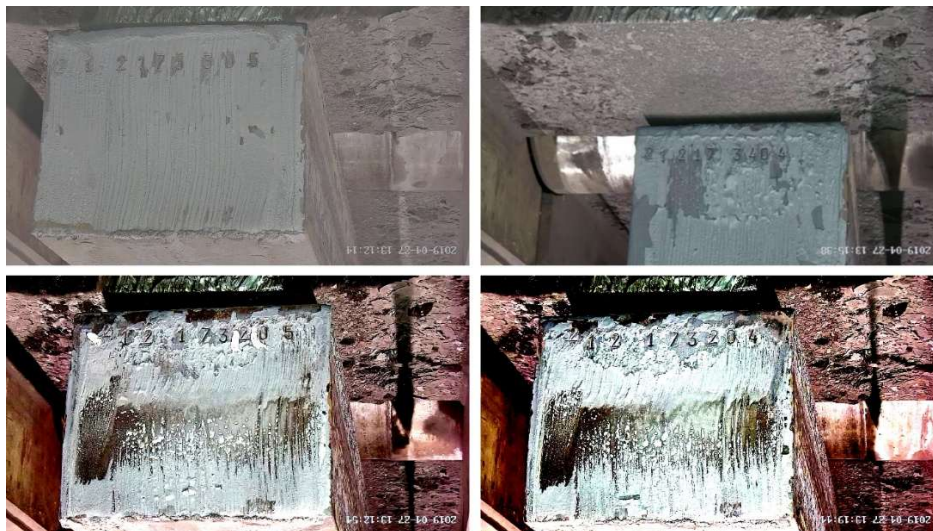


Рис. 62 – Полученные изображения

Смотря на процесс, обучения представленный на рисунке 63 можно увидеть, что, как и при обучении на оригинальной базе потери RPN сети близятся к 0, а потери классификации и локализации около 0,05. При этом после получения обученной модели и тестирования её на изображениях клейма, не участвующих в обучении результат корректного распознавания, равняется 95,24%.

В сравнении с первой сетью точность увеличилась на 0,87%.

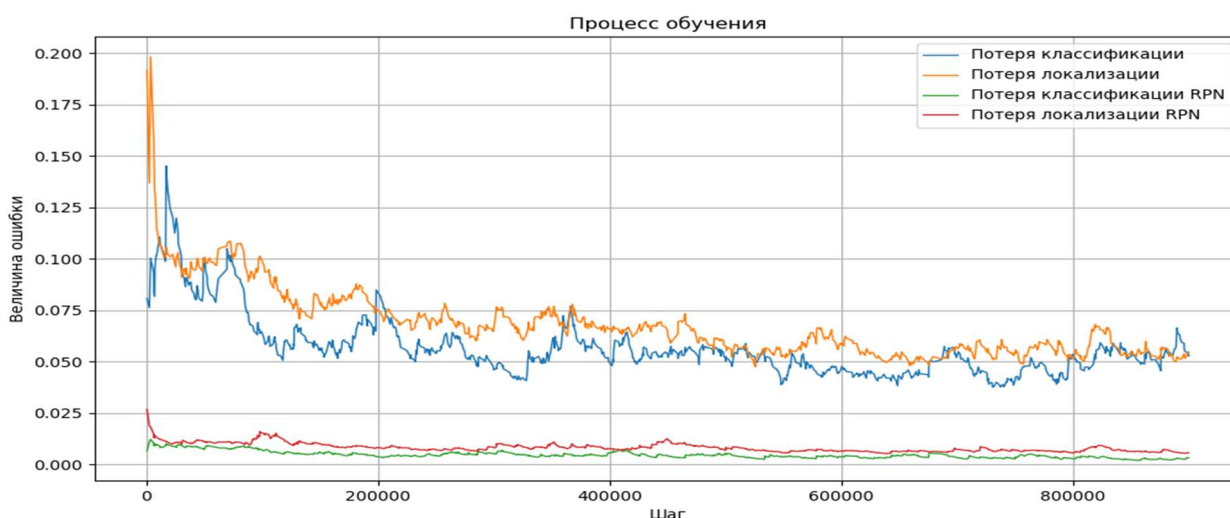


Рис. 63– График потерь при обучении

3.1.6 Способ оптимизации времени обучения

В ходе проведения экспериментов по аугментации и долгого обучения нейронных сетей возникла необходимость уменьшить время, затрачиваемое на процесс обучения. Инструментарий TensorFlow позволяет использовать в одной итерации обучения сразу несколько изображений. Данный приём, за счёт увеличения подаваемой информации в такт, позволяет увеличить обобщающую способность сети и соответственно увеличить качество обучения. Но подать большое количество изображений невозможно так как даже самые мощные графические процессоры имеют относительно не большой объём памяти.

С целью обойти данное ограничение была сформирована база данных каждое из изображений которых содержит десять клейм. Клейма были вырезаны и помещены на фон близкий в тому, как выглядит торец литой заготовки.

Для формирования новой базы использовалась оригинальная база данных, которая использовалась для экспериментов с аугментацией, из 10000 изображений было получено 2000.

Пример обучающей выборки представлен на рисунке 64.

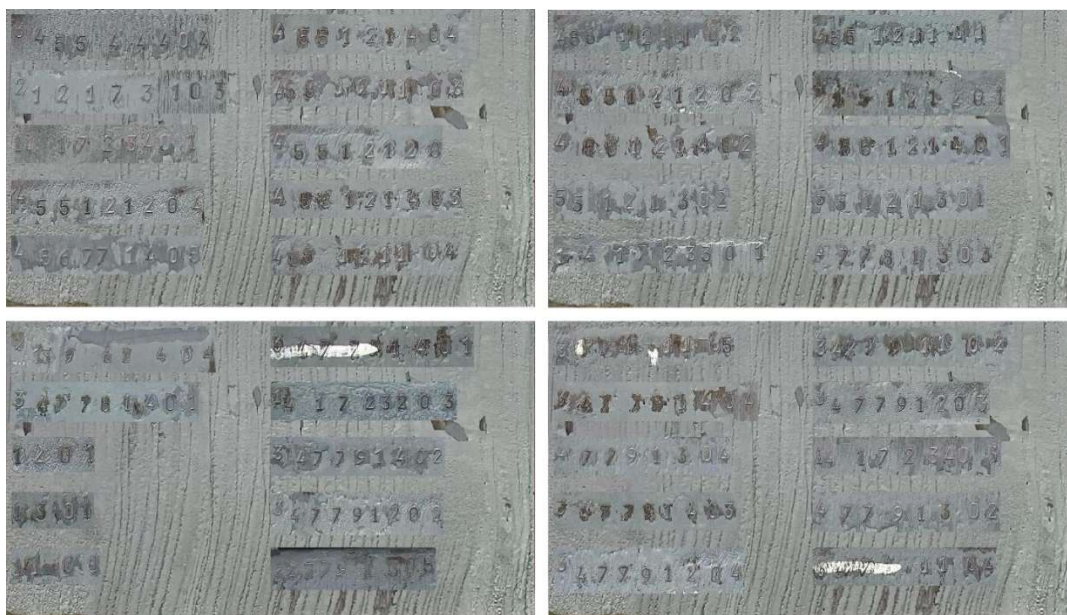


Рис. 64 – Полученные изображения

Процесс обучения представлен на рисунке 65 показывает минимальную ошибку для классификации и локализации по сравнению со всеми остальными процессами обучения, ошибка опустилась ниже 0,025.

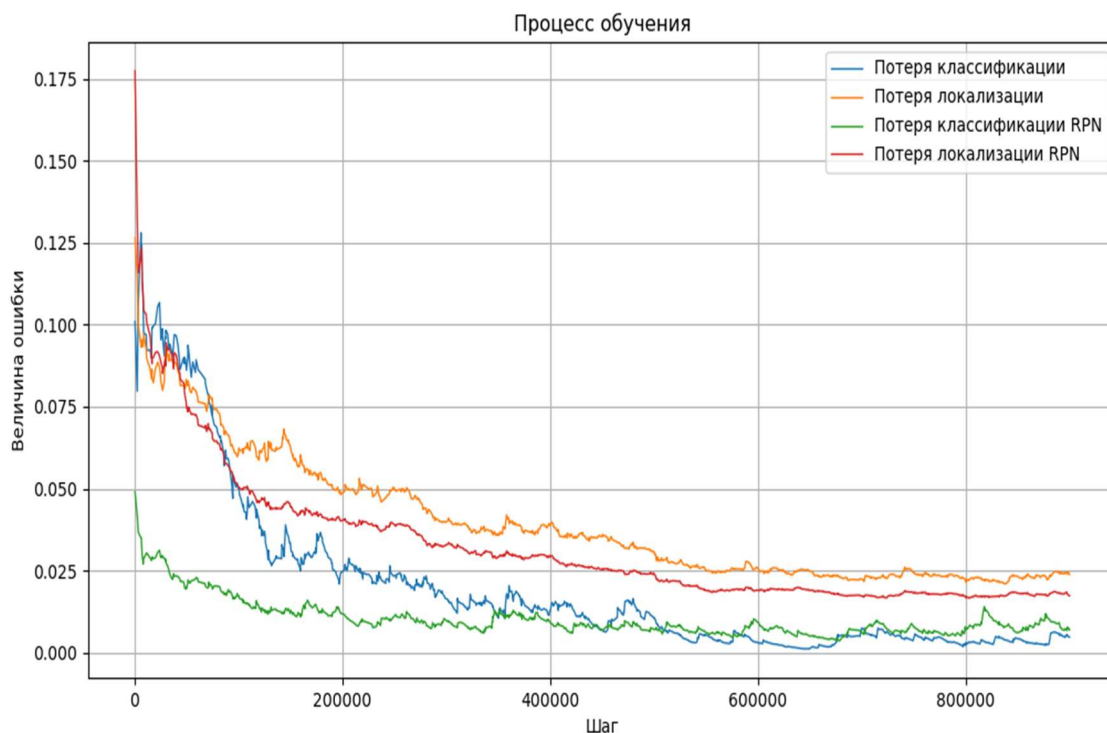


Рис. 65 – График потерь при обучении

При тестировании обученной модели результат распознавания был менее 10%. Анализируя то, как сеть распознаёт объекты на изображении (рисунок 80) видно, что модель выделяет объекты в случайных областях и при этом присваивает им высокую вероятность нахождения в ограничивающей рамке.

Данный эффект, судя по всему, возник из-за того, что в обучающей выборке была большая концентрация объектов на изображении, и эти объекты были равномерно распределены по всей площади поиска. Этот факт негативно сказался на работе сети региональных предложений, что и повлекло подобный результат.

Следовательно, увеличить скорость обучения нейронной сети можно либо за счёт применения более мощного графического процессора, либо выбора менее глубоких архитектур нейронных сетей.



Рис. 66 – Результат распознавания

3.2 Результаты аугментации

Сводный итог тестирования нейронной сети, обученной на выборках, сформированных с помощью разных методов аугментации представлен на изображении 82.

В ходе эксперимента по оценке качества влияния разных методов аугментации на качество работы нейронной сети видно, что единственным методом, давшим отрицательный прирост, выступил метод поворота, остальные же методы в разной мере позволили повысить качество работы нейронной сети.

Анализируя результаты, которые показал каждый из рассматриваемых методов, можно заявить, что использование метода поворота не целесообразно в ситуациях, когда все интересующие нас элементы имеют чёткую геометрическую ориентацию, так как этот метод нагружает нейронную сеть дополнительными признаками, не помогающими найти и распознать элемент на изображении, за счёт чего происходит уменьшение значимости тех признаков, которые способствуют нахождению и распознаванию.

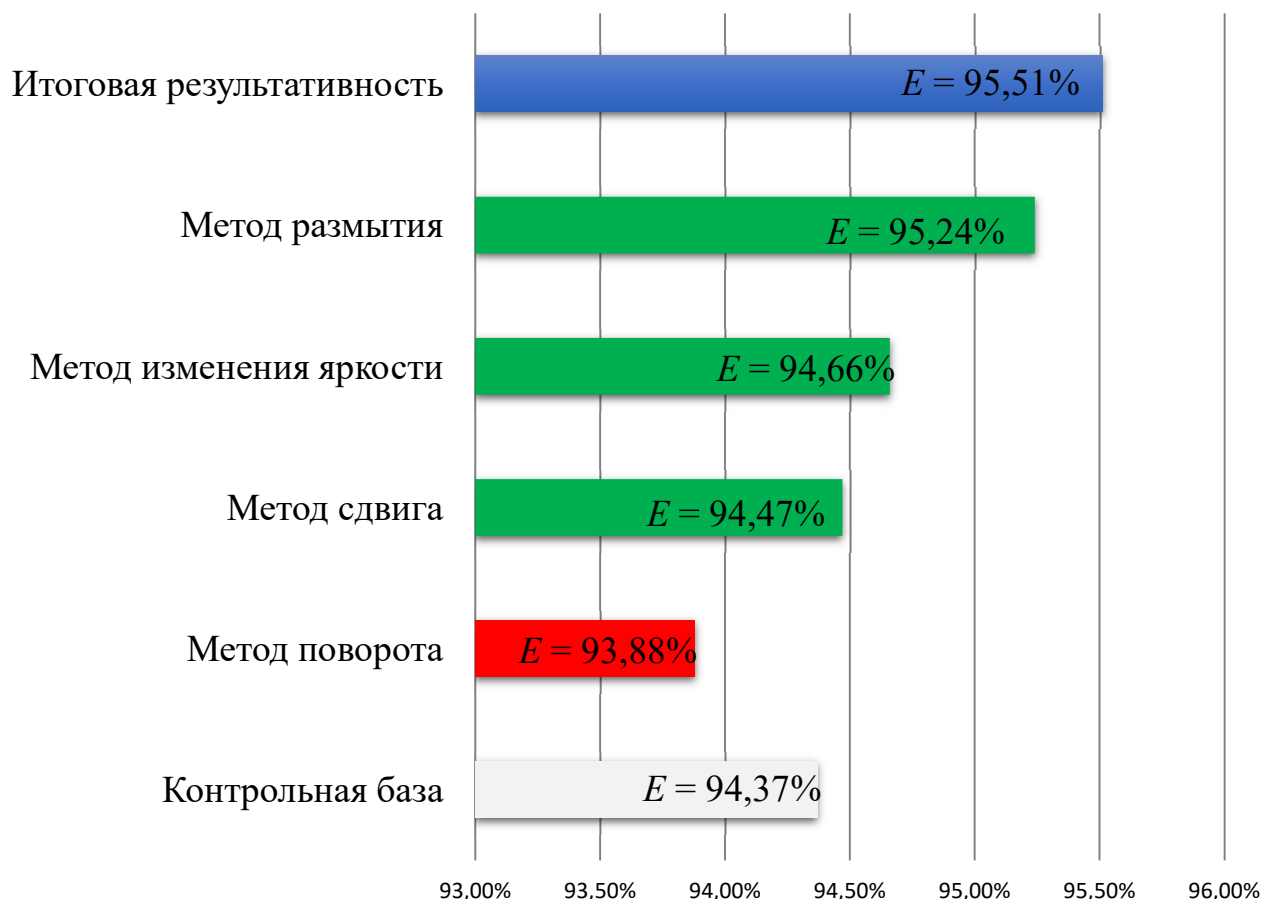


Рис. 67 – Результативность работы сети обученной на разных методах аугментации

Итогов результативность рассчитывалась согласно формуле (23):

$$V_{\text{итог}} = V_{\text{ярк}} \left(\sum_{i=1}^4 (0,25 \cdot V_{\text{исх}}(\beta_i)) \right) + V_{\text{сдв}} \left(\sum_{i=1}^4 (0,25 \cdot V_{\text{исх}}((\lambda_x, \lambda_y)_i)) \right) + V_{\text{разм}} \left(\sum_{i=1}^4 (0,25 \cdot V_{\text{исх}}(\sigma_i)) \right) \quad (31)$$

где $(\lambda_x, \lambda_y) = [0.01, 0.01; 0.01, -0.01; -0.01, 0.01; -0.01, -0.01]$,

$\beta = [0.3; 0.5; 1.4; 1.8]$, $\sigma = [0.5; 2.5; 3; 4]$.

Согласно формуле (23), использовались только те методы, которые показали положительную результативность относительно контрольной базы, а именно: $V_{\text{ярк}}$, $V_{\text{сдв}}$, $V_{\text{разм}}$ – методы аугментации по изменению яркости, сдвига и размытия соответственно, где β , λ_x и λ_y , σ – коэффициенты вышеописанных методов, а $V_{\text{исх}}$ – исходная обучающая выборка.

Метод сдвига показал минимальный прирост в результативности, он не формирует новые признаки, а лишь сдвигает ограничивающую рамку.

Использование данного метода для рассматриваемой задачи так же не целесообразно, так как польза от него минимальна.

Методы изменения яркости и контрастности обеспечили существенный прирост распознавания. Данный результат обоснован тем, что изображение торца литой заготовки не всегда имеет одинаковую яркость (это может быть связано со временем суток или качеством работы осветительного прибора) и чёткость (на это влияет расстояние от торца заготовки до камеры и степень запылённости объектива), а добавление в обучающую выборку данных с подобными признаками расширили обобщающую работу нейронной сети.

3.3 Каскад малых нейронных сетей

Анализируя результаты работы нейросетевого детектора было замечено, что нередко случаи, когда искомый элемент выделяется на изображении, но идентифицируется некорректно, или же выделив элемент, нейросетевой детектор не может с высокой долей вероятности отнести его к какому либо из классов и пропускает его.

Для решения этой проблемы предлагается использовать малые нейронные сети, которые будут применяться в тех случаях, когда есть сомнение в корректности идентификации или же когда вероятность нахождения цифры клейма в ограничивающей рамке мала, для того чтобы точно идентифицировать его.

Для определения наилучшей архитектуры нейронной сети было проведённое шесть экспериментов с тремя архитектурами нейронной сети прямого распространения и тремя архитектурами свёрточной нейронной сети. В качестве значимых параметров рассматривается точность распознавания и быстроедействие. [75]

3.3.1 Тестирование нейронных сетей прямого распространения

Вначале были обучены и протестированы архитектуры с одним скрытым слоем. Количество нейронов в скрытых слоях подбиралось интуитивно и

обоснованно желанием охватить максимально возможный диапазон.

Описание архитектур тестируемых сетей представлены в таблице 5.

Таблица 5

Номер сети	Нейронов в скрытом слое	Номер сети	Нейронов в скрытом слое	Номер сети	Нейронов в скрытом слое
1	100	10	1000	19	3000
2	200	11	1200	20	3500
3	300	12	1400	20	3500
4	400	13	1600	21	4000
5	500	14	1800	22	5000
6	600	15	2000	23	7500
7	700	16	2250	24	10000
8	800	17	2500	25	100000
9	900	18	2750		

Результаты тестирования представлены на рисунке 68. По левой оси, ниже представленной гистограммы, отображено общее количество ситуаций, когда нейронная сеть присвоила поданному элементу ошибочный класс. По правой оси отображается процент ошибки от общего количества в тестовой выборке.

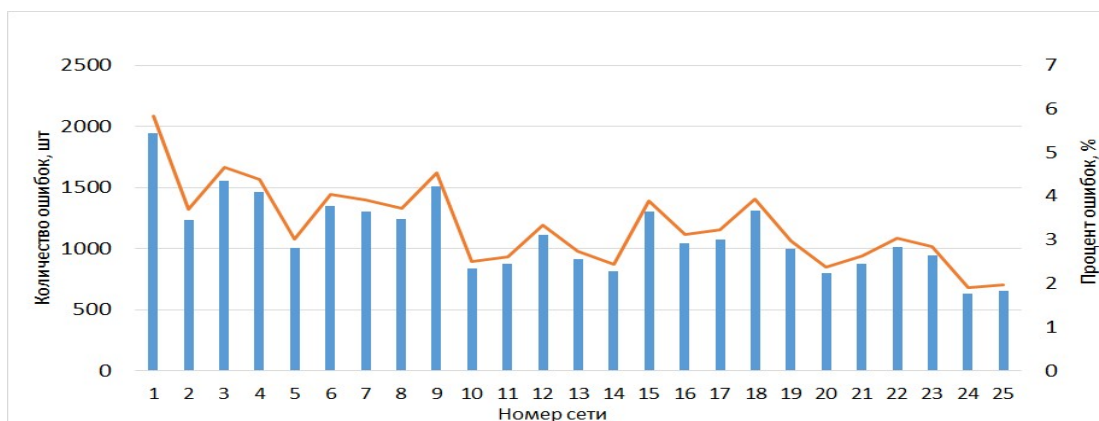


Рис. 68 – Результаты тестирования нейронных сетей с одним скрытым слоем

По результатам тестирования видно, что наихудший результат показала архитектура со ста нейронами в скрытом слое (1948 ошибок), а лучший результат показала архитектура с десятью тысячами нейронов в скрытом слое (636 ошибок).

Анализируя результаты обучения видно, что качество идентификации растёт с ростом количества нейронов в скрытом слое, но при этом растёт и время, затрачиваемое на распознавание.

Далее тестировались архитектуры с двумя скрытыми слоями. Описание архитектур тестируемых сетей представлены в таблице 6.

Таблица 6

Номер сети	Нейронов в скрытом слое №1	Нейронов в скрытом слое №2	Номер сети	Нейронов в скрытом слое №1	Нейронов в скрытом слое №2
1	100	100	14	1000	1600
2	100	400	15	1000	2500
3	100	1000	16	1600	100
4	100	1600	17	1600	400
5	100	2500	18	1600	1000
6	400	100	19	1600	1600
7	400	400	20	1600	2500
8	400	1000	21	2500	100
9	400	1600	22	2500	400
10	400	2500	23	2500	1000
11	1000	100	24	2500	1600
12	1000	400	25	2500	2500
13	1000	1000			

Результаты тестирования представлены на рисунке 69.

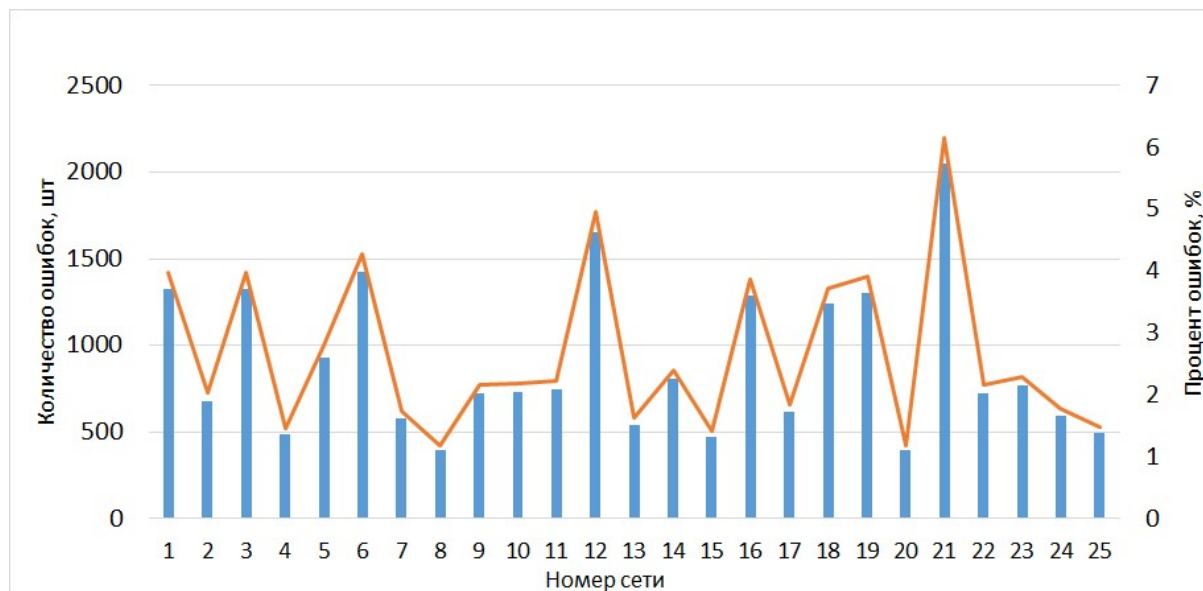


Рис. 69 – Результаты тестирования нейронных сетей с двумя скрытыми слоями

По результатам тестирования видно, что наихудший результат составил 2050 ошибок, а наилучший 392 ошибки. Архитектуры с двумя слоями показали себя лучше, при чём тенденции, выявленные для предыдущего эксперимента, сохраняются и в данной серии.

Далее тестировались архитектуры с тремя скрытыми слоями. Описание архитектур тестируемых сетей представлены в таблице 7.

Таблица 7

Номер сети	Нейронов в скрытом слое №1	Нейронов в скрытом слое №2	Нейронов в скрытом слое №3	Номер сети	Нейронов в скрытом слое №1	Нейронов в скрытом слое №2	Нейронов в скрытом слое №3
1	100	100	100	14	3500	3500	5000
2	400	100	100	15	10000	5000	3500
3	400	400	100	16	3500	5000	10000
4	400	400	400	17	800	400	200
5	100	400	400	18	200	400	800
6	100	100	400	19	1600	800	400
7	100	400	1000	20	400	800	1600
8	1000	400	100	21	3200	1600	800
9	3500	3500	3500	22	800	1600	3200
10	5000	3500	3500	23	2000	500	125
11	5000	5000	3500	24	125	500	2000
12	5000	5000	5000	25	4000	1000	250
13	3500	5000	5000				

Результаты тестирования представлены на рисунке 70.

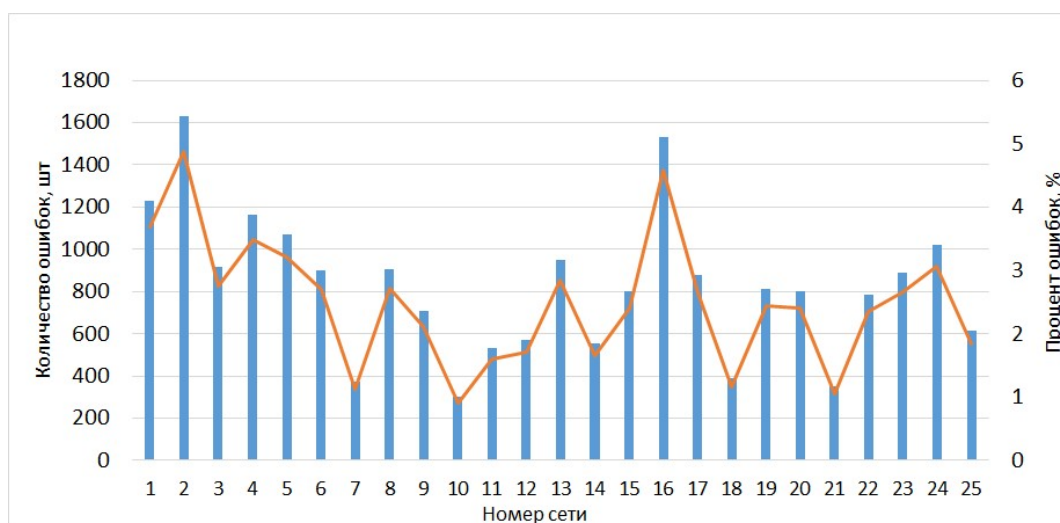


Рис. 70 – Результаты тестирования нейронных сетей с тремя скрытыми слоями

По результатам тестирования видно, что наихудший результат составил 1629 ошибок, а наилучший 304 ошибки. Несмотря на уменьшение количества некорректных распознаваний, дальнейшее увеличение количества скрытых слоёв нецелесообразно ввиду того, что прирост корректных распознаваний значительно замедлился, а время на идентификацию начинает существенно замедлять работу всей системы.

Нейронные сети прямого распространения в целом показали себя не удовлетворительно, ошибка идентификации варьируется от одного, до шести процентов относительно объёма тестовой выборки. Данный факт скорее всего связан с плохим качеством изображений из-за множества шумовых признаков, мешающими определить класс объекта. Следовательно, целесообразно выполнить эксперименты со свёртовыми нейронными сетями.

3.3.2 Тестирование свёрточных нейронных сетей

В качестве архитектур свёрточных нейронных сетей были взяты 6 архитектур с одним скрытым слоем и 36 с двумя свёрточными слоями. Размер карт свёртки был принят кратным степеням двойки. Роли слоёв классификации выступают два полносвязных слоя с 1000 и 600 слоями соответственно. Описание архитектур тестируемых сетей представлены в таблице 8.

Таблица 8

Номер сети	Слой свёртки №1	Слой свёртки №2	Слой свёртки №3	Слой свёртки №4	Нейронов в скрытом слое №1	Нейронов в скрытом слое №2
1	8	8	-	-	1000	600
2	8	8	8	8	1000	600
3	8	8	16	16	1000	600
4	8	8	32	32	1000	600
5	8	8	64	64	1000	600
6	8	8	128	128	1000	600
7	8	8	256	256	1000	600
8	16	16	-	-	1000	600
9	16	16	8	8	1000	600
10	16	16	16	16	1000	600
11	16	16	32	32	1000	600
12	16	16	64	64	1000	600
13	16	16	128	128	1000	600
14	16	16	256	256	1000	600
15	32	32	-	-	1000	600
16	32	32	8	8	1000	600
17	32	32	16	16	1000	600
18	32	32	32	32	1000	600
19	32	32	64	64	1000	600

20	32	32	128	128	1000	600
21	32	32	256	256	1000	600
22	64	64	-	-	1000	600
23	64	64	8	8	1000	600
24	64	64	16	16	1000	600
25	64	64	32	32	1000	600
26	64	64	64	64	1000	600
27	64	64	128	128	1000	600
28	64	64	256	256	1000	600
29	128	128	-	-	1000	600
30	128	128	8	8	1000	600
31	128	128	16	16	1000	600
32	128	128	32	32	1000	600
33	128	128	64	64	1000	600
34	128	128	128	128	1000	600
35	128	128	256	256	1000	600
36	256	256	-	-	1000	600
37	256	256	8	8	1000	600
38	256	256	16	16	1000	600
39	256	256	32	32	1000	600
40	256	256	64	64	1000	600
41	256	256	128	128	1000	600
42	256	256	256	256	1000	600

Результаты тестирования представлены на рисунке 71.

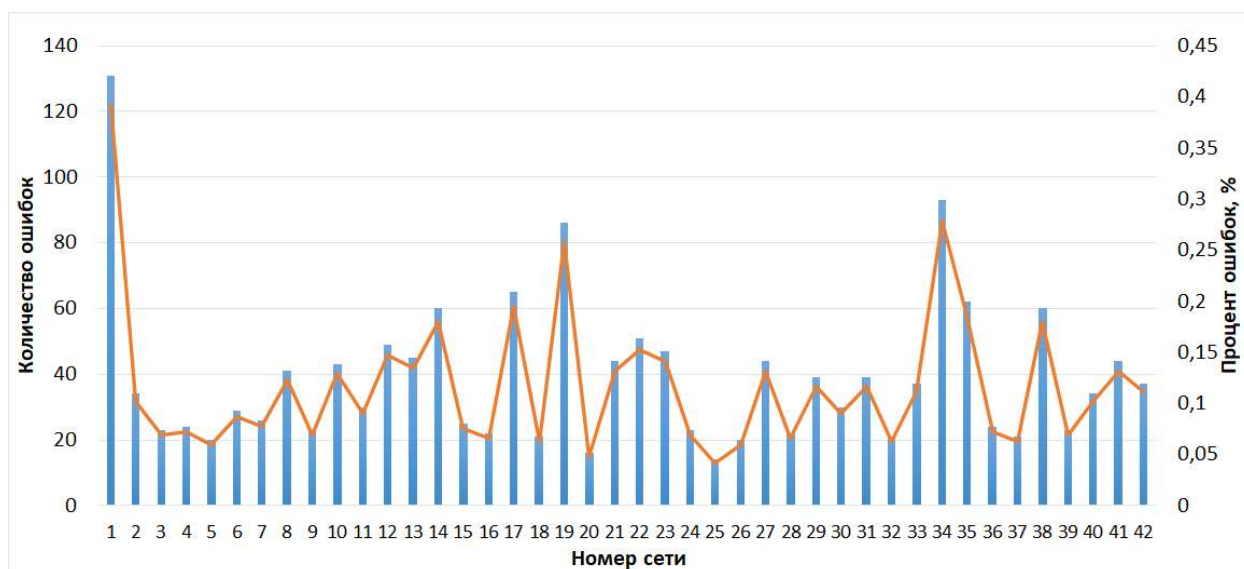


Рис. 71 – Результаты тестирования свёрточных нейронных сетей

Свёрточная нейронная сеть показала значительно лучшие результаты, так наихудший результат составил 131 ошибку, а лучший 14 ошибок. Следовательно, в разрабатываемой надстройке целесообразно использовать именно этот класс нейронных сетей.

3.3.3 Тестирование нейронных сетей на двух классах

Использование свёрточных нейронных сетей позволило сократить ошибку идентификации до 0,042%. Но анализируя ошибки, допускаемые при распознавании, выяснилось, что часть классов нейронная сеть путает чаще чем остальные. А именно такие классы как 0 и 1, 2 и 7, 3 и 5. Скорее всего это обоснованно тем, что эти классы визуальны похожи друг на друга и за счёт нахождения нейронной сетью схожих признаков происходит ошибка при их идентификации.

Для уменьшения влияния данного фактора, обучили ещё три нейронных сети, но уже не для десяти классов, а для двух, для тех, что сети часто путают между собой.

В качестве архитектур использовались архитектуры из предыдущих экспериментов, за исключением того, что на выходе производится прогноз для двух классов, а не для десяти.

Результаты тестирования представлены в таблицах 9-11.

Таблица 9

Номер сети	Ошибка для "0"	Ошибка для "1"	Сумма ошибок	Номер сети	Ошибка для "0"	Ошибка для "1"	Сумма ошибок
1	9	1	10	22	12	2	14
2	0	11	11	23	4	18	22
3	0	3	3	24	0	1	1
4	0	3	3	25	1	26	27
5	2	3	5	26	2	4	6
6	0	3	3	27	0	5	5
7	12	4	16	28	0	4	4
8	2	3	5	29	3	4	7
9	0	1	1	30	1	2	3
10	3	6	9	31	9	3	12
11	1	2	3	32	0	2	2
12	12	14	26	33	41	2	43
13	1	11	12	34	0	9	9
14	26	2	28	35	0	5	5
15	3	3	6	36	5	8	13
16	0	5	5	37	0	3	3
17	2	1	3	38	9	3	12
18	1	2	3	39	0	8	8

19	0	2	2	40	1	5	6
20	10	5	15	41	3	3	6
21	1	3	4	42	0	4	4

Таблица 10

Номер сети	Ошибка для "2"	Ошибка для "7"	Сумма ошибок	Номер сети	Ошибка для "2"	Ошибка для "7"	Сумма ошибок
1	22	2	24	22	0	1	1
2	1	5	6	23	0	3	3
3	0	0	0	24	0	0	0
4	0	0	0	25	0	1	1
5	0	0	0	26	0	0	0
6	0	3	3	27	0	21	21
7	19	1	20	28	0	2	2
8	0	4	4	29	0	1	1
9	0	1	1	30	1	0	1
10	0	4	4	31	0	3	3
11	2	0	2	32	0	2	2
12	20	4	24	33	3	0	3
13	3	9	12	34	0	1	1
14	0	9	9	35	0	13	13
15	1	1	2	36	3	4	7
16	3	3	6	37	15	20	35
17	1	0	1	38	0	0	0
18	12	0	12	39	0	1	1
19	0	0	0	40	0	1	1
20	0	0	0	41	0	6	6
21	0	0	0	42	2	10	12

Таблица 11

Номер сети	Ошибка для "3"	Ошибка для "5"	Сумма ошибок	Номер сети	Ошибка для "3"	Ошибка для "5"	Сумма ошибок
1	5	3	8	22	3	11	14
2	3	1	4	23	27	4	31
3	12	1	13	24	6	4	10
4	2	1	3	25	4	3	7
5	1	7	8	26	9	2	11
6	4	14	18	27	5	2	7
7	14	2	16	28	3	4	7
8	7	1	8	29	8	4	12
9	8	3	11	30	15	20	35
10	4	0	4	31	5	4	9
11	7	2	9	32	1	3	4
12	5	5	10	33	3	2	5
13	4	5	9	34	7	5	12
14	14	7	21	35	4	7	11

15	6	2	8	36	9	3	12
16	12	2	14	37	4	3	7
17	13	0	13	38	4	2	6
18	3	8	11	39	3	2	5
19	5	3	8	40	3	11	14
20	3	2	5	41	4	3	7
21	3	1	4	42	1	2	3

Анализируя полученные данные, видно, при обучении сети на двух классах, практически исключается вероятность ошибки. Так для классов «0» и «1» наилучший результат составил одну ошибку при объёме тестовой выборки 27044 элементов, для классов «2» и «7» наилучший результат составил ноль ошибок при объёме тестовой выборки 7210 элементов, для классов «3» и «5» наилучший результат составил три ошибки при объёме тестовой выборки 13128 элементов.

3.4 Алгоритм повышенной эффективности идентификации кодов литых заготовок

В ходе тестирования нейронных сетей была предложена следующая схема обновлённой системы распознавания клейма.

Согласно представленной на рисунке 72 схеме результат основного нейросетевого детектора (ОНД) анализируется на пропущенные элементы посредством быстрого нейросетевого детектора (БНД). Далее выделенные рамки, в которых потенциально могут быть пропущенные элементы подаются на ансамбль малых нейронных сетей (АМНС), который и производит корректировку конечного результата.

Согласно разработанному алгоритму обновленной системы распознавания, приведенному на рисунке 73, изображение поступает на основной детектор.

ОНД обрабатывает изображение и выделяет на нём области, содержащие интересующие объекты. Для каждого объекта определяются координаты, что позволяет точно указать его местоположение на

изображении. На выходе ОНД формируется результат в виде списка объектов с их координатами.

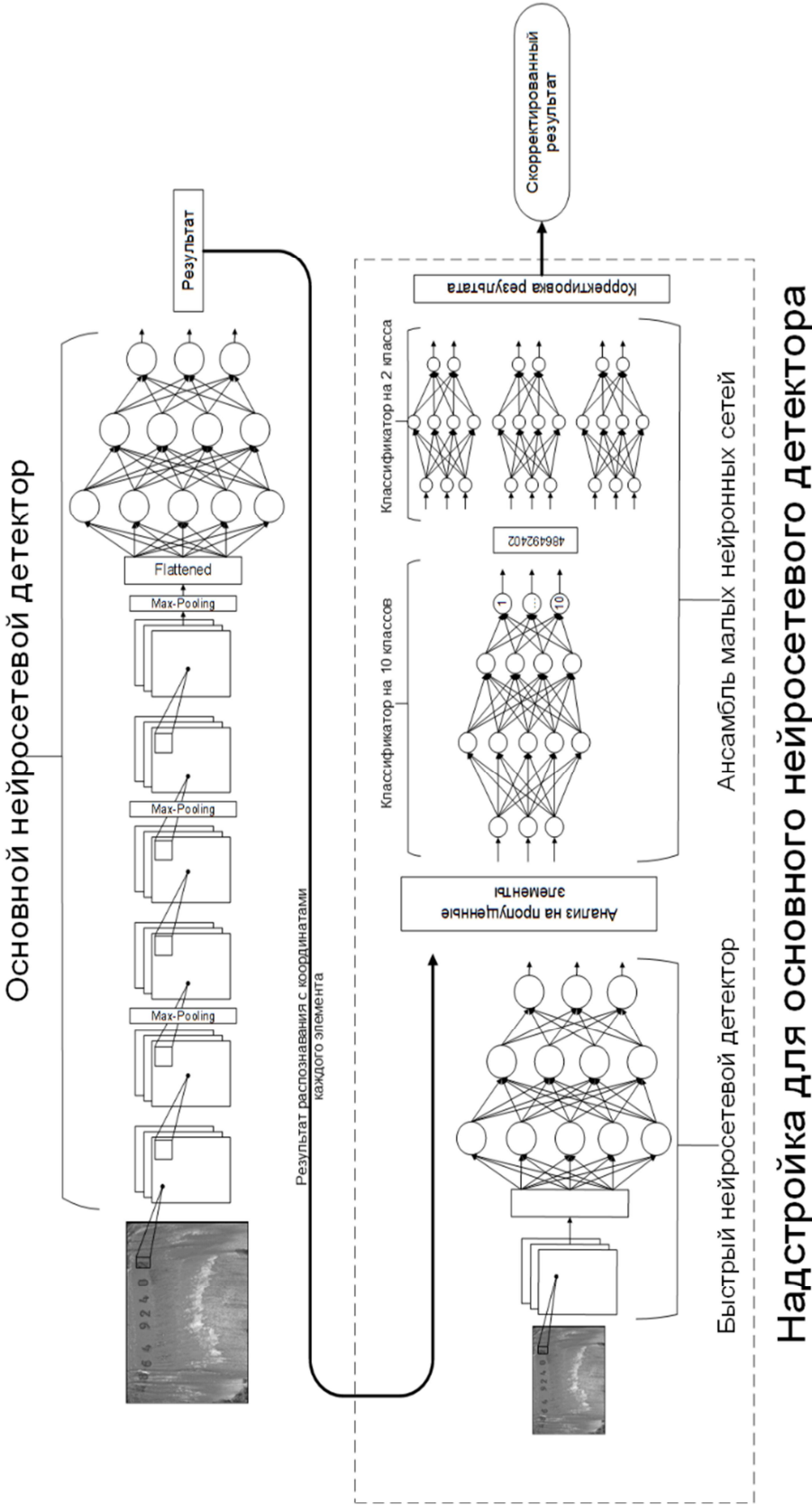


Рис. 72 – Схема обновлённой системы распознавания

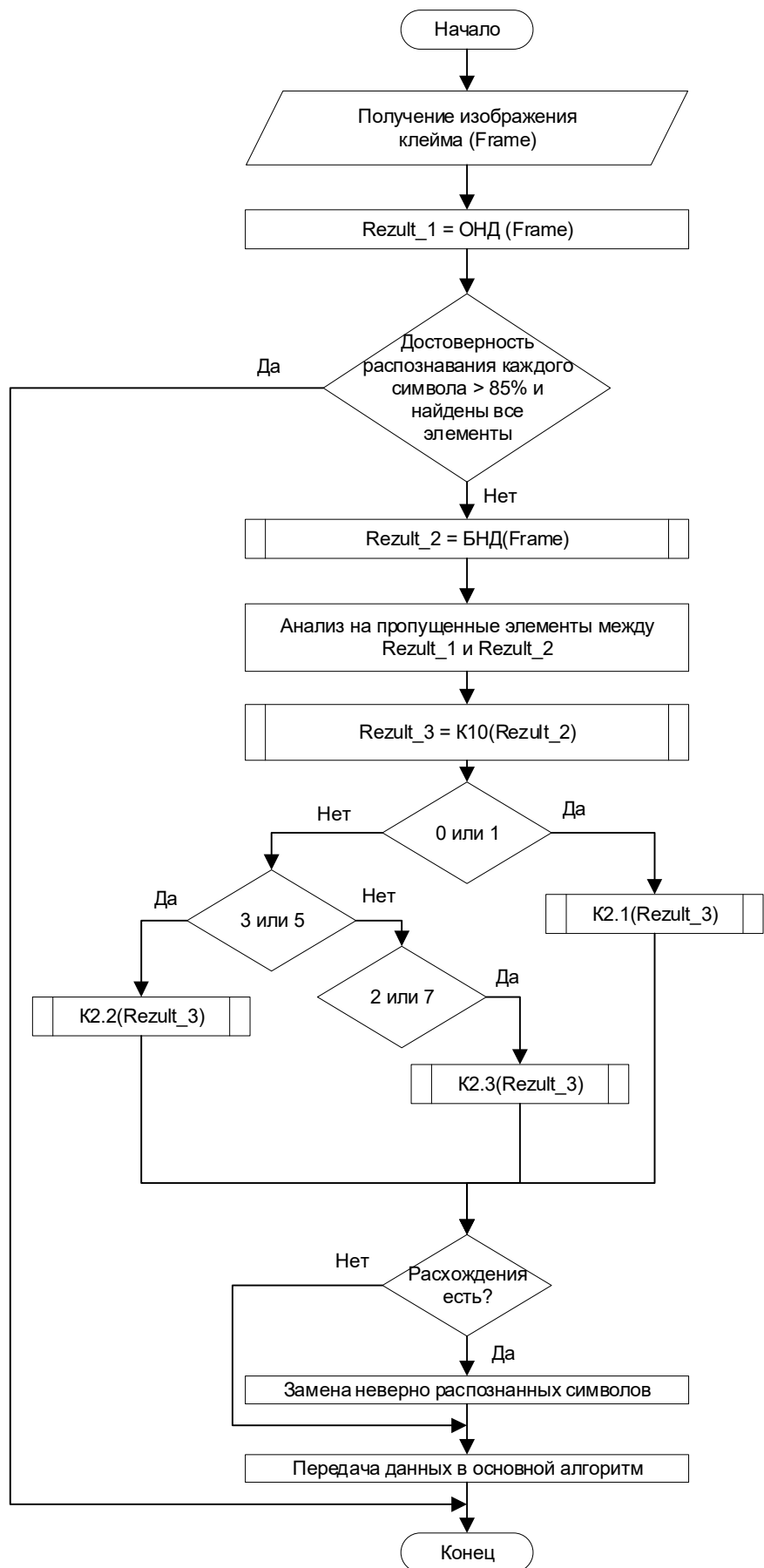


Рис. 73 – Алгоритм уточнения кода клейма

В то же время БНД анализирует это же изображение и выделяет потенциальные области, которые могут содержать объекты. Результаты работы ОНД и БНД сравниваются на возможность наличия пропуска искомым объектов или некорректной классификации выделенных. Выделенные области, в которых содержатся найденные - с низкой степенью уверенности ($<85\%$), или пропущенные объекты поступают на ансамбль малых нейронных сетей, в котором выделенные области подаются на нейронную сеть прямого распространения (K10), где производится первичная классификация, а потом в зависимости от результата на один из бинарных нейросетевых классификаторов (K2.1, K2.2, K2.3) для корректировки результата. В итоге, если какой-либо из объектов был пропущен или неверно классифицирован, производится корректировка результата работы ОНД.

3.5 Результаты работы

Для тестирования обновлённой системы была использована та же тестовая выборка, что и в других экспериментах.

По результатам работы алгоритма, прирост в результативности за счёт используемых в данной главе методов, составил 3,23%.

4. Выводы по главе 3

В третьей главе была выполнена работа по повышению эффективности распознавание кодов литых заготовок путем совершенствования алгоритмов обработки изображений клейма заготовки.

Посредством методов аугментации удалось улучшить набор данных, по следующим параметрам:

- 1) Значительное увеличения объёма изображений в наборе
- 2) Нормализация элементов по классам
- 3) Увеличение вариативности признаков содержащихся в наборе элементов

Также за счёт надстройки к уже разработанному алгоритму удалось повысить распознавание наиболее подверженных деформации клейм. Несмотря на кажущимся малым прирост, стоит отметить, что распознанные данной надстройкой имеют сильно деформированные элементы клейма и задача их распознавания значительно сокращает участие оператора поста контроля в проблеме идентификации заготовки.

Оставшиеся же нераспознанные клейма имеют деформацию, которая настолько искажает изображение, что идентифицировать код можно только путём дополнительной отчистки.

Глава 4. Апробация разработанной системы распознавания кода клейма литой заготовки

4.1. Техническое обеспечение системы

При реализации и внедрении (приложение 2) в опытно-промышленную эксплуатацию системы распознавания кода клейма используются, как вновь приобретенные технические средства, так и уже присутствующие на объекте.

Так для реализации проекта закуплено техническое обеспечение, представленное в таблице 12.

Таблица 12

№	Наименование оборудования
1	Видеокамера Optimus IP-P082.1 (10x)
2	Системный блок в сборе: ASUS TUF B360-PLUS / Intel Core i5 8500 / DDR4 8192Mb / SSD WD 240Gb M.2 / NVidia ASUS 4096Mb GTX1050TI / 600W
3	Видеокарта Asus PCI-E 4.0 ROG-STRIX-RTX2060S-O8G-EVO-GAM nVidia GeForce RTX 2060SUPER 8192Mb 256bit GDDR6 1470/14000/HDMIx2/DPx2/Type-Cx1/HDCP Ret
4	Видеокамера Axis Q1645-LE

Видеокамера (поз. 1 таблице 12) необходима для непосредственного контроля за приходящими заготовками и передачи полученного видеопотока для детектирования клейм на рабочую станцию, установленную в машзале K1 сортопрокатного цеха №1.

Рабочая станция (поз. 2 таблицу 12), собранная для выполнения детектирования клейма, имеет в своем составе видеокарту NVidia ASUS 4096Mb GTX1050TI. В процессе выполнения работ было установлено, что технические характеристики данной карты не способны обеспечивать выполнение поставленной задачи в заданные временные ограничения. В этой связи была закуплена более мощная видеокарта (поз. 3 таблице 12) и установлена в рабочую станцию вместо базовой карты NVidia ASUS 4096Mb

GTX1050TI. Она смогла обеспечить выполнение задачи детектирования в заданные временные ограничения: менее 2 сек на обработку одного изображения.

На рисунке 74 показана схема комплекса технических средств решаемой задачи.

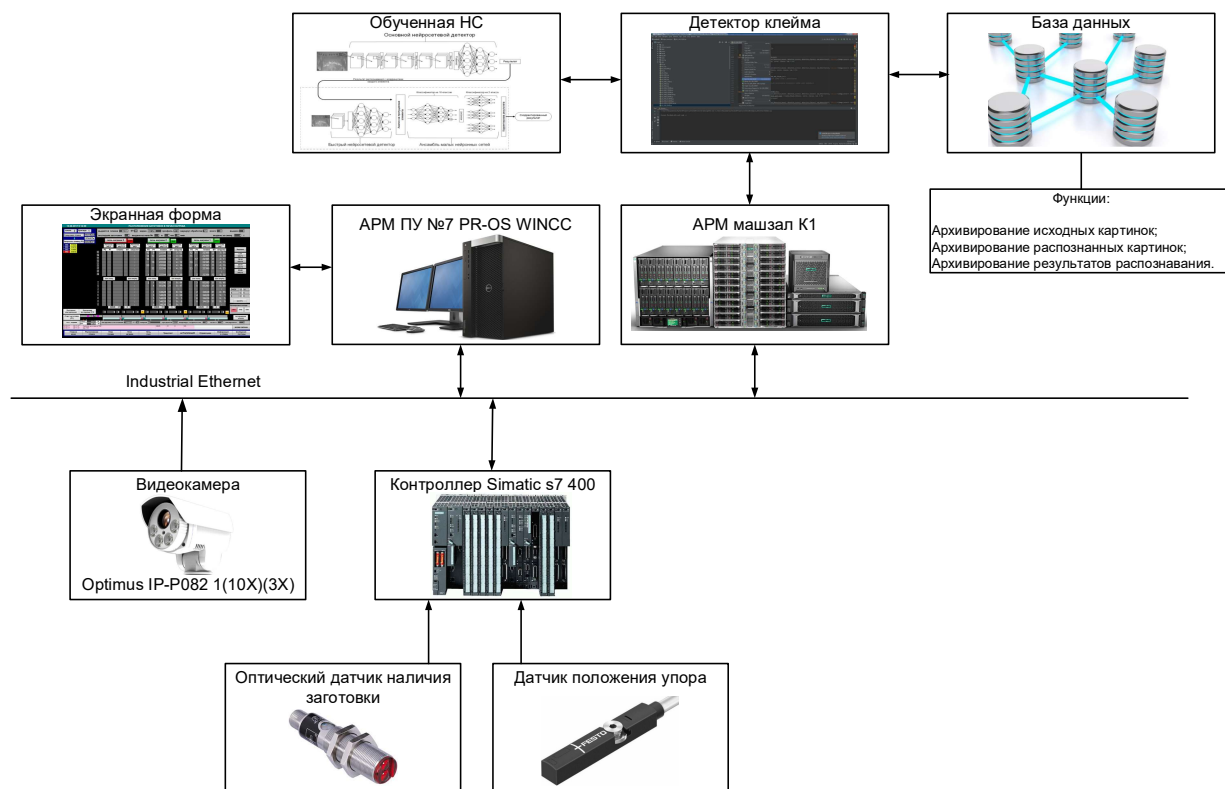


Рис. 74 – Схема комплекса технических средств

Из схемы видно, что система детектирования клейма взаимодействует с системой управления комбината, используя для своей работы два сигнала: дискретный сигнал наличия заготовки на конвейере перед упором и дискретный сигнал положения упора. При этом в существующую АС УТП была добавлена рабочая станция с разработанным программным комплексом и установлена IP-камера.

Первоначально в системе предполагалось использовать детектирование наличия заготовки по анализу видеоряда, приходящего с камеры. В процессе анализа видеоряда был замечен эффект, что приходящая заготовка отражает свет от лампы дополнительного освещения и засвечивает изображение на камере, создавая мощный световой импульс. Именно на данный импульс,

диагностируемый с помощью эмпирически определенного порога световой интенсивности, ориентировалась ранняя версия детектора. Однако, при интеграции ПО детектора в АСУТП комбината и первичных пусконаладочных работах оказалось, что данный подход работает с задержками, иногда приводящими к пропуску заготовок. Также присутствовала реакция на подъем упора при отсутствии заготовки, что вызывало лишний запуск программы распознавания. Если темп стана был достаточно высоким, то, как раз из-за данного ложного срабатывания на движение упора, программа детектор могла пропустить пришедшую в данный момент заготовку. Вследствие указанных недостатков от данного подхода было принято решение отказаться в пользу анализа сигналов с датчиков наличия заготовки и состояния упора.

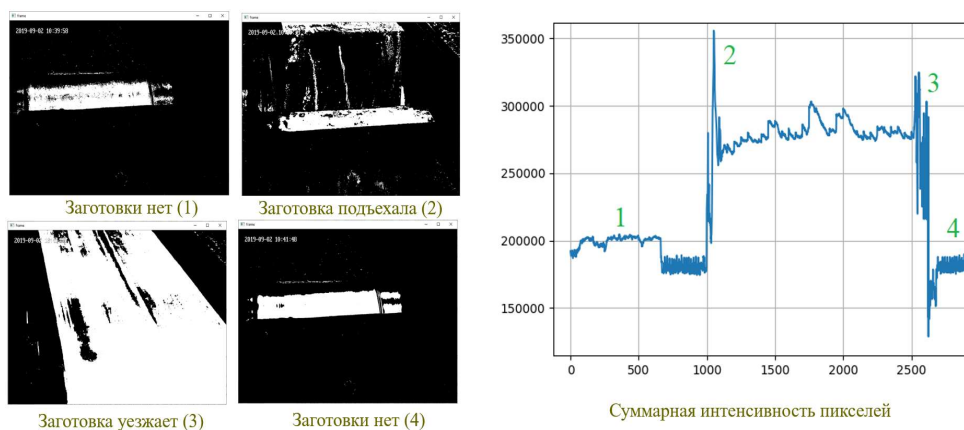


Рис. 75 – Пороговая световая интенсивность цикла детектирования заготовки

Совокупность данных сигналов используется для запуска системы распознавания: при наличии заготовки на рольганге и при опущенном упоре (дискретный сигнал нижнего положения) программой-детектором производится снимок торца заготовки (рисунок 75) с последующим распознаванием снимка с целью выделения кода клейма.

Для получения снимка заготовки используется видеокамера Optimus IP-P082.1(10x). Она подключена к источнику питания DC 12В от 1А, который запитан от сети 220В. Для подключения камеры к локальной сети комбината используется восьмижильная витая пара (ее длина должна быть не более 90 м) с разъемами RJ-45.

4.2. Системное программное обеспечение

Для функционирования программной части системы распознавания клейма требуется рабочая станция с установленными версиями системного ПО. В качестве операционной системы используется Windows 10 Prof, на которую требуется установить:

1. Python-3.6.6
2. Visual C++ redistributable
3. Драйвера для видеокарты
4. CUDNN
5. CUDA
6. Ряд библиотек Python

Все программы, устанавливаемые на операционную систему Windows, находятся в свободном доступе, либо полностью бесплатны, либо имеют бесплатные версии и не требуют закупки лицензий. Бесплатные версии ПО, описываемые ниже и используемые для реализации программы системы распознавания клейма, имеют весь необходимый функционал для обеспечения его работы.

Список библиотек для установки:

1. tensorflow-gpu==1.12.0
2. keras==2.2.5
3. pillow==6.0.0
4. imageio==2.5.0
5. matplotlib==3.0.3
6. numpy==1.16.3
7. opencv-python==3.3.1.11
8. scikit-learn==0.21.3
9. scikit-image==0.15.0
10. pandas==0.25.3

4.2.1 Специальное программное обеспечение

Для выполнения задачи проекта по распознаванию клейма написана специальная программа системы распознавания клейма на языке Python 3.6. Данная программа позволяет взаимодействовать с цеховым контроллером Simatic S7, формировать результаты распознавания и сохранять их в файлы на жесткий диск, и собственно ядро системы - модуль, отвечающий за распознавание - «Нейросетевой детектор клейма».

«Нейросетевой детектор клейма» — это функция программы системы распознавания клейма, состоящая из двух нейронных сетей и логики формирования конечного результата в виде кода распознанного клейма литой заготовки.

Одна быстрая сеть класса `faster_rcnn_inception_v2_soco` со скоростью одного инференса 58 ms для определения положения машинописных цифр на торце заготовки (они могут быть перевернуты) и позиционирования фотографии (машинописные цифры должны читаться обычным способом – не перевернуты) для подачи на нейронную сеть, используемую для распознавания.

Одна сеть класса `faster_rcnn_inception_resnet_v2_atrous_soco` с временем инференса 620 ms натренирована на детектирование цифр, нанесенных машинописным способом.

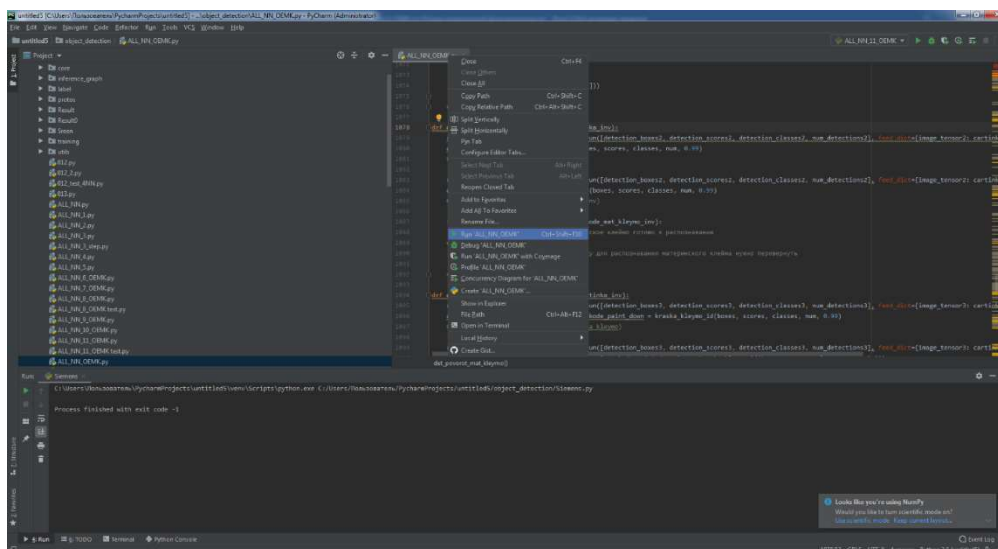


Рис. 76 – Запуск системы распознавания клейма

При первоначальном запуске программы системы распознавания клейма производится подключение к контроллеру и видеокамере.

Установление связи с видеокамерой осуществляется автоматически с параметрами, указанными в конфигурационном файле. В случае если подключиться не удастся или нарушается связь с видеопотоком в процессе работы системы распознавания клейма, то пользователю формируется сообщение в информационное поле IDE PyCharm:

‘Пропала связь с видеопотоком камеры’

Программа системы распознавания клейма автоматически пытается установить связь с камерой и когда ей это удастся вывод надписи о отсутствии связи прекращается.

В результате экспериментов установлено, что связь с первого раза может не установиться, но может установиться со второй и очень редко с третьей попытки. Для автоматизации установления связи с контроллером, чтобы не перезапускать программу вручную в системе распознавания клейма добавлена опция, по которой, в случае отсутствия связи с контроллером, программа самостоятельно, не «уходя в ошибку», пытается установить связь еще раз. Таких попыток предусмотрено 4, после чего, если все же связь не установлена, программа завершает свою работу.

При установлении связи пользователю формируется сообщение в информационное поле IDE PyCharm о номере попытки из 4-х и об ее успешности:

‘Номер попытки подключения к контроллеру: <номер фактической попытки> из 4-х’

В случае установления связи:

‘Связь с контроллером установлена’

В случае отсутствия связи с контроллером:

‘Нет связи с контроллером’

И если за 4 попытки связь установить не удалось, то выводится сообщение:

'Не удалось установить связь с контроллером'

'Повторите запуск программы'

После чего программа завершает свою работу.

Программа системы распознавания клейма после обработки изображения формирует выход в виде кода распознанного клейма, который пересылается в память цехового контроллера Simatic S7 в специализированную область памяти DB<адрес> (по умолчанию DB97). Запись кода осуществляется посимвольно, каждая цифра клейма в свою ячейку памяти блока DB<адрес>, начиная с начального адреса (по умолчанию DB97.DBW4). В блоке DB<адрес> код занимает девять двухбайтовых ячеек типа int и соответственно 18 байт памяти.

Также программа передает в контроллер еще два параметра – это признак распознавания клейма и тактовый импульс. Значение первого записывается по адресу DB97.DBW2, а второго DB97.DBW50.

Признак распознавания клейма используется как идентификатор полноты клейма:

0 – ожидание результата распознавания. Формируется, если длина кода = 0, как правило, когда камера сделала снимок без заготовки,

1 – клеймо состоит из девяти цифр, «успешное чтение»,

2 – клеймо состоит менее чем из девяти цифр ($1 \leq \text{размер клейма} \leq 8$), «сбой/неуспешное чтения».

Данный параметр передается в контроллер после каждого распознавания снимка с камеры «Нейросетевым детектором клейма», выдерживается в течении 2 сек, а затем устанавливается в «0».

Тактовый импульс формируется в программе с дискретностью две секунды и предназначается для проверки связи системы распознавания клейма с промышленным контроллером Simatic S7. Если в течении 60 сек. не поступает ни одного импульса, то связь признается нарушенной (данная функция производится средствами контроллера и в системе распознавания клейма не реализовывалась). Вследствие того, что функция тактирования

встроена в основной алгоритм «Нейросетевого детектора клейма», возможны незначительные задержки в формировании импульсов, связанные с выполнением основных функций распознавания. В таком случае импульсы интегрируются в буфере памяти и после завершения распознавания пересылаются в контроллер.

Из контроллера программа системы распознавания клейма производит считывание признака наличия заготовки из адреса DB97.DBW46. При наличии переднего фронта импульса по данному сигналу запускается выполнение «Нейросетевого детектора клейма». Как уже отмечалось, для запуска системы распознавания используются сигналы наличия заготовки на рольганге и состояния упора (опущен) (сбор и анализ данных сигналов производится средствами контроллера и в системе распознавания клейма не реализовывался).

Также одним из выходов программы системы распознавания клейма являются два файла:

1. С фактическим изображением заготовки с камеры, для которого выполняется распознавание «Нейросетевым детектором клейма», пример на рисунке 77. Файл записывается в каталог:

C:\Users\Dmitry\PycharmProjects\untitled\Result\<имя_файла.jpg>.

Формат имени файла следующий:

REZ_041720074404_isx_410384402.jpg.

Здесь после слова «REZ_» указывается системное время в момент записи: 04 – месяц, 17 – день, 20 – год, 07 – час, 44 – минута, 02 – секунда (на указанной в примере фотографии наблюдается присутствующая рассинхронизация в 21 сек. между временем в камере и операционной системой компьютера). После слова «_isx_» указывается распознанный «Нейросетевым детектором клейма» код.



Рис. 77 – Фактическое изображение заготовки с камеры

2. С изображением результатов работы «Нейросетевого детектора клейма» по клейму с машинописными цифрами, пример на рисунке 78. Здесь распознанные цифры машинописного клейма обведены квадратами разных цветов с указанием достоверности, сформированной нейронной сетью, принадлежности распознанной цифры к соответствующему классу (классов 10 по числу цифр от 0 до 9).



Рис. 78 – Изображение с детектированием машинописных цифр клейма

Файл записывается в каталог:

C:\Users\Dmitry\PycharmProjects\untitled\Result\<имя_файла.jpg>.

Формат имени файла следующий:

REZ_041720074404_kod_410384402.jpg.

Здесь после слова «REZ_» указывается системное время в момент записи: 04 – месяц, 17 – день, 20 – год, 07 – час, 44 – минута, 02 – секунда (на указанной в примере фотографии наблюдается присутствующая рассинхронизация в 21 сек. между временем в камере и операционной системой компьютера). После слова «_kod_» указывается распознанный «Нейросетевым детектором клейма» код.

4.2.2 АСУТП «Система слежения за металлом на участке печей СПЦ-1 PR-OS».

Система PR-OS это основной инструмент слежения за материалом на участке печей, с которой непосредственно взаимодействует оператор ПУ-7.

Система «Распознавания клейма» интегрирована в PR-OS как система поддержки принятия решения – результат распознавания от «системы распознавания клейма» передается в PR-OS, где происходит сравнение номера плавки на посадке с распознанным системой номером. Для SCADA системы PR-03 в рамках внедрения и для помощи оператору по результатам работы системы распознавания кода клейма разработаны экранные формы с рекомендуемыми действиями. Здесь учтены все возможные ситуации, которые могут случиться на посту оператора при управлении движением заготовками с точки зрения контроля кода клейма литой заготовки.

По результату сравнения система формирует для оператора следующие сообщения:

А) При неуспешном распознавании (система распознавания клейма не смогла в виду различных причин прочитать 9 цифр номера плавки и заготовки) оператору выводится сообщение «Смотри сам».



Рис.79 – Сообщение «Смотри сам»

Б) При успешном распознавании и несовпадению прочитанного номера плавки с плавкой на посадке по системе PROS, система выдаст оператору предупреждающее сообщение «Другая плавка».



Рис.80 – Сообщение «Другая плавка»

В) При успешном распознавании и несовпадении прочитанного номера плавки с плавкой на посадке по системе PROS, система проводит сравнение на номер следующей плавки (назначенную ПУ-4). В случае совпадения, оператору будет выведено сообщение «Заготовка из следующей плавки».



Рис.85 – Сообщение «Заготовка из следующей плавки»

Г) В случае совпадения плавки на посадке с распознанным номером, система произведет поиск номера заготовки в списке заготовок на посадке. В случае успешного нахождения номера заготовки, система выдаст сигнал оператору «ОК» и подсветит зеленым распознанный номер заготовки.

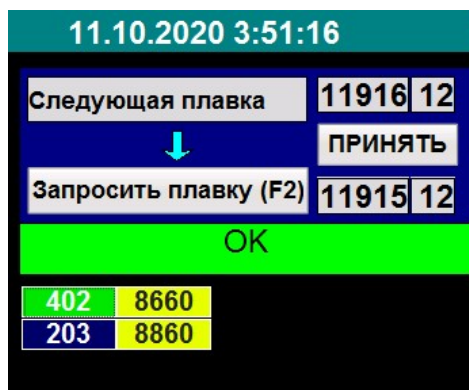


Рис.86 – Сообщение «ОК» и подсвеченная заготовка

При этом, задача квитирования (подтверждения) номера заготовки с оператора не снимается. Система используется как система поддержки принятия решения, предупреждая оператора о возможных сбоях и облегчая выбор заготовки в случае успешного чтения. Основной функционал и технологическая инструкция оператора остались без изменений.

4.3 Итоговый алгоритм работы системы

Камера видеонаблюдения передаёт данные в OpenCV для предварительной обработки, обработанные данные передаются в нейросетевой модуль для детектирования клейма с использованием нейронных сетей глубокого обучения и ансамбля малых нейронных сетей. Результаты детектирования клейма литой заготовки передаются на контроллер, с которого информация поступает в SCADA-систему и передаётся в цеховую систему управления производственными процессами. PyCharm и 7SNAP обеспечивают разработку и интеграцию всех компонентов системы [76-79].

Разработанная система представляет собой многоуровневый программно-аппаратный комплекс, функционирующий в режиме реального времени. Общая схема системы (рис. 87) отражает последовательную цепочку

обработки данных от момента захвата изображения до передачи управляющего сигнала в производственную инфраструктуру.

Первым звеном схемы выступает камера видеонаблюдения, установленная над технологической линией. Она обеспечивает непрерывный захват видеопотока, в котором присутствуют литые заготовки с нанесёнными клеймами. Выбор камеры выполнен с учётом требований к разрешению, частоте кадров и условиям освещения производственного помещения.

Полученный видеокادر поступает в модуль предварительной обработки, реализованный средствами библиотеки OpenCV. На данном этапе выполняются следующие операции:

- коррекция яркости и контрастности для компенсации неравномерного освещения;
- удаление шумов с использованием фильтров (например, двустороннего или медианного);
- геометрическая нормализация и приведение изображения к единому масштабу, заданному требованиями входного слоя нейронной сети;
- выделение области интереса (ROI) для сокращения вычислительной нагрузки на последующие этапы.

Далее обработанное изображение направляется в нейросетевой модуль, который является ядром всей системы. Этот модуль включает в себя два компонента:

1. Основной детектор на основе глубокой свёрточной нейронной сети, задачей которого является обнаружение клейма на изображении заготовки и определение его координат (ограничивающего прямоугольника).
2. Ансамбль малых нейронных сетей, предназначенный для классификации распознанного символа или кода клейма. Использование ансамбля позволяет повысить надёжность классификации.

Результатом работы нейросетевого модуля является структурированная информация: координаты клейма, его идентификационный код, а также степень уверенности в правильности детектирования.

Полученный результат передаётся на промышленный контроллер, который выполняет функцию связующего звена между системой машинного зрения и верхним уровнем управления. Контроллер осуществляет:

- проверку результата на достоверность;
- временную буферизацию данных для синхронизации с тактом технологического процесса;
- формирование стандартизированного протокола передачи.

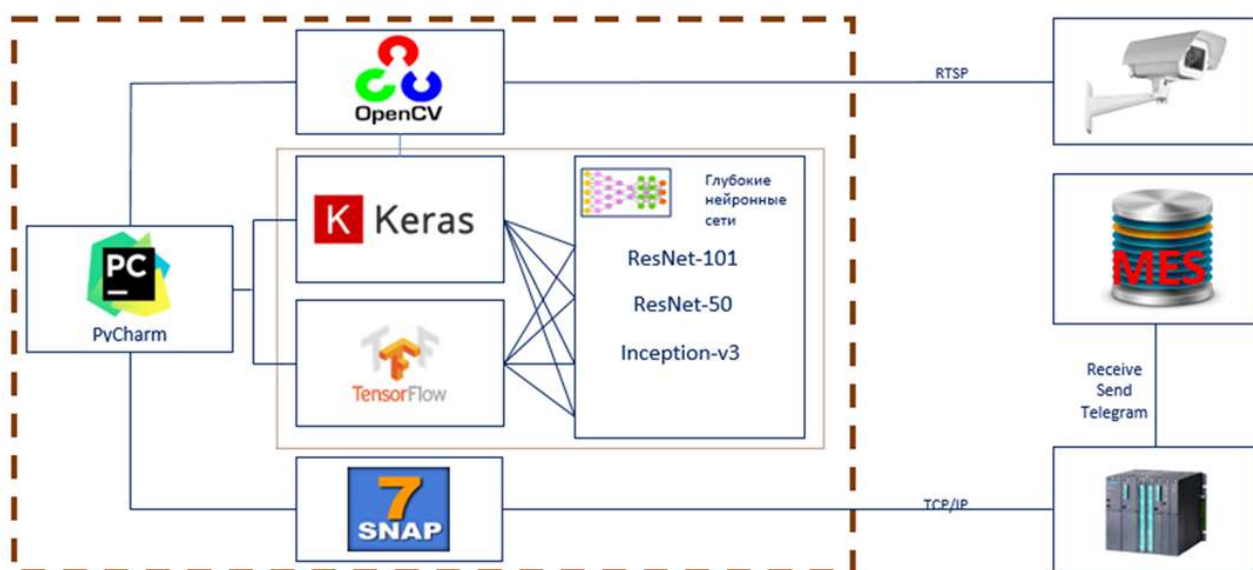


Рис. 87 – Разработанная система

Далее информация по промышленным сетям поступает в SCADA-систему, где отображается оператору в виде мнемосхем и журналов событий, а также передаётся в цеховую систему управления производственными процессами (MES/ERP). Это позволяет автоматически регистрировать факт обработки каждой заготовки, контролировать брак и вести статистику выпуска продукции.

Разработка и интеграция всех модулей осуществлялась с использованием средств PyCharm как основной IDE для написания кода на Python, а также платформы 7SNAP, которая обеспечивает визуальное проектирование логики управления, настройку потоков данных и бесшовную стыковку нейросетевых компонентов с промышленным оборудованием.

Итоговый алгоритм работы системы представлен на рисунке 88.

В начале алгоритма производится загрузка нейросетевого детектора и информация о количестве прошедших заготовок и сколько из них были распознаны некорректно.

Далее следует первый блок алгоритма, в котором основной нейросетевой детектор распознаёт код лито заготовки. В начале идёт проверка условий по наличию заготовки на рольганге и положению упора. Если заготовка присутствует, а упор опущен, то из видеопотока вычленяется кадр и подаётся на детектор. После нахождения цифр на изображении клейма литой заготовки производится проверка по ориентации заготовки и если выясняется, что она была перевёрнута, алгоритм переворачивает изображение и производит процедуру распознавания заново.

Далее следует второй блок алгоритма, в котором производится уточнение кода клейма. Уточнение кода клейма выполняется по средствам ансамбля малых нейронных сетей, согласно алгоритму представленному на рисунке 73.

В третьем блоке происходит формирование сообщения оператору. Так в зависимости от результата распознавания алгоритм может выдать следующие сообщения:

- Смотри сам;
- ОК;
- Другая плавка;
- Заготовка следующей плавки.

В четвертом блоке алгоритма производится расчёт погрешности работы системы распознавания. После квитирования заготовки оператором алгоритм добавляет единицу к переменной, отвечающей за количество прошедших заготовок. Но в ситуации, когда заготовка была распознана не корректно, производится добавление единицы к переменной, отвечающей за количество ошибочных распознавания, и производится расчёт процента ошибочных распознаваний. Так если величина ошибки превысит 5 процентов, оператору будет выведено сообщение о необходимости до обучения детектора.

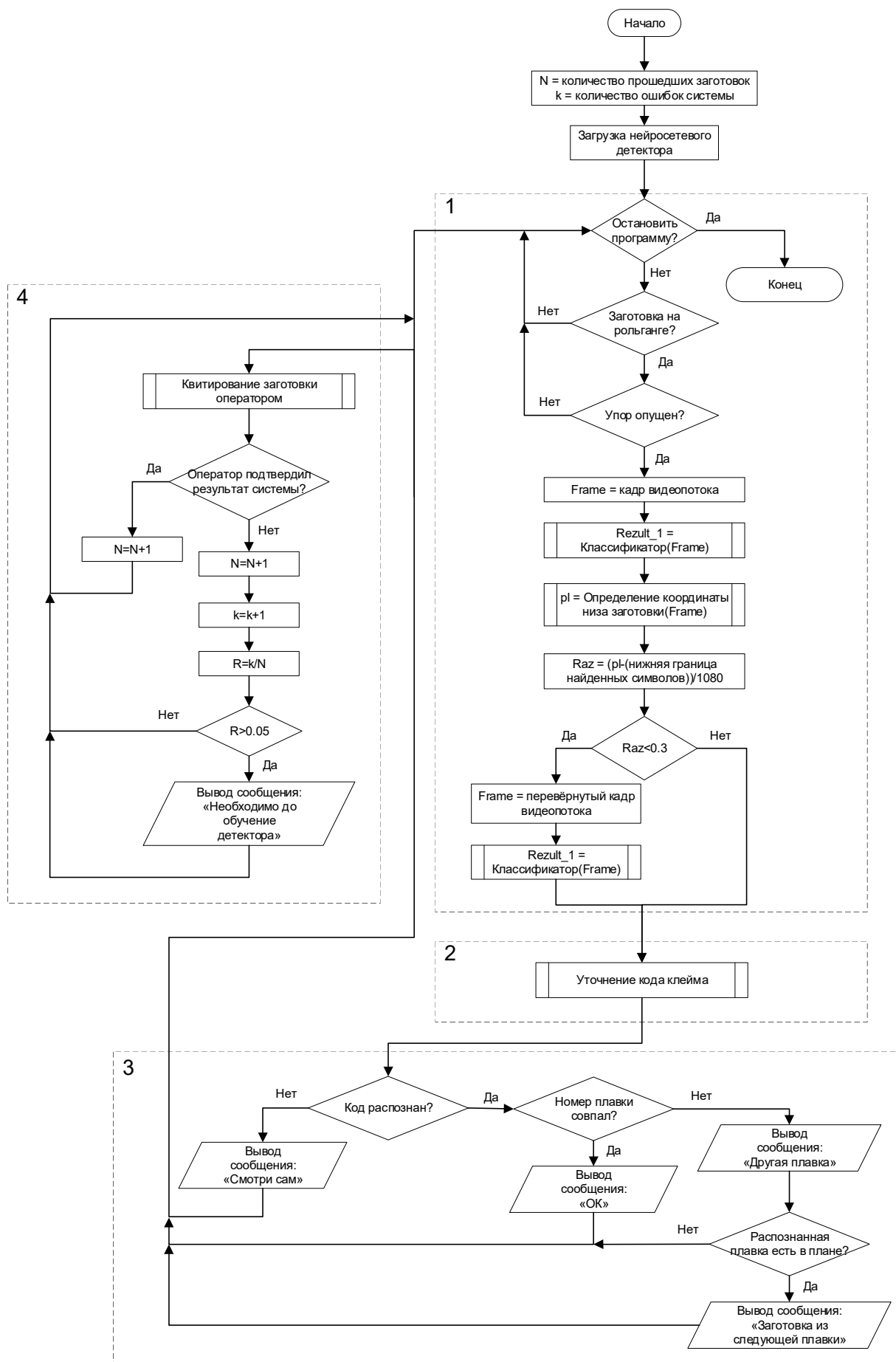


Рис. 88 – Алгоритм детектора

Сравнительная характеристика результативности подходов к распознаванию клейма литой заготовки представлена на рисунке 89.

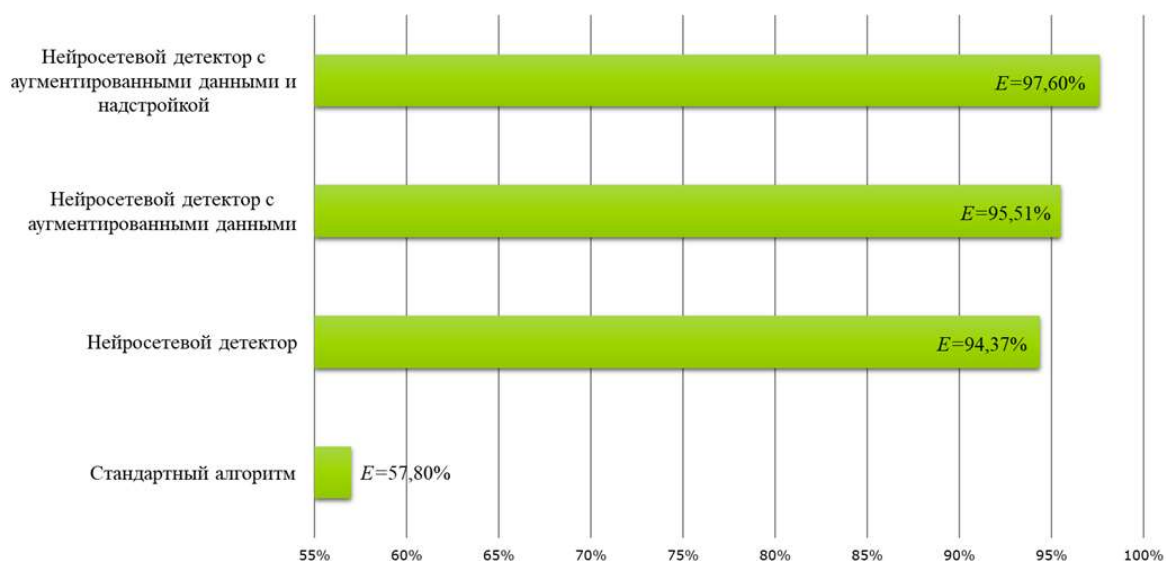


Рис. 89 – Результативность работы подходов к распознаванию кода клейма

4.4 Испытания системы распознавания клейма, интегрированного в АСУ СПЦ-1 АО «ОЭМК»

Начиная с 08.04.2020 года системы распознавания клейма была интегрирована в АСУТП СПЦ-1 силами работников комбината. Рабочая станция, установленная в машзале PRO помещения К1, принимала сигнал с видеокамеры обрабатывала его и выдавала в контроллер S7-400 системы PR-OS распознанный «Нейросетевым детектором клейма» девятизначный цифровой код.

В процессе интеграции была доработана программная составляющая систему распознавания клейма в соответствии с требованиями персонала. Они касались по большей части интерфейса взаимодействия с системой комбината.

Как уже было описано выше на данном этапе согласно требованиям заказчика, алгоритм запуска «Нейросетевого детектора клейма» на распознавание был заменен с подхода по определению момента прихода заготовки по анализу видеоряда на сигнал с контроллера службы АСУ комбината (опускание упора при наличии заготовки на рольганге).

За период 5 дней начиная с 17.04.2020 по 22.04.2020 было собрано и обработано 4027 фотографий. Из них 8.9 % (362 фотографий) признаны бракованными по различным причинам.

Среди оставшихся 3665 фотографий анализ выявил:

- прочитано 96.1 % (3522 клейма),
- не прочитано 3.9 % (143 клейма);

В большинстве случаев ошибочного распознавания «Нейросетевой детектор клейма» ошибочно интерпретирует одну – две цифры, но это не позволяет признать такое клеймо верным.

5. Выводы по главе 4

В четвертой главе был представлен отчёт о работе разработанного детектора в условиях реального производства. В рамках опытной эксплуатации разработанной системы было установлено, что она обеспечивает точность более 90% при распознавании кода клейма, при быстродействии 1,5 секунд на одно изображение.

Внедрение данной системы обеспечило положительный экономический эффект, за счет:

- Снижения вероятности смешивания металла и сопутствующих потерь.
- Создания системы контроля действий оператора.
- Снижения нагрузки на оператора.

Также были представлены финансовые затраты на материально техническую основу разработанной системы.

Заключение

В диссертационной работе удалось добиться повышения эффективности системы контроля за движением заготовок на посту управления металлургического предприятия за счет внедрения интеллектуальной системы поддержки принятия решений способной автоматически распознавать код клейма, нанесенного машинописным способом на литую заготовку.

В ходе достижения полученной цели в рамках исследования были решены следующие основные задачи:

1. Выполнен анализ и проведено исследование эффективности существующих наборов данных изображений цифр и методов распознавания образов на изображении.

2. Собран в промышленных условиях, аннотирован и расширен с применением методов аугментации собственный уникальный набор изображений цифр клейма литой заготовки.

3. Выполнен анализ и проведено исследование эффективности существующих архитектур искусственных нейронных сетей глубокого обучения для применения в реальных условиях для задач распознавания цифр кодов литых заготовок.

4. Разработан алгоритм автоматического распознавания машинописных цифровых кодов на литых заготовках для действующей АСУТП, использующий основной нейросетевой детектор и ансамбль малых нейронных сетей, обеспечивающий адаптивную поддержку его работы.

5. Разработана автоматизированная система поддержки принятия решений оператором, включающая интеллектуальное распознавание номеров заготовок, верификацию в базе данных цеховой АСУТП и выдачу рекомендаций в режиме реального времени.

6. Создано программное обеспечение для АСУТП, объединяющее контроль клейма литых заготовок, работу с контроллером, SCADA-системой

и модулем интеллектуального детектирования с поддержкой промышленных протоколов.

7. Проведена экспериментальная оценка эффективности контроля за движением литых заготовок на посту управления металлургического предприятия разработанной и внедренной системы поддержки принятия решений, способной автоматически распознавать код клейма, нанесенного машинописным способом.

Список использованной литературы

1. Marr D. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. — San Francisco: W.H. Freeman, 1982. — 397 p.
2. Poggio T., Torre V., Koch C. Computational vision and regularization theory // Nature. — 1985. — Vol. 317, No. 6035. — P. 314–319.
3. Marroquin J., Mitter S., Poggio T. Probabilistic Solution of Ill-Posed Problems in Computational Vision // Journal of the American Statistical Association. — 1987. — Vol. 82, No. 397. — P. 76–89.
4. Szeliski R. Computer Vision: Algorithms and Applications. — 2nd ed. — Cham: Springer, 2022. — 925 p.
5. Face Recognition Applications in AI // Ultralytics Blog. — 2025 [Электронный ресурс]. — URL: <https://www.ultralytics.com/ru/blog/facial-recognition-applications-in-ai> (дата обращения: 15.09.2026).
6. Szeliski R. Computer Vision: Algorithms and Applications. — 2nd ed. — Cham: Springer, 2022. — 925 p.
7. Gonzalez R.C., Woods R.E. Digital Image Processing. — 4th ed. — Pearson, 2018. — 1019 p.
8. Shapiro L.G., Stockman G.C. Computer Vision. — Upper Saddle River: Prentice Hall, 2001. — 580 p.
9. Liu L., Ouyang W., Wang X., Fieguth P., Chen J., Liu X., Pietikäinen M. Deep Learning for Generic Object Detection: A Survey // International Journal of Computer Vision. — 2020. — Vol. 128. — P. 261–318.
10. Paul M., Burlina P. Robustness and Adaptation to Hidden Factors of Variation // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). — 2022. — P. 1–10.
11. Bishop, C. M. *Pattern recognition and machine learning*. Information science and statistics. Softcover published in 2016. New York, NY: Springer, 2006.

12. Nielsen, M. A. *Neural networks and deep learning*. Vol. 2018. Determination press San Francisco, CA, USA: 2015.
13. HUANG, J., RATHOD, V., SUN, C., ZHU, M., KORATTIKARA, A. et al. Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). July 2017.
14. W. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics., vol. 5, pp. 115 133, 1940.
15. F. Rosenblatt, a probabilistic model for information storage and organization in the brain, Psychological Review., vol. 65, pp. 386 408, 1958.
16. P. J. Werbos, Beyond regression: New tools for prediction and analysis in the behavioral sciences., PhD thesis, Harvard University, 1974.
17. D. Kriesel, A Brief Introduction to Neural Networks. Germany: www.dkriesel.com, 2005.
18. C. Peterson and T. Rognvaldsson, An introduction to artificial neural networks, Proc. 1991 CERN Summer School of Computing, pp. 113 170, 1992.
19. Zeiler, M. D. and Fergus, R. Visualizing and Understanding Convolutional Networks. arXiv (2013).
20. Smith, S. W. CHAPTER 6 - Convolution. Digital Signal Processing. Ed. by S. W. Smith. Boston: Newnes, 2003, 107–122.
21. Nello Cristianini and John Shawe-Taylor. An introduction to support vector machines and other kernel-based learning methods. Cambridge university press, 2000.
22. Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467, 2016.

23. François Chollet. Keras (2015). URL <http://keras.io>, 2017.
24. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.-Y., Berg A.C. SSD: Single Shot MultiBox Detector // European Conference on Computer Vision (ECCV). — 2016. — P. 21–37.
25. imagenet. Imagenet Large Scale Visual Recognition Challenge 2016 (ILSVRC2016). <http://image-net.org/challenges/LSVRC/2016/results>. [Online; accessed 06-July-2019].
26. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. <http://www.deeplearningbook.org>.
27. Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A. Going Deeper with Convolutions // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2015.
28. Yangqing Jia Pierre Sermanet Scott E. Reed Dragomir Anguelov Dumitru Erhan Vincent Vanhoucke Christian Szegedy, Wei Liu and Andrew Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842, 2014.
29. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // Proceedings of the 32nd International Conference on Machine Learning (ICML). — 2015. — P. 448–456.
30. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR, abs/1502.03167, 2015.
31. Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z. Rethinking the Inception Architecture for Computer Vision // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2016.
32. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. CoRR, abs/1512.00567, 2015.

33. Chen-Yu Lee, Saining Xie, Patrick W. Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. CoRR, abs/1409.5185, 2014.
34. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.
35. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2016. — P. 770–778.
36. 11. Szegedy C., Ioffe S., Vanhoucke V., Alemi A.A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning // Proceedings of the AAAI Conference on Artificial Intelligence. — 2017. — Vol. 31, No. 1. — P. 4278–4284.
37. Christian Szegedy, Sergey Io_e, and Vincent Vanhoucke. Inception-v4, inceptionresnet and the impact of residual connections on learning. CoRR, abs/1602.07261, 2016.
38. He, K.; Zhang, X.; Ren, S.; aj.: Deep Residual Learning for Image Recognition. 2015, arXiv:1512.03385.
39. Szegedy, C.; Ioffe, S.; Vanhoucke, V.: Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. 2016, arXiv:1602.07261.
40. Simard, P. Y., Steinkraus, D. and Platt, J. C. (2003) ‘Best practices for convolutional neural networks applied to visual document analysis’, in null, p. 958.
41. Valle, E., Fornaciali, M., Menegola, A., Tavares, J., Bittencourt, F.V., Li, L.T. and Avila, S., 2017. Data, depth, and design: learning reliable models for melanoma screening. arXiv preprint arXiv:1711.00441, 1..
42. Asperti, A. and Mastronardo, C. (2017) ‘The effectiveness of data augmentation for detection of gastrointestinal diseases from endoscopical images’, arXiv preprint arXiv:1712.03689.
43. Bloice, M. D., Stocker, C. and Holzinger, A. (2017) ‘Augmentor: an image augmentation library for machine learning’, arXiv preprint arXiv:1708.04680.

44. Jung, A. (2017) 'imgaug: Image augmentation for machine learning experiments.', p. Accessed 3 April 2017. Available at: <https://github.com/aleju/imgaug>.
45. Chollet, F. (2015) Keras: Deep Learning for humans, Github. Available at: <https://github.com/keras-team/keras>
46. Abadi, M. et al. (2016) 'Tensorflow: Large-scale machine learning on heterogeneous distributed systems', arXiv preprint arXiv:1603.04467.
47. Ye Q., Doermann D. Text Detection and Recognition in Imagery: A Survey // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2015. — Vol. 37, No. 7. — P. 1480–1500.
48. Simon M. Lucas, Alex Panaretos, Luis Sosa, Anthony Tang, Shirley Wong, Robert Young, Kazuki Ashida, Hiroki Nagai, Masayuki Okamoto, Hiroaki Yamamoto, Hidetoshi Miyao, JunMin Zhu, WuWen Ou, Christian Wolf, Jean-Michel Jolion, Leon Todoran, Marcel Worring, and Xiaofan Lin. ICDAR 2003 robust reading competitions: entries, results, and future directions. International Journal of Document Analysis and Recognition (IJ DAR), 7(2):105–122, 2005. ISSN 1433-2825. doi: 10.1007/s10032-004-0134-3. URL <http://dx.doi.org/10.1007/s10032-004-0134-3>.
49. D. Karatzas, S. R. Mestre, J. Mas, F. Nourbakhsh, and P. P. Roy. ICDAR 2011 robust reading competition - challenge 1: Reading text in born-digital images (web and email). In Proceedings of the 2011 International Conference on Document Analysis and Recognition (ICDAR), pages 1485–1490, Sept 2011. doi: 10.1109/ICDAR.2011.295.
50. D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. ICDAR 2015 competition on robust reading. In Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR), pages 1156–1160, Aug 2015. doi: 10.1109/ICDAR.2015.7333942.

51. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.
52. M. Niclsen, *Neural Networks and Deep Learning*. Determination Press, 2015
53. Y. LeCun, L. Jackel, L. Brunot, A. Cortes, C. Denker, H. Drucker, I. Guyon, I. Muller, E. Sackinger, P. Simard, and V. Vapnik, Comparison of learning algorithms for handwritten digit recognition, *International Conference of Artificial Neural Networks*, pp. 5360, 1995.
54. Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.
55. Shorten C., Khoshgoftaar T.M. A survey on Image Data Augmentation for Deep Learning // *Journal of Big Data*. — 2019. — Vol. 6, No. 1. — P. 1–48.
56. SHORTEN, Connor a Taghi KHOSHGOFTAAR. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data* [online]. Cham: Springer International Publishing, 2019, 6(1), 1-48 [cit. 2019-12-29]. DOI: 10.1186/s40537-019-0197-0. Dostupné z: <https://journalofbigdata.springeropen.com/>
57. Qiangguo Jin, Zhaopeng Meng, Tuan D Pham, Qi Chen, Leyi Wei, and Ran Su. “DUNet: A deformable network for retinal vessel segmentation”. In: *Knowledge-Based Systems* 178 (2019), pp. 149–162.
58. ZHANG, Aston, Zachary LIPTON a Alexander SMOLA. 13.1. Image Augmentation. *Dive into Deep Learning* [online]. [cit. 2019-12-29]. Dostupné z: <https://d2l.ai/>
59. UHER, Václav. Trénovatelné metody pro automatické zpracování biomedicínských obrazů [online]. Brno, 2018 [cit. 2019-12-29]. Dostupné z: <http://hdl.handle.net/11012/137277>. Disertační práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústa telekomunikací. Vedoucí práce Radim Burget.

60. Rafael C. Gonzalez and Richard E. Woods. Digital image processing. Pearson Prentice Hall, 2010.
61. Jakub Nalepa, Michal Marcinkiewicz, and Michal Kawulok. “Data augmentation for brain-tumor segmentation: a review”. In: *Frontiers in computational neuroscience* 13 (2019), p. 83.
62. ARDENSON, Richard a Peter KROGH. Color space and color profiles. DpBestflow.org [online]. 2015 [cit. 2020-01-04]. Dostupné z: <https://www.dpbestflow.org/color/color-space-and-color-profiles>
63. <https://github.com/tzutalin/labelImg> (дата обращения: 20.05.2022)
64. Ankit Chadha, Neha Satam, «A Robust Rapid Approach to Image Segmentation with Optimal Thresholding and Watershed Transform», 2013.
65. Shilpa Kamdi, R.K.Krishna, «Image Segmentation and Region Growing Algorithm», 2012.
66. Jianbo Shi, Jitendra Malik, «Normalized Cuts and Image Segmentation», 2000.
67. Полещенко Д.А., Зорин И.С. Нейросетевая идентификация клейма заготовок СПЦ-1 АО «ОЭМК» // Современные сложные системы управления (HTCS'2018) : труды XIII Международной научно-практической конференции / под общ. ред. Ю.И. Еременко. – 2018. – С. 81–84.
68. Зорин И.С., Полещенко Д.А. Нейросетевая идентификация клейма заготовок СПЦ-1 АО «ОЭМК» // Современные проблемы горно-металлургического комплекса. Наука и производство : сборник материалов Пятнадцатой Всероссийской научно-практической конференции / редкол.: Ю.И. Еременко [и др.]. – 2018. – С. 313–318.
69. Зорин И.С., Полещенко Д.А. Применение предобученных нейросетевых детекторов для идентификации кодов литой заготовки // Современные проблемы горно-металлургического комплекса. Наука и производство :

- XVII Всероссийская научно-практическая конференция. – Старый Оскол, 2021. – С. 284–289.
70. Зорин И.С., Полещенко Д.А. О применении сети глубокого обучения Faster R-CNN для идентификации клейма заготовок СПЦ-1 АО «ОЭМК» // Современные проблемы горно-металлургического комплекса. Наука и производство : материалы XVI Всероссийской научно-практической конференции с международным участием. – 2019. – С. 348–353.
71. Полещенко Д.А., Зорин И.С. Нейросетевая идентификация клейма заготовок СПЦ-1 АО «ОЭМК» // Приборы и системы. Управление, контроль, диагностика. – 2020. – № 6. – С. 32–39.
72. Фомин А.В., Глущенко А.И., Полещенко Д.А., Зорин И.С. Идентификация маркировки стальных заготовок в СПЦ-1 АО «Оскольский электрометаллургический комбинат им. А.А. Угарова» на основе нейросетевого подхода // Управление большими системами : труды XVII Всероссийской школы-конференции молодых ученых. – М., 2021. – С. 178–189.
73. Полещенко Д.А., Фомин А.В., Глущенко А.И., Зорин И.С. Разработка и промышленная эксплуатация интеллектуальной системы детектирования клейма литой заготовки в АО «ОЭМК им. А.А. Угарова» // Metallurg. – 2022. – № 3. – С. 97–103.
74. Poleshchenko D., Zorin I. Analysis of Methods of Augmentation of Images of the Branding of Cast Billet at A. A. Ugarov Oskol Electrometallurgical Plant // Proceedings of 5th International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency. – IEEE, 2023. – P. 315–319.
75. Poleshchenko D.A., Fomin A.V., Zorin I.S. The use of a cascade of small neural networks to increase the efficiency of the neural network detector for the stamp of cast blanks SPC-1 JSC “OEMK named after A.A. Ugarov” // Metallurgist. – 2026. – № 3. – P. 124–130.

76. Программа системы считывания клейма литой заготовки с адаптивным порогом распознавания. Зорин И.С., Полещенко Д.А. Свидетельство о регистрации программы для ЭВМ RU 2020662616, 16.10.2020. Заявка № 2020661937 от 08.10.2020.
77. Классификатор цифр на видео клейма литой заготовки на базе нейросетевого детектора. Зорин И.С., Полещенко Д.А. Свидетельство о регистрации программы для ЭВМ RU 2019662151, 17.09.2019. Заявка № 2019661140 от 10.09.2019.
78. Классификатор цифр на фотографии клейма литой заготовки на базе детектора. Зорин И.С., Полещенко Д.А. Свидетельство о регистрации программы для ЭВМ RU 2019662152, 17.09.2019. Заявка № 2019661141 от 10.09.2019.
79. Классификатор цифр на фотографии клейма литой заготовки на базе сверточной нейронной СЕТИ. Зорин И.С., Полещенко Д.А. Свидетельство о регистрации программы для ЭВМ RU 2018664334, 14.11.2018. Заявка № 2018661498 от 22.10.2018.

```

import os
import cv2
import numpy as np
import tensorflow as tf
import time
from utils import label_map_util
from utils import visualization_utils as vis_util
from PIL import Image

def func_lower_limit(cartinka):
    lower_limit = 0
    gray_rotation = cv2.cvtColor(cartinka, cv2.COLOR_BGR2GRAY)
    Gauss_rotation = cv2.GaussianBlur(gray_rotation, (5, 5), 0)
    ret, thresh_rotation = cv2.threshold(Gauss_rotation, 100, 255, 0)
    th_rotation = thresh_rotation[0:1080, 600:1200]
    th_rotation = np.sum(th_rotation, axis=1)
    th_rotation = np.convolve(th_rotation, np.ones((10,))/10)[(10-1):]
    for i in reversed(range(700)):
        if th_rotation[i] < 40000:
            print('Граница низа заготовки', i / 1080)
            lower_limit = i / 1080
            break
    cv2.imwrite("Sreen\REZ({0}).jpg".format(1488), thresh_rotation)
    return lower_limit

def func_detector(cartinka_R):
    probability = 0.9
    startTime = time.time()
    image_expanded = np.expand_dims(cartinka_R, axis=0)
    (boxes, scores, classes, num) = sess.run([detection_boxes, detection_scores,
    detection_classes, num_detections], feed_dict={image_tensor: image_expanded})

    while probability > 0.5:
        box = np.zeros((1, 20, 4))
        clas = np.zeros((1, 20))
        kol_scores = 0
        for poz in range(20):
            if scores[0][poz] > probability:
                kol_scores = kol_scores + 1
                box[0][poz][:] = boxes[0][poz][:]
                if classes[0][poz] == 10:
                    clas[0][poz] = 0

```

```

        else:
            clas[0][poz] = classes[0][poz]
    if kol_scores == 9:
        break
    else:
        probability = probability - 0.05

MASSIV = np.concatenate((box[0][:][:], np.reshape(clas, (20, 1))), axis=1)
ind_c = np.argsort(MASSIV, axis=0)
string_20 = ind_c[0:20, 1]
MASSIV = MASSIV[string_20, :]
n = 0
for poz in range(20):
    if MASSIV[n, 0] == 0:
        n = n + 1
    else:
        break
S = []
for h in range(MASSIV[n:20, 4].shape[0]):
    f = int(MASSIV[n:20, 4][h])
    S += '{0}'.format(f)
S = ".join(S)
print('Время:', time.time() - startTime)
return cartinka_R, MASSIV[n:20, :], S, n # вывод функции
if 0 == 0:
    cap = cv2.VideoCapture('D:\PycharmProjects\Video\2019-09-
02\10.50.82.144_00_00_20190902104334509.mp4')
    cap.set(cv2.CAP_PROP_POS_MSEC, 0)
    cap.set(cv2.CAP_PROP_FPS, 1)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)
    dim = (1920, 1080)
    font = cv2.FONT_HERSHEY_SIMPLEX
    bottomLeftCornerOfText = (500, 200)
    fontScale = 5
    fontColor = (0, 0, 0)
    lineType = 20
    MODEL_NAME = 'inference_graph'
    CWD_PATH = os.getcwd()
    PATH_TO_CKPT = os.path.join(CWD_PATH, MODEL_NAME,
'600_4bt_frozen_inference_graph.pb')
    PATH_TO_LABELS = os.path.join(CWD_PATH, 'training', 'labelmap.pbtxt')
    NUM_CLASSES = 10

```

```

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES,use_display_name=True)
category_index = label_map_util.create_category_index(categories)
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
    sess = tf.Session(graph=detection_graph)
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')
num_detections = detection_graph.get_tensor_by_name('num_detections:0')

ret, frame_1 = cap.read()
image_expanded = np.expand_dims(frame_1, axis=0)
(boxes, scores, classes, num) = sess.run(
    [detection_boxes, detection_scores, detection_classes, num_detections],
    feed_dict={image_tensor: image_expanded})
NOMER = 1
display = 350
otsch = 0
flag = 0
sglad = np.zeros(10)
ret, frame_1 = cap.read()
ret, frame_2 = cap.read()
gray_1 = cv2.cvtColor(frame_1, cv2.COLOR_BGR2GRAY)
gray_2 = cv2.cvtColor(frame_2, cv2.COLOR_BGR2GRAY)
while(True):
    ret, frame_3 = cap.read()
    gray_3 = cv2.cvtColor(frame_3, cv2.COLOR_BGR2GRAY)
    difference_01 = np.uint8((np.float32(gray_1) - np.float32(gray_3))/20)
    gray_1 = gray_2
    gray_2 = gray_3
    Gauss_01 = cv2.GaussianBlur(difference_01, (15,15), 0)
    ret, thresh_01 = cv2.threshold(Gauss_01, 125, 255, 0)
    if display != 350:
        cv2.putText(frame_3, s, bottomLeftCornerOfText, font, fontScale, fontColor,
lineType)
        display = display + 1

```

```

cv2.imshow('dist', cv2.resize(frame_3, dim, interpolation = cv2.INTER_AREA))
th_01 = sum(sum(thresh_01[0:300,0:1920]))
sglad[0:9] = sglad[1:10]
sglad[9] = th_01
summa = (sum(sglad) / 10)**3
if summa > 2*10**15:
    flag = 1
if flag == 1:
    otsch = otsch + 1
    if otsch == 150:
        otsch = 0
        flag = 0
        ret, cartinka_ = cap.read()
        cartinka = cv2.flip(cartinka_, -1)
        rezult_cartinka_01, rezult_MASSIV_01, rezult_S_01, rezult_n_01 =
func_detector(cartinka)
        rezult_cartinka_02, rezult_MASSIV_02, rezult_S_02, rezult_n_02 =
func_detector(cartinka_)
        N_01 = 20 - rezult_n_01
        N_02 = 20 - rezult_n_02
        if N_01 == 9 and N_02 != 9:
            s = rezult_S_01
            np.save('Sreen\{0}'.format(NOMER), rezult_MASSIV_01)
            cv2.imwrite('Sreen\{0}.jpg'.format(NOMER), rezult_cartinka_01)
            NOMER = NOMER + 1
            display = 0
        else:
            if N_01 != 9 and N_02 == 9:
                s = rezult_S_02
                np.save('Sreen\{0}'.format(NOMER), rezult_MASSIV_02)
                cv2.imwrite('Sreen\{0}.jpg'.format(NOMER), rezult_cartinka_02)
                NOMER = NOMER + 1
                display = 0
            else:
                if N_01 == 9 and N_02 == 9:
                    l_limit = func_lower_limit(cartinka)
                    if (rezult_MASSIV_01[2, 0] - l_limit) < 0.3 and l_limit != 0:
                        s = rezult_S_01
                        np.save('Sreen\{0}'.format(NOMER), rezult_MASSIV_01)
                        cv2.imwrite('Sreen\{0}.jpg'.format(NOMER),
rezult_cartinka_01)
                        NOMER = NOMER + 1
                        display = 0
                    else:

```

```

        s = result_S_02
        np.save('Screen\{0}'.format(NOMER), result_MASSIV_02)
        cv2.imwrite('Screen\{0}.jpg'.format(NOMER),
result_cartinka_02)
        NOMER = NOMER + 1
        display = 0

    if cv2.waitKey(1) & 0xFF == ord(' '):
        break
cap.release()
cv2.destroyAllWindows()

```

АКТ

о внедрении результатов исследования Зорина Ивана Сергеевича, направленных на решение актуальной научно-технической проблемы повышения эффективности контроля за движением заготовок на посту управления металлургического предприятия за счет внедрения в АСУТП системы поддержки принятия решений, способной автоматически распознавать код клейма, нанесенного машинописным способом на литую заготовку.

АО «Оскольский электрометаллургический комбинат имени А.А. Угарова» подтверждает, что следующие результаты, полученные лично Зориным Иваном Сергеевичем, использовались при выполнении НИОКР на тему «Система автоматизированного распознавания клейма литой заготовки на посту управления №7 сортопрокатного цеха №1 АО «ОЭМК им. А.А. Угарова» по договору от 18.12.2018 № 377343 внедренной в производство согласно акта от 29.10.2020.

Личный вклад Зорина И.С. в создание системы заключается в следующем:

- разработка алгоритма поддержки оператора в режиме советчика при распознавании цифрового кода, нанесённого машинописным способом на литую заготовку в АСУТП металлургического предприятия, включающий разработанную процедуру ансамблирования нейросетевого детектора каскадом малых нейронных сетей, адаптивно подключающихся для выполнения уточняющего анализа;

- собран и размечен набор из более чем 41 000 изображений клейма (около 370 000 аннотированных цифр), полученных в промышленных условиях СПЦ-1. Была разработана и реализована процедура аугментации, для расширения выборки и повышения устойчивости сети к дефектам клеймения;

- программной реализации модулей системы, использующих интеллектуальное распознавание номера литой заготовки с последующей его верификацией в БД АСУТП металлургического комбината с выдачей рекомендаций оператору в SCADA систему, позволяющая решать поставленную задачу в режиме реального времени.

Технический директор



П.В. Гулов

«25» 02 2026 г.

Наименование организации
АО «ОЭМК им. А.А. Угарова»
Структурное подразделение СПЦ-1

Приложение к учетной политике
Форма № 1-НИОКТР

УТВЕРЖДАЮ
Управляющий директор АО «ОЭМК им. А.А. Угарова»

С.И. Шишковец
« 02 » 2021 г.

Номер документа	Дата составления
-----------------	------------------

АКТ
обследования (внедрения, не внедрения) НИОКТР
(нужное подчеркнуть)

Настоящий акт составлен на предмет технического освидетельствования НИОКТР: «Разработка и внедрение системы автоматизированного распознавания клейма литой заготовки на загрузке печей нагрева СПЦ-1»

по договору от 18.12.2018 № 377343

с контрагентом СТИ НИТУ «МИСиС»

Наименование объекта, на котором осуществлено внедрение СПЦ-1

При обследовании установлено, что:

Научно-исследовательская работа на тему: «Разработка и внедрение системы автоматизированного распознавания клейма литой заготовки на загрузке печей нагрева СПЦ-1» с целью и задачами:

Выполненные этапы:

1. Произведен сбор данных об объекте автоматизации и проведение необходимых научно-исследовательских работ, для реализации требований Технического задания.
2. Разработан опытный образец системы распознавания клейма, наносимого клеймовочной машиной.
3. Разработан опытный образец системы распознавания рукописной маркировки.
4. Внедрение опытных образцов
5. Надзор за опытной эксплуатацией систем. По результатам проведенной работы предоставлен отчет, с разработанной инструкцией по эксплуатации систем считывания и передача данных.

Выполненные этапы (без НДС)

1 этап:

2 этап:

3 этап:

4 этап:

5 этап:

Заключение (краткое описание, преимущества выполненной работы): Разработана и внедрена система, обеспечивающая автоматическую идентификацию номера литой заготовки, включающую в себя операции распознавания клейма и систему распознавания клейма и рукописной маркировки, для исключения смешивания металла при посяде в печи нагрева СПЦ-1. Также разработана инструкция по эксплуатации системы. Предоставлены отчеты со статистическими данными и показателями работы автоматизированной системы, включающей в себя операции распознавания клейма и рукописной маркировки.

Дата завершения работы « 29 » октября 2020 г.

Дата начала использования результатов НИОКТР

« 01 » 02 2021 г.

Условный срок полезного использования

5 лет

Общая сумма расходов на НИОКТР

Председатель комиссии

Главный инженер
(должность)

Члены комиссии:

Технический директор

(должность)

Начальник СПЦ-1

(должность)

Начальник УТА

(должность)

Директор регионального ИТ -

Центра ООО «Металло-Тех»

г. Старый Оскол

(должность)

К.А. Чернов
(расшифровка подписи)

С.Л. Евсеев

(расшифровка подписи)

Е.В. Горетый

(расшифровка подписи)

В.Т. Севостьянов

(расшифровка подписи)

А.А. Кохтенко

(расшифровка подписи)

Присвоен инвентарный номер

Расходы по НИОКТР подлежат списанию на

(субсчет, участок затрат, статья расходов, МБЗ)

Безуличный номер
(должность)

(подпись)

Иван Москвитин
(расшифровка подписи)

С.И. Шишковец
С.И. Шишковец

С.И. Шишковец
С.И. Шишковец

ПРИЛОЖЕНИЕ 3

УТВЕРЖДАЮ



Заместитель директора СТИ НИТУ «МИСИС»
по науке и инновациям, д.т.н.
А.А. Кожухов
«11» 03 2026 г.

Акт

о внедрении в образовательный процесс результатов кандидатской
диссертационной работы Зорина Ивана Сергеевича

Настоящий акт подтверждает, что результаты исследований, проведённых в диссертационной работе Зорина Ивана Сергеевича, где разработаны процедура аугментации изображений для обучения глубоких нейронных сетей, алгоритм распознавания кодов литых заготовок на основе ансамбля нейросетевого детектора и каскада малых нейронных сетей, а также система поддержки принятия решений оператора АСУ ТП металлургического предприятия, используются в виде отдельных программных модулей в учебном процессе в СТИ НИТУ «МИСИС» на кафедре «Автоматизированных и информационных систем управления им. Ю.И. Еременко» в курсе предметов «Нейросетевые системы» и «Нейросетевые технологии в прикладных задачах». Использование результатов диссертационной работы позволяет углубить знания студентов в сфере применения методов искусственного интеллекта в управлении сложными технологическими процессами.

заместитель директора
СТИ НИТУ «МИСИС»

по учебно-методической работе  д.э.н., профессор Ильичева Е.В.

И.о. зав. кафедрой
АИСУ им. Ю.И. Еременко
СТИ НИТУ «МИСИС»



к.т.н., доцент Полещенко Д.А.