МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

ПРИМЕНЕНИЕ МАШИННОГО ОБУЧЕНИЯ В ЗАДАЧЕ КЛАССИФИКАЦИИ ПРОТЕКТОРА ШИНЫ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Монастырской Арины Олеговны

Научный руководитель
зав. каф. техн. пр., к. ф.-м. н.,
доцент _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С 2019 года в Российской Федерации существует национальный проект «Безопасные качественные дороги», направленный на повышение качества жизни населения и обеспечение необходимого уровня безопасности дорожного движения. На выполнении поставленных в рамках Проекта задач негативно сказалась пандемия — были сокращены объемы финансирования. Было спровоцировано увеличение спроса на поддержанные малобюджетные транспортные средства и сокращение расходов на их содержание. Повысился процент ДТП по причине неисправности автомобилей. Надлежащее состояние шин является одним из главных составляющих безопасного движения [1]. В правилах дорожного движения прописаны условия использования шин, их состояние также проверяют на плановых технических осмотрах. Данный вопрос остается открытым, так как отслеживать актуальное состояние протекторов таким образом не удается.

Использование машинного обучения, в перспективе в камерах видеонаблюдения, на дорогах даст возможность усовершенствовать отслеживание текущего состояния протектора.

Цель данной работы состоит в том, чтобы, с помощью создания собственной архитектуры модели последовательной сверточной нейронной сети, определять состояние протектора шины на загружаемом изображении (поврежденное или нет), упростив при этом взаимодействие пользователя с нейросетью посредством разработки веб-приложения.

Ниже приведены задачи для достижения поставленной цели:

1. Найти и загрузить изображения для классификации;
2. Предобработать полученные изображения — аугментировать их;
3. Создать архитектуру модели сверточной нейронной сети;
4. Обучить модель на тренировочной выборке;
5. Проверить работу классификатора на тестовой выборке и проанализировать результаты;
6. Сохранить модель;
7. Создать веб-приложение для удобного взаимодействия с моделью.

Работа выполнена на 45 страницах машинописного текста, состоит из введения, теоретической, практической частей и заключения, содержит 1 таблицу, 34 рисунка, 10 приложений, список литературных источников содержит 22 наименования.

1 Теоретическая часть

1.1 Архитектура нейронной сети

Сверточная нейросеть на сегодняшний день является наиболее эффективной в обработке изображений среди других видов нейросетей. Идея сверточной нейросети состоит в выделении признаков исходных изображений. Данные выделения признаков осуществляются посредством реализаций операций сверток и субдискретизаций [2].

Топология используемой архитектуры сверточной нейронной сети представлена на изображении 1.

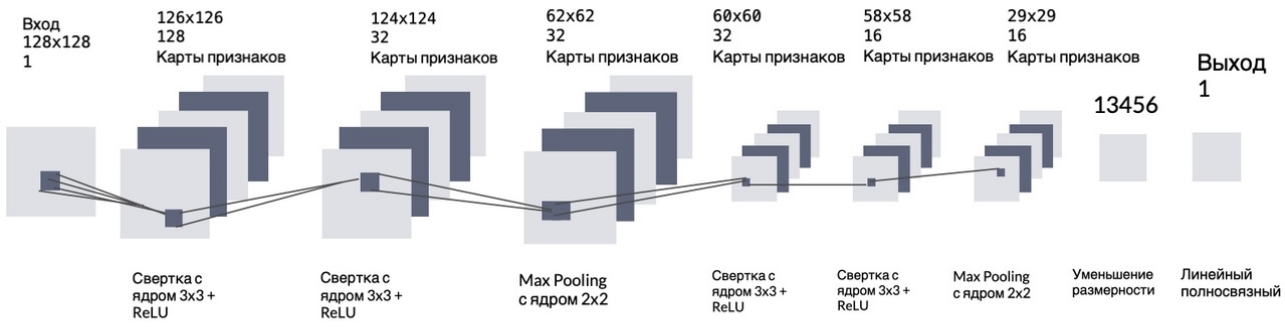

Рисунок 1 – Топология сверточной нейронной сети

— Первый сверточный слой.
  1. Слой двумерной свертки.

     128 ядер. Ядро размера 3 x 3 с шагом по умолчанию 1. Функция активации ReLU. Значение padding по умолчанию.
  2. Слой двумерной свертки.

     32 ядра. Ядро размера 3 x 3 с шагом по умолчанию 1. Функция активации ReLU. Значение padding по умолчанию.
  3. Слой MaxPooling.

     32 ядра. Ядро размера 2 x 2 с шагом по умолчанию 1.
— Второй сверточный слой.
  1. Слой двумерной свертки.

     32 ядра. Ядро размера 3 x 3 с шагом по умолчанию 1. Функция активации ReLU. Значение padding по умолчанию.
  2. Слой двумерной свертки.

     16 ядер. Ядро размера 3 x 3 с шагом по умолчанию 1. Функция

активации ReLU. Значение padding по умолчанию.

3. Слой MaxPooling.

   16 ядер. Ядро размера 2 x 2 с шагом по умолчанию 1.

— Слой Flatten — без параметров — выравнивание входа.

— Выходной слой — плотно связанный слой.

  Размерность выходного пространства — 1, так как бинарная классификация. Функция активации — Sigmoid.

1.2 Веб-приложение

Веб-приложение в представленной работе было построено по принципу Клиент-Сервер. Клиент-серверная архитектура состоит из трех компонентов:

— Клиент — локальный компьютер на стороне пользователя. Клиент отправляет запрос к серверной части.

— Сервер — системное оборудование, предназначенное для выполнения конкретных запросов, поступающих с клиентской части: предоставление доступа пользователю к системным ресурсам, взаимодействие с базой данных.

— База данных — разрешение проблем безопасности, так как она располагается отдельно от сервера.

1.2.1 Django

Django — высокоуровневый Python Web Framework. Приложение Django приложение включает в себя четыре компонента:

— Модели данных для взаимодействия с базой данных — models. Python-класс, который обращается к данным при запросе. ORM (связь баз данных с объектно-ориентированными языками программирования — объектно-реляционное отображение) обеспечивает модели непосредственный доступ к базе данных;

— Представления — views, принимают запросы со стороны клиента. Есть возможность использовать инструмент устранения дублирования в представлениях — ViewSets — высокоуровневых классов, объединяющих функциональности разных View-классов;

— Шаблоны — формы представления данных. Один из основных средств вывода;

— Внешние доступы к представлениям — urls.

1.2.2 React

Ключевая концепция фреймворка React заключается в разбиении веб-страницы на части — компоненты, которые описываются функциями или классами языка JavaScript, реализующими метод render(). Этот метод может возвратить только класс JavaScript или React-элемент. Компоненты являются самодостаточными элементами. В зависимости от способа реализации в React существует два типа компонент: функциональные и классовые.Основным отличием двух типов является возможность классового компонента хранить внутреннее состояние, которое позволяет обновлять пользовательское представление на основе событий [3].

1.2.3 Требуемые модули и библиотеки

Для возможности реализации обмена данными между frontend- и backend-частями используется технология совместного использования ресурсов между разными источниками — CORS (Cross-Origin Resource Sharing). Данная технология позволяет вносить коррективы в ограничения, обеспечивая при этом безопасный доступ. Для того, чтобы браузер разрешил доступ к ресурсам из другого источника, он должен получить определенные заголовки в ответе от сервера, которые указывают, разрешает ли сервер запросы из других источников. Эти заголовки — corsheaders, прописываются в настройках приложения Django. Модуль corsheaders сообщает браузеру, что веб-приложения запущены на одном источнике и ошибки не возникает [4].

2 Практическая часть

2.1 Загрузка и предобработка данных

2.1.1 Описание датасета

Набор изображений включает в себя разделенные на два класса в зависимости от поврежденности протекторов шин кадры с различными разрешениями. Набор данных был предварительно разделен на тренировочную, валидационную и тестовую выборки. Всего набор содержал 1019 элементов.

2.1.2 Предобработка изображений

Изображения были объединены в две директории, разграничивающие данные на два класса — damaged и normal, в зависимости от поврежденности протекторов шин. Данные были разделены на тренировочную и тестовую выборки в процентном соотношении 80% на 20%, соответственно.

Так как сверточная нейронная сеть требует большого количества данных для обучения, требовалось провести аугментацию изображений, находящихся в тренировочной выборке. Размеры датасета были увеличены практически в 10 раз с помощью применения различных преобразований к изображениям.

Перед созданием архитектуры сверточной нейронной сети к загруженным изображениям были добавлены метки, показывающие, к какому классу относится изображение. Были определены две метки: «1» или «0», в зависимости от поврежденности протектора.

После чего данные тренировочной выборки были нормализованы.

2.1.3 Обучение модели нейронной сети

Модель была связана с метриками, оптимизатором и функцией потерь. Обучение модели сверточной нейронной сети было структурировано по эпохам. Всего в процессе обучения было использовано 15 эпох. Размер батчей был задан по умолчанию — 64.

Используемые данные представляют из себя Numpy-массивы, следовательно, в таком случае была предоставлена возможность указания процента имеющихся данных для использования их в качестве проверочной выборки. В работе были взяты последние 30% наблюдений для контроля процесса обучения — валидации.

2.1.4 Результаты обучения модели нейронной сети

На тренировочной выборке модель показала следующие результаты: точность составила 91.93%, функция потерь — 19.81%. На валидационной выборке: точность — 86%, функция потерь — 37%.

Данные для тестирования были приведены к тому же виду, что и тренировочная выборка. Модель была оценена на тестовых данных: точность составила 89%, а результат функции потерь — 27%. Таким образом, на тестовых данных последовательная модель сверточной нейронной сети показала результаты, сравнимые с результатами, представленными на тренировочной выборке, следовательно, возможного переобучения не произошло.

Обучение модели на основе динамики метрики точности и функции потерь по эпохам (по оси X — от 0 до 15) для тренировочной и валидационной выборок представлена на изображении 2.

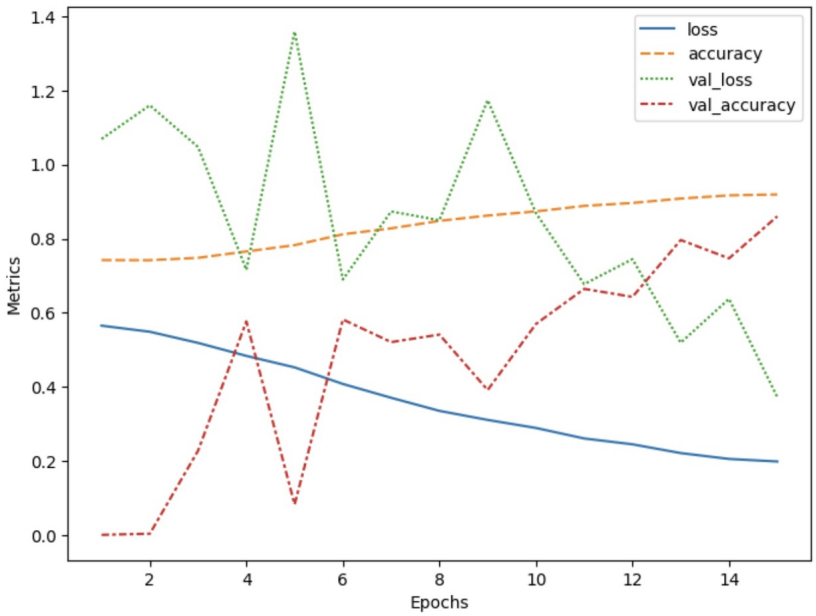

Рисунок 2 – Динамика обучения модели

Из графика видно, что функция потерь для валидационной и тренировочной выборок уменьшались, в то время как точность постепенно росла — положительная динамика, обучение прошло успешно.

После проведенного анализа модель была сохранена на локальный диск с расширением .h5 для дальнейшего использования в веб-приложении.

2.2 Веб-приложение

Для создания веб-приложения, были созданы: виртуальное окружение, базовые React- и Django-приложения. Для возможности обмена данными между frontend- и backend-частями был определен модуль corsheaders.

2.2.1 Backend веб-приложения

Ниже представлено описание основных компонент backend-а приложения, выполненного с помощью Django:

- модель протектора шины определяет функционал, задающий основную логику работы классификатора и получения конечных результатов.
- класс сериализаторов переводит изображение в нужный для взаимодействия с ним формат — строку байт.
- файл представлений отвечает за обработку входящих запросов.
- файл URL-маршрутов принимает запросы от клиента, за обработку которых отвечает класс представления.
- файл настроек Django-проекта, в том числе взаимодействие с frontend-частью.

2.2.2 Frontend веб-приложения

После создания backend части, реализованной с помощью Django, был создан функциональный React компонент ImagePicking, который отвечает за загрузку изображения с локального диска на веб-страницу, его замену и отправку на вход классификатору и визуализацию результата работы классификатора — таким образом, данный компонент реализует всю логику пользовательского представления.

Данный функционал реализован через объявления простых компонент, которые исполняются только при наступлении конкретных событий — нажатий кнопок в JSX-части [5].

За основной JavaScript компонент отвечает файл приложения $App.js$, в котором прописывается основной React компонент App, использующий функциональный компонент ImagePicking и реализующий метод render(), в котором формируется описание его представления. Данный метод возвращает выражение JSX — HTML-разметку для ее вывода на веб-страницу.

2.3 Результаты

В результате выполненной работы были созданы веб-приложение и модель сверточной нейронной сети, а также реализовано их взаимодействие.

Frontend-часть веб-приложения была реализована с помощью JavaScript-библиотеки React и расположена по адресу 127.0.0.1:3000, Backend-часть была реализована с помощью набора инструментов Django REST framework и расположена по адресу 127.0.0.1:8000.

В серверную часть веб-приложения также была загружена предварительно обученная и локально сохраненная последовательная модель сверточной нейросети, созданная для определения класса изображения в зависимости от категории представленного на нем протектора шины — «0» или «1» (проверка на наличие повреждений — изображению с поврежденным протектором присваивается значение класса «1», с неповрежденным — «0») с точностью до 89%.

2.3.1 Пример работы классификатора для изображения с неповрежденным протектором шины

Результатом работы последовательной модели сверточной нейронной сети после нажатия на кнопку «Отправить классификатору» для приведенного выше примера загруженного изображения является всплывающее окно с ответом «На изображении представлен исправный протектор», как представлено на изображении 3.

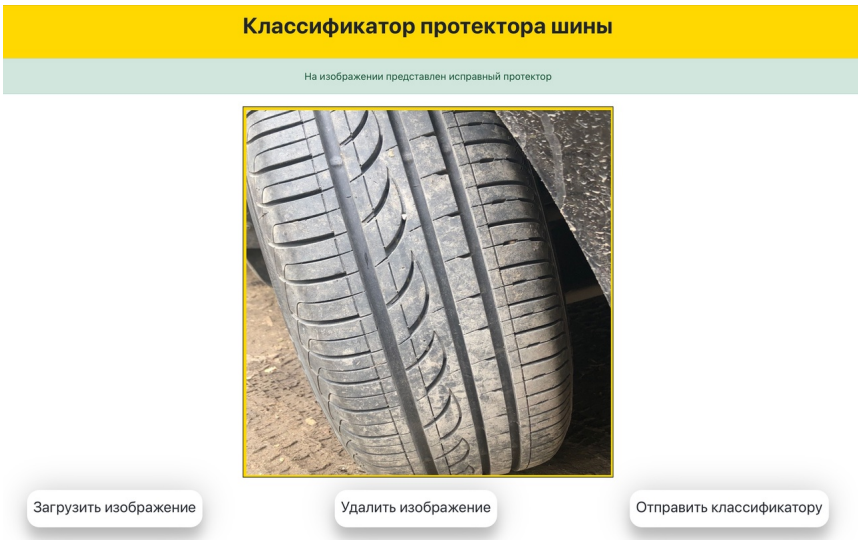

Рисунок 3 – Пример результата работы модели сверточной нейронной сети

Таким образом, результат работы модели сверточной нейронной сети является корректным — протектор шины на изображении является неповрежден-

ным.

2.3.2 Пример работы классификатора для изображения с поврежденным протектором шины

Так как в научной работе был реализован бинарный классификатор, то необходимо было проверить и предоставить результаты работы модели сверточной нейронной сети для двух классов.

Результатом работы последовательной модели сверточной нейронной сети после нажатия на кнопку «Отправить классификатору» для загруженного с локального диска реального изображения с поврежденным протектором является всплывающее окно с ответом «На изображении представлен поврежденный протектор», как представлено на изображении 4.

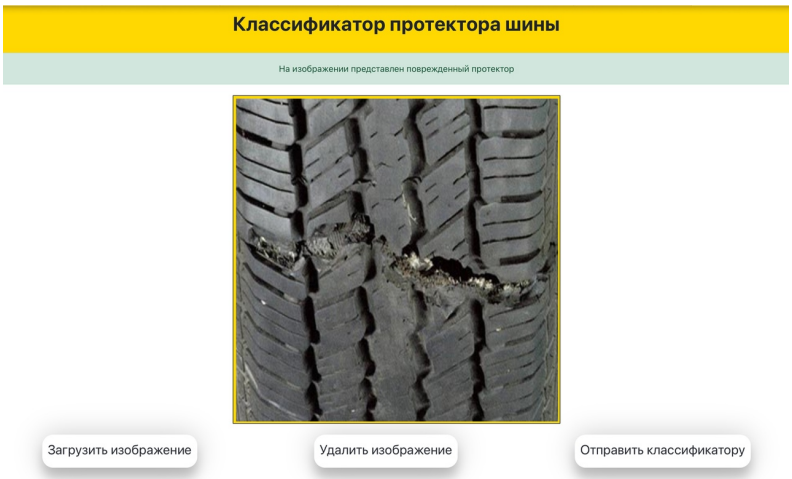

Рисунок 4 – Пример результата работы модели сверточной нейронной сети

На представленном изображении видно, что результат работы модели сверточной нейронной сети является корректным — протектор шины является поврежденным.

Таким образом, на примерах реальных данных классификатор показал корректные ответы.

Заключение

В ходе представленной работы были изучены и использованы:

- CNN — сверточная нейронная сеть для определения класса изображения;
- Django для реализации backend-части веб-приложения;
- React для реализации frontend-части веб-приложения.

Было создано веб-приложение, определяющее состояние протектора шины на любом выбранном изображении.

Проведенное исследование в дальнейшем может быть использовано в качестве базового функционала для внедрения моделей классификации изображений в камеры видео-наблюдений на автомобильных дорогах, которые способны фиксировать неисправность и присылать уведомление об этом владельцу автомобиля, вне зависимости от времени прохождения технического обслуживания.

Модели классификаторов могут использоваться как в фото-, так и в видеофиксации, то есть могут быть усовершенствованы до определения состояния протектора шины в режиме реального времени.

Внедрение подобных сверточных нейронных сетей в повседневную жизнь способно сократить процент дорожно-транспортных происшествий по причине неисправных протекторов, в том числе во время неблагоприятных погодных условий.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Масленникова, Е.* Проблемы реализации национального проекта "Безопасные и качественные автомобильные дороги"[Электронный ресурс]. / Е. Масленникова // *Cyber-Leninka*. — 2021. — URL: https://cyberleninka.ru/article/n/problemy-realizatsii-natsionalnogo-proekta-bezopasnye-i-kachestvennye-avto\mobilnye-dorogi/viewer (Дата обращения 23.01.2023). Загл. с экр. Яз. рус.

2 *Goodfellow, I.* Deep Learning / I. Goodfellow, Y. Bengio, A. Courville. — Massachusetts: The MIT Press, 2016. — 502 P.

3 *Хортон, А.* Разработка веб-приложений в ReactJS / А. Хортон. — Москва: ДМК Пресс, 2017. — 256 С.

4 Django CORS Guide: What It Is and How to Enable It [Электронный ресурс]. — URL: https://www.stackhawk.com/blog/django-cors-guide/ (Дата обращения 02.03.2023). Загл. с экр. Яз. англ.

5 *Mangabo, K.* Full Stack Django and React / K. Mangabo. — Packt Publishing, 2023. — 432 P.