Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательно учреждение высшего образования

«Южно-Уральский государственный университет

(национальный исследовательский университет)»

Политехнический институт

Факультет машиностроения

Кафедра «Мехатроника и автоматизация»

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, к.т.н., доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Р. Гасияров

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ МАГИСТРА

ЮУрГУ 15.04.06.2021.215 ПЗ (ВКР)

Модернизация мехатронной системы участка транспортировки за чистовой группой клетей стана 2300 ПАО «ЧМК»

|  |  |  |
| --- | --- | --- |
| Нормоконтролер |  | Руководитель работы |
| Преподаватель |  | Доцент, к.т.н. |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.А. Гасиярова |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.С. Маклаков |

|  |  |  |
| --- | --- | --- |
| Консультант по проверке ВКР на оригинальность |  | Автор работы  студент группы П-264 |
| Ст. преподаватель |  |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.С. Воронин |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А Лисов  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. |

Челябинск 2021

Министерство НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ российской федерации

Федеральное государственное автономное образовательно учреждение высшего образования

«Южно-Уральский государственный университет

(национальный исследовательский университет)»

**Институт** Политехнический институт

**Факультет** Машиностроения

**Кафедра** Мехатроника и автоматизация

**Направление** 15.04.06 Мехатроника и робототехника

**УТВЕРЖДАЮ**

**Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_**\_\_\_\_ В.Р. Гасияров

подпись

« 3 »декабря 2020г.

**ЗАДАНИЕ**

**НА ВЫПОЛНЕНИЕ ВЫПУСКНОЙ**

**КВАЛИФИКАЦИОННОЙ РАБОТЫ МАГИСТРА**

**Студенту группы** П-264 Лисову Андрею Анатольевичу

1. Тема работы

Модернизация мехатронной системы участка транспортировки за чистовой группой клетей стана 2300 ПАО «ЧМК»

утверждена приказом по университету от \_\_\_\_\_\_\_\_\_\_20\_\_г. №\_\_\_\_\_\_\_\_\_\_\_\_\_

(утверждена распоряжением по факультету от \_\_\_\_\_\_\_\_20\_\_ г. №\_\_\_\_\_\_\_\_\_)

1. Срок сдачи студентом законченной работы 27.05.2021
2. Исходные данные к работе материалы преддипломной практики
3. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

|  |
| --- |
| 1. Описание и принцип работы мехатронной системы участка транспортировки;  2. Существующая проблема. Цель и задачи модернизации;  3. Требования к приводам мехатронной системы участка транспортировки;  4. Технические параметры, кинематическая, структурная и функциональная схемы мехатронной системы участка транспортировки;  5. Разработка электропривода рольганга;  6. Требования к проектируемой системе распознавания дефектов на поверхности металла;  7. Разработка алгоритм работы программы для распознавания дефектов на поверхности металла;  8. Разработка программы для ЭВМ на языке программирования Python |

1. Перечень обязательного графического и иллюстративного материала

|  |
| --- |
| 1. Цели и задачи  2. Технологическая схема  3. Требования к мехатронной системе участка транспортировки  4. Кинематическая, структурная и функциональная схемы мехатронной системы  5. Тахограмма и нагрузочная диаграмма электропривода  6. Результаты моделирования в Matlab Simulink  7. Требования к системе распознавания дефектов  8. Принцип работы алгоритма распознавания  9,10. Алгоритм работы программы  11,12. Результаты работы программы  13. Выводы по ВКР |

Всего 13 листов

**Руководитель** \_\_\_\_\_\_\_\_\_\_\_\_ доцент, кандидат технических наук Маклаков А.С.

подпись должность, звание Фамилия И.О.

**Дата выдачи задания** 3.12.2020

**Задание принял к исполнению студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лисов А.А.

подпись Фамилия И.О.

АННОТАЦИЯ

Лисов А.А. Модернизация мехатронной системы участка транспортировки за чистовой группой клетей стана 2300 ПАО «ЧМК». – Челябинск: ЮУрГУ, П-264, 89 с., 38 ил., 6 табл., библиогр. список – 22 наим. 2 прил.

В данной работе была рассмотрена модернизация существующего привода участка транспортировки за чистовой группой клетей стана 2300 ПАО «ЧМК» на современную систему преобразователь частоты – асинхронной двигатель, с расчётом математической модели и её моделирования в среде Mathlab Simulink.

Также в качестве дополнительной задачи улучшения качества выпуска готовой продукции горячего проката была разработана система компьютерного зрения, основанная на свёрточной нейронной сети.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc72764702)

[1 ОПИСАНИЕ И ПРИНЦИП РАБОТЫ МЕХАТРОННОЙ СИСТЕМЫ УЧАСТКА ТРАНСПОРТИРОВКИ 8](#_Toc72764703)

[1.1 Разогрев 8](#_Toc72764704)

[1.2 Черновая обработка 10](#_Toc72764705)

[1.3 Обрезка 11](#_Toc72764706)

[1.4 Чистовая обработка 11](#_Toc72764707)

[1.5 Плоскость и корона 12](#_Toc72764708)

[1.6 Ламинарное охлаждение 13](#_Toc72764709)

[1.7 Намотка 15](#_Toc72764710)

[2 СУЩЕСТВУЮЩАЯ ПРОБЛЕМА. ЦЕЛЬ И ЗАДАЧИ МОДЕРНИЗАЦИИ 16](#_Toc72764711)

[3 ТРЕБОВАНИЯ К МЕХАТРОННОЙ СИСТЕМЕ УЧАСТКА ТРАНСПОРТИРОВКИ. 18](#_Toc72764712)

[4 ТЕХНИЧЕСКИЕ ПАРАМЕТРЫ, КИНЕМАТИЧЕСКАЯ, СТРУКТУРНАЯ И ФУНКЦИОНАЛЬНАЯ СХЕМА МЕХАТРОННОЙ СИСТЕМЫ УЧАСТКА ТРАНСПОРТИРОВКИ 19](#_Toc72764713)

[4.1 Технические характеристики механизма рольганга 19](#_Toc72764714)

[4.2 Кинематическая схема участка транспортировки 20](#_Toc72764715)

[4.3 Структурная схема электропривода рольганга 20](#_Toc72764716)

[4.4 Функциональная схема 21](#_Toc72764717)

[5 РАЗРАБОТКА ЭЛЕКТРОПРИВОДА РОЛЬГАНГА 23](#_Toc72764718)

[5.1 Выбор и расчет электродвигателя 23](#_Toc72764719)

[5.2 Выбор преобразователя частоты 31](#_Toc72764720)

[5.3 Система управления объектом, расчет регуляторов 34](#_Toc72764721)

[5.4 Расчёт параметров регулятора 36](#_Toc72764722)

[5.5 Моделирование в MathLab Simulink 40](#_Toc72764723)

[6 ТРЕБОВАНИЯ К ПРОЕКТИРУЕМОЙ СИСТЕМЕ РАСПОЗНАВАНИЯ ДЕФЕКТОВ НА ПОВЕРХНОСТИ МЕТАЛЛА 42](#_Toc72764724)

[7 РАЗРАБОТКА АЛГОРИТМА РАБОТЫ ПРОГРАММЫ ДЛЯ РАСПОЗНАВАНИЯ ДЕФЕКТОВ НА ПОВЕРХНОСТИ МЕТАЛЛА 45](#_Toc72764725)

[7.1 Обзор существующих алгоритмов машинного обучения 45](#_Toc72764726)

[7.2 Свёрточная нейронная сеть (ConvNet / CNN) 48](#_Toc72764727)

[7.3 Обзор существующих инструментов и библиотек для проектирования компьютерного зрения 52](#_Toc72764728)

[7.4 Алгоритм работы программы 55](#_Toc72764729)

[7.5 Расчёт высоты камеры над поверхностью 61](#_Toc72764730)

[8 РАЗРАБОТКА ПРОГРАММЫ ДЛЯ ЭВМ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON 65](#_Toc72764731)

[8.1 О Google Colab 65](#_Toc72764732)

[8.2 Разработка свёрточной нейронной сети на Python в Google Colab 66](#_Toc72764733)

[ЗАКЛЮЧЕНИЕ 85](#_Toc72764734)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 87](#_Toc72764735)

[Приложение А Листинг кода на Python в Google Colab](#_Toc72764735) 89

ВВЕДЕНИЕ

Прокатка является основным видом обработки металлов давлением. Около 75 процентов стали, выплавляемой на металлургических заводах, обрабатывается на прокатных станах и выпускается в виде готового проката: листов, сортовых профилей, труб и т.п.

В современных механизированных прокатных станах с поточным технологическим процессом обработки металла рольганги являются одним из наиболее распространенных вспомогательных механизмов, от которых в большой степени зависит производительность и бесперебойная работа прокатного стана в целом. Производительность прокатного стана может оказаться невысокой, если хотя бы один из его механизмов не в состоянии выполнить соответствующее количество операций в заданное время. К таковым относятся и транспортные рольганги.

Транспортными рольгангами называют все рольганги, установленные перед рабочей клетью и за ней и связывающие между собой, отдельные вспомогательные машины и устройства стана. На этапе транспортировки листопроката также производится визуальная проверка качества проката специалистами по качеству. Заменить визуальный контроль человека можно при помощи компьютерных технологий. В настоящее время компьютерные технологии являются неотъемлемой частью жизни практически всех окружающих нас людей. Компьютерные технологии стали более доступными и появились узкие сферы компьютерной деятельности, а также в индустрии 4.0.

Индустрия 4.0 – это создание «умной фабрики», производственной площадки, которая использует данные от различных типов датчиков и всех доступных источников для оптимизации процессов. Компьютерное зрение является частью комплексного подхода к интеллектуальному производству, который позволяет компьютерам и машинам «видеть» физический мир, позволяя им извлекать, обрабатывать и анализировать информацию из визуальных входов.

Рынок компьютерного зрения претерпевает неуклонную трансформацию, постоянно создавая новые решения и технологические достижения. Индустрия 4.0 еще больше выиграла от этой технологии во время пандемии, поскольку она широко используется для целей инвентаризации. В 2018 году мировой рынок компьютерного зрения составлял более 9,2 миллиарда долларов, а к 2025 году ожидается, что он превысит 13,0 миллиарда долларов. И Северная Америка, и Европа являются лидерами внедрения компьютерного зрения в производство и ряд других отраслей.

Наряду с растущим интересом к технологиям и ростом рынка компании уделяют больше внимания технологическим достижениям, которые предлагает искусственный интеллект. Согласно исследованию, компьютерное зрение – одна из наиболее широко используемых технологий. Его используют не менее 20 процентов компаний по всему миру.

Цель данной работы – модернизация существующей системы электропривода на более современную с последующим расчётом математической модели её моделированием в среде Matlab Simulink, а также реализация системы компьютерного зрения для повышения качества выпускаемой продукции.

Задачи:

- сформулировать требования к приводам мехатронной системы участка транспортировки;

- найти технические параметры модернизируемой системы, а также разработать кинематическую, структурную и функциональную схему мехатронной системы участка транспортировки;

- разработать новый электропривод рольганга с системой управления ПЧ-АД;

-сформулировать требования к проектируемой системе распознавания дефектов на поверхности металла;

- разработать алгоритм работы программы для распознавания дефектов на поверхности металла;

- разработать программу для ЭВМ на языке программирования Python.

1 ОПИСАНИЕ И ПРИНЦИП РАБОТЫ МЕХАТРОННОЙ СИСТЕМЫ УЧАСТКА ТРАНСПОРТИРОВКИ

### 1.1 Разогрев

По мере надобности слябов, составляются графики и материал размещается с помощью железнодорожных вагонов и мостовых кранов на складе слябов в западном конце стана горячей прокатки. Слябы укладываются по одной на линию прокатки (рисунок 1.1). Технологическая схема участка показана на рисунке 1.2.

Размеры и вес сляба являются утвержденными, горячий прокат начинается с разогрева стальных полуфабрикатов почти до температуры плавления, затем прокатка их через 12 последовательных клетей до всё более тонкого состояния [1]. Когда в печи есть место, электромеханические «толкатели» включаются для перемещения слябов в печь.



Рисунок 1.1. Укладка слябов на линию прокатки

Оказавшись внутри, слябы поддерживаются на высоте примерно 2,5 метров от пола печи водоохлаждаемыми трубами с огнеупорным покрытием, называемыми «салазками». Чтобы минимизировать холодные пятна («следы скольжения»), оставленные на слябе, расстояние между полозьями изменяется примерно на две трети пути через печь.

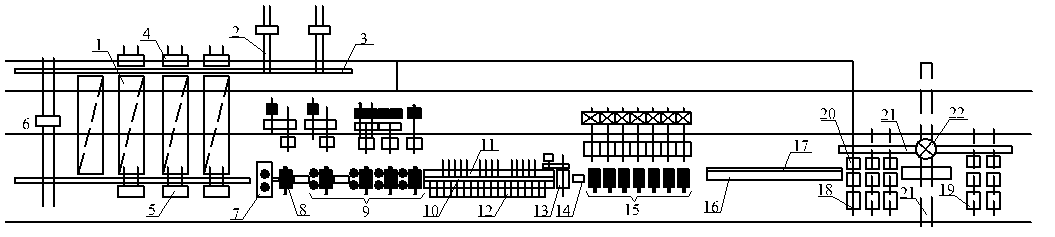


Рисунок 1.2. Технологическая схема участка. 1 – нагревательные печи, 2 – устройство для подачи слябов 3 – загрузочный рольганг, 4 – печной толкатель, 5 – приёмники слябов, 6 – передаточная тележка для слябов, 7 – вертикальная клеть окалиноломатель, 8 – черновая клеть дуос, 9 – черновые универсальные клети, 10 – промежуточный рольганг, 11 – сбрасыватель, 12 – карман для толстых полос, 13 – летучие ножницы, 14 – чистовой роликовый окалиносниматель, 15 – чистовая группа клетей, 16 – отводящий рольганг, 17 – система водяного охлаждения полос, 18 – моталки для смотки тонких полос, 19 – привод моталок, 20 – толкатель рулонов, 21 – контейнеры транспортёров рулонов, 22 – подъёмно-поворотный стол

Два независимых комплекта полозьев, один фиксированный, другой шагающий, по очереди поддерживают сляб, когда он проходит через печь массивным подрамником, приводимым в действие двумя гидроцилиндрами.

Большая часть предварительного нагрева стали достигается за счет того, что горячие отходящие газы печи устремляются мимо слябов на пути к «рекуператорам» над дверцей загрузки. Любое тепло, оставшееся в отходящих газах, в этих массивных теплообменниках предварительно нагревает входящий воздух для горения до температуры более 500 0С [2]. Напротив, в зоне нагрева сталь в первую очередь нагревается раскаленными стенками печи. В зоне выдержки многочисленные горелки меньшего размера стремятся поддерживать равномерную температуру внутри зон, чтобы уравновесить любые холодные точки в слябах.

Постоянно оценивая температурный градиент по профилю каждого сляба в печи, при помощи специального оборудования помогает оператору выбрать скорость производства и заданные значения температуры для зон слябов.

Когда сляб достигает конца печи определяется достаточно ли он нагрет, если да, то открывается люк печи, и далее мостовом краном поднимаются с рамы. Далее горячий сляб помещается на рольганг, который переносит его на черновой стан.

### 1.2 Черновая обработка

Черновой стан состоит из шести независимых прокатных клетей, последние четыре из которых включают небольшие вертикальные прокатные станы, называемые кромкообрезными станками. Слябы, нагретые в печи до яркого оранжево-желтого цвета, прокатываются через одну клеть за раз для получения так стержней, пригодных для чистовой прокатки. Водоструйные форсунки высокого давления очищают окисленное железо или окалину с поверхности по пути проката.

Когда передаточная штанга выходит из последней черновой клети стана, оценивается толщина передней кромки прутка. Точно так же пирометр измеряет температурный профиль стержня от головы до хвоста, а специальная камера фотографирует оба конца. В зависимости от толщины, ширины и марки

прокатываемого продукта средняя температура прутка на выходе из последнего чернового стана обычно составляет от 1000 до 1150 0С. Далее начинается вычисление скорости и величины зазора между валками. Последние четыре черновых стана включают в себя кромкообрезные станки для контроля ширины и прокатки прутка толщиной от 12 до 15 сантиметров до примерно 3 см, в зависимости от заказанной клиентом ширины, толщины и марки стали. Как упоминалось ранее, каждая третья и пятая черновые станы имеют коллекторы для удаления окалины под высоким давлением, работающие под давлением 4500 кг на квадратный сантиметр.

### 1.3 Обрезка

Поскольку передняя часть прокатной полосы имеет важное значение для правильной настройки величины зазора валков чистовых станов и разматывателей, а также из-за того, что неровная задняя часть может повредить поверхности рабочих валков или вызвать проблемы с резьбой затяжных винтов валков для следующего производственного процесса, передняя и задняя часть обрезается парой больших стальных барабанов, каждый из которых оснащен ножом, проходящим вдоль его длины. Когда полоса движется по роликовому столу со скоростью около 30 метров в минуту, датчики определяют ее положение и скорость, чтобы синхронизировать срезные барабаны для оптимизации количества обрезки [1].

### 1.4 Чистовая обработка

Стан горячей прокатки включает шесть чистовых станов, которые уменьшают толщину полосы до толщины, необходимой заказчику или следующему процессу (рисунок 1.3). Скорость прокатки установлена таким образом, чтобы последняя клеть могла выполнить окончательное обжатие при конечной температуре от 800 до 900 0С, для достижения определенных механических свойств.

К этому моменту сляб из стали превратился в плоский лист длиной 60 метров. В отличие от черновых станов, чистовые станы прокатывают передаточную штангу в тандеме, что означает, что каждая полоса будет прокатываться через все шесть клетей одновременно. Горячая сталь довольно хрупкая, поэтому при прокатке натяжение между клетями чистового стана необходимо тщательно контролировать на очень низком уровне, чтобы избежать растяжения или разрыва полосы.

Перед чистовой прокаткой передняя и задняя части полосы будут обрезаны, чтобы выровнять их. Последняя двухэтапная операция по удалению окалины выполняется для удаления окалины, образовавшейся на прокате во время черновой обработки.

Регулировки производится по мере необходимости для обеспечения правильного прохождения нитей полосы без образования петель и складок, растяжения и разрывов. Положение каждого валка передается в сложную систему автоматизации чистового стана, которая вместе с информацией от датчиков нагрузки, которые контролируют усилие прокатки, и от рентгеновского датчика, измеряющего конечную толщину полосы, плавно регулирует зазоры и скорости валков для поддержания необходимой толщины, несмотря на колебания температуры в каждом прутке [3].

### 1.5 Плоскость и корона

В дополнение к степени растяжения клетей прокатного стана, как описано ранее, валки будут отклоняться или изгибаться под нагрузкой, поскольку они раздвигаются посередине полосой, но на концах поддерживаются подшипниками будет появляться отклонение, которое является выпуклостью полосы, обычно называемой короной.

Воронки полосы начинаются на черновых станах и продолжаются через каждую последующую клеть прокатного стана. Корона полосы измеряется на выходе из чистовых станов с помощью второго сканирующего рентгеновского датчика, который пересекает полосу взад-вперед по ширине полосы по мере прокатки стали. Измеренная им толщина сравнивается с толщиной, измеренной первичным рентгеновским датчиком, контролирующим осевую линию по всей длине полосы, и разница затем наносится на график как запись качества продукта. Обычно на стане горячей прокатки производится материал с венцом от 0,025 до 0,076 мм в зависимости от ряда факторов, включая толщину, ширину и сорт готового продукта. После выхода из чистовых станов полоса проходит через валки с индивидуальным приводом, через группы струй воды большого объема с низким давлением, которые охлаждают раскаленную полосу до заданной температуры наматывания (рисунок 1.4).



Рисунок 1.3. Чистовая обработка

### 1.6 Ламинарное охлаждение

Критически важным для горячекатаной стали является температура наматывания, так как рулон будет охлаждаться от этой температуры до температуры окружающей среды в течение трех дней. По сути, это термообработка, сравнимая с отжигом, напряжения, передаваемые стали во время обжатия с 20 сантиметров до заказанной толщины, получают возможность по мере охлаждения рулона снять это внутреннее напряжение.

Охлаждение стали до 200 0С, когда она движется со скоростью до 800 метров в минуту, требует огромного количества воды, индивидуально регулируемой и контролируемой системой автоматизации, смачивания стали сверху и снизу водяной завесой [4].



Рисунок 1.4. Охлаждение полосы

Горячекатаный лист наматывается при относительно высоких температурах, чтобы попытаться максимально снять внутреннее напряжение встали. И наоборот, намотка при относительно низкой температуре позволяет сортам стали сохранять более высокие уровни внутренних напряжений и ограничивает размер отдельных кристаллов и карбидов, которые образуются внутри и между кристаллами; каждый из этих факторов способствует повышению прочности готовой горячекатаной полосы.

### 1.7 Намотка

Конфигурация стана горячей прокатки состоит из двух моталок (рисунок 1.5). Между ними существуют незначительные различия, но оба работающих намотчика начинаются с пары прижимных валков, которые захватывают головной конец полосы и создают натяжение через выходной стол и обратно к чистовым станам. Головной конец отклоняется заслонкой вниз, связанной с моталкой, и направляется вокруг оправки с помощью пневматически приводимых в действие оберточных валков, соединенных фартуками.



Рисунок 1.5. Намотка полосы на моталку

После выхода полосы из чистового стана прижимные валки продолжают удерживать обратное натяжение, чтобы рулон не распускался; перед тем, как конец ленты будет протянут через прижимные ролики, оберточные ролики снова входят в зацепление. Гидравлическая тележка с катушкой перемещается под катушку и, после того, как поднимается, чтобы поддержать объем рулона, снимает катушку с оправки и помещает ее в положение для транспортировки к процедурам маркировки и автоматического оклейки.

2 СУЩЕСТВУЮЩАЯ ПРОБЛЕМА. ЦЕЛЬ И ЗАДАЧИ МОДЕРНИЗАЦИИ

На текущий момент в прокатном цеху №4, на участке транспортировки за чистовой группой клетей стана 2300 ПАО «ЧМК» применяются устаревшие технологии производственных процессов. В качестве основного электропривода применяется система тиристорный преобразователь – двигатель постоянного тока с независимым возбуждением, а системой контроля качества выпускаемого продукта по-прежнему представлена человеком.

Главными недостатками, кроме морального устаревания данный технологий и процессов производств можно отнести:

-двигатели постоянного тока требуют обслуживания коллекторно-щёточного узла [5], для этого обслуживания требуется остановка производственного процесса, что приносит большой ущерб в условиях непрерывного производства;

-тиристорный преобразователь существующего типа наводит высокие гармоники в сеть питания цеха, что ухудшает общее качество этой сети [6];

-человек, в отличии от машины не способен работать в непрерывном режиме на монотонной работе, поэтому поверхность проката физически не может контролироваться в полной мере. С данной проблемой может справиться автоматическая система распознавания дефектов, имеющая высокую точность распознавания и отсутствие эффекта усталости от монотонной работы, а также имеющая очень высокое быстродействие по сравнению с человеком.

-модернизация существующей системы также позволит повысить эффективность работы, за счёт применения более современного оборудования, имеющего более высокий коэффициент полезного действия, повысить качество выпускаемой продукции, за счёт внедрения системы компьютерного зрения и снизить стоимость постоянного обслуживания существующего электропривода, путём замены двигателя постоянного тока с независимым возбуждением на асинхронный двигатель с короткозамкнутым ротором.

Для разработки системы электропривода асинхронный двигатель – преобразователь частоты (ПЧ-АД) и системы компьютерного зрения, основанной на алгоритмах машинного обучения (свёрточные нейронные сети) требуется решить следующие задачи:

- сформулировать требования к приводам мехатронной системы участка транспортировки;

-произвести поиск исходных чертежей производственного участка, модернизируемого узла;

- на основе найденной документации составить структурную, функциональную и кинематическую схему исходного привода, а также найти технические параметры модернизируемой системы, такие как: скорость движения проката, масса проката, линейные размеры рольганга и подвижных элементов;

- разработать новый электропривод рольганга с системой управления ПЧ-АД;

-сформулировать требования к проектируемой системе распознавания дефектов на поверхности металла, в соответствии с действующими нормами и правилами;

-провести анализ существующих библиотек и фреймворков для глубокого и машинного обучения и выбрать наиболее оптимальный вариант, удовлетворяющий всем обозначенным требованиям.

-провести анализ на предмет существующих датасетов дефектов на поверхности горячего проката;

- разработать алгоритм работы программы для распознавания дефектов на поверхности металла в виде блок-схем;

- разработать программу для ЭВМ на языке программирования Python, используя ранее выбранные библиотеки и фреймворки для работы с машинным обучением на основе ранее найденного или созданного датасета изображений поверхностных дефектов на поверхности горячего проката.

3 ТРЕБОВАНИЯ К МЕХАТРОННОЙ СИСТЕМЕ УЧАСТКА ТРАНСПОРТИРОВКИ

Исходные параметры для проектирования привода.

-электропривод должен быть подобран исходя из тех условий, что напряжение цеховой магистрали переменного тока составляет 380 В, а частота 50 Гц;

- среда использования характеризуется общей запылённостью цеха, поэтому электропривод должен иметь защиту не меньше IP54.

-производство проката носит непрерывный характер, поэтому режим работы электродвигателя – S1;

-климатическое исполнение электродвигателя не ниже У2;

-скорость движения проката после выхода из чистовых клетей 1,25 м/с, максимальное допустимое ускорение заготовки равное 1 м/с2, величина ошибки регулирования скорости не должна превышать 1% от заданной.

- величина ошибки регулирования момента не должна превышать 5% от заданного значения.

-механическая часть электропривода остается неизменной, за исключением редуктора.

Требования к системе управления

-система регулирования является автоматической;

-система управления электропривода должна успешно интегрироваться в существующий ПЛК Siemens S7–1200;

- размещение пульта управления – автономное, вынесенное из конструктива мехатронной системы.

-поскольку для участка транспортировки применяется 3 группы одинаковых электроприводов, то преобразователь частоты должен быть выполнен в виде модуля, с управлением от общего контроллера Siemens S7–1200;

4 ТЕХНИЧЕСКИЕ ПАРАМЕТРЫ, КИНЕМАТИЧЕСКАЯ, СТРУКТУРНАЯ И ФУНКЦИОНАЛЬНАЯ СХЕМА МЕХАТРОННОЙ СИСТЕМЫ УЧАСТКА ТРАНСПОРТИРОВКИ

### 4.1 Технические характеристики механизма рольганга

Для существующего рольганга участка транспортировки листопроката после чистовых клетей стана 2300 ПАО «ЧМК» известны следующие характеристики, приведённые в таблице 4.1.

Таблица 4.1. Технические характеристики механизма рольганга

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Численное значение |
| Масса ролика | mp, кг | 767,3 |
| Диаметр ролика | D, м | 0,25 |
| Момент инерции ролика | Jp, кг·м2 | 5,99 |
| Число роликов | N | 17 |
| Масса листопроката | m3, кг | 1870 |
| Длина заготовки | L, м | 30 |
| Скорость движения | Vp, м/с | 1,25 |
| Время работы рольганга | tp, с | 24 |
| Число циклов работы | z, 1/ч | 130 |

Принятые обозначения:

аДОП = 0,5 м/с2 – максимальное допустимое ускорение заготовки;

dшр = 0,095 м – диаметр шейки ролика;

f = 0,005 – коэффициент трения качения;

μ = 0,015 – коэффициент трения скольжения;

JШ = 0,4 кг∙м2 – момент инерции тормозного шкива;

CК = 50 МН∙м/рад – крутильная жесткость;

JВ = 3 кг∙м2 – момент инерции продольного вала.

### 4.2 Кинематическая схема участка транспортировки

Кинематическая схема рольганга показана на рисунке 4.1, участок транспортировки разделён на три. Каждый участок включает в себя систему из 17 роликов (3), двигателя (4), редуктора (6) и продольного вала (2), передающего вращающий момент на ролики через коническую шестерню (1). Торможение двигателя производится при помощи тормозного шкива (5), установленного на вал асинхронного двигателя. В существующей системе требуется заменить электропривод и как следствие редуктор, остальная механическая часть остаётся прежней.

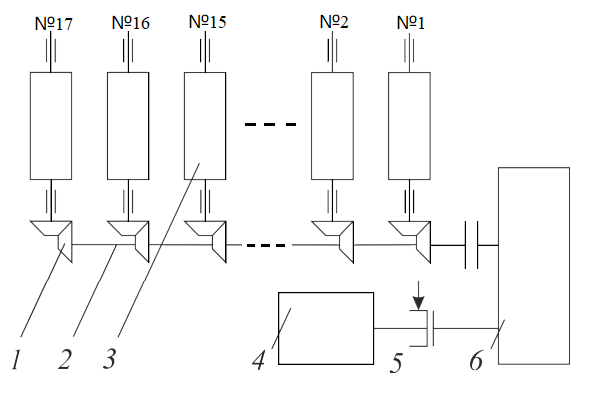


Рисунок 4.1. Кинематическая схема участка транспортировки. 1 – Коническая шестерня, 2 – Продольный вал, 3 – Ролик, 4 – Электродвигатель, 5 – Тормозной шкив, 6 – Редуктор

### 4.3 Структурная схема электропривода рольганга

Структурная схема электропривода транспортного рольганга показана на рисунке 4.2. Управление двигателем (существующим) производится при помощи тиристорного преобразователя (ТП), управляемого при помощи «СИФУ» система импульснофазового управления). Сигналы обратной связи по скорости и току поступают на вход системы управления (СУ), представленного двухконтурной системой подчинённого регулирования с внешним контуром скорости.

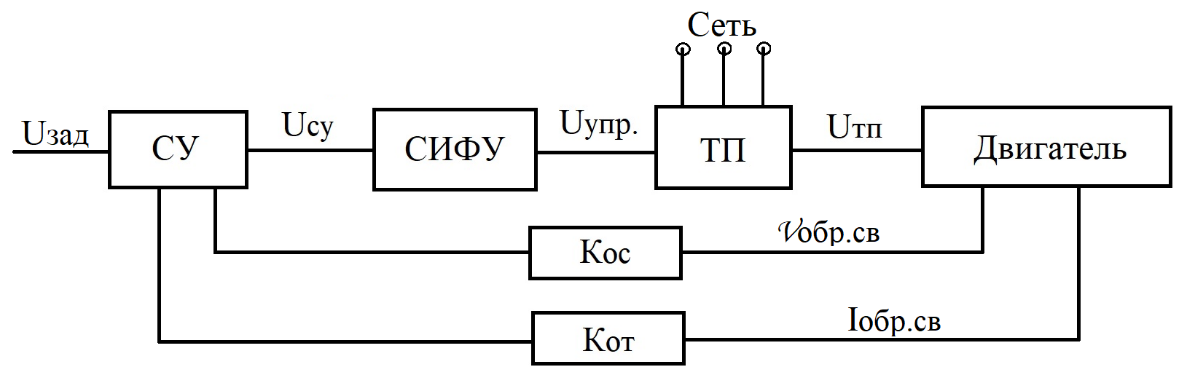


Рисунок 4.2. Структурная схема электропривода транспортного рольганга

### 4.4 Функциональная схема

Функциональная схема существующего электропривода рольганга показана на рисунке 4.3. Напряжение из сети через автоматический выключатель (Q) передается на трансформатор напряжения, после на 2 комплекта тиристоров (V11 – V16, V21 – V26), позволяющих осуществить движение двигателя как вперёд, так и назад, для защиты вентилей применяются плавкие предохранители (Z11 – Z16, Z21 – Z26).

Комплект вентилей управляется при помощи СИФУ путём изменения момента включения тиристора относительно точки естественного открытия, опорная частота сети снимается с двух трансформаторов тока (Т1, Т2). Система управления электроприводом (СУ) построена на двухконтурной системе регулирования с внутренним контуром тока. Регулирование осуществляется через образную связь по скорости путём снятия напряжения с тахогенератора (BR) и тока с шунта (Rш). Существующий двигатель является двигателем с независимым возбуждением, поэтому к его обмотке возбуждения (ОВ) подведено питание от независимого источника.

Поскольку данная система устарела и требует замены (причины во 2-й главе), то она будет заменена на более современную – преобразователь частоты – асинхронный двигатель (ПЧ – АД, функциональная схема данного типа управления представлена на рисунке 5.3).

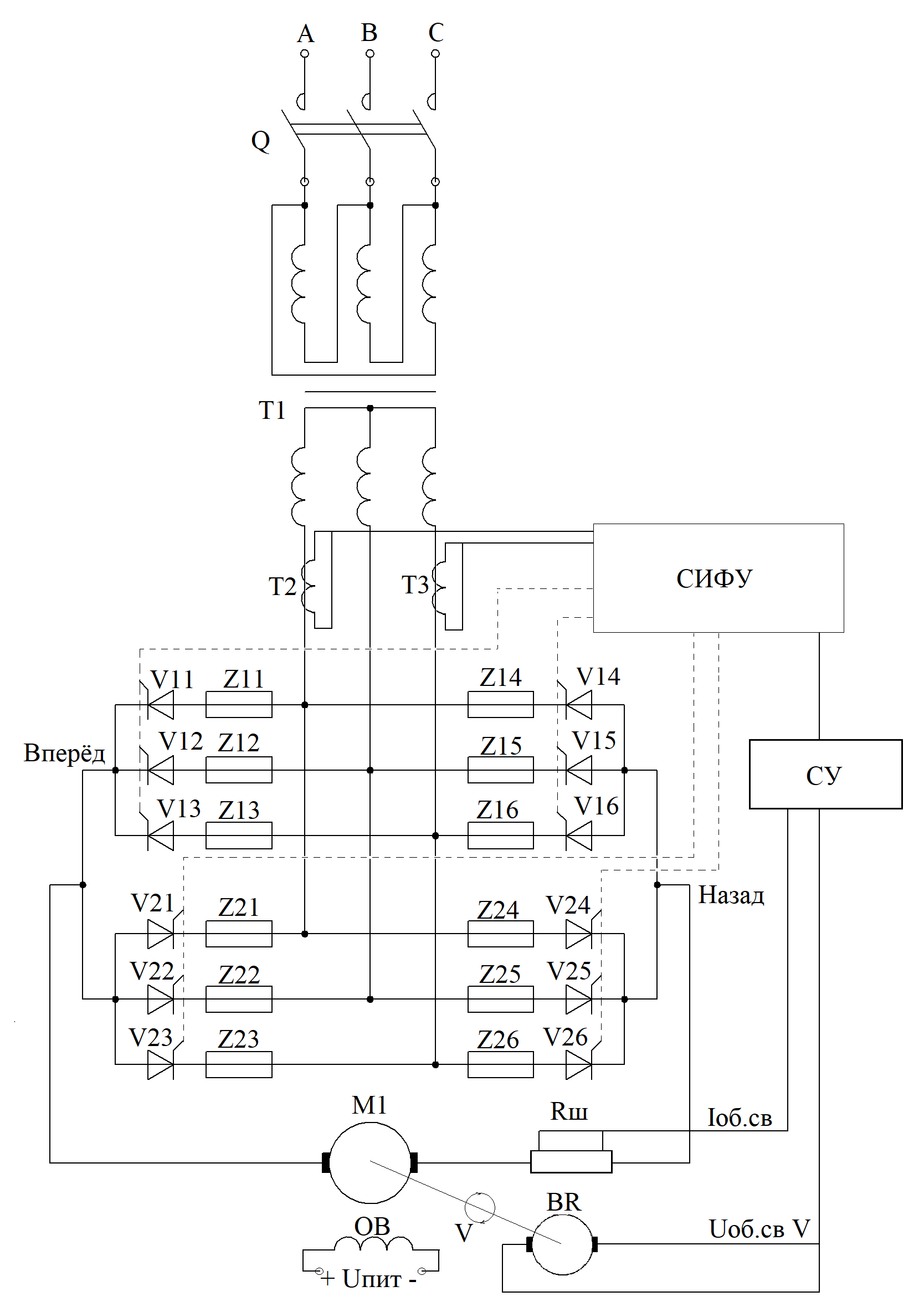


Рисунок 4.3. Функциональная схема существующего электропривода

5 РАЗРАБОТКА ЭЛЕКТРОПРИВОДА РОЛЬГАНГА

### 5.1 Выбор и расчет электродвигателя

На основе исходных технических данных из таблицы 4.1, а также методики расчета из [8] определим время разгона tп, и время торможения tт электропривода:

(5.1)

Путь, проходимый за время пуска (торможения) листопроката:

(5.2)

Время установившегося режима движения с заданной скоростью Vу:

(5.3)

Тогда получим:

Для каждого участка определим статические моменты сопротивления движению по формуле:

(5.4)

где момент сил трения в подшипниках;

(5.5)

где g = 9,8 м/с² - ускорение свободного падения, тогда:

Момент силы трения качения :

(5.6)

где *f* – коэффициент трения качения.

Рассчитаем Мрост [8]

Кратковременная нагрузка при проскальзывании заготовки:

Момент инерции рабочего органа:

(5.7)

Динамические моменты для каждого режима работы:

(5.8)

Тогда полный момент рабочего органа:

(5.9)

Определим полный момент для каждого режима работы:

- установившийся режим:

(5.10)

- пуск:

(5.11)

- торможение:

(5.12)

Среднеквадратичное значение момента электропривода вычислим при помощи следующей формулы [8]:

(5.13)

где Mk – момент двигателя на k-м участке (k=1, … , n); tk – длительность k-го участка.

Время цикла при заданном числе циклов работы машины в час и фактическое значение относительной продолжительности включения:

(5.15)

(5.14)

И, наконец, определяем расчетную мощность двигателя:

(5.16)

где υ0 – скорость движения рабочего органа, в нашем случае υ0 = υВ; ПВКАТ – ближайшее к ПВф каталожное значение относительной продолжительности включения. ПВКАТ = 1; k1 – коэффициент, учитывающий динамические нагрузки, обусловленные вращающимися элементами электропривода, а также потери мощности в редукторе, принимаем k1 = 1,5.

Тогда:

Выбор электродвигателя и редуктора

Для непрерывного режима работы следует выбирать двигатели специальных серий, предназначенных для этого режима (режим работы S1). Двигатель выбирается по каталогу таким образом ПВКАТ ≥ ПВф. Выбираем двигатель, основные технические характеристики которого приведены в таблице 5.1.

Таблица 5.1. Технические данные двигателя АИР 132S6

|  |  |  |
| --- | --- | --- |
| Параметр | Обозначение | Значение |
| Мощность, кВт | Р | 5,5 |
| Частота вращения, об/мин | n | 960 |
| Ток номинальный, А | IН | 12,9 |
| КПД | 𝜂 | 0,84 |
| Коэффициент мощности | cosφ | 0,77 |
| Пусковой ток, А | IП | 6,5 IН |
| Номинальный момент, Н∙м | МНОМ | 54,6 |
| Пусковой момент, Н∙м | МП | 2,1 МП |
| Максимальный момент, Н∙м | Мкр | 2,1 МП |
| Момент инерции ротора, кг∙м2 | Jдв | 0,05 |

Передаточное число редуктора определяется по номинальной скорости вращения выбранного двигателя ωН н и основной скорости движения исполнительного органа υ0.

(5.17)

На основании расчётных данных выбираем цилиндрический двухступенчатый редуктор типа РЦД-350 с передаточным числом jр = 10, предельной скоростью вращения быстроходного вала nб = 1000 об/мин, допускаемой нагрузкой Тт = 1200 Н∙м и КПД ηр= 97 процентов.

Статический момент приведённый к валу двигателя:

(5.18)

Статический момент приведённый к валу двигателя (двигательный режим):

(5.19)

Статический момент приведённый к валу двигателя (тормозной режим)

(5.20)

Приведенные статические моменты системы электропривода рассчитывают для каждого участка с учетом режима работы электропривода по формуле:

(5.21)

где Мхх – момент потерь холостого хода, который принимаем равным 5 процентам от номинального момента двигателя, т.е.

(5.22)

Тогда статические моменты (двигательный режим):

в тормозном режиме:

Суммарный приведенный к валу двигателя момент инерции системы:

(5.23)

где Jд – момент инерции ротора двигателя; δ – коэффициент, учитывающий момент инерции остальных элементов электропривода: муфт, тормозного шкива, редуктора и др., принимаем δ= 1,5; Jпр – приведенный к валу двигателя суммарный момент инерции движущихся исполнительных органов рабочей машины и связанных с ними движущихся масс.

(5.24)

тогда получаем:

Для возможности учета влияния упругостей в механизме указана крутильная жесткость Ск, отнесенная к рабочему валу. Приведенную к валу двигателя жесткость упругой механической связи Спр рассчитаем по следующей формуле:

(5.25)

Для каждого участка работы найдем значения установившейся скорости:

Теперь определим динамический момент:

(5.27)

(5.26)

Пусковой момент:

(5.28)

Момент торможения:

(5.29)

Проверка двигателя по нагреву и производительности.

Для приближенного расчета времени переходного процесса оценивают средний момент двигателя Мср.  Время разгона до скорости Vр:

(5.30)

Время торможения проката

(5.31)

Угол поворота вала двигателя за время переходных процессов:

(5.33)

(5.32)

Найдем угол поворота вала двигателя, соответствующий величине перемещения в данном режиме:

(5.34)

Время работы двигателя с установившейся скоростью ωс = 100 рад/с,

(5.35)

Проверка двигателя по характеристикам заключается в сравнении суммарного фактического времени работы электропривода в цикле tф с заданным значением времени работы tp в исходных проектных данных. Для обеспечения требуемой производительности фактическое время работы должно быть меньше указанного.

Тогда:

(5.36)

Условие производительности выполнено, поскольку tф < tр. Далее следует проверка двигателя по нагреву путём сравнениния среднеквадратичного момента с допускаемым моментом двигателя при его работе с фактической продолжительностью включения ПВф.

Фактическая продолжительность включения:

(5.37)

Затем момент двигателя при ПВкат = 100%, ближайшем к ПВф:

(5.38)

Теперь найдем момент двигателя, допускаемый по нагреву для рассчитанного (фактического) графика нагрузки:

(5.39)

Рассчитываем среднеквадратичный момент двигателя при фактическом графике нагрузки по результатам предварительного расчета:

(5.40)

Так как выполнено условие Мсркв < Мдоп (27,7425 Н∙м < 53,9993 Н∙м), выбранный двигатель проходит по условиям нагрева, поэтому приступаем к детальному точному расчету выбранного электропривода. На основании расчётных данных, полученных выше построим нагрузочные диаграммы скоростей и моментов рабочего органа для каждого режима работы (рисунки 5.1, 5.2).

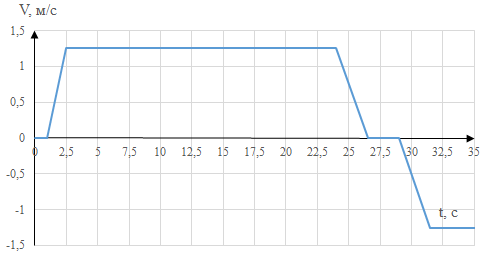


Рисунок 5.1. Тахограмма скорости от времени

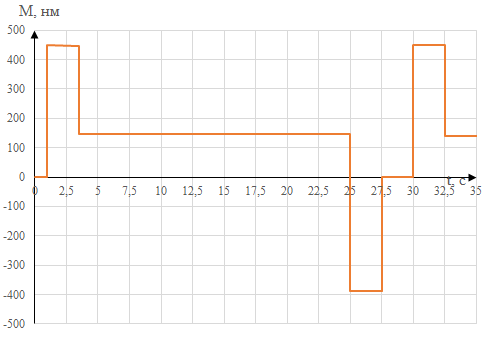


Рисунок 5.2. Нагрузочная диаграмма моментов от времени

### 5.2 Выбор преобразователя частоты

На рисунке 5.3 представлена функциональная двухзвенного преобразователя частоты (ПЧ) с неуправляемым выпрямителем и автономным инвертором напряжения со скалярной системой управления. Где ЗИ – задатчик интенсивности, ПИ – пропорционально-интегральный регулятор скорости, ФП – функциональный преобразователь, Блок ШИМ – производит расчёт скважностей для инвертора, АД – асинхронный двигатель с короткозамкнутым ротором.

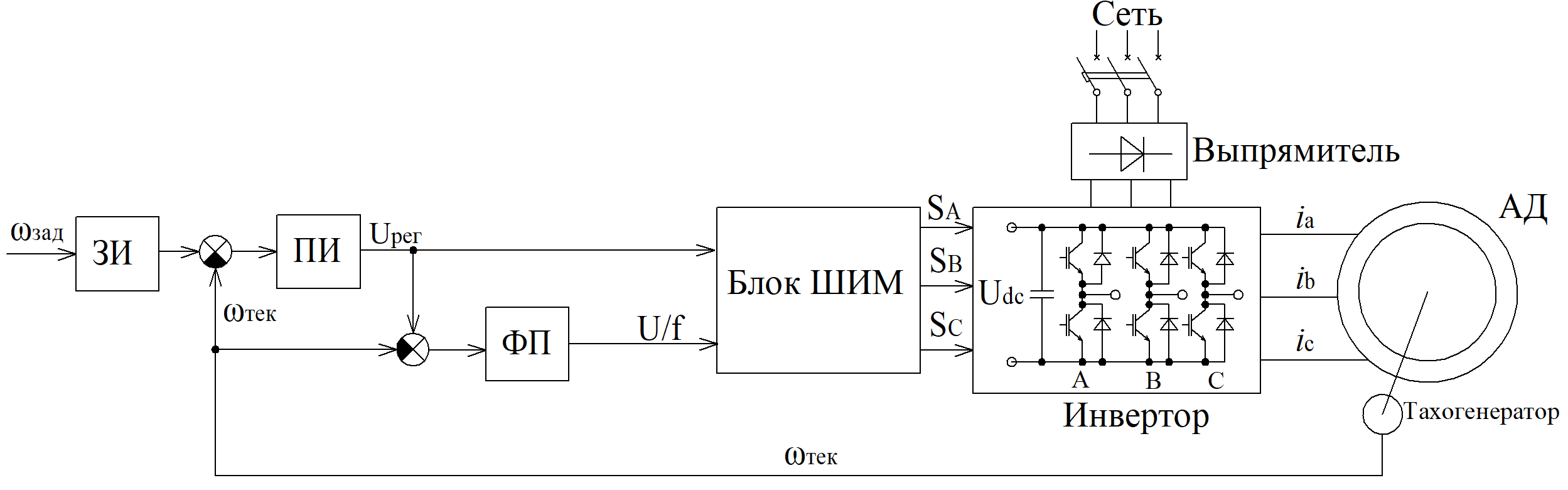


Рисунок 5.3. Функциональная схема электропривода рольганга

Скалярное управление (частотное) – метод управления бесщеточным электродвигателем переменного тока, который заключается в том, чтобы поддерживать постоянным отношение напряжение/частота во всем рабочем диапазоне скоростей, при этом контролируется только величина и частота питающего напряжения.

Основными техническими параметрами преобразователя частоты выбираются по следующим формулам:

(5.41)

где − напряжение питания преобразователя частоты, В; − напряжение питания двигателя, В.

(5.42)

где − номинальный ток ПЧ, А; − номинальный ток двигателя, А.

(5.43)

где − номинальная мощность ПЧ, кВт; − мощность двигателя, кВт.

На основании расчётов, технических требований к приводу, описанных в главе 3 данной работы и специфики привода рольганга, выберем силовой модуль преобразователя частоты фирмы Siemens модели SINAMICS G120 POWER (6SL3210-1NE23-2AG1) технические данные которого представлены в таблице 5.2.

Таблица 5.2. Технические данные преобразователя частоты

|  |  |
| --- | --- |
| Наименование, обозначение, ед. измерения | Величина |
| Номинальная мощность ПЧ, Рпч, кВт | 15 |
| Номинальный ток ПЧ, Iвх.пч, A | 32 |
| Номинальное напряжение питания, Uпч, В | 380-480 |
| КПД, η, % | 97 |
| Коэффициент мощности, cosφ | 0,9 |
| Диапазон изменения выходной частоты при векторном управлении fmin…fmax, Гц | 0 … 200 |
| Диапазон изменения выходной частоты при скалярном управлении fmin…fmax, Гц | 0 … 550 |
| Вес, кг | 5,1 |
| Степень защиты | IP20 |
| Наличие тормозного блока | Да |
| Габариты (ШхВхГ), мм: | 140х355х165 |
| Интерфейс | Modbus TCP, Modbus последоват., Ethernet |

Частотные преобразователи Siemens модели SINAMICS G120 POWER (6SL3210-1NE23-2AG1) с векторным и скалярным способом управления применяются для регулирования скорости вращения двигателей с высокой динамикой и повышенными требованиями к моменту и перегрузке. В отличие от системы тиристорный преобразователь – двигатель постоянного тока, система преобразователь частоты – асинхронный двигатель имеет ряд преимуществ, описанных ниже.

Регулируемый предел крутящего момента.

После регулировки скорости путем преобразования частоты можно установить соответствующий предел крутящего момента, чтобы защитить двигатель от повреждений. В настоящее время технологии преобразования частоты позволяет регулировать не только предел крутящего момента, но и точность регулирования крутящего момента, достигающую примерно 3-5 процентов.

Управляемый режим остановки.

Как и управляемое ускорение, режимом остановки также можно управлять в системе управления частотой, и можно выбирать различные режимы остановки (замедленная остановка, обычная остановка, замедленная остановка + торможение постоянным током). Это помогает уменьшить воздействие на механические компоненты и двигатели, тем самым делая всю систему более надежной и соответственно продлевая срок службы.

Энергосбережение

Энергосбережение преобразователя частоты в основном проявляется при использовании вентиляторов, насосов и других устройств, работающих в непрерывном режиме, к которым также относится рольганг.

Простой реверс двигателя

При использовании преобразователя частоты нет необходимости в дополнительных реверсивных устройствах для реализации реверса, так как преобразователь частоты в базовом оснащении имеет возможность изменения последовательности фаз выходного напряжения, что снижает затраты на техническое обслуживание экономит место для установки.

### 5.3 Система управления объектом, расчет регуляторов

Как правило, в каталогах [9], [10] на асинхронные двигатели приводятся следующие технические данные:

Рдв − номинальная мощность двигателя, кВт;

UH − номинальное фазное напряжение, В;

IH − номинальный ток статора двигателя, А;

nH − номинальная частота вращения, об⁄мин;

SH − номинальное скольжение, о.е.;

ηH − коэффициент полезного действия электродвигателя в режиме номинальной мощности (100 процентная нагрузка), %;

cosφH − коэффициент мощности в режиме номинальной мощности, о. е.;

ki – кратность пускового тока, о. е.;

kп – кратность пускового момента, о. е.;

kmax – кратность максимального момента, о. е.;

ki – кратность минимального момента, о. е.;

На основе каталожных данных этих данных производиться расчёт электрических параметров асинхронного двигателя.

Определение номинального скольжения:

(5.43)

Расчет величины критического скольжения:

(5.44)

Конструктивный коэффициент. Первоначально задается значение коэффициента C1 в диапазоне от 1,02 до 1,05 для предварительного расчета параметров схемы замещения.

После расчета производится сравнение полученного значения с первоначально выбранными и при необходимости расчет повторяется для другого значения C1.

Приведенное активное сопротивление ротора:

где ΔPмех=0,01·Pдв – механические потери Вт;

(5.45)

Пренебрегая потерями в ферромагнитном сердечнике и добавочными потерями в машине, можно считать, что мощность, выделяемая в активном сопротивлении статорной обмотки:

(5.46)

Отсюда найдем величину активного сопротивления статора:

(5.47)

Приведенная индуктивность рассеяния статора и ротора:

где f – частота питающей сети, Гц.

(5.48)

Индуктивность обмотки статора и приведенная индуктивность обмотки ротора:

где – критический момент двигателя.

(5.49)

Индуктивность контура намагничивания:

(5.50)

Проверка правильности выбора коэффициента приведения:

(5.51)

Коэффициент С1 равен значению принятого коэффициента С1 = 1,02, значит расчёты были проведены верно, сведём расчётные данные электрических параметров двигателя в одну таблицу (таблица 5.2).

Таблица 5.2. Обмоточные данные асинхронного двигателя

|  |  |
| --- | --- |
| Параметр | Значение |
| Активное сопротивление статора R1, Ом | 1,5122 |
| Активное приведенное сопротивление ротора R'2, Ом | 0,4636 |
| Индуктивность статора Ls, Гн | 0,0849 |
| Приведенная индуктивность ротора L'r, Гн | 0,0849 |
| Индуктивность рассеяния статора L1𝜎 и ротора L2𝜎, Гн | 0,002 |
| Взаимная индуктивность Lµ, Гн | 0,0829 |

Расчёт системы управления электропривода

Структурная схема векторной системы управления представлена на рисунке 5.4. Канал потокосцепления ротора состоит из регулятора потока Wрп, регулятора тока Wрт и преобразователя частоты в виде инерционного звена первого порядка. Контур регулирования скорости ротора состоит из регуляторов скорости Wрс, регулятора тока Wрт и преобразователя частоты приведённым к виду инерционного звена первого порядка.

### 5.4 Расчёт параметров регулятора

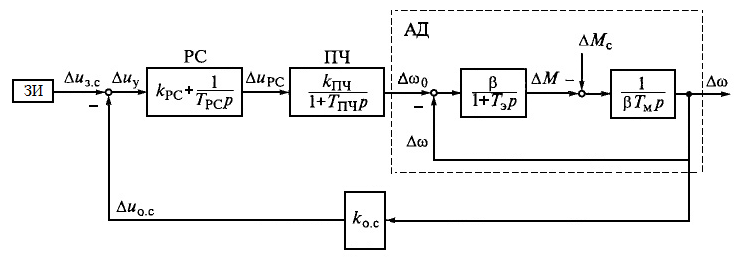
Структурная схема системы преобразователь частоты – асинхронный двигатель представлена со скалярной системой управления на рисунке 5.4.

Передаточную функцию преобразователя частоты представлена в следующем виде:

(5.52)

где − эквивалентный передаточный коэффициент преобразователя частоты; − эквивалентная постоянная времени преобразователя.

Эквивалентный передаточный коэффициент преобразователя частоты найдём из соотношения:



(5.53)

Рисунок 5.4. Структурная схема ПЧ-АД при скалярной системе управления

Значение эквивалентной постоянной времени преобразователя частоты:

(5.54)

где − время задержки ШИМ; − эквивалентная постоянная времени преобразователя частоты, с; − время затрачиваемое процессором на вычисление и преобразование сигналов

Время задержки ШИМ:

(5.55)

где − частота ШИМ, ; Частота ШИМ , тогда:

(5.56)

Численное значение постоянной времени преобразователя представлено ниже:

Тогда передаточная функция преобразователя частоты:

(5.57)

Сопротивление ротора двигателя Rp, Ом; − большая постоянная времени электродвигателя, с.

(5.58)

где − активное сопротивление на выходе автономного инвертора напряжения выходного фильтра, Ом − активное сопротивление обмотки статора электродвигателя, Ом; − приведенное активное сопротивление обмотки ротора электродвигателя,Ом; − коэффициент электромагнитной связи ротора асинхронного двигателя, Ом.

Активное сопротивление на выходе автономного инвертора напряжения выходного фильтра  Коэффициент электромагнитной связи ротора электродвигателя определяется по следующей формуле:

(5.59)

где − взаимная индуктивность контура намагничивания, Гн; L'r − полная индуктивность фазы ротора двигателя, Гн.

Индуктивность рассеяния электродвигателя определяется по формуле:

(5.60)

где − полная индуктивность фазы статора, Гн.

Сопротивления асинхронного двигателя:

(5.61)

Некомпенсируемая постоянная времени, Электромагнитная постоянная времени электродвигателя:

Электромагнитный момент АД:

(5.62)

(5.63)

где − число пар полюсов обмотки статора.

Передаточная функция механической части электродвигателя:

(5.64)

где − суммарный момент инерции АД,

(5.65)

Суммарный момент инерции АД:

(5.66)

где − момент инерции асинхронного двигателя − момент инерции механизма участка рольгангаТогда:

(5.67)

Передаточная характеристика механической части АД:

(5.68)

Вычислим коэффициент изменения момента двигателя при отклонении его скорости при U = const.

(5.69)

Рассчитаем электромагнитную постоянную времени двигателя, а также дополнительные вспомогательные:

(5.71)

(5.70)

Коэффициент обратной связи по скорости:

(5.72)

(5.73)

где − номинальная скорость электродвигателя, об/мин. Тогда:

(5.74)

Постоянная времени регулятора скорости:

(5.75)

где – коэффициент компенсации электромагнитной переменной = 3.

Тогда:

(5.76)

Коэффициент регулятора скорости:

(5.77)

Тогда передаточная функция регулятора скорости:

(5.78)

А передаточная функция фильтра на входе контура скорости:

(5.79)

### 5.6 Моделирование в MathLab Simulink

Математическая модель скалярной системы управления электроприводом рольганга приведена на рисунке 5.5. График угловой скорости представлен на рисунке 5.6, на рисунке 5.7 показан график момента на выходе редуктора. В соответствии с выдвинутыми требованиями ошибка регулирования по скорости не должна превышать 5%, при моделировании получили перерегулирование по скорости на уровне 4,3%, а ускорение равным 0,5 м/с2, при максимально допустимом 1 м/с2.

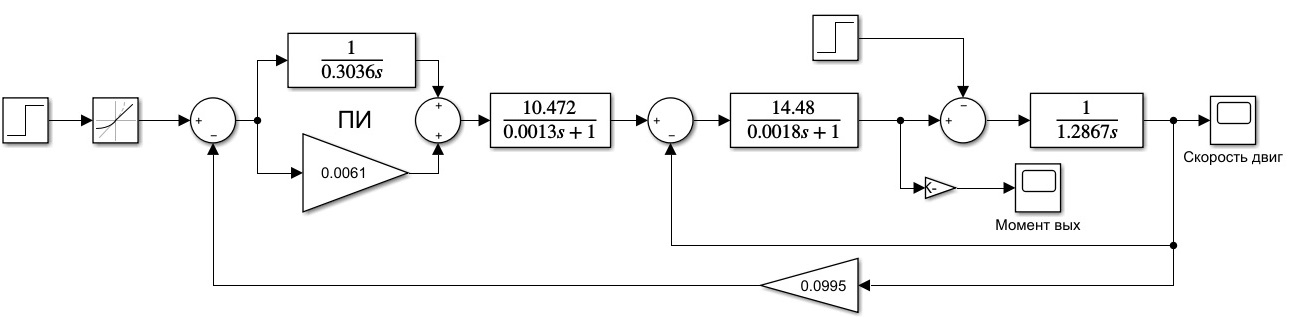


Рисунок 5.5. Математическая модель системы управления

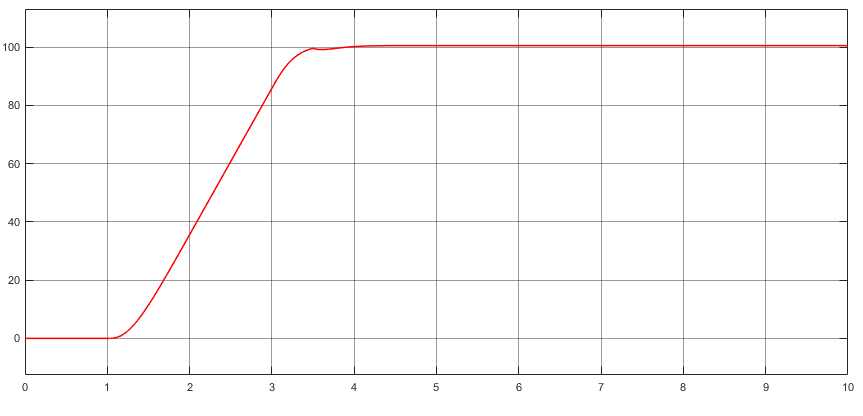


Рисунок 5.6. График угловой скорости двигателя

Требований по моменту в данной работе не выдвигалось, из рисунка 5.7 можем увидеть перерегулирование на уровне равное 17,98%, пульсации по моменту не критичны для нормальной работы привода и системы распознавания дефектов.

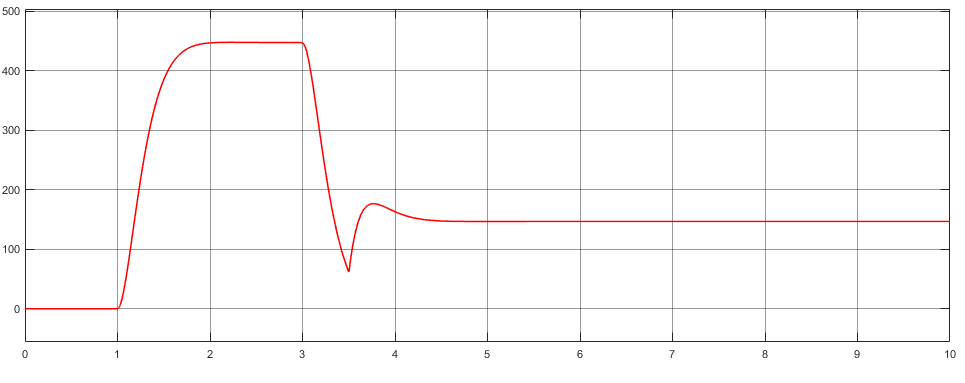


Рисунок 5.7. График момента на выходе редуктора

6 ТРЕБОВАНИЯ К ПРОЕКТИРУЕМОЙ СИСТЕМЕ РАСПОЗНАВАНИЯ ДЕФЕКТОВ НА ПОВЕРХНОСТИ МЕТАЛЛА

Требования к точности.

- согласно ГОСТ 52246-2016 присваивать определённую группу качества проката с точностью не менее 90%, в соответствии с полученными данными (таблица 6.1) для цинкового покрытия (Ц), железоцинкового покрытия (ЖЦ), цинкалюминиевое покрытия (ЦА), цинкалюмомагниевое (ЦАМ).

Таблица 6.1. Допускаемые дефекты покрытия

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Показатель покрытия | | | Общие  допускаемые  дефекты | Дефекты, допускаемые в зависимости от типа, вида и группы отделки покрытия |
| Тип | Вид | Группа  отделки |
| Ц | Н | Обычного  качества | Следы от перегибов полосы и  регулирующих роликов;  царапины и потертость (темные  точки и пятна), не нарушающие  сплошность покрытия;  светлые и матовые пятна;  неравномерность окраски пассивной пленки | Наплывы, натеки, наслоения без растрескивания, местная шероховатость покрытия (сыпь); крупинки; неравномерность кристаллизации покрытия. Наплывы без растрескивания, местная шероховатость покрытия (сыпь);крупинки цинка в виде отдельных точек; неравномерность кристаллизации покрытия |
| ЖЦ,  ЦА,  ЦАМ | - |
| Ц | М | То же, что и для цинкового покрытия вида Н, а также узор кристаллизации ≤ 20 мм от кромок, темные вкрапления и лунки ≤5 мм в виде  отдельных участков и полос, отдельные  кристаллы цинка по всей поверхности |
| Показатель покрытия | | | Общие  допускаемые  дефекты | Дефекты, допускаемые в зависимости от типа, вида и группы отделки покрытия |
| Тип | Вид | Группа  отделки |
| Ц | М | У | Следы от перегибов полосы и регулирующих роликов; царапины и потертость (темные точки и пятна), не  нарушающие  сплошность покрытия; светлые и матовые пятна; неравномерность окраски пассивной пленки | То же, что и для цинкового покрытия  вида Н улучшенной отделки У, а также темные точки и дорожки (следы) от деформированных вкраплений, лунок, отдельных кристаллов; Размытый узор кристаллизации на расстоянии не более 20 мм от кромок |
| Ц | Н | У | Темные точки и дорожки (следы) от деформационных наплывов, натеков, наслоений, крупинок без растрескивания; местная шероховатость покрытия (сыпь); матовый и размытый узор кристаллизации покрытия |
| ЖЦ,  ЦА,  ЦАМ | - | В |
| ЖЦ,  ЦА,  ЦАМ | - | В | Допустимые дефекты для лицевой стороны проката устанавливают по  согласованию сторон. Обратная сторона проката должна соответствовать требованиям к качеству поверхности покрытий группы отделки У |
| Ц | М |

- в соответствии и ГОСТ 20847-75 определять с точностью выше 85% следующие типы самых распространённых дефектов на поверхности горячекатаных листов: вдавленные окалины, пятна, волосные трещины, рябизна, посторонние включения и царапины.

Продолжение таблицы 6.1 Допускаемые дефекты покрытия

Требования к результатам анализа

-по окончанию анализа распознанные дефекты, их количество относительно друг друга и чистой поверхности должны быть представлены в удобном для визуального распознавания виде.

-на основе распознанных данных программе следует составить рекомендации для текущего оборудования (шлифовка валков, лучшая зачистка слябов и т.д.).

-названия распознаваемых дефектов следует классифицировать в соответствии с ГОСТ 21014-88.

Требования к времени распознавания.

Производство проката носит непрерывный характер, поэтому система компьютерного зрения должна успевать распознавать дефекты, во время движения листопроката с номинальной скоростью 1,25 м/с.

Требования к экономической составляющей.

- система компьютерного зрения должна обладать минимальными требованиями к обслуживанию и иметь возможность быстрой замены этого самого оборудования;

- система должна иметь дешёвую возможность переобучения для возможности расширения классификации дефектов, без потери точности;

- система должна давать меньше ошибок или отклонений в сравнении с человеком для повышения качества обработки информации.

Требования к встраиваемости в существующую систему и интерфейсу.

- разработанная система должна легко встраиваться в существующую систему, без её кардинальной переработки.

- система компьютерного зрения должна позволять накапливать собранную информацию на внешних накопителях и существующих локальных дисках;

- передача данных должна производиться через последовательное соединение и может осуществляться в форме обычного выхода RS232, Ethernet.

- система компьютерного зрения должна удовлетворять современным требованиям визуального и эстетического оформления для лёгкой навигации внутри программы и возможности быстрого обучения сотрудников.

7 РАЗРАБОТКА АЛГОРИТМА РАБОТЫ ПРОГРАММЫ ДЛЯ РАСПОЗНАВАНИЯ ДЕФЕКТОВ НА ПОВЕРХНОСТИ МЕТАЛЛА

### 7.1 Обзор существующих алгоритмов машинного обучения

Линейная регрессия

Этот метод используется для оценки реальных значений (стоимость домов, количество звонков, общий объем продаж и т.д.) на основе непрерывных переменных (рисунок 7.1). Здесь мы устанавливаем взаимосвязь между независимыми и зависимыми переменными, подбирая лучшую линию. Эта линия наилучшего соответствия известна как линия регрессии и представлена ​​линейным уравнением:

(7.1)

где y(x) – зависимая переменная; *а* – наклон; *x* – независимая переменная; *b* – смещение.

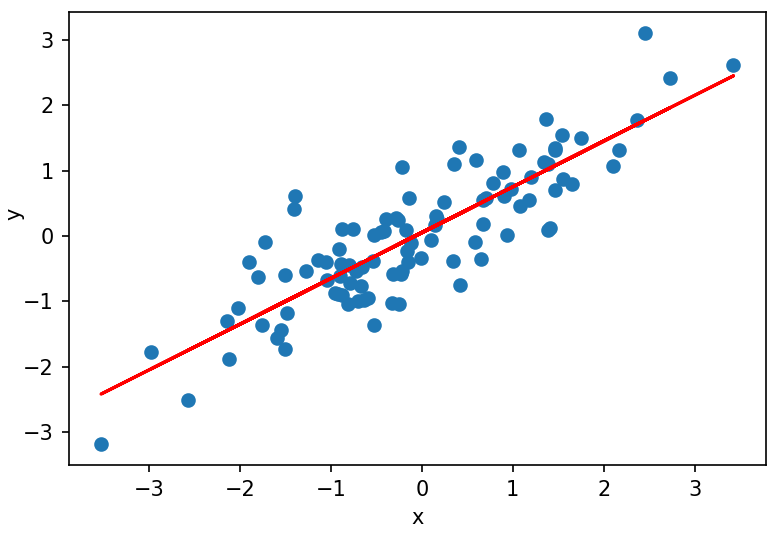


Рисунок 7.1. Линейная регрессия

Логистическая регрессия

Данный метод используется для оценки дискретных значений (на основе заданного набора независимых переменных. Проще говоря, он предсказывает вероятность возникновения события, подгоняя данные к логистической функции (рисунок 7.2).

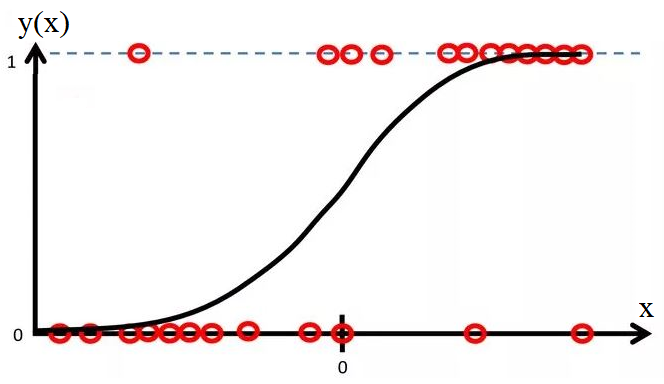


Рисунок 7.2. Логическая регрессия

Дерево решений

Алгоритм дерева решений принадлежит к семейству алгоритмов обучения с учителем. В отличие от других алгоритмов контролируемого обучения, алгоритм дерева решений также может использоваться для решения задач регрессии и классификации. Целью использования дерева решений является создание модели обучения, которую можно использовать для прогнозирования класса или значения целевой переменной путем изучения простых правил принятия решений, выведенных из предшествующих данных (данных обучения). В Дереве решений для прогнозирования метки класса для записи мы начинаем с корня дерева. Мы сравниваем значения корневого атрибута с атрибутом записи. На основе сравнения мы переходим по ветви, соответствующей этому значению, и переходим к следующему узлу (рисунок 7.3).

SVM (машина опорных векторов)

Задача алгоритма машины опорных векторов – найти гиперплоскость в N-мерном пространстве (N - количество функций), которая четко классифицирует точки данных.

Чтобы разделить два класса точек данных, можно выбрать множество возможных гиперплоскостей. Наша цель – найти плоскость с максимальным запасом, то есть максимальным расстоянием между точками данных обоих классов. Увеличение максимального расстояния дает некоторое подкрепление, так что будущие точки данных можно классифицировать с большей уверенностью.

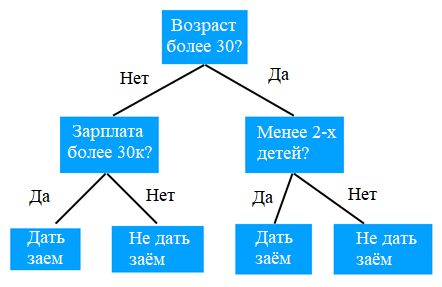
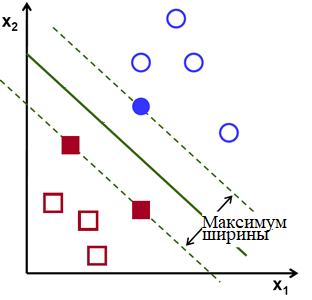


Рисунок 7.3. Дерево решений



Оптимум

Рисунок 7.4. Машина опорных векторов

KNN (метод K-ближайших соседей)

KNN часто используется в простых системах b технологиях распознавания изображений и моделях принятия решений. Как и почти все остальное, KNN работает благодаря глубоко укоренившимся математическим теориям, которые он использует:

(7.2)

При реализации KNN первым шагом является преобразование точек данных в векторы признаков или их математические значения. Затем алгоритм работает, находя расстояние между математическими значениями этих точек. Самый распространенный способ найти это расстояние – это евклидово расстояние. Далее точки объединяются в группы (рисунок 7.5).

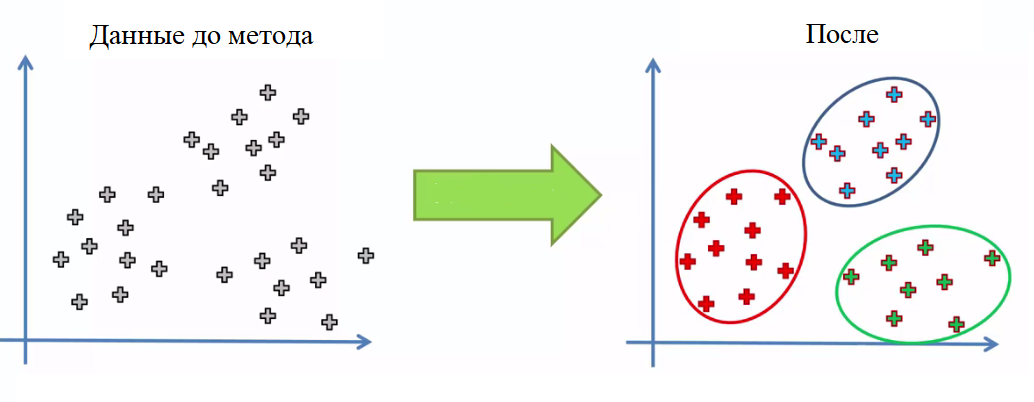


Рисунок 7.5. Метод k-ближайших соседей

### 7.2 Свёрточная нейронная сеть (ConvNet / CNN)

Алгоритмы, описанные выше, позволяют решить задачу классификации дефектов на поверхности листопроката, но точность предсказаний при этом будет не высока [15]. Поэтому для решения поставленной задачи требуется такой алгоритм, который будет производить анализ намного точнее и иметь гибкость настройки, в данном случае подходит алгоритм, основанный на свёрточных нейронных сетях [16]. Архитектура данной сети представлена на рисунке 7.6.

Сверточная нейронная сеть (ConvNet / CNN) – это алгоритм глубокого обучения, который может принимать входное изображение, назначать важность различным аспектам или объектам на изображении и иметь возможность отличать один от другого. Предварительная обработка, требуемая в ConvNet, намного ниже по сравнению с другими алгоритмами классификации.

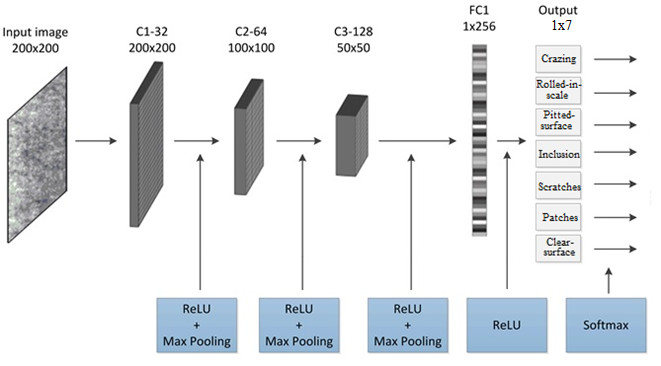


Рисунок 7.6. Архитектура CNN для задачи классификации

CNN может успешно фиксировать пространственные и временные зависимости в изображении с помощью применения соответствующих фильтров. Архитектура лучше соответствует набору данных изображения за счет уменьшения количества задействованных параметров и возможности повторного использования весов. Другими словами, сеть можно обучить, чтобы лучше понимать сложность изображения.

Входное изображение

На рисунке 7.7 представлено RGB изображение, разделенное тремя цветовыми плоскостями – красной, зеленой и синей. Роль CNN состоит в том, чтобы преобразовать изображения в форму, которую легче обрабатывать, без потери функций, которые имеют решающее значение для получения хорошего прогноза. Это важно, когда мы должны разработать архитектуру, которая не только хороша для изучения функций, но и масштабируется до массивных наборов данных [15].

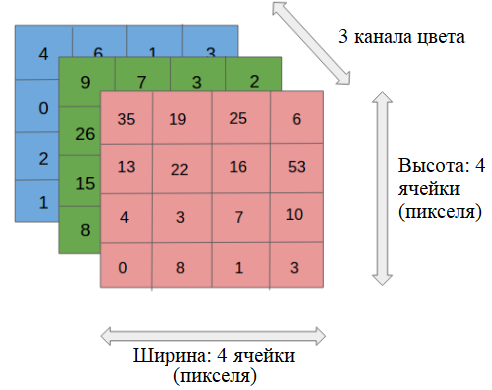


Рисунок 7.7. Входное RGB-изображение 4х4х3

Слой свертки – ядро

Целью операции свертки является извлечение высокоуровневых функций, таких как края, из входного изображения. Свёрточные сети не обязательно должны быть ограничены только одним сверточным слоем. Обычно первый ConvLayer отвечает за захват низкоуровневых функций, таких как края, цвет, ориентация градиента и т.д. С добавлением слоев архитектура также адаптируется к высокоуровневым функциям, давая нам сеть, которая имеет полное понимание изображений в наборе данных.

Есть два типа результатов операции: один, в котором размерность свернутого элемента уменьшается по сравнению с исходным, и другой, в котором размерность либо увеличивается, либо остается прежней.

Уровень объединения

Подобно сверточному слою, уровень объединения отвечает за уменьшение пространственного размера свернутого элемента. Это необходимо для уменьшения вычислительной мощности, необходимой для обработки данных за счет уменьшения размерности.

Существует два типа объединения: максимальное объединение и среднее объединение. Max Pooling возвращает максимальное значение из части изображения, покрытой ядром. С другой стороны, Average Pooling возвращает среднее значение всех значений из части изображения, покрытой ядром.

Max Pooling также действует как подавитель шума. Он полностью отбрасывает шумные активации, а также выполняет устранение шума вместе с уменьшением размерности. С другой стороны, Average Pooling просто выполняет уменьшение размерности как механизм подавления шума. Следовательно, мы можем сказать, что Max Pooling работает намного лучше, чем Average Pooling.

Сверточный уровень и уровень объединения вместе образуют i-й уровень сверточной нейронной сети. В зависимости от сложности изображений количество таких слоев может быть увеличено для еще большего захвата низкоуровневых деталей, но за счет большей вычислительной мощности. После прохождения, описанного выше процесса мы успешно позволили модели понять главные особенности и патерны изображения. Двигаясь дальше, мы собираемся сгладить окончательный результат и передать его в обычную нейронную сеть для целей классификации [16].

Классификация – полностью подключенный уровень (уровень FC)

Добавление полностью связанного слоя (обычно) простейший способ изучения нелинейных комбинаций высокоуровневых функций, представленных выходными данными сверточного слоя. Теперь, когда мы преобразовали наше входное изображение в подходящую форму для нашего многоуровневого персептрона, мы сгладим изображение в вектор-столбец. Сглаженный вывод подается в нейронную сеть с прямой связью и с использованием алгоритма обратного распространения ошибки, применяемого к каждой итерации обучения. В течение ряда эпох модель способна различать доминирующие и некоторые низкоуровневые функции в изображениях и классифицировать их с помощью техники Softmax Classification.

### 7.3 Обзор существующих инструментов и библиотек для проектирования компьютерного зрения

Как и любое технологическое решение в наши дни, компьютерное зрение на производстве может иметь довольно универсальный технический стек. В этом разделе мы собираемся взглянуть на некоторые из наиболее широко используемых инструментов и библиотек, которые используются в CV.

OpenCV – один из наиболее широко известных, используемых инструментов компьютерного зрения. Он написан на C ++ и имеет множество языковых привязок: Python, Java, Matlab, JavaScript. Он имеет библиотеку функций и алгоритмов для компьютерного зрения, обработки изображений и числовых алгоритмов общего назначения с открытым исходным кодом, которые позволяют пользователям интерпретировать изображения, калибровать камеру по заранее установленному стандарту, уменьшать или даже устранять оптические искажения и анализировать движение объекта. Он также имеет ряд других удобных функций, таких как 3D-реконструкция, сегментация объектов, распознавание жестов и т.д.

Фреймворки AForge.NET/Accord.NET и песочница компьютерного зрения – хороший набор инструментов для решений .NET. Оба поставляются с широким спектром доступных инструментов, предоставляющих пользователям доступ к резюме, библиотекам обработки изображений и видео, расширенным алгоритмам анализа и многому другому. CV Sandbox, программный пакет с открытым исходным кодом, позволяет решать различные задачи, связанные с CV (например, автоматизация машинного зрения, обработка изображений / видео, многокамерный просмотр, сканирование штрих-кода и т.д.)

SimpleCV. SimpleCV – это фреймворк для создания приложений CV. Благодаря быстрому созданию прототипов и взаимодействию со многими другими инструментами SimpleCV является отличным вариантом для технологий компьютерного зрения на производстве. Однако его производительность может быть не такой высокой, как у OpenCV, особенно в среде с высокой нагрузкой. Другая проблема – это довольно небольшое сообщество, которому может не хватать поддержки с точки зрения документации и решения проблем.

PyTorch – это библиотека Machine Learning (ML) с открытым исходным кодом, которая часто используется для компьютерного зрения и обработки естественного языка. Она подойдет для разработки на Python и C ++.

Компьютерное видение –это междисциплинарная область. Несмотря на то, что эти инструменты и платформы имеют более широкую область применения, они также могут стать мощным инструментом для решений компьютерного зрения в Индустрии 4.0.

CUDA – это платформа для параллельных вычислений, которая помогает пользователям запускать сложные процессы в графических процессорах, а не в центральном процессоре. Его инструментарий включает библиотеку NVIDIA Performance Primitives, которая представляет собой сборник различных функций обработки изображений, видео и сигналов. CUDA хорошо работает с множеством библиотек и инструментов, что делает его идеальным инструментом для ускорения алгоритмов CV.

TensorFlow. набирает все большую популярность благодаря своей мощности и простоте использования. Помимо некоторых хороших инструментов для обработки или классификации изображений, он также помогает вам внедрить глубокое обучение в компьютерное зрение. С помощью Python API можно выполнять обнаружение лиц и выражений лица. К недостаткам TensorFlow можно отнести а) высокое энергопотребление, б) необходимость журнала ресурсов и в) необходимость более высокой квалификации разработчиков.

Keras – это библиотека нейронной сети с открытым исходным кодом, которая может работать поверх TensorFlow, Microsoft Cognitive Toolkit, R, Theano или PlaidML. Это удобный, модульный и расширяемый инструмент, который расширяет возможности при работе с глубокими нейронными сетями.

Библиотека загрузки данных Nvidia (DALI) – это портативная библиотека с открытым исходным кодом, которая обеспечивает декодирование и увеличение изображений, видео и речи для приложений глубокого обучения.

H2O.ai – это платформа машинного обучения с открытым исходным кодом и линейной масштабируемостью. Он поддерживает большинство статистических алгоритмов и алгоритмов машинного обучения, а также предлагает функции AutoML.

OpenVINO (визуальный вывод и оптимизация нейронной сети) – это набор инструментов, написанный на C ++ и Python. Он состоит из множества предварительно обученных моделей, что позволяет вам оптимизировать и развертывать их, а не разрабатывать.

Kubeflow – это бесплатная платформа машинного обучения с открытым исходным кодом, которая была разработана, чтобы упростить и скоординировать выполнение рабочих процессов машинного обучения в кластерах Kubernetes.

Microsoft Cognitive Toolkit (ранее CNTK) – это набор инструментов с открытым исходным кодом для решений глубокого обучения. Это также один из первых наборов инструментов, поддерживающих формат Open Neural Network Exchange ONNX.

Matlab - отличный инструмент для создания приложений для обработки изображений. Этот инструмент помогает создавать ясный и краткий код (по сравнению с C ++), упрощая чтение и отладку, а значит, и поддержку. Это также позволяет быстро создавать прототипы. С другой стороны, а) Matlab – платный инструмент, и б) он может работать довольно медленно во время выполнения вычислений.

FastAI для PyTorch. Эта библиотека, разработанная некоммерческой группой исследователей, упрощает обучение быстрых и точных нейронных сетей с упором на глубокое обучение.

### 7.4 Алгоритм работы программы

Виды дефектов горячекатаных листов и полос очень многочисленны. Некоторые из них специфичны и относятся только к какому-либо конкретному виду продукции, поэтому далее будут рассмотрены самые распространённые из них (рисунок 7.8), для данных дефектов возьмём датасет NEU (Northeastern University) surface defect database – набор данных по поверхностным дефектам горячего проката.

Все дефекты разделены на шесть категорий и одну дополнительную, составленную вручную:

* царапины;
* вдавленные окалины;
* пятна;
* волосные трещины;
* рябизна;
* посторонние включения.
* чистая поверхность (дополнение датасета)

В наборе для каждого класса имеется 300 изображений, размером 200 × 200 пикселей, а также 300 изображений чистой поверхности. Первоначально требуется обучить свёточную сеть при помощи размеченных данных из используемого датасета, а после удачной итерации обучения сохранить веса и архитектуру полученной модели [17]. Алгоритм работы обучения вёрточной нейронной сети приведён на рисунке 7.8.

Для начала подключается Гугл.Диск, на который будут сохраняться модели и с которого будет загружаться датасет и тестовые изображения для проверки работоспособности сети, также подключается дополнительные библиотеки машинного обучения, работы с файлами, визуализации результатов и для облегчения выполнения математических операций с матрицами и векторами. Архитектура сети по большей части определяется эмпирическим путём, так как увеличение параметров сети не всегда ведёт к улучшению итогового результата. Обучение будет остановлено если точность распознавания превысит 93% (Е) или если сеть закончит тренировку на заданном количестве эпох обучении (Epochs), после идёт сохранение весов и архитектуры модели на облачном хранилище, для последующей загрузки без необходимости обучать сеть заново.

Данный этап может повторяться несколько раз, пока не будет достигнут оптимальный результат – параметры и гиперпараметры сети представлены в минимально необходимом количестве (минимально нужное количество весов модели), а также количество эпох обучения не позволяет переобучить систему и ухудшить конечный результат.

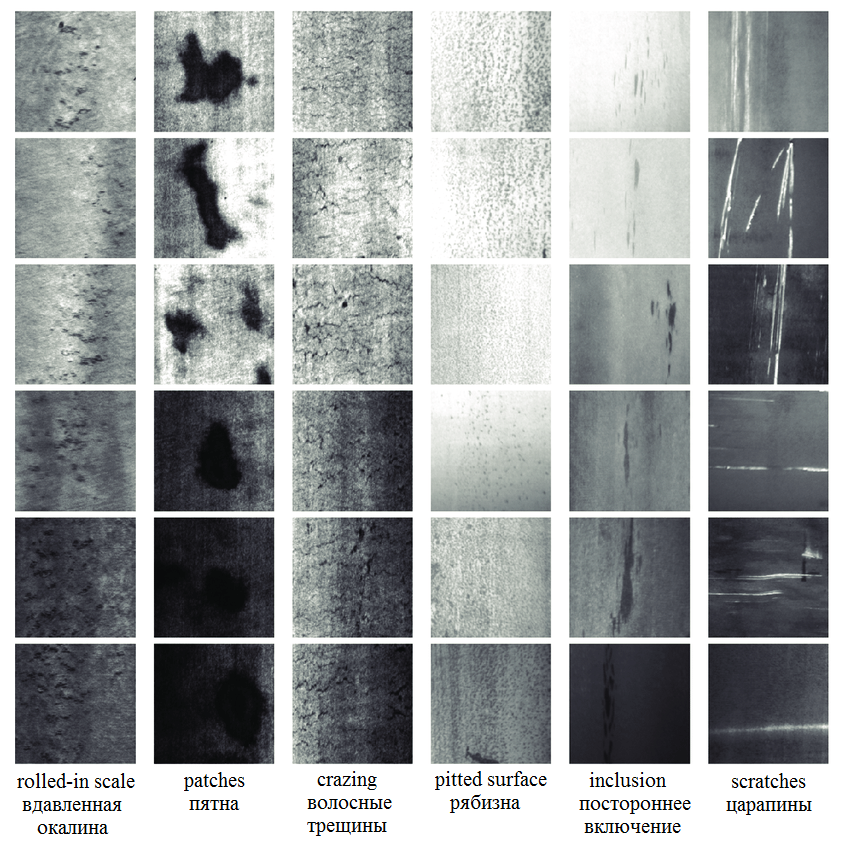


Рисунок 7.8. Примеры дефектов металла

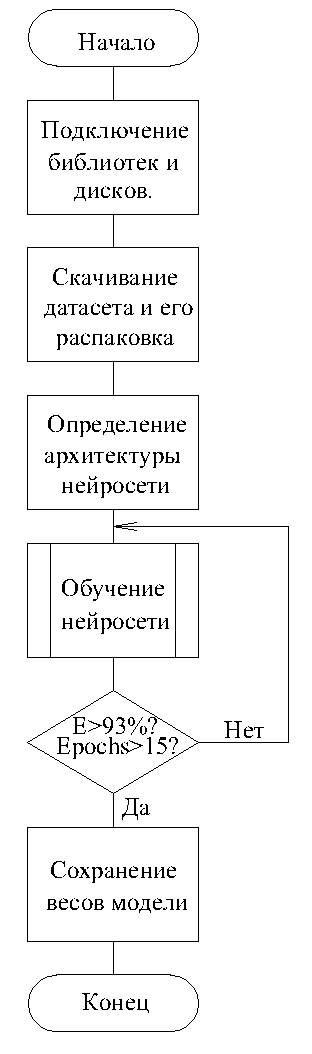


Рисунок 7.8. Алгоритм обучения свёрточной нейронной сети

После сохранения весов и архитектуры, следует этап загрузки данной модели, данная операция требуется из соображений оптимизации времени, т.к. обучение модели проходит примерно за 40 минут и конечный результат неопределённа из-за стохастического эффекта определения весов, целесообразно сохранить удавшуюся модель, чтобы в будущем была возможность её загрузки без дополнительных затрат времени. Далее на загруженной модели проводится ряд тестов для определения её качества и скорости работы. Алгоритм работы на данном этапе представлен на рисунке 7.9.

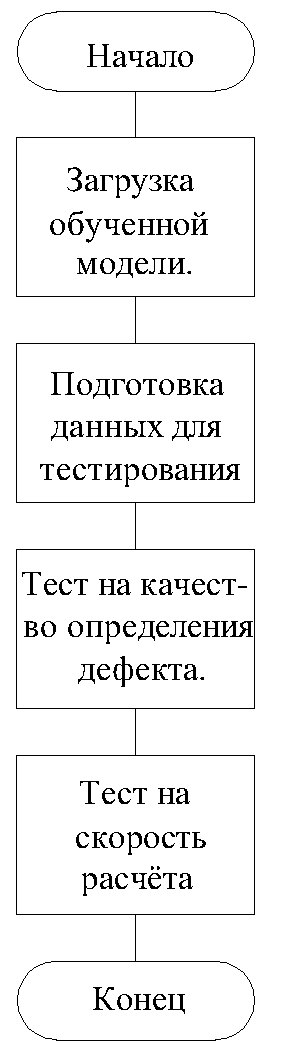


Рисунок 7.9. Алгоритм тестирования обученной нейронной сети

После обучения и проверки свёрточой нейронной сети производится работа над основным участком работы программы, а именно определение дефектов на листе проката с помощью специализированной камеры для машинного обучения. Особенность данных камер заключается в более прочном и защищенном исполнении специально для производственных условий, а также способности передавать «сырое», необработанное изображение высокой чёткости и контрастности. Расчёт периодичности снимков камеры, её высоту над поверхностью проката, выполняются на основании исходных параметров камеры такие как разрешение и угол обзора. Алгоритм работы представлен на рисунке 7.10.

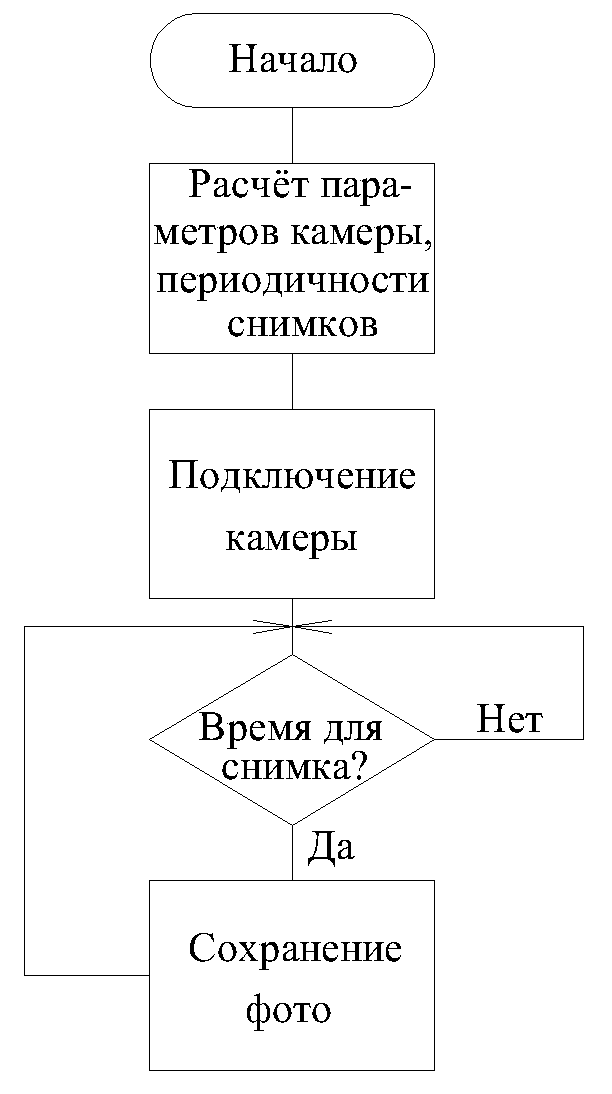


Рисунок 7.10. Алгоритм работы камеры

Снимки с камеры, полученные на предыдущем этапе отправляются на предобработку и далее в свёрточную нейросеть для определения дефекта. Предобработка заключается в том, что исходное изображение разбивается на фрагменты, подробнее процесс описан в главе ниже. Результаты обработки сохраняются в специальный массив, чтобы после на основе этих данных построить графики, гистограммы, удобные для восприятия человеком, а также вынести рекомендации по эксплуатации оборудования. Алгоритм работы представлен на рисунке 7.11.

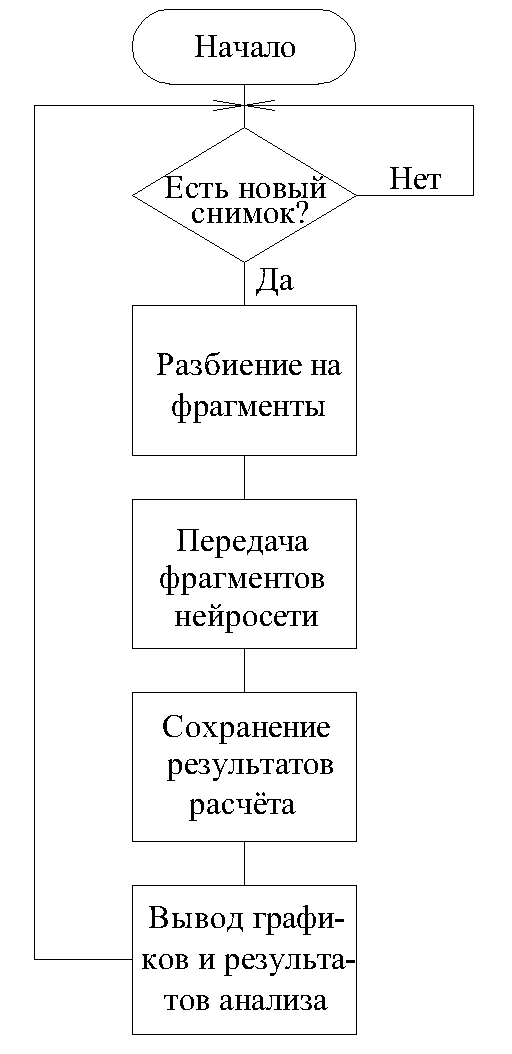


Рисунок 7.11. Алгоритм работы свёрточной нейронной сети при распознавании дефектов на поверхности металла

Также дополнительно потребуется составить 300 изображений чистой поверхности металла и дополнить ими датасет. Сам алгоритм заключается в распознавании отдельных фрагментов кадра, полученного с камеры. Исходное изображение разбивается на участки 200х200 пикселей (рисунок 7.12), фрагменты отправляются на нейросеть, а после результат анализа записывается в память, для последующей обработки. Обработка заключается в том, что на основе полученных данных строятся графики и диаграммы распространённости дефектов и выносятся рекомендации относительно прокатного оборудования, а также в соответствии с ГОСТ 5246-2016, табл. 10 присваивается категория качества поверхности листопроката.

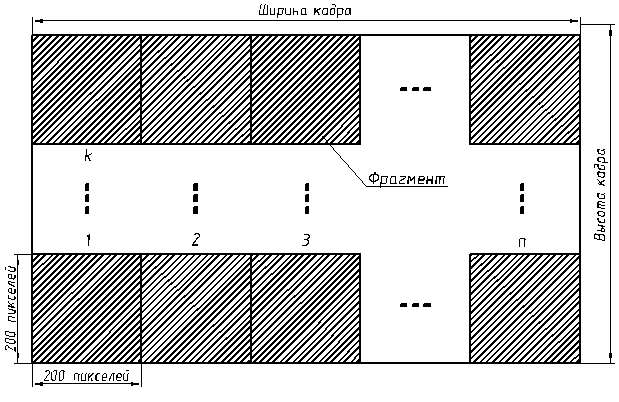


Рисунок 7.12. Разбиение кадра на фрагменты

### 7.5 Расчёт высоты камеры над поверхностью

Для более полноценного использования оборудования целесообразно, чтобы в объектив камеры попадал весь участок листопроката, но при этом без рамок вокруг. Для выбранной камеры TDS-VLXT-123C-FO с параметрами, приведёнными в таблице 7.1 вычислим высоту над полотном при помощи формул ниже.

Так как камера используется в условиях повышенных температур, то чтобы предотвратить повреждение оборудования, камера оснащена с аварийным отключением. Функция Device Temperature Status Transition Selector позволяет выбирать различные пороговые значения для температур: Normal To High: свободно программируемое значение, High To Exeeded: фиксированное значение (при превышении запись изображения останавливается), Exeeded To Normal: свободно программируемое значение температуры для безошибочного повторного включения камеры.

Таблица 7.1. Параметры камеры TDS-VLXT-123C-FO

|  |  |
| --- | --- |
| Модель матрицы | Sony IMX253 |
| Разрешение | 4096 × 3000 пикселей |
| Угол обзора | 720 |
| Тип матрицы | Цифровая RGB |
| Частота съёмки | 46 кад/сек, при глубине цвета 12 бит |
| Буфер изображений | 1024 Мб |
| Интерфейс передачи сигнала | 10 Гбит Ethernet |
| Температура работы | -10…+70 0С |
| Степень защиты | IP68 |

Согласно таблице 7.1 ширина кадра камеры равна 4096 пикселей, а высота кадра равна 3000 пикселей, угол камеры равен 720, а также на основании рисунка 7.13 вычислим высоту камеры над поверхностью. Построим пирамиду с основанием в виде прямоугольника ABCE (рисунок 7.14), где АВ – ширина кадра, АЕ – высота, а АН высота камеры над поверхностью и выделим треугольник MON (рисунок 7.15), из которого и найдём ОН – высоту камеры над поверхностью.

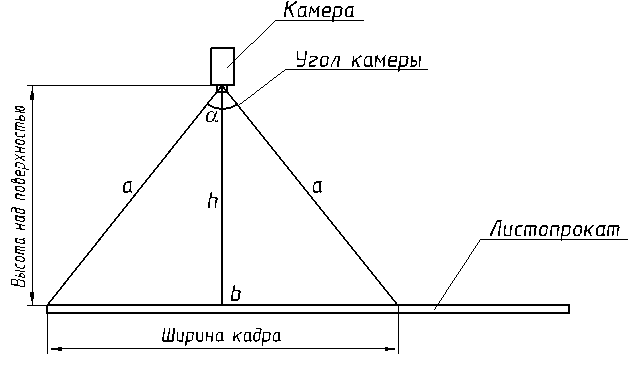


Рисунок 7.13. Камера. Вид сбоку

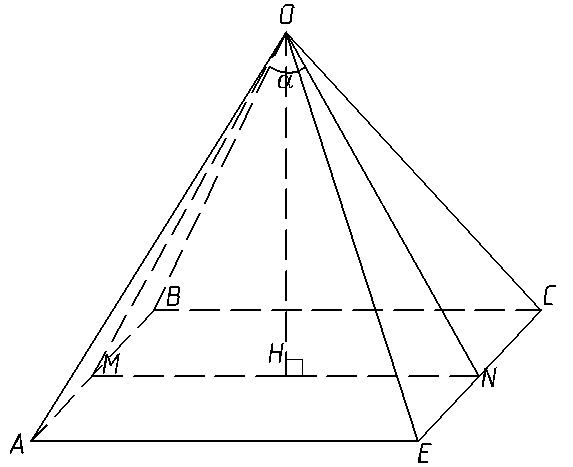


Рисунок 7.14. Построение пирамиды OABCE

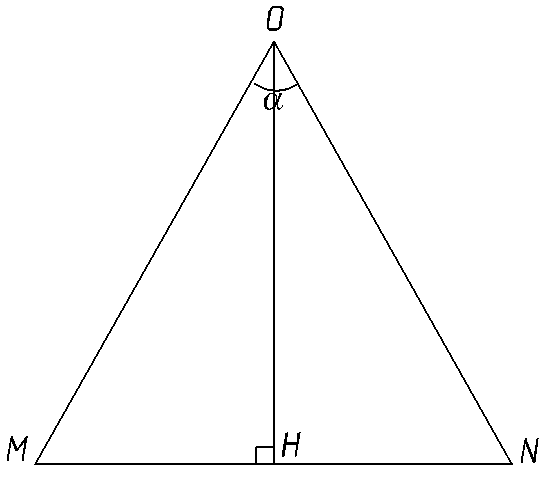


Рисунок 7.15. Треугольник MОN, выделенный из пирамиды OABCE

Вычислим ОН:

(7.1)

Вычислим АЕ, исходя из соотношения ширины кадра к высоте (в пикселях) и перейдём к реальным величинам до множив на ширину проката (1,9 м):

(7.2)

Тогда АЕ:

Вычислим сторону ОМ треугольника, применив теорему косинусов:

(7.3)

Поскольку MON равнобедренный, то OM = ON, MN = AE, тогда получим:

(7.4)

Тогда корень квадратного уравнения найдём из выражения ниже:

Тогда ОМ будет равно:

(7.5)

Тогда ОН:

(7.6)

Таким образом камера, которая расположена на данной высоте будет захватывать всю область листопроката и тогда будет достигнута максимальная эффективность распознавания. Также данные расчёты будут воссозданы в программном варианте, чтобы была возможность пересчитать высоту при замене камеры или изменении размера проката. Расчёт параметров происходит однократно при настройке сети и камеры, на основе этих вычислений организуется работа системы распознавания.

8 РАЗРАБОТКА ПРОГРАММЫ ДЛЯ ЭВМ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON

### 

### 8.1 О Google Colab

Colaboratory, или сокращенно Colab – продукт компании Google Research. Colab позволяет любому писать и выполнять произвольный код Python через браузер и особенно хорошо подходит для машинного обучения, анализа данных и образования. С технической точки зрения Colab – это размещенная на хосте служба Jupyter для скриптов, которая не требует настройки для использования, но при этом предоставляет бесплатный доступ к вычислительным ресурсам, включая графические процессоры.

Скрипты Colab хранятся на Google Диске или могут быть загружены с GitHub. Скриптами Colab можно делиться так же, как документами или таблицами Google. Код выполняется на виртуальной машине, принадлежащей текущей учетной записи. Виртуальные машины удаляются, если они бездействуют в течение некоторого времени.

Типы графических процессоров, доступных в Colab, со временем меняются. Это необходимо, чтобы Colab могла бесплатно предоставлять доступ к этим ресурсам. Графические процессоры, доступные в Colab, включают в себя Nvidia K80s, T4s, P4s и P100s. Нельзя выбрать, к какому типу графического процессора можно подключиться, пользователи, которые заинтересованы в более надежном доступе к самым быстрым графическим процессорам Colab, могут использовать Colab Pro.

Объем памяти, доступной в виртуальных машинах Colab, меняется со временем (но остается стабильным в течение всего срока работы виртуальной машины). Иногда объём может автоматически назначаться дополнительная память, когда Colab обнаруживает, что она может понадобиться.

### 8.2 Разработка свёрточной нейронной сети на Python в Google Colab

Для начала подключим Гугл Диск, чтобы была возможность загружать датасет и сохранять готовые модели. Для удобства использования введём номер текущей итерации, чтобы сохранять или загружать модели под этим номером, так как для получения хорошего результата нужно провести обучение несколько раз, варьируя параметры сети, поэтому возможность сохранения и загрузки удачных моделей имеет важное значение. Также сохранённые модели можно импортировать в другие языки программирования, которые в отличии от Python являются компилируемыми, например, С++ и JavaScript.

from google.colab import drive

drive.mount('/content/drive')

number\_of\_interations = 10

print('Текущий номер для сохранения модели, чекпоинтов, а также загрузки существующей модели с облака: ', number\_of\_interations)

Далее подключим библиотеки для машинного обучения и компьютерного зрения, а также загрузим архив датасета изображений дефектов для обучения нейросети и распакуем его во временную директорию.

import zipfile

import os

import tensorflow as tf

from tensorflow.keras.optimizers import RMSprop

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow import keras

from \_\_future\_\_ import absolute\_import, division, print\_function, unicode\_literals

import time

!pip install -q pyyaml h5py  # Требуется для сохранения модели в формате HDF5

zip\_file = '/content/drive/My Drive/Files for colab/NEU3.zip'

z = zipfile.ZipFile(zip\_file, 'r')

z.extractall()

print(os.listdir)

Разделим изображения на две большие категории: тестовые и для обучения, в каждой категории определим 7 классов изображений: царапины, вдавленные окалины, пятна, волосные трещины, рябизна, посторонние включения, чистая поверхность (по именам папок, в которых лежат изображения), в случае если данные каталоги уже созданы будет выведено соответствующее сообщение.

#перед этим создайте папку с данными train и в папке train создайте еще шесть папок для шести дефектов

import shutil

import numpy as np

try:

    source1 = "NEU/train"

    os.mkdir("NEU/test")

    dest11 = "NEU/test"

    files = os.listdir(source1)

    for f in files:

        os.mkdir(dest11 + '/'+ f)

        spilt\_num = int(len(os.listdir(source1 + '/'+ f))\*0.08)

        for i in os.listdir(source1 + '/'+ f)[spilt\_num:]:

            shutil.move(source1 + '/'+ f +'/'+ i, dest11 + '/'+ f +'/'+ i)

except:

    print("\nВсе уже есть в каталоге. Вам не нужно запускать эту ячейку")

Теперь программно разделим тестовые и тренировочные выборки в 2 больших каталога, в случае если данные каталоги уже созданы будет выведено соответствующее сообщение.

import shutil

import numpy as np

try:

    source1 = "NEU/test"

    os.mkdir("NEU/valid")

    dest11 = "NEU/valid"

    files = os.listdir(source1)

    for f in files:

        os.mkdir(dest11 + '/'+ f)

        spilt\_num=int(len(os.listdir(source1 + '/'+ f))\*0.5)

        for i in os.listdir(source1 + '/'+ f)[spilt\_num:]:

            shutil.move(source1 + '/'+ f +'/'+ i, dest11 + '/'+ f +'/'+ i)

except:

    print("\nВсе уже есть в каталоге. Вам не нужно запускать эту ячейку")

Проведём предварительную подготовку изображений, приведя их к одинаковому размеру. Метод batch сообщает методу тренировки model.fit использовать блоки по 10 изображений и метки при обновлении внутренних переменных модели [18].

# Все изображения будут смасштабированы к 1./255

train\_datagen = ImageDataGenerator(rescale = 1. / 255, shear\_range = 0.2, zoom\_range = 0.2, horizontal\_flip = True)

test\_datagen = ImageDataGenerator(rescale=1./255)

# Обучающие изображения партиями по 10 с использованием генератора train\_datagen

train\_generator = train\_datagen.flow\_from\_directory('NEU/train', target\_size = (200, 200), batch\_size = 10, class\_mode = 'categorical')

# Тестовая выборка по 10 пакетов с использованием генератора train\_datagen

validation\_generator = test\_datagen.flow\_from\_directory('NEU/valid', target\_size = (200, 200), batch\_size = 10, class\_mode = 'categorical')

Создадим класс колбека, который будет прерывать обучение нейросети, если будет достигнута точность распознавания в 93 и более процента, данный класс вызывается после каждой итерации обучения сети.

class myCallback(tf.keras.callbacks.Callback):

    def on\_epoch\_end(self, epoch, logs={}):

        if(logs.get('val\_accuracy') > 0.93 ):

            print("\nТочность расчёта более 93%, расчёт был остановлен!")

            self.model.stop\_training = True

Определим архитектуру свёрточной нейронной сети. Процесс преобразования 2D-изображения в вектор называется сглаживанием (flattening) и реализуется посредством сглаживающего слоя — flatten-слоя. ReLU — функция активации нейрона. Функция ReLU возвращает 0, если в качестве входного значения было отрицательное значение или ноль, во всех остальных случаях функция вернёт исходное входное значение. Softmax — функция, которая вычисляет вероятности для каждого возможного выходного класса.

model = tf.keras.models.Sequential([

    tf.keras.layers.Conv2D(8, (2,2), activation='relu', input\_shape=(200, 200, 3),

    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(32, (2,2), activation='relu'),

    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64, (2,2), activation='relu'),

    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(256, activation='relu'),

    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(7, activation='softmax')

])

Скомпилируем модель и проведём её резюме, показанное на рисунке 8.1.

model.compile(loss = 'categorical\_crossentropy', optimizer = 'rmsprop', metrics = ['accuracy'])

print('Модель свёрточной сети создана')

model.summary()

Далее идёт процесс обучения нейросети на данных из датасета [19], [20], с сохранением под номером текущей операции чекпоинтов и модели нейросети, которая содержит в себе всю архитектуру сети и веса для нейронов на Гугл Диске. Для текущей итерации (10) и параметров сети получена валовая точность в 92,31 процента.

callbacks = myCallback()

checkpoint\_path = "/content/drive/My Drive/Files for colab/Checkpoint\_for\_machine\_learning\_" + str(number\_of\_interations) + ".ckpt"

checkpoint\_dir = os.path.dirname(checkpoint\_path)

# Создаем коллбек сохраняющий веса модели

cp\_callback = tf.keras.callbacks.ModelCheckpoint(filepath = checkpoint\_path, save\_weights\_only = True, verbose = 1)

history = model.fit(train\_generator, batch\_size = 16, epochs = 15, validation\_data = validation\_generator, callbacks=[cp\_callback], verbose = 1, shuffle = True)

# Сохраним всю модель в  HDF5 файл

model.save("/content/drive/My Drive/Files for colab/Machine\_learning\_model\_" + str(number\_of\_interations) + ".h5")

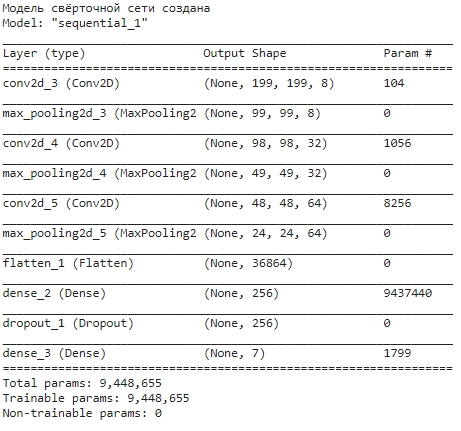


Рисунок 8.1. Резюме скомпилированной CNN

Построим графики точности и потерь в процессе обучения нейросети (рисунок 8.2), используя библиотеку для построения графиков «matplotlib». «Loss» определяется как разница между значением, прогнозируемым моделью, и истинным значением, а «accuracy» как валовая точность определения дефекта. Величина первого параметра должна стремиться к нулю, а второго к единице.

import matplotlib.pyplot as plt

plt.figure(1)

# обобщение истории для точности

plt.subplot(211)

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

 # обобщение истории для потерь

plt.subplot(212)

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc = 'upper left')

plt.show()

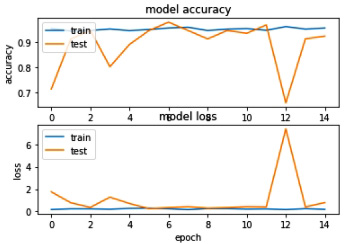


Рисунок 8.2. Графики процесса обучения

Загрузим сохранённую модель сети под текущим номером итерации (10) и выведем её резюме, чтобы удостовериться в правильности загрузки.

# Восстановим в точности ту же модель, включая веса и оптимизатор

Loading\_model = keras.models.load\_model("/content/drive/My Drive/Files for colab/Machine\_learning\_model\_" + str(number\_of\_interations) + ".h5")

Loading\_model.summary()

Загрузим датасет с диска, чтобы провести тест работоспособности.

# Загрузим имена файлов и их соответствующие целевые метки в массив numpy

from sklearn.datasets import load\_files

import numpy as np

test\_dir = 'NEU/test'

def load\_dataset(path):

    data = load\_files(path)

    files = np.array(data['filenames'])

    targets = np.array(data['target'])

    target\_labels = np.array(data['target\_names'])

    return files,targets,target\_labels

x\_test, y\_test,target\_labels = load\_dataset(test\_dir)

no\_of\_classes = len(np.unique(y\_test))

no\_of\_classes

Загрузим вспомогательную библиотеку для категоризации элементов.

from keras.utils import np\_utils

y\_test = np\_utils.to\_categorical(y\_test,no\_of\_classes)

Подгоним каждый пиксель изображения к требуемому для программы размеру и преобразуем данные в массив.

# У нас есть только имена файлов в наборе x. Загрузим изображения и преобразуем их в массив.

from keras.preprocessing.image import array\_to\_img, img\_to\_array, load\_img

def convert\_image\_to\_array(files):

    images\_as\_array=[]

    for file in files:

        # Преобразование в массив Numpy

        images\_as\_array.append(img\_to\_array(load\_img(file)))

    return images\_as\_array

x\_test = np.array(convert\_image\_to\_array(x\_test))

print('Test set shape : ',x\_test.shape)

x\_test = x\_test.astype('float32') / 255

Визуализируем предсказания нейросети в виде 16 случайных изображений из тестовой категории (рисунок 8.3). Если надпись зелёного цвета, значит нейросеть верно определила дефект, если красным, то неверно (правильный ответ в скобках).

y\_pred = Loading\_model.predict(x\_test)

# построение случайной выборки тестовых изображений, их предсказанные метки и  истинного значения

fig = plt.figure(figsize = (16, 9))

for i, idx in enumerate(np.random.choice(x\_test.shape[0], size = 16, replace = False)):

    ax = fig.add\_subplot(4, 4, i + 1, xticks=[], yticks=[])

    ax.imshow(np.squeeze(x\_test[idx]))

    pred\_idx = np.argmax(y\_pred[idx])

    true\_idx = np.argmax(y\_test[idx])

    ax.set\_title("{} ({})".format(target\_labels[pred\_idx], target\_labels[true\_idx]), color = ("green" if pred\_idx == true\_idx else "red"))

Для теста отдельного изображения реализован следующий блок кода. Тест ведётся по тестовой категории изображений от 0 до 83, результат для изображения под номером 8 показан на рисунке 8.4, также дополнительно добавим библиотеку времени, чтобы отследить сколько времени занимает определение дефекта. Для тестового изображения под номером 8, время выполнения блока кода составило 0,02268 секунд. Для других изображений время выполнения лежит в пределах ±5 процентов, от времени, полученного выше.

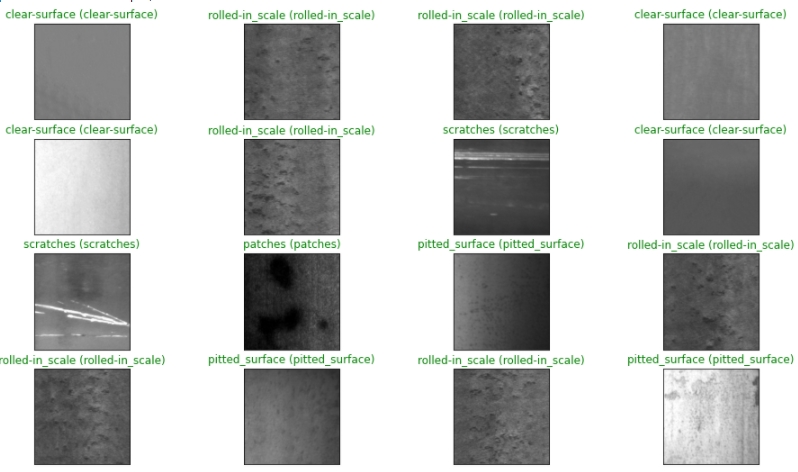


Рисунок 8.3. Визуализация результатов предсказаний нейросети

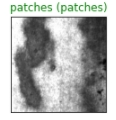


Рисунок 8.4. Предсказание нейросети для изображения №8

import matplotlib.pyplot as plt

print('Введите номер изображения, для теста из тестовой выборки (от 0 до 83)')

idx = int(input())

start\_time = time.time()

fig = plt.figure(figsize=(16, 9))

ax = fig.add\_subplot(4, 4, i + 1, xticks=[], yticks=[])

ax.imshow(np.squeeze(x\_test[idx]))

pred\_idx = np.argmax(y\_pred[idx])

true\_idx = np.argmax(y\_test[idx])

ax.set\_title("{} ({})".format(target\_labels[pred\_idx], target\_labels[true\_idx]), color=("green" if pred\_idx == true\_idx else "red"))

print("Время выполнения операции = ", (time.time() - start\_time), ' c')

Реализуем проверку всех изображений из тестовой выборки с отображением результата следующим блоком кода, представляющий из себя прогон всех изображений по циклу с отправкой на нейросеть:

import matplotlib.pyplot as plt

print('Прогон всех тестовых изображений по порядку')

for k in range(84):

  fig = plt.figure(figsize=(16, 9))

  ax = fig.add\_subplot(4, 4, 2, xticks=[], yticks=[])

  ax.imshow(np.squeeze(x\_test[k]))

  pred\_idx = np.argmax(y\_pred[k])

  true\_idx = np.argmax(y\_test[k])

  ax.set\_title("{} ({})".format(target\_labels[pred\_idx], target\_labels[true\_idx]), color=("green" if pred\_idx == true\_idx else "red"))

Поскольку при работе алгоритма распознавания не требуется визуализировать результат, то преобразуем часть кода, чтобы вычислить лишь то время, которое затрачивается для определения дефекта. В данном случае затраченное время равняется 0,001384 секунды. Для других изображений время выполнения также, как и ранее лежит в пределах ±5 процентов, от времени, полученного в результате выполнения блока кода. Полученное время является временем, затрачиваемым на определение номера дефекта, и оно будет использоваться для дальнейших расчётов быстродействия системы в целом.

print('Введите номер изображения, для теста из тестовой выборки (от 0 до 83)')

idx = int(input())

start\_time = time.time()

pred\_idx = np.argmax(y\_pred[idx])

print('Номер дефекта: ', pred\_idx)

print("Время выполнения операции = ", (time.time() - start\_time), ' c')

time\_for\_operation = (time.time() - start\_time)

Теперь рассчитаем параметры для правильной работы камеры, подробнее о методике расчета можно узнать предыдущей главе, результат показан на рисунке 8.5. Для выбранной камеры промежуток времени, через который будет делаться кадр равняется 2,0753 с.

import math

width = 1.9 #Ширина листопроката

cam\_length = 4096 #Разрешение камеры по длине

cam\_width = 3000 # Разрешение камеры по высоте

alpha = 72 #Угол обзора камеры

b = width \* cam\_length / cam\_width  #Длина участка полосы проката в м

alpha\_rad = alpha / 59.29577 # Из градусов в радианы

edge\_of\_the\_pyramid = b \* math.sqrt(2 - 2\* math.cos(alpha\_rad)) / (2 - 2\* math.cos(alpha\_rad)) #Вычисление ребра пирамиды (стороны треугольника MON)

cam\_height = math.sqrt( edge\_of\_the\_pyramid\*edge\_of\_the\_pyramid - (b / 4)) #Высота камеры над листом ОН

length\_for\_pixel = b/cam\_length #Соотношение одного пикселя к длине полосы

width\_for\_pixel = b/cam\_width #Соотношение одного пикселя к ширине полосы

print('При длине полосы, попавшей в объектив = ' + str(round(b, 3)) + ' м')

print('Высота камеры над листом = ', str(round(cam\_height, 3)) + ' м')

print('Один пиксель соответствует длине листа в ' + str (round (length\_for\_pixel, 6)) + ' м')

print('Один пиксель соответствует ширине листа в ' + str (round (width\_for\_pixel, 6)) + ' м')

#Посчитаем количество фрагментов для 1 кадра, для анализа

number\_of\_fragments\_in\_length = int(cam\_length / 200)

print('Фрагментов по длине = ', int(number\_of\_fragments\_in\_length))

number\_of\_fragments\_in\_width = int(cam\_width / 200)

print('Фрагментов по ширине = ', number\_of\_fragments\_in\_width)

print('Всего фрагментов = ', number\_of\_fragments\_in\_length \* number\_of\_fragments\_in\_width)

time\_for\_frame\_operation = number\_of\_fragments\_in\_length \* number\_of\_fragments\_in\_width \* time\_for\_operation

print('Время на обработку одного кадра = ', time\_for\_frame\_operation, 'c')

#Расчёт промежутка времени через который будет делаться 1 кадр

velocity = 1.25 #Скорость листопроката м/с

time\_frame = b / velocity #Промежуток времени через который будет делаться снимок

print('Промежуток времени через который будет делаться снимок = ' + str(round (time\_frame, 4)) + ' c')

print('Запас или недостаток (знак минус) по времени = ', (time\_frame - time\_for\_frame\_operation ), ' c')

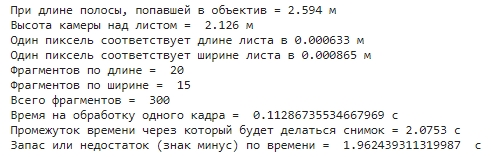


Рисунок 8.5. Результат расчёта параметров работы камеры

Создадим директорию, в которую будут сохраняться фрагменты изображения, полученные после обработки снимка с камеры, в случае если каталог уже имеется будет выдано соответствующее сообщение.

try:

  os.mkdir("Fragments/") # Создание пустого каталога (папки), в которую будут сохраняться фрагменты

except:

    print("\nВсе уже есть в каталоге. Вам не нужно запускать эту ячейку")

Реализуем подключение камеры, которая будет делать снимок, с заданным периодом времени (рассчитанным в блоке выше) и сохранять полученное изображение.

import cv2

import time

timing = time.time()

cap = cv2.VideoCapture(0) #Подключаемся (захватываем) к камере. 0 — это индекс камеры, если их несколько то будет 0 или 1 и т.д.

time\_frame = round(time\_frame, 3) #Округлим время работы

iteration = 1

while True:

    if time.time() - timing > time\_frame:

      iteration += 1

      ret, img = cap.read() # Читаем с устройства кадр, метод возвращает флаг ret (True , False) и img — саму картинку (массив numpy)

      name = 'Test' +  str(iteration) + '.jpg' #Генерация имени файла

      cv2.imwrite("/content/drive/My Drive/Files for colab/TestPicture/" + str(name), img)

cap.release()

Разобьем изображение на фрагменты 200х200 пикселей по всей длине и ширине кадра и сохраним их в ранее созданную папку.

import cv2

import matplotlib

import os

image = cv2.imread("/content/drive/My Drive/Files for colab/TestPicture/Test2.jpg") #Тестовое изображение, загружаемое с Гугл.Диска

#Пересчёт количества фрагментов, на тот случай, если загружаемое изображение имеет иные размеры

length = image.shape[0] #Высота изображения

width = image.shape[1] #Ширина изображения

#Посчитаем количество фрагментов для 1 кадра для анализа

number\_of\_fragments\_in\_length\_img = int(length/ 200)

number\_of\_fragments\_in\_width\_img = int(width / 200)

number\_of\_fragments\_img = number\_of\_fragments\_in\_length\_img \* number\_of\_fragments\_in\_width\_img

number = 0 #Вспомогательная переменная

x\_massive = [] #Ось х для построения графиков

#Дробление изображения

for i in range (number\_of\_fragments\_in\_length\_img):

  for k in range(number\_of\_fragments\_in\_width\_img):

    number+=1

    x\_massive.append(number)

    cropped = image[(0 + 200 \* i) : (200 + 200 \* i), (0 + 200 \* k) : (200 + 200 \* k)] # Вырезаем изображения кусочками по 200х200 пикселей по всей площади

    name = "Cropped\_" + str(number) + '.jpg'

    cv2.imwrite("Fragments/" + str(name), cropped)

Подготовим фрагменты для дальнейшей обработки нейросетью, подогнав их под нужный размер и преобразовав в массив NumPy.

# Загрузим имена файлов и их соответствующие целевые метки в массив numpy

from sklearn.datasets import load\_files

import numpy as np

test\_dir\_2 = 'Fragments/ForTest'

def load\_dataset\_2 (path):

    data\_2 = load\_files(path)

    files\_2 = np.array(data\_2['filenames'])

    targets\_2 = np.array(data\_2['target'])

    target\_labels\_2 = np.array(data\_2['target\_names'])

    return files\_2,targets\_2,target\_labels\_2

x\_test\_2, y\_test\_2, target\_labels\_2 = load\_dataset\_2(test\_dir\_2)

no\_of\_classes\_2 = len(np.unique(y\_test\_2))

# У нас есть только имена файлов в наборе x. Загрузим изображения и преобразуем их в массив.

from keras.preprocessing.image import array\_to\_img, img\_to\_array, load\_img

def convert\_image\_to\_array(files):

    images\_as\_array\_2=[]

    for file in files:

        # Преобразование в массив Numpy

        images\_as\_array\_2.append(img\_to\_array(load\_img(file)))

    return images\_as\_array\_2

x\_test\_2 = np.array(convert\_image\_to\_array(x\_test\_2))

x\_test\_2 = x\_test.astype('float32')/255

Передадим все фрагменты изображения из папки «Fragments» на свёрточную нейросеть, сохраним результаты анализа в массив, а также высчитаем время на выполнение данного блока, время работы составило примерно 0,13 секунды.

import matplotlib.pyplot as plt

y\_pred\_2 = Loading\_model.predict(x\_test\_2)

start\_time = time.time()

massive\_of\_defects = []

# построение случайной выборки тестовых изображений, их предсказанные метки и  истинного значения

fig = plt.figure(figsize = (16, 9))

for idx\_2 in range(number\_of\_fragments\_img):

    pred\_idx\_2 = np.argmax(y\_pred\_2[idx\_2])

    print('Номер дефекта: '+ str(pred\_idx\_2) + ' для ' + str(idx\_2 + 1) + ' изображения')

    if (pred\_idx\_2 != 7):

      massive\_of\_defects.append(pred\_idx\_2)

print("Время выполнения операции = ", (time.time() - start\_time), ' c')

print("Расшифровка: crazing = 1, inclusion = 2, patches = 3, plitted surface = 4, rolled in scale = 5, scratches = 6, clear surface = 7")

На тот случай, если потребуется очистить содержимое папки «Fragments», добавим следующие строчки кода в отдельную ячейку.

import os

import glob

print("Удалить содержимое папки Fragments? 1/0")

delete = int(input())

if (delete == 1):

  for f in files:

    try:

        os.remove(f)

    except OSError as e:

        print("Ошибка: %s : %s" % (f, e.strerror))

else:

  print('Ок, не удаляем')

Построим гистограмму найденных дефектов, относительно друг друга, классический график распределения дефектов, а также круговую диаграмму, основываясь на полученном ранее массиве распознанных дефектов. Результаты показаны на рисунках 8.6 – 8.8. В данном блоке кода массив используется номеров дефектов, полученный свёрточной нейросетью ранее. Для построения графика и гистограммы используется «очищенный» массив, то есть в массив количества дефектов без учёта фрагментов чистой поверхности, но при построении круговой диаграммы количество чистых фрагментов уже учитывается.

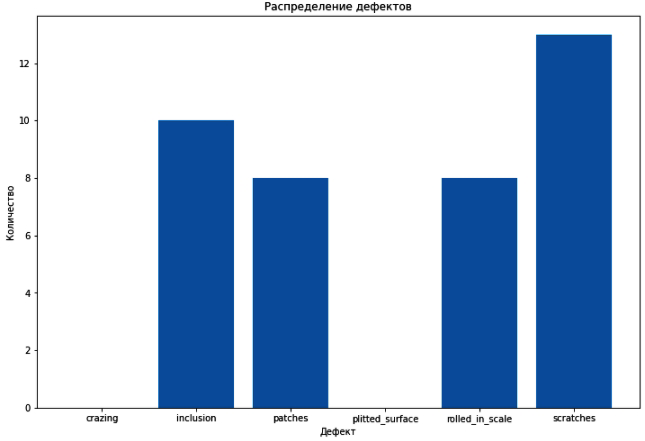


Рисунок 8.6. Гистограмма распределения дефектов



Рисунок 8.7. График распределения дефектов

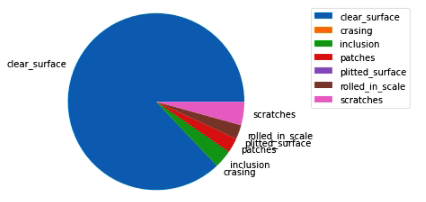


Рисунок 8.8. Круговая диаграмма распределения дефектов

import seaborn as sns

import matplotlib.pyplot as plt

x = ['crazing', 'inclusion', 'patches', 'plitted\_surface', 'rolled\_in\_scale', 'scratches']

y = []

crazing = 0

inclusion  = 0

patches = 0

plitted\_surface = 0

rolled\_in\_scale = 0

scratches = 0

#Посчитаем количество дефектов каждого вида

for iter in range(len(massive\_of\_defects)):

  if massive\_of\_defects[iter] == 1:

    crazing += 1

  elif massive\_of\_defects[iter] == 2:

    inclusion += 1

  elif massive\_of\_defects[iter] == 3:

    patches += 1

  elif massive\_of\_defects[iter] == 4:

    plitted\_surface += 1

  elif massive\_of\_defects[iter] == 5:

    rolled\_in\_scale += 1

  elif massive\_of\_defects[iter] == 6:

    scratches += 1

#добавим количество в массив

y.append(crazing)

y.append(inclusion)

y.append(patches)

y.append(plitted\_surface)

y.append(rolled\_in\_scale)

y.append(scratches)

#Построим гистограмму

fig, ax = plt.subplots()

ax.bar(x, y)

plt.title('Распределение дефектов')

plt.xlabel('Дефект')

plt.ylabel('Количество')

ax.set\_facecolor('white')

fig.set\_facecolor('white')

fig.set\_figwidth(12) #Ширина графика

fig.set\_figheight(8) #Высота графика

#plt.grid(True) #Раскоментировать, если требуется сетка

#Построим классический график

fig = plt.figure(figsize=(10, 6))

plt.grid()

plt.title('Распределение дефектов')

plt.xlabel('Дефект')

plt.ylabel('Количество')

plt.plot(x, y)

#Построим круговую диаграмму

vals = [] #Массив процентов

persent\_of\_clear\_surface = ((number\_of\_fragments\_img) - len(massive\_of\_defects)) / (number\_of\_fragments\_img) \* 100 #Процент чистых фрагментов

persent\_of\_crasing = round ((y2[0] / 3), 3) #Процент волосных трещин

persent\_of\_inclusion = round ((y2[1] / 3), 3) #Процент посторонних включений

persent\_of\_patches = round ((y2[2] / 3), 3) #Процент пятен

persent\_of\_plitted\_surface = round ((y2[3] / 3), 3) #Процент рябизны

persent\_of\_rolled\_in\_scale = round ((y2[4] / 3), 3) #Процент вдавленных окалин

persent\_of\_scratches = round ((y2[5] / 3), 3) #Процент царапин

vals.append(persent\_of\_clear\_surface)

vals.append(persent\_of\_crasing)

vals.append(persent\_of\_inclusion)

vals.append(persent\_of\_patches)

vals.append(persent\_of\_plitted\_surface)

vals.append(persent\_of\_rolled\_in\_scale)

vals.append(persent\_of\_scratches)

labels\_for\_round\_diagram = ["clear\_surface", "crasing", "inclusion", "patches", "plitted\_surface", "rolled\_in\_scale", "scratches"]

fig, ax = plt.subplots()

ax.pie(vals, labels = labels\_for\_round\_diagram)

ax.legend(loc = 'upper left', bbox\_to\_anchor = (1.0, 1.0))

ax.axis("equal")

print("Процент чистой поверности: ", persent\_of\_clear\_surface)

print("Процент волосных трещин: ", persent\_of\_crasing)

print("Процент посторонних включений: ", persent\_of\_inclusion)

print("Процент пятен: ", persent\_of\_patches)

print("Процент рябизны: ", persent\_of\_plitted\_surface)

print("Процент вдавленных окалин: ", persent\_of\_rolled\_in\_scale)

print("Процент царапин: ", persent\_of\_scratches)

Основываясь на [9] и [13], вынесем рекомендации по корректировке оборудования и материалов для повышения качества выходного продукта. Рекомендации выносятся на основе наиболее часто встречающихся дефектов определённого типа. Волосные трещины и рябизна свидетельствуют о высоком износе поверхности прокатных валков, посторонние включения и вдавленные окалины показывают, что требуется проверить работу окалиноломателей и гидросбива. При преимущественном наличии царапин выносится такая рекомендация – рекомендуется удалить острые части направляющей арматуры или очистить поверхность валков чистовой группы.

#Объявим переменные

max = y[0]

position = 0

#Найдём индекс наибольшего значения в массиве

for i in range(len(y)):

    if y[i]>max: max = y[i]; position = i

#Вынесение рекомендаций

if (y[0] > 1 or y[3] > 1 ):

  print('Внимание! Имеются волосные трещины и рябизна. Для повышения качества проката до обычного требебуется провести шлифовку прокатных валов чистовой группы')

elif (position == 1 or position == 4):

  print('Преимущественный дефекты - посторонние включения и вдавленные окалины. Для улучшения качества поверности рекомендуется улучшить зачистку слябов, настроить работу горизонтального и вертикального окалиноломателя перед черновой группой клетей')

elif(position == 2):

  print("Преимущественный дефект - пятна на поверности металла. Для улучшения качества поверхности рекомендуется проверить работу окалиноломателя и гидросбива")

elif (position == 5):

  print('Преимущественный дефект - царапины на поверности металла. Для улучшения качества поверности рекомендуется удалить острые части направляющей арматуры или очистить поверхность валков чистовой группы')

Далее, основываясь на ГОСТ 5246-2016, табл. 10 присвоим категорию качества поверхности горячего проката. Точность присвоения категории составила 90,5%.

if (y[0] > 1 and y[3] == 0):

  print('Категория качества по ГОСТ 5246-2016, табл. 10  - У')

elif (y[0] > 1 and y[3] > 1):

  print('Категория качества по ГОСТ 5246-2016, табл. 10  - В')

else:

  print('Категория качества по ГОСТ 5246-2016, табл. 10  - Обычная')

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы для прокатного цеха №4, участка транспортировки за чистовой группой клетей стана 2300 ПАО «ЧМК» на основе найденной документации была составлена структурная, функциональная и кинематическая схемы исходного привода, а также на основе найденных технических параметров модернизируемой системы, такие как: скорость движения проката, масса проката, линейные размеры рольганга и подвижных элементов была разработана более современная система электропривода – асинхронный двигатель – преобразователь частоты.

Для заданных значений величины ошибки регулирования по скорости не превышающей 5% от заданного, максимального допустимого ускорения заготовки равным 1 м/с2, получили, что ошибка регулирования по скорости составила 4,3%, максимальное ускорение равным 0,5 м/с2.

Выбранный двигатель АИР 132S6 также удовлетворяет всем поставленным условиям: напряжение питания – 380 В, а частота 50 Гц; защита от влаги и пыли IP55; режим работы электродвигателя – S1; климатическое исполнение электродвигателя У2.

Преобразователь частоты SINAMICS G120 POWER также удовлетворяет всем поставленным условиям: преобразователь выполнен в виде модуля и успешно интегрируется в существующую систему управления, основанную на ПЛК Siemens S7–1200.

В результате исследования существующих библиотек и фреймворков и методов для глубокого и машинного обучения был выбран наиболее оптимальный вариант для распознавания дефектов – свёрточная нейронная сеть, в качестве основных библиотек машинного обучения были выбраны TensorFlow и Keras, а качестве фреймворка для работы выступил Google Colaboratory.

В качества обучающего датасета для сети выл выбран NEU (Northeastern University) surface defect database представляющий собой набор данных по самым распространённым поверхностным дефектам горячего проката. Также данный датасет был дополнен вручную тремястами дополнительными изображениями чистой поверхности горячего проката.

Для разработанной программы для ЭВМ на языке программирования Python, используя ранее выбранные библиотеки и фреймворк для работы с машинным обучением на основе найденного и дополненного датасета изображений поверхностных дефектов горячего проката были выполнены сформулированные требования. Программа в соответствии с ГОСТ 20847-75 определяет с точностью 92,31% дефекты и чистую поверхность, при заданной точности в 85%. Согласно ГОСТ 52246-2016 присваивать определённую группу качества проката с точностью 90,5% (заданная 90%), в соответствии с полученными данными после анализа нейросетью фрагментов изображения, поступающего с камеры.

Скорости работы нейросети хватает, чтобы за 1,7 секунды успеть обработать исходное изображение, полученное с камеры в фрагменты 200х200 пикселей, отправить их на нейросеть, обработать результаты и вывести результаты на экран, при том что всё это нужно успеть примерно за 2 секунды остается запас порядка 15% для выполнения дополнительных задач.

По окончанию анализа распознанные дефекты, их количество относительно друг друга и чистой поверхности представляются в удобном для визуального распознавания виде – графики, круговые диаграммы и гистограммы.

Основываясь на [9] и [13], выносятся рекомендации по корректировке оборудования и материалов для повышения качества выходного продукта. Рекомендации выносятся на основе наиболее часто встречающихся дефектов определённого типа.

Созданная система также имеет дешёвую возможность переобучения для возможности расширения классификации дефектов, без значительной потери точности распознавания.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Фотиев М.М.Электропривод и электрооборудование металлургических цехов: учеб. пособие для вузов – М.: Металлургия, 1990. – 352 с.

2. Грудев А.П. Технология прокатного производства – М.: Изд-во «Металлургия», 1994. – 326 с.

3. Глазов В.С. Машины непрерывного транспортирования для прокатки станов. – М.: «Металлургия», 1979. – 248с.

4. Зюзин В.И. Технология прокатного производства. Справочник в 2 ч. – М.: Изд-во «Металлургия», 1991. – 855 с.

5. Чиликин М. Г. Общий курс электропривода: учебник для вузов. – М.: Энергоиздат, 1981. – 576 с.

6. Копылов И. П. Справочник по электрическим машинам: в 2 т – М.: Энергоатомиздат, 1989. – 688 с.

7. Королев А.А. Конструкция и расчет машин и механизмов прокатных станов: учеб. пособие для вузов М.: «Металлургия», 1985. – 376с.

8. Гуревич А.Е. Справочные данные по электрооборудованию. – Т. 2. Электрооборудование для тяжелого машиностроения и металлургических – М.: Энергия, 1965. – 480 с.

9. Анурьев В.И. Справочник конструктора – машиностроителя в 3-х томах – М.: Изд-во «Машиностроение», 1999. – 585 с.

10. Кравчик А. Э. Асинхронные двигатели серии 4А. Справочник – М.: Энергоатомиздат, 1982. – 380 с.

11. ГОСТ 52246-2016. Прокат листовой горячеоцинкованный М.: Изд-во стандартов, 2017. – 39 с.

12. ГОСТ 21014-88 Прокат чёрных металлов. Термины и определения дефектов поверхности М.: Изд-во стандартов, 1990. – 96 с.

13. ГОСТ 7564-97 Общие правила отбора проб, заготовок и образцов для механических и технологических испытаний М.: Изд-во стандартов, 1990. – 16 с.

14. ГОСТ 19.701-90 Схемы алгоритмов, программ, данных и систем М.: Изд-во стандартов, 1992. – 21 с.

15. Бархатов В.А. Распознавание дефектов с помощью искусственной нейронной сети специального типа – М.: Дефектоскопия – 2006. – № 2. – с. 28―39.

16. Бархатов В.А. Обнаружение сигналов и их классификация с помощью распознавания образов – М.: Дефектоскопия – 2006. – № 4 – с. 14―27.

17. Fukushima, K. Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position // Biological Cybernetics, vol. 36 – 1980 – № 4 – pp. 193–202.

18. Behnke, S. Hierarchical Neural Networks for Image Interpretation // ser. Lecture Notes in Computer Science. Springer – 2003 – vol. 2766.

19. Jingwen Fu. Recognition Of Surface Defects On Steel Sheet Using Transfer Learning // Xian Jiaotong University – 2020.

20. Yiping Gao, Liang Gao, Xinyu Li, and Xuguo Yan. A Semisupervised convolutional neural network-based method for steel surface defect recognition // Robotics and Computer Integrated Manufacturing – 2020 – vol. 61:101825.

21. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks // In International Conference on Neural Information Processing Systems – 2012.

22. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition // Computer Science – 2014.

ПРИЛОЖЕНИЕ А

Листинг кода на Python в Google Colab

1 ячейка

from google.colab import drive

drive.mount('/content/drive')

number\_of\_interations = 10

print('Текущий номер для сохранения модели, чекпоинтов, а также загрузки существующей модели с облака: ', number\_of\_interations)

2 ячейка

import zipfile

import os

import tensorflow as tf

from tensorflow.keras.optimizers import RMSprop

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow import keras

from \_\_future\_\_ import absolute\_import, division, print\_function, unicode\_literals

import time

!pip install -q pyyaml h5py  # Требуется для сохранения модели в формате HDF5

zip\_file = '/content/drive/My Drive/Files for colab/NEU3.zip'

z = zipfile.ZipFile(zip\_file, 'r')

z.extractall()

print(os.listdir)

3 ячейка

#перед этим создайте папку с данными train и в папке train создайте еще шесть папок для шести дефектов

import shutil

import numpy as np

try:

    source1 = "NEU/train"

    os.mkdir("NEU/test")

    dest11 = "NEU/test"

    files = os.listdir(source1)

    for f in files:

        os.mkdir(dest11 + '/'+ f)

        spilt\_num = int(len(os.listdir(source1 + '/'+ f))\*0.08)

        for i in os.listdir(source1 + '/'+ f)[spilt\_num:]:

            shutil.move(source1 + '/'+ f +'/'+ i, dest11 + '/'+ f +'/'+ i)

except:

    print("\nВсе уже есть в каталоге. Вам не нужно запускать эту ячейку")

4 ячейка

import shutil

import numpy as np

try:

    source1 = "NEU/test"

    os.mkdir("NEU/valid")

    dest11 = "NEU/valid"

    files = os.listdir(source1)

    for f in files:

        os.mkdir(dest11 + '/'+ f)

        spilt\_num=int(len(os.listdir(source1 + '/'+ f))\*0.5)

        for i in os.listdir(source1 + '/'+ f)[spilt\_num:]:

            shutil.move(source1 + '/'+ f +'/'+ i, dest11 + '/'+ f +'/'+ i)

except:

    print("\nВсе уже есть в каталоге. Вам не нужно запускать эту ячейку")

5 ячейка

# Все изображения будут смасштабированы к 1./255

train\_datagen = ImageDataGenerator(rescale = 1. / 255, shear\_range = 0.2, zoom\_range = 0.2, horizontal\_flip = True)

test\_datagen = ImageDataGenerator(rescale=1./255)

# Обучающие изображения партиями по 10 с использованием генератора train\_datagen

train\_generator = train\_datagen.flow\_from\_directory('NEU/train', target\_size = (200, 200), batch\_size = 10, class\_mode = 'categorical')

# Тестовая выборка по 10 пакетов с использованием генератора train\_datagen

validation\_generator = test\_datagen.flow\_from\_directory('NEU/valid', target\_size = (200, 200), batch\_size = 10, class\_mode = 'categorical')

6 ячейка

class myCallback(tf.keras.callbacks.Callback):

    def on\_epoch\_end(self, epoch, logs={}):

        if(logs.get('val\_accuracy') > 0.92 ):

            print("\nТочность расчёта более 93%, расчёт был остановлен!")

            self.model.stop\_training = True

7 ячейка

model = tf.keras.models.Sequential([

    tf.keras.layers.Conv2D(8, (2,2), activation='relu', input\_shape=(200, 200, 3)),

    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(32, (2,2), activation='relu'),

    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64, (2,2), activation='relu'),

    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(256, activation='relu'),

    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(7, activation='softmax')

])

8 ячейка

model.compile(loss = 'categorical\_crossentropy', optimizer = 'rmsprop', metrics = ['accuracy'])

print('Модель свёрточной сети создана')

model.summary()

9 ячейка

callbacks = myCallback()

checkpoint\_path = "/content/drive/My Drive/Files for colab/Checkpoint\_for\_machine\_learning\_" + str(number\_of\_interations) + ".ckpt"

checkpoint\_dir = os.path.dirname(checkpoint\_path)

# Создаем коллбек сохраняющий веса модели

cp\_callback = tf.keras.callbacks.ModelCheckpoint(filepath = checkpoint\_path, save\_weights\_only = True, verbose = 1)

history = model.fit(train\_generator, batch\_size = 16, epochs = 15, validation\_data = validation\_generator, callbacks=[cp\_callback], verbose = 1, shuffle = True)

# Сохраним всю модель в  HDF5 файл

model.save("/content/drive/My Drive/Files for colab/Machine\_learning\_model\_" + str(number\_of\_interations) + ".h5")

10 ячейка

import matplotlib.pyplot as plt

plt.figure(1)

# обобщение истории для точности

plt.subplot(211)

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

 # обобщение истории для потерь

plt.subplot(212)

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()

11 ячейка

# Восстановим в точности ту же модель, включая веса и оптимизатор

Loading\_model = keras.models.load\_model("/content/drive/My Drive/Files for colab/Machine\_learning\_model\_" + str(number\_of\_interations) + ".h5")

Loading\_model.summary()

12 ячейка

# Загрузим имена файлов и их соответствующие целевые метки в массив numpy

from sklearn.datasets import load\_files

import numpy as np

test\_dir = 'NEU/test'

def load\_dataset(path):

    data = load\_files(path)

    files = np.array(data['filenames'])

    targets = np.array(data['target'])

    target\_labels = np.array(data['target\_names'])

    return files,targets,target\_labels

x\_test, y\_test,target\_labels = load\_dataset(test\_dir)

13 ячейка

no\_of\_classes = len(np.unique(y\_test))

print('Количество классов: ', no\_of\_classes)

14 ячейка

from keras.utils import np\_utils

y\_test = np\_utils.to\_categorical(y\_test,no\_of\_classes)

15 ячейка

# У нас есть только имена файлов в наборе x. Загрузим изображения и преобразуем их в массив.

from keras.preprocessing.image import array\_to\_img, img\_to\_array, load\_img

def convert\_image\_to\_array(files):

    images\_as\_array=[]

    for file in files:

        # Преобразование в массив Numpy

        images\_as\_array.append(img\_to\_array(load\_img(file)))

    return images\_as\_array

x\_test = np.array(convert\_image\_to\_array(x\_test))

print('Размеры тестовой выборки (кол-во изображений, размеры, кол-во каналов): ', x\_test.shape)

16 ячейка

x\_test = x\_test.astype('float32')/255

17 ячейка

import matplotlib.pyplot as plt

y\_pred = Loading\_model.predict(x\_test)

start\_time = time.time()

# построение случайной выборки тестовых изображений, их предсказанные метки и  истинного значения

fig = plt.figure(figsize = (16, 9))

for i, idx in enumerate(np.random.choice(x\_test.shape[0], size = 16, replace = False)):

    ax = fig.add\_subplot(4, 4, i + 1, xticks=[], yticks=[])

    ax.imshow(np.squeeze(x\_test[idx]))

    pred\_idx = np.argmax(y\_pred[idx])

    true\_idx = np.argmax(y\_test[idx])

    ax.set\_title("{} ({})".format(target\_labels[pred\_idx], target\_labels[true\_idx]), color = ("green" if pred\_idx == true\_idx else "red"))

print("Время выполнения операции = ", (time.time() - start\_time), ' c')

18 ячейка

import matplotlib.pyplot as plt

print('Введите номер изображения, для теста из тестовой выборки (от 0 до 83)')

idx = int(input())

start\_time = time.time()

fig = plt.figure(figsize=(16, 9))

ax = fig.add\_subplot(4, 4, i + 1, xticks=[], yticks=[])

ax.imshow(np.squeeze(x\_test[idx]))

pred\_idx = np.argmax(y\_pred[idx])

true\_idx = np.argmax(y\_test[idx])

ax.set\_title("{} ({})".format(target\_labels[pred\_idx], target\_labels[true\_idx]), color=("green" if pred\_idx == true\_idx else "red"))

print("Время выполнения операции = ", (time.time() - start\_time), ' c')

19 ячейка

import matplotlib.pyplot as plt

print('Прогон всех тестовых изображений по порядку')

for k in range(84):

  fig = plt.figure(figsize=(16, 9))

  ax = fig.add\_subplot(4, 4, 2, xticks=[], yticks=[])

  ax.imshow(np.squeeze(x\_test[k]))

  pred\_idx = np.argmax(y\_pred[k])

  true\_idx = np.argmax(y\_test[k])

  ax.set\_title("{} ({})".format(target\_labels[pred\_idx], target\_labels[true\_idx]), color=("green" if pred\_idx == true\_idx else "red"))

20 ячейка

print('Введите номер изображения, для теста из тестовой выборки (от 0 до 83)')

idx = int(input())

start\_time = time.time()

pred\_idx = np.argmax(y\_pred[idx])

print('Номер дефекта: ', pred\_idx)

print("Время выполнения операции = ", (time.time() - start\_time), ' c')

time\_for\_operation = (time.time() - start\_time)

print("Расшифровка: crazing = 1, inclusion = 2, patches = 3, plitted surface = 4, rolled in scale = 5, scratches = 6, clear surface = 7")

21 ячейка

import math

width = 1.9 #Ширина листопроката

cam\_length = 4096 #Разрешение камеры по длине

cam\_width = 3000 # Разрешение камеры по высоте

alpha = 72 #Угол обзора камеры

b = width \* cam\_length / cam\_width  #Длина участка полосы проката в м

alpha\_rad = alpha / 59.29577 # Из градусов в радианы

edge\_of\_the\_pyramid = b \* math.sqrt(2 - 2\* math.cos(alpha\_rad)) / (2 - 2\* math.cos(alpha\_rad)) #Вычисление ребра пирамиды (стороны треугольника MON)

cam\_height = math.sqrt( edge\_of\_the\_pyramid\*edge\_of\_the\_pyramid - (b / 4)) #Высота камеры над листом ОН

length\_for\_pixel = b/cam\_length #Соотношение одного пикселя к длине полосы

width\_for\_pixel = b/cam\_width #Соотношение одного пикселя к ширине полосы

print('При длине полосы, попавшей в объектив = ' + str(round(b, 3)) + ' м')

print('Высота камеры над листом = ', str(round(cam\_height, 3)) + ' м')

print('Один пиксель соответствует длине листа в ' + str (round (length\_for\_pixel, 6)) + ' м')

print('Один пиксель соответствует ширине листа в ' + str (round (width\_for\_pixel, 6)) + ' м')

#Посчитаем количество фрагментов для 1 кадра, для анализа

number\_of\_fragments\_in\_length = int(cam\_length / 200)

print('Фрагментов по длине = ', int(number\_of\_fragments\_in\_length))

number\_of\_fragments\_in\_width = int(cam\_width / 200)

print('Фрагментов по ширине = ', number\_of\_fragments\_in\_width)

print('Всего фрагментов = ', number\_of\_fragments\_in\_length \* number\_of\_fragments\_in\_width)

time\_for\_frame\_operation = number\_of\_fragments\_in\_length \* number\_of\_fragments\_in\_width \* time\_for\_operation

print('Время на обработку одного кадра = ', time\_for\_frame\_operation, 'c')

#Расчёт промежутка времени через который будет делаться 1 кадр

velocity = 1.25 #Скорость листопроката м/с

time\_frame = b / velocity #Промежуток времени через который будет делаться снимок

print('Промежуток времени через который будет делаться снимок = ' + str(round (time\_frame, 4)) + ' c')

print('Запас или недостаток (знак минус) по времени = ', (time\_frame - time\_for\_frame\_operation ), ' c')

22 ячейка

import cv2

import time

timing = time.time()

cap = cv2.VideoCapture(0) #Подключаемся (захватываем) к камере. 0 — это индекс камеры, если их несколько то будет 0 или 1 и т.д.

time\_frame = round(time\_frame, 3) #Округлим время работы

iteration = 1

while True:

    if time.time() - timing > time\_frame:

      iteration += 1

      ret, img = cap.read() # Читаем с устройства кадр, метод возвращает флаг ret (True , False) и img — саму картинку (массив numpy)

      name = 'Test' +  str(iteration) + '.jpg' #Генерация имени файла

      cv2.imwrite("/content/drive/My Drive/Files for colab/TestPicture/" + str(name), img)

cap.release()

23 ячейка

try:

  os.mkdir("Fragments/") # Создание пустого каталога (папки), в которую будут сохраняться фрагменты

except:

    print("\nВсе уже есть в каталоге. Вам не нужно запускать эту ячейку")

24 ячейка

import cv2

import matplotlib

import os

image = cv2.imread("/content/drive/My Drive/Files for colab/TestPicture/Test12.jpg") #Тестовое изображение, загружаемое с Гугл.Диска

#Пересчёт количества фрагментов, на тот случай, если загружаемое изображение имеет иные размеры

length = image.shape[0] #Высота изображения

width = image.shape[1] #Ширина изображения

#Посчитаем количество фрагментов для 1 кадра для анализа

number\_of\_fragments\_in\_length\_img =  int(length/ 200)

number\_of\_fragments\_in\_width\_img = int(width / 200)

number\_of\_fragments\_img = number\_of\_fragments\_in\_length\_img \* number\_of\_fragments\_in\_width\_img

number = 0 #Вспомогательная переменная

x\_massive = [] #Ось х для построения графиков

#Дробление изображения

for i in range (number\_of\_fragments\_in\_length\_img):

  for k in range(number\_of\_fragments\_in\_width\_img):

    number+=1

    x\_massive.append(number)

    cropped = image[(0 + 200 \* i) : (200 + 200 \* i), (0 + 200 \* k) : (200 + 200 \* k)] # Вырезаем изображения кусочками по 200х200 пикселей по всей площади

    name = "Cropped\_" + str(number) + '.jpg'

    cv2.imwrite("Fragments/ForTest/" + str(name), cropped)

print("Всего фрагментов: ", number\_of\_fragments\_img)

25 ячейка

# Загрузим имена файлов и их соответствующие целевые метки в массив numpy

from sklearn.datasets import load\_files

import numpy as np

test\_dir\_2 = 'Fragments/ForTest'

def load\_dataset\_2 (path):

    data\_2 = load\_files(path)

    files\_2 = np.array(data\_2['filenames'])

    targets\_2 = np.array(data\_2['target'])

    target\_labels\_2 = np.array(data\_2['target\_names'])

    return files\_2,targets\_2,target\_labels\_2

x\_test\_2, y\_test\_2, target\_labels\_2 = load\_dataset\_2(test\_dir\_2)

no\_of\_classes\_2 = len(np.unique(y\_test\_2))

26 ячейка

# У нас есть только имена файлов в наборе x. Загрузим изображения и преобразуем их в массив.

from keras.preprocessing.image import array\_to\_img, img\_to\_array, load\_img

def convert\_image\_to\_array(files):

    images\_as\_array\_2=[]

    for file in files:

        # Преобразование в массив Numpy

        images\_as\_array\_2.append(img\_to\_array(load\_img(file)))

    return images\_as\_array\_2

x\_test\_2 = np.array(convert\_image\_to\_array(x\_test\_2))

x\_test\_2 = x\_test.astype('float32')/255

27 ячейка

import matplotlib.pyplot as plt

y\_pred\_2 = Loading\_model.predict(x\_test\_2)

start\_time = time.time()

massive\_of\_defects = []

# построение случайной выборки тестовых изображений, их предсказанные метки и  истинного значения

fig = plt.figure(figsize = (16, 9))

for idx\_2 in range(number\_of\_fragments\_img):

    pred\_idx\_2 = np.argmax(y\_pred\_2[idx\_2])

    print('Номер дефекта: '+ str(pred\_idx\_2) + ' для ' + str(idx\_2 + 1) + ' изображения')

    if (pred\_idx\_2 != 7):

      massive\_of\_defects.append(pred\_idx\_2)

print("Время выполнения операции = ", (time.time() - start\_time), ' c')

print("Расшифровка: crazing = 1, inclusion = 2, patches = 3, plitted surface = 4, rolled in scale = 5, scratches = 6, clear surface = 7")

28 ячейка

import os

import glob

print("Удалить содержимое папки Fragments? 1/0")

delete = int(input())

if (delete == 1):

  for f in files:

    try:

        os.remove(f)

    except OSError as e:

        print("Ошибка: %s : %s" % (f, e.strerror))

else:

  print('Ок, не удаляем')

29 ячейка

import seaborn as sns

import matplotlib.pyplot as plt

x = ['crazing', 'inclusion', 'patches', 'plitted\_surface', 'rolled\_in\_scale', 'scratches']

y = []

crazing = 0

inclusion  = 0

patches = 0

plitted\_surface = 0

rolled\_in\_scale = 0

scratches = 0

#Посчитаем количество дефектов каждого вида

for iter in range(len(massive\_of\_defects)):

  if massive\_of\_defects[iter] == 1:

    crazing += 1

  elif massive\_of\_defects[iter] == 2:

    inclusion += 1

  elif massive\_of\_defects[iter] == 3:

    patches += 1

  elif massive\_of\_defects[iter] == 4:

    plitted\_surface += 1

  elif massive\_of\_defects[iter] == 5:

    rolled\_in\_scale += 1

  elif massive\_of\_defects[iter] == 6:

    scratches += 1

#добавим количество в массив

y.append(crazing)

y.append(inclusion)

y.append(patches)

y.append(plitted\_surface)

y.append(rolled\_in\_scale)

y.append(scratches)

#Построим гистограмму

fig, ax = plt.subplots()

ax.bar(x, y)

plt.title('Распределение дефектов')

plt.xlabel('Дефект')

plt.ylabel('Количество')

ax.set\_facecolor('white')

fig.set\_facecolor('white')

fig.set\_figwidth(12) #Ширина графика

fig.set\_figheight(8) #Высота графика

#plt.grid(True) #Раскоментировать, если требуется сетка

#Построим классический график

fig = plt.figure(figsize=(10, 6))

plt.grid()

plt.title('Распределение дефектов')

plt.xlabel('Дефект')

plt.ylabel('Количество')

plt.plot(x, y)

#Построим круговую диаграмму

vals = [] #Массив процентов

persent\_of\_clear\_surface = ((number\_of\_fragments\_img) - len(massive\_of\_defects)) / (number\_of\_fragments\_img) \* 100 #Процент чистых фрагментов

persent\_of\_crasing = round ((y2[0] / 3), 3) #Процент волосных трещин

persent\_of\_inclusion = round ((y2[1] / 3), 3) #Процент посторонних включений

persent\_of\_patches = round ((y2[2] / 3), 3) #Процент пятен

persent\_of\_plitted\_surface = round ((y2[3] / 3), 3) #Процент рябизны

persent\_of\_rolled\_in\_scale = round ((y2[4] / 3), 3) #Процент вдавленных окалин

persent\_of\_scratches = round ((y2[5] / 3), 3) #Процент царапин

vals.append(persent\_of\_clear\_surface)

vals.append(persent\_of\_crasing)

vals.append(persent\_of\_inclusion)

vals.append(persent\_of\_patches)

vals.append(persent\_of\_plitted\_surface)

vals.append(persent\_of\_rolled\_in\_scale)

vals.append(persent\_of\_scratches)

labels\_for\_round\_diagram = ["clear\_surface", "crasing", "inclusion", "patches", "plitted\_surface", "rolled\_in\_scale", "scratches"]

fig, ax = plt.subplots()

ax.pie(vals, labels = labels\_for\_round\_diagram)

ax.legend(loc = 'upper left', bbox\_to\_anchor = (1.0, 1.0))

ax.axis("equal")

print("Процент чистой поверности: ", persent\_of\_clear\_surface)

print("Процент волосных трещин: ", persent\_of\_crasing)

print("Процент посторонних включений: ", persent\_of\_inclusion)

print("Процент пятен: ", persent\_of\_patches)

print("Процент рябизны: ", persent\_of\_plitted\_surface)

print("Процент вдавленных окалин: ", persent\_of\_rolled\_in\_scale)

print("Процент царапин: ", persent\_of\_scratches)

30 ячейка

#Объявим переменные

max = y[0]

position = 0

#Найдём индекс наибольшего значения в массиве

for i in range(len(y)):

    if y[i]>max: max = y[i]; position = i

#Вынесение рекомендаций

if (y[0] > 1 or y[3] > 1 ):

  print('Внимание! Имеются волосные трещины и рябизна. Для повышения качества проката до обычного требебуется провести шлифовку прокатных валов чистовой группы')

elif (position == 1 or position == 4):

  print('Преимущественный дефекты - посторонние включения и вдавленные окалины. Для улучшения качества поверности рекомендуется улучшить зачистку слябов, настроить работу горизонтального и вертикального окалиноломателя перед черновой группой клетей')

elif(position == 2):

  print("Преимущественный дефект - пятна на поверности металла. Для улучшения качества поверности рекомендуется")

elif (position == 5):

  print('Преимущественный дефект - царапины на поверности металла. Для улучшения качества поверности рекомендуется удалить острые части направляющей арматуры или очистить поверхность валков чистовой группы')

31 ячейка

if (y[0] > 1 and y[3] == 0):

  print('Категория качества по ГОСТ 5246-2016, табл. 10  - У')

elif (y[0] > 1 and y[3] > 1):

  print('Категория качества по ГОСТ 5246-2016, табл. 10  - В')

else:

  print('Категория качества по ГОСТ 5246-2016, табл. 10  - Обычная')