



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)»

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
НА ТЕМУ:

**«Метод обнаружения сфальсифицированных видео на основе
нейронных сетей»**

Студент группы ИУ7-82Б

(Подпись, дата)

Куликов Д. А.

(И.О. Фамилия)

Руководитель ВКР

(Подпись, дата)

Тассов К. Л.

(И.О. Фамилия)

Нормоконтролер

(Подпись, дата)

Мальцева Д. Ю.

(И.О. Фамилия)

2022 г.

РЕФЕРАТ

Расчетно-пояснительная записка 81 с., 38 рис., 5 табл., 61 ист.

Ключевые слова: нейронные сети, метод обнаружения сфальсифицированных видео, сверточные нейронные сети, капсульные нейронные сети, рекуррентные нейронные сети, генеративно-состязательные сети.

Объектом разработки является система для обнаружения сфальсифицированных видео.

Цель работы — разработать и реализовать метод обнаружения сфальсифицированных видео на основе нейронных сетей.

Поставленная цель достигается благодаря использованию нейронных сетей, выявляющих манипуляции на видео. Для обнаружения сфальсифицированных видео предложено использование модели ансамбля сетей MesoNet и CapsNet на основе слабых экспертов, использующей подход стекинга.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	9
1 Аналитический раздел	14
1.1 Постановка задачи	14
1.2 Обзор существующих решений	14
1.3 Сверточные нейронные сети	16
1.4 Капсулльные нейронные сети	18
1.4.1 Обзор структуры капсулльной нейронной сети	19
1.4.2 Алгоритм динамической маршрутизации	20
1.5 Реккурентные нейронные сети	22
1.6 Генеративно-состязательные сети	23
1.7 Ансамбли нейросетевых классификаторов	24
1.7.1 Бэггинг	25
1.7.2 Бустинг	29
1.7.3 Стекинг	31
1.8 Создание сфальсифицированных видео, основанных на замене лиц	33
1.9 Существующие методы обнаружения сфальсифицированных видео, основанных на замене лиц	35
1.9.1 Метод обнаружения на основе оценки частоты моргания глаз	36
1.9.2 Метод обнаружения на основе оценки частоты сердечных сокращений .	38
1.9.3 Метод обнаружения на основе оценки лицевых ориентиров	38
1.9.4 Метод обнаружения с помощью сверточной сети MesoNet	41
1.9.5 Метод обнаружения на основе капсулльных нейронных сетей	43
1.10 Сравнение методов обнаружения сфальсифицированных видео	47
1.11 Выводы	48
2 Конструкторский раздел	49
2.1 Декомпозиция задачи	49
2.2 Модифицированный алгоритм динамической маршрутизации	49
2.3 Проектирование ансамбля нейронных сетей	50
2.4 Проектирование архитектуры программного продукта	51
2.5 Выводы	52

3 Технологическая часть	54
3.1 Средства разработки для построения нейронных сетей	54
3.2 Средства разработки для реализации web-приложения	54
3.3 Подготовка данных для обучения	55
3.4 Валидация данных	57
3.5 Используемые метрики в задаче классификации	58
3.6 Реализация капсулной нейронной сети	59
3.7 Реализация нейронной сети MesoNet	60
3.8 Демонстрация работы программного обеспечения	61
3.9 Выводы	66
4 Исследовательский часть	67
4.1 Определение коэффициента скорости обучения	67
4.2 Определение размера пакета	69
4.3 Сравнение реализованных методов обнаружения	71
4.4 Выводы	72
ЗАКЛЮЧЕНИЕ	73
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	75

ТЕРМИНЫ И УСЛОВНЫЕ ОБОЗНАЧЕНИЯ

Искусственный интеллект (ИИ) (англ. Artificial Intelligence, AI) — свойство интеллектуальных компьютерных систем, обладающих возможностями выполнять творческие задачи, которые считаются прерогативой человека [1].

Машинное обучение (англ. Machine Learning, ML) — область искусственного интеллекта, изучающий различные способы построения обучающихся алгоритмов. Среди множества парадигм и подходов в машинном обучении выделяются нейронные сети [2].

Компьютерное зрение (англ. Computer Vision, CV) — область искусственного интеллекта, которое занимается задачами, связанными с анализом изображений и видео [3].

Нейронная сеть (также искусственная нейронная сеть, ИНС) (англ. Artificial Neural Network, ANN) — математическая модель, а также ее программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей, используемая для решения задач ИИ [4].

Датасет (англ. Dataset) — набор данных, выборка [5].

Дипфейк (англ. Deepfake) — методика создания и замены элементов на существующих фотографиях и видео при помощи ИИ [6].

Сокращения названий архитектур нейронных сетей:

- сверточная нейронная сеть (англ. convolutional neural network, CNN);
- капсульная нейронная сеть (англ. capsule neural network, CapsNet);
- рекуррентная нейронная сеть (англ. recurrent neural network, RNN);
- рекуррентная нейронная сеть с долгой краткосрочной памятью (англ. long short-term memory, LSTM);
- генеративно-состязательная сеть (англ. generative adversarial network, GAN).

ВВЕДЕНИЕ

Глубокое обучение успешно применяется для решения различных сложных задач, начиная от анализа больших данных и заканчивая компьютерным зрением. Достижения в этой области используются практически везде, например:

- в медицине, для проведения диагностики и выявления болезней;
- в интернет-магазинах и социальных сетях, для создания рекомендательных систем;
- в кино и рекламе, для преобразования изображений;
- в машиностроении, для создания беспилотного управления и т.д.

Одной из недавно появившихся технологий, основанных на глубоком обучении, является дипфейк (англ. Deepfake) — методика создания и замены элементов на существующих фотографиях и видео при помощи искусственного интеллекта.

В связи с постоянным усовершенствованием компьютерных и мобильных устройств, а также ростом использования социальных сетей и порталов обмена медиа, в сети ежедневно увеличивается количество сфальсифицированных фотографий и видео, сделанных с помощью технологии Deepfake. До недавнего времени количество поддельных видеороликов и степень их реалистичности были ограничены отсутствием сложных инструментов редактирования, высоким спросом на экспертные знания в предметной области, а также сложным и трудоемким процессом. Однако в последние годы время изготовления и обработка видео значительно сократилось благодаря доступу к большим объемам обучающих данных и высокопроизводительным вычислительным мощностям, но в большей степени из-за развития методов машинного обучения и компьютерного зрения, которые устраниют необходимость в ручном редактировании.

Deepfake активно используются в рекламе, кино, моде и обучении. Так, например, Первый канал для продолжения сериала «Диверсант» воссоздал, с

помощью данной технологии, персонажа умершего актера Владислава Галкина — Григория Калтыгина (см. рисунок 0.1). Причем его роль используется не просто в эпизоде, а на протяжении 8 минут [7].



Рисунок 0.1 – Первый канал создал дипфейк актера Владислава Галкина.

Ещё один яркий пример — новогодняя реклама Сбербанка, в которой герой фильма «Иван Васильевич меняет профессию» Жорж Милославский оказался в 2020 году (см. рисунок 0.2). Тот самый герой, призывавший советских граждан хранить деньги в сберегательной кассе, попал в будущее с его моментальными платежами, доставкой еды, заказом такси и другими сервисами экосистемы банка [8].



Рисунок 0.2 – Сбербанк «вернул» Жоржа Милославского на экраны в своей новогодней рекламе.

Оценивая будущее технологии, сложно игнорировать новостные ленты, которые всё чаще сообщают о вреде ее использования. ИТ-аналитики заявля-

ют, что Deepfake может стать самой опасной технологией в цифровой сфере за последние десятилетия [9]. С распространением дипфейков участились случаи дискредитации людей. Больше 90% подделок созданы, чтобы навредить репутации [10]. Например, мошенники создали дипфейк с Олегом Тиньковым (см. рисунок 0.3) для рекламы поддельной страницы «Тинькофф Инвестиций» [11].



Рисунок 0.3 – Дипфейк с Олегом Тиньковым для рекламы поддельной страницы «Тинькофф Инвестиций».

В мире политики технология Deepfake может быть использована, как мощное оружие против отдельных деятелей и целых партий, чтобы манипулировать общественным восприятием [12]. Например, в ходе спецоперации на Украине в интернете резко увеличилось число сфальсифицированных видео [13]. Президент Украины Владимир Зеленский стал жертвой дипфейка (см. рисунок 0.4) — в сети появилось поддельное видео с его изображением, призывающим сложить оружие [14]. Такие сообщения провоцируют опрометчивые поступки, угрожают здоровью людей, вызывают беспорядки в обществе.

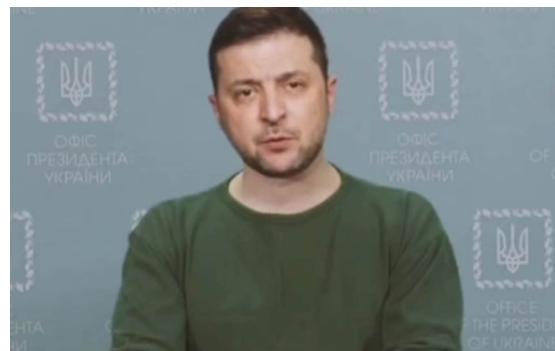


Рисунок 0.4 – Дипфейк президента Украины Владимира Зеленского.

В настоящее время нет прямого запрета в сети на использование данной

технологии. Однако размещение поддельных роликов регулируются крупнейшими социальными сетями, включая Reddit, Twitter, Facebook и TikTok [15]. Для борьбы с поддельными видеороликами крупные ИТ-компании периодически объявляют конкурс на создание лучшего Deepfake-детектора [16]. Вопросом выявления дипфейков занимаются также государственные органы разных стран. Так, в феврале 2021 года конкурс на создание эффективного детектора объявило МВД России. Начальная сумма контракта — 4,8 млн рублей [17].

На данный момент существуют несколько методов, позволяющих распознать сфальсифицированные видео. Однако, точность обнаружения и скорость определения поддельных видеороликов не всегда позволяют использовать их в системах автономного управления. В настоящее время российский рынок ИБ не предлагает специальных технологий и решений с открытым исходным кодом для защиты от дипфейков. Таким образом, задача создания качественного алгоритма распознавания сфальсифицированных видео является актуальной.

Сфальсифицированные видео можно разделить на:

- видео, основанные на соединении кадров;
- видео, основанные на копировании и перемещении объектов;
- видео, основанные на замене лиц.

В выпускной квалификационной работе будут рассмотрены существующие методы обнаружения сфальсифицированных видео, основанных на замене лиц, так как именно они представляют наибольшую угрозу для безопасности людей.

Цель работы — разработать и реализовать метод обнаружения сфальсифицированных видео на основе нейронных сетей.

Для достижения поставленной цели необходимо решить следующие задачи:

- проанализировать предметную область;
- выполнить обзор существующих подходов обнаружения поддельных видео с помощью нейронных сетей;

- в результате полученных во время анализа данных разработать метод обнаружения сфальсифицированных видео на основе нейронных сетей;
- реализовать разработанный метод в программном продукте;
- провести исследование работоспособности реализованного метода обнаружения.

1 Аналитический раздел

1.1 Постановка задачи

В соответствии с заданием на выпускную квалификационную работу необходимо разработать метод для обнаружения сфальсифицированных видео.

На вход алгоритму обнаружения подаются видеокадры. Ограничение на загрузку видеофайлов: допустимыми видеоформатами являются .AVI и .MP4, так как они являются самыми распространенными и кроссплатформерными.

Выходом алгоритма является спрогнозированный класс видеоролика (настоящий или поддельный).

Постановка задачи представлена на рисунке 1.5.

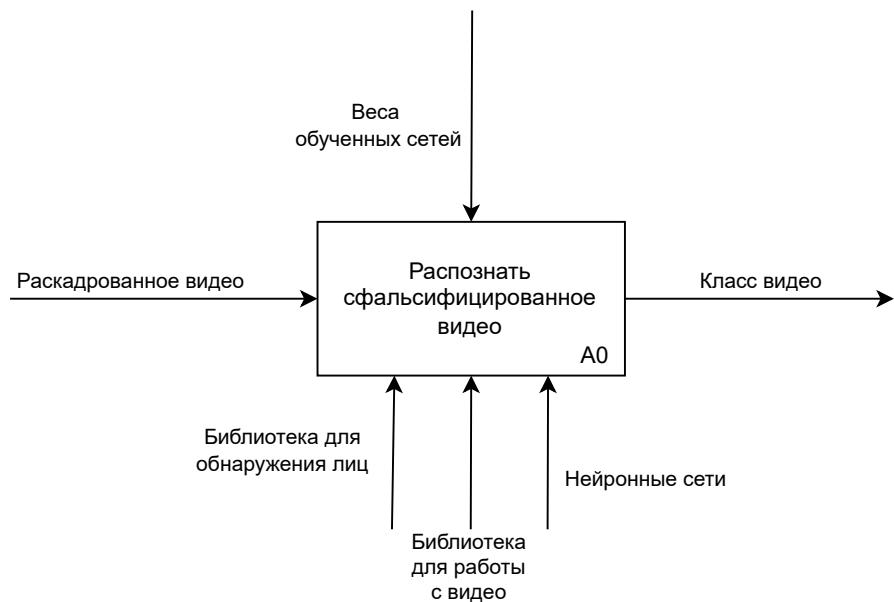


Рисунок 1.5 – Постановка задачи.

1.2 Обзор существующих решений

На данный момент существует несколько сервисов, предоставляющих возможность проверки манипуляции над видео.

KaiCatch — корейский платный сервис, с помощью которого можно распознать поддельную фотографию или видеозапись с точностью до 90%. В основе алгоритмов распознавания дипфейков лежат технологии ИИ. Стоимость

одной проверки картинки или видео составляет около 1,7\$. У KaiCatch нет своего сайта. Доступ к сервису можно получить только через Android-приложение. Язык софта корейский. В будущем планируется перевод приложения на английский, японский и китайский языки, а также разработка приложение под платформу iOS.

Deepware — сервис с открытым исходным кодом, который бесплатно анализирует видео на дипфейки. На данный момент, сервис находится в стадии бета-тестирования, поэтому он работает не стабильно и результаты обнаружения не всегда являются удовлетворительными. Также существует мобильное приложение Deepware на Android.

Microsoft Video Authenticator — программное обеспечение для борьбы с дипфейками, разрабатываемое компанией Microsoft. Программа определяет вероятность монтажа на фото и видео. Обучение модели обнаружения происходит с помощью общедоступных наборов данных FaceForensic++ и DFDC. На данный момент приложение не является общедоступным. Когда компания выпустит технологию в общее пользование, неизвестно.

Сравнительный анализ существующих решений может быть представлен в таблице 1.

Сервис	Стоимость	Общедоступность	Точность обнаружения
KaiCatch	1.7\$ за один прогноз	+	Высокая, 90%
Deepware	Бесплатно	+	Низкая, работает нестабильно
Microsoft Video Authenticator	-	-	Неизвестно

Таблица 1 – Сравнение методов обнаружения сфальсифицированных видео.

1.3 Сверточные нейронные сети

Использование традиционных нейронных сетей для работы с изображениями часто затруднено из-за большого размера входного вектора и количества нейронов в промежуточных слоях. Для обучения такой сети требуются значительные затраты вычислительных ресурсов. Сверточные нейронные сети в меньшей степени, обладают недостатками, описанными выше.

Сверточная нейронная сеть представляет собой специальную архитектуру, нацеленную на эффективную обработку изображений [18], которая включает в себя три типа слоев: сверточный, подвыборочный и полносвязный. Основная идея заключается в том, что в начале чередуются несколько сверточных и подвыборочных слоев, а затем на выходе присутствуют некоторое количество полносвязных слоев. Данная сеть является односторонней.

Пример структуры сверточной нейронной сети представлен на рисунке 1.6.

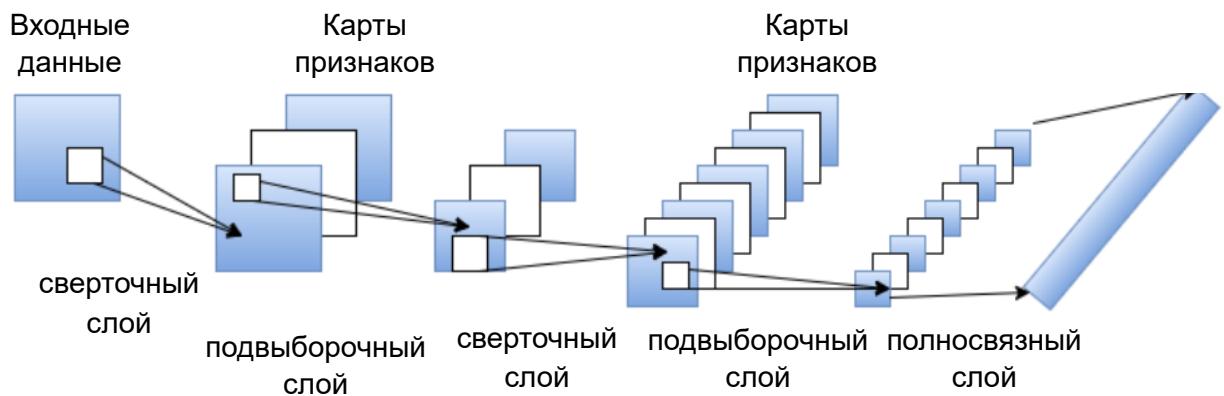


Рисунок 1.6 – Пример структуры сверточной нейронной сети.

Сверточный слой представляет собой применение операции свертки к выходам из предыдущего слоя, формируя при этом карту признаков. Ядро свертки или фильтр, представляет собой матрицу весов, которая кодирует какой-либо признак, например, наличие края, угла или линии. Фильтр «проходит» над изображением, поэтапно выполняя операцию умножения с той частью вход-

ных данных, над которой он сейчас находится, а затем суммирует полученные значения в один выходной пиксель. Признаки на выходе являются взвешенными суммами признаков на входе, расположенных примерно в таком же месте, что и выходной пиксель. При этом размерность выхода уменьшается. Веса ядра свертки неизвестны и корректируются в процессе обучения [18].

На рисунке 1.7 представлен пример операции свертки и получения карты признаков.

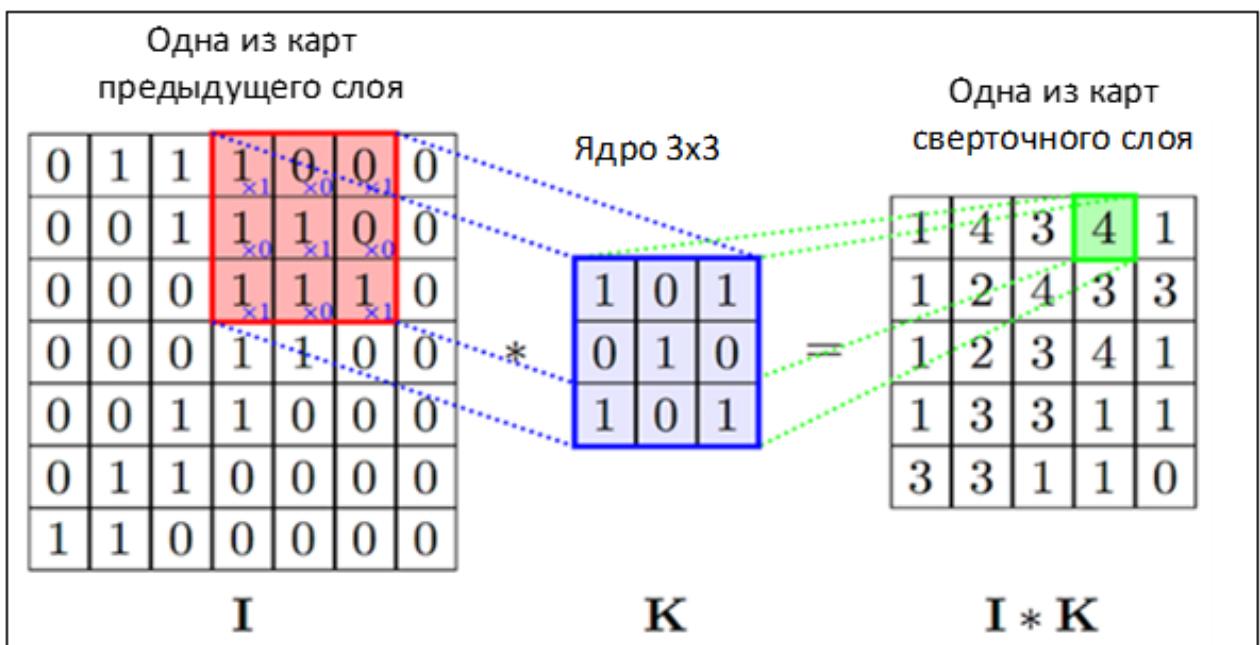


Рисунок 1.7 – Операция свертки и получение значений карты признаков.

Подвыборочный или пулинговый слой (англ. pooling) представляет собой нелинейное уменьшение размерности карт признаков, при этом группа пикселей сжимается до одного. Если на предыдущей операции свертки были выявлены какие-либо признаки, то для дальнейшей обработки исходное изображение уже не нужно, и оно сжимается до менее подробного. При этом часто используется функция максимума (англ. max-pooling), где из небольшой прямоугольной или квадратной области выбирается пиксель, имеющий максимальное значение. Иногда используется функция нахождения среднего значения или L2-нормирования. Подвыборочный слой существенно уменьшает исходный размер изображения.

На рисунке 1.8 представлен пример пулинга с функцией максимума и фильтром 2×2 с шагом 2.

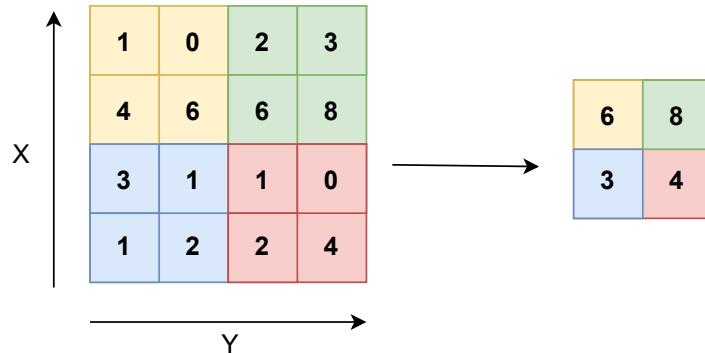


Рисунок 1.8 – Пулинг с функцией максимума и фильтром 2×2 с шагом 2

Стоит отметить, что современные CNN отказываются от использования подвыборочных слоев в пользу чередующихся сверток, в целях более агрессивного уменьшения размера представлений на выходе [19].

После прохождения нескольких операций свертки изображения и подвыборочных слоев сеть перстраивается от конкретных пикселей к более абстрактным картам признаков. Полученные данные объединяются и передаются на полно связные слои, которые обладают небольшой размерностью по отношению к исходному количеству пикселей.

1.4 Капсулевые нейронные сети

В 2017 году Джекфри Хинтон, ученый в области машинного обучения и основоположник алгоритма обратного распространения ошибок, опубликовал статью [20], в которой описал принцип работы капсулевой нейронной сети (англ. CapsNet) и предложил использование алгоритма динамической маршрутизации между капсулами, который позволил достичь более высокой производительности и превзойти CNNs в задачах классификации объектов [22, 23, 24]. Капсулевые сети появились в результате обнаруженных недостатков сверточных нейронных сетей. Внутреннее представление данных CNN не учитывает пространственные расположения между простыми и сложными объектами. Например, если на изображении в случайном порядке расположены глаза, нос или

губы, то для CNN это признак наличия лица (см. рисунок 1.9).

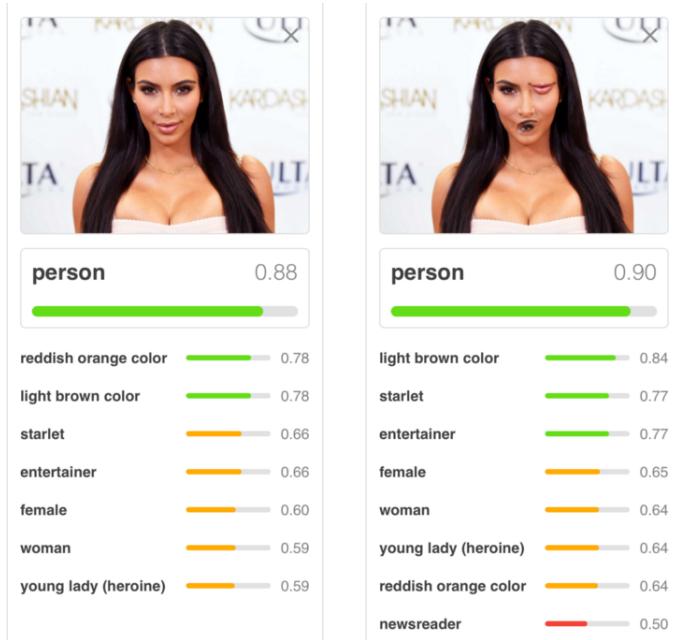


Рисунок 1.9 – Несовершенство CNN в распознавании человека.

Использование CapsNet решает данную проблему. Капсулы в процессе определения лица не только выявляют наличие или отсутствие черт, но и учитывают параметры, в которых они организованы. Это означает, что сеть обнаружит лицо только в том случае, если артефакты, выявленные капсулами, присутствуют в правильном порядке. Стоит отметить, что для обучения распознавания объекта с различных ракурсов с помощью CNN понадобится большое количество данных, тем временем капсулевые сети увеличивают точность распознавания объекта в другом ракурсе на 45% [20].

1.4.1 Обзор структуры капсулой нейронной сети

Основным элементом CapsNet является капсула — вектор нейронов, который описывает существование некоторого объекта. Они могут определять позицию, поворот, размеры объекта и т.д. По сравнению с нейронами, которые выдают скалярное число, капсулы имеют возможность отслеживать направление функции. Следовательно, если изменится положение объекта, то значение вектора останется прежним, а направление укажет на изменение его положения.

Общую структуру капсулой нейронной сети можно разделить на две части:

- 1) Кодер, который принимает входные данные изображения и отображает их в векторной форме, содержащей все параметры объекта, необходимые для рендеринга изображения. Кодер дополнительно инкапсулирует:
 - свёрточный слой, цель которого состоит в извлечении из входного изображения базовых паттернов;
 - слой первичных капсул (PrimaryCaps), цель которого состоит в создании комбинации более сложных форм на основе ранее найденных признаков, обнаруженных свёрточным слоем;
 - капсулый слой (DigitCaps), который содержит все параметры для создания объекта. Чтобы определить, какая капсула из PrimaryCaps поступает в капсулой слой, ее вес должен совпадать с капсулой из DigitCaps, называемой родительской.
- 2) Декодер, работа которого заключается в декодировании вектора из капсулого слоя в изображение. Декодер изучает особенности выходного вектора из капсулого слоя, который используются для восстановления, или реконструкции, изображения. Для этого применяются полно связанные слои.

1.4.2 Алгоритм динамической маршрутизации

Вход капсулы высокого уровня s_j представляет из себя взвешенную сумму по каждому вектору предсказания капсул более низкого уровня:

$$s_j = \sum_{i=1} c_{ij} \hat{u}_{j|i}, \quad (1)$$

где c_{ij} – коэффициент связи капсулы нижнего уровня i с капсулой более высокого уровня j .

Вес c_{ij} является неотрицательным числом. Для каждой капсулы нижнего уровня i сумма всех весов c_{ij} равна единице, т. е. $\sum_{i=1} c_{ij} = 1$. Эти веса вычисляются с помощью алгоритма динамической маршрутизации.

Таким образом, вычисляется распределение вероятности выхода капсулы низкого уровня, принадлежащее капсule более высокого уровня.

На рисунке 1.10 представлен псевдокод алгоритма динамической маршрутизации.

Procedure 1 Routing algorithm.

```

1: procedure ROUTING( $\hat{\mathbf{u}}_{j|i}$ ,  $r$ ,  $l$ )
2:   for all capsule  $i$  in layer  $l$  and capsule  $j$  in layer  $(l + 1)$ :  $b_{ij} \leftarrow 0$ .
3:   for  $r$  iterations do
4:     for all capsule  $i$  in layer  $l$ :  $\mathbf{c}_i \leftarrow \text{softmax}(\mathbf{b}_i)$             $\triangleright \text{softmax}$  computes Eq. 3
5:     for all capsule  $j$  in layer  $(l + 1)$ :  $\mathbf{s}_j \leftarrow \sum_i c_{ij} \hat{\mathbf{u}}_{j|i}$ 
6:     for all capsule  $j$  in layer  $(l + 1)$ :  $\mathbf{v}_j \leftarrow \text{squash}(\mathbf{s}_j)$             $\triangleright \text{squash}$  computes Eq. 1
7:     for all capsule  $i$  in layer  $l$  and capsule  $j$  in layer  $(l + 1)$ :  $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j$ 
return  $\mathbf{v}_j$ 
```

Рисунок 1.10 – Алгоритм динамической маршрутизации.

На вход алгоритм принимает капсулу $\hat{\mathbf{u}}_{j|i}$ нижнего уровня l и количество итераций маршрутизации r . На выходе из алгоритма получаются капсula более высокого уровня v_j .

Во второй строке инициализируется нулем коэффициенты временного значения b_{ij} , которые в ходе выполнения алгоритма будут итеративно обновляться и после завершения процедуры их значение будет сохранено в c_{ij} .

Третья строка описывает то, что шаги алгоритма 4-7 будут повторяться r раз. Рекомендуется использовать три итерации маршрутизации. Чрезмерная итерация может привести к переобучению и снижению производительности сети.

В четвертой строке вычисляются веса маршрутизации c_i для капсулы нижнего уровня. При этом используется функция softmax, которая нужна для того, чтобы каждый вес c_{ij} был неотрицательным числом, а их сумма равнялась единице.

В пятой строке вычисляется линейная комбинация входных векторов капсулы нижнего уровня, которые взвешены с помощью коэффициентов маршрутизации c_{ij} , рассчитанных на предыдущем шаге.

В шестой строке векторы из последнего шага проходят через нелинейное преобразование (функцию сжатия, англ. squash), обеспечивающее сохранение направления вектора, длина которого не должна превышать единицы. Этот шаг создает выходной вектор v_j для всех более высоких уровней капсулы.

В седьмой строке происходит обновление весов. Новое значение веса равно сумме предыдущего значения и скалярного произведения текущего выхода капсулы v_j и входа в неё капсулы более низкого уровня $\hat{u}_{j|i}$. Скалярное произведение отражает согласованность двух капсул.

1.5 Рекуррентные нейронные сети

Рекуррентная нейронная сеть (англ. recurrent neural network, RNN) — архитектура нейронных сетей, в которой связи между элементами образуют направленную последовательность [21]. Рекуррентные сети часто применяются для решения задачи выявления и классификации действий на видео.

Основная идея рекуррентных нейронных сетей заключается в последовательном использовании информации. В обычных нейронных сетях подразумевается, что все входы и выходы независимы. Но для многих задач это не подходит. Если требуется предсказать результат, то нужно учитывать данные до него. Такие сети называются рекуррентными, потому что они выполняют одну и ту же задачу для каждого элемента последовательности, причем его выход зависит от предыдущих вычислений.

Обучение RNN похоже на обучение обычной нейронной сети. Используется алгоритм обратного распространения ошибки, но с некоторым изменением. Поскольку одни и те же параметры используются во всех временных шагах в сети, градиент на каждом выходе зависит не только от расчетов текущих вычислений шагов, но и от предыдущих временных этапов (см. рисунок 1.11). Например, чтобы вычислить градиент для пятого элемента последовательности, необходимо «распространить ошибку» на 4 шага и суммировать градиенты. Такой подход называется алгоритмом обратного распространения ошибки во времени (англ. Backpropagation Through Time, BPTT).

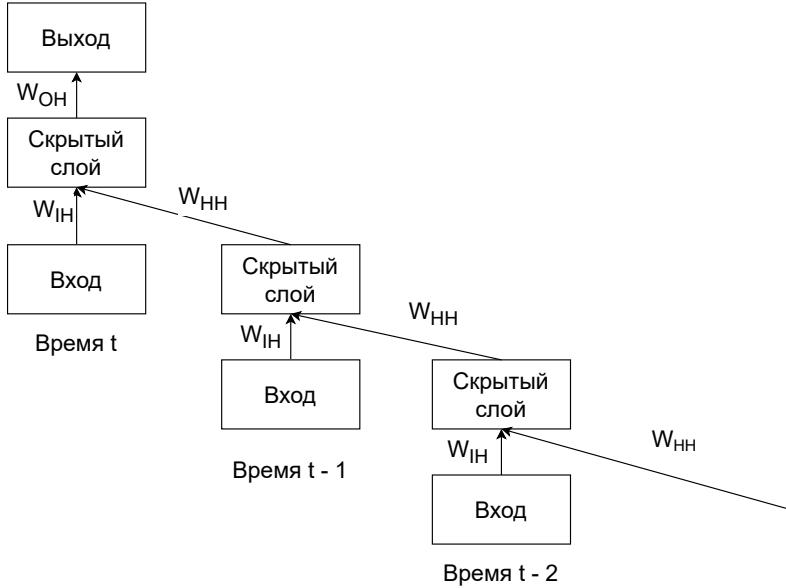


Рисунок 1.11 – Развёртка рекуррентной нейронной сети.

Наиболее часто используемым типом RNN являются сети с долгой краткосрочной памятью (англ. Long short term memory, LSTM), которые лучше хранят долгосрочные зависимости, чем RNN. LSTM используют другую функцию для вычисления скрытого состояния. Память в LSTM называется ячейками, которые принимают на вход предыдущее состояние и текущее входное значение. Внутри эти ячейки решают, какую память хранить, а какую удалять. Затем они объединяют предыдущее состояние, текущую память и входное значение.

1.6 Генеративно-состязательные сети

Генеративно-состязательная сеть (англ. Generative adversarial network, GAN) — архитектура нейронной сети, состоящая из генератора и дискриминатора, которые настроены на работу друг против друга [25].

Сеть генератора (англ. Generator) представляет собой своего рода обратную сверточную сеть, которая генерирует примеры, смешивая несколько исходных и используя вектор случайного шума.

Сеть дискриминатора (англ. Discriminator) — сверточная сеть, которая классифицирует примеры на правильные и неправильные. Результаты различия снова подаются на вход генератору, таким образом, чтобы он смог подобрать лучший набор параметров, а дискриминатор уже не мог бы отличить вновь сгенерированные данные от реальных.

нерированный правильный пример от неправильного.

Таким образом, целью генератора является повысить процент ошибок дискриминатора, цель которого наоборот повысить точность распознавания. В процессе совместного конкурентного обучения, если система достаточно сбалансирована, достигается состояние равновесия, в котором обе сети значительно улучшили свое качество.

GAN обучают создавать структуры, похожие на сущности из мира в области изображений, видео, музыки, речи и прозы.

На рисунке 1.12 представлена структура генеративно-состязательной сети.

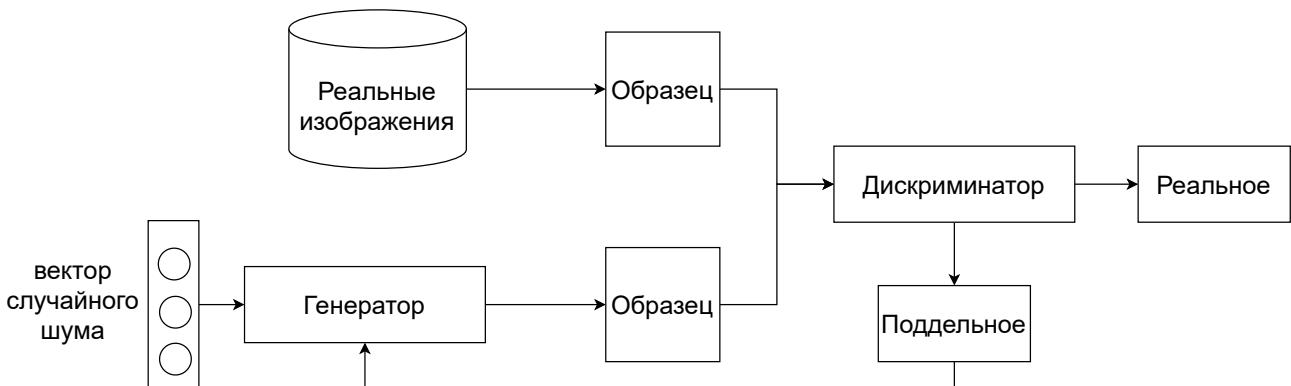


Рисунок 1.12 – Структура генеративно-состязательной сети.

1.7 Ансамбли нейросетевых классификаторов

Глубокое обучение нейронных сетей являются нелинейными методами. Они предлагают повышенную гибкость и могут масштабироваться пропорционально количеству доступных данных обучения. Недостатком этой гибкости является то, что они обучаются с помощью стохастического алгоритма обучения, это означает, что они чувствительны к специфике обучающих данных и могут находить различный набор весов каждый раз, когда они обучаются, что приводит к различным прогнозам. Как правило, это называется нейронными сетями, имеющими высокую дисперсию, что может привести к неудачам при попытке разработать окончательную модель для прогнозирования. Успешный подход к уменьшению дисперсии заключается в обучении нескольких моделей

вместо одной и объединении прогнозов. Такой подход называется ансамблевым обучением [32].

Ансамблевые методы — это подход машинного обучения, в котором несколько моделей (часто называемых «слабыми экспертами») обучаются для решения одной и той же проблемы и объединяются для получения лучших результатов. Основная идея состоит в том, что при правильном сочетании слабых моделей получаются более точные и надежные модели ансамбля («сильный эксперт»). Ансамблевое обучение не только уменьшает дисперсию прогнозов, но также может привести к прогнозам, которые лучше, чем любая отдельная модель.

Основные типы методов, которые направлены на объединение слабых экспертов:

- Бэггинг (англ. bagging). Рассматриваются однородных слабые эксперты, обучение которых происходит параллельно и независимо друг от друга, а затем их результаты объединяют, следуя некоторому детерминированному процессу усреднения. Известным алгоритмом реализации данного метода является случайный лес.
- Бустинг (англ. boosting). Рассматриваются однородные слабые эксперты, обучение которых происходит последовательно адаптивным способом (слабый эксперт зависит от предыдущих), а затем их результаты объединяют, следуя детерминированной стратегии. Известными алгоритмами реализации данного метода являются градиентный и адаптивный бустинг.
- Стекинг (англ. stacking). Рассматриваются разнородные отдельно взятые слабые эксперты, обучение которых происходит параллельно. Существует мета-модель, которой на вход подаются базовые модели, а выходом является итоговый прогноз.

1.7.1 Бэггинг

Бэггинг (от «bootstrap aggregation») — это один из самых простых видов ансамблей, основанный на статистическом методе бутстрэпа, который позволяет оценивать многие статистики сложных распределений.

Метод бутстрэпа заключается в генерации выборок размера B (так называемых бутстрэп выборок) из исходного набора данных размера N путем случайного выбора элементов с повторениями в каждом из наблюдений B . Это означает, что N раз выбирается произвольный объект выборки (считается, что каждый объект выбирается с одинаковой вероятностью $\frac{1}{N}$), причем каждый раз выбирается из всех исходных N объектов. Стоит отметить, что из-за возвращения среди них окажутся повторы. При некоторых допущениях эти выборки имеют довольно хорошие статистические свойства: их можно рассматривать как репрезентативные и независимые выборки истинного распределения данных. Для этого необходимо, чтобы:

- Размер исходного датасета должен быть достаточно большим, чтобы выборка из датасета была хорошим приближением к выборке из реального распределения (репрезентативность).
- Размер исходного датасета должен быть достаточно большим по сравнению с размером бутстрэп выборок B , чтобы выборки не слишком сильно коррелировали друг с другом (независимость).

На рисунке 1.13 представлена иллюстрация процесса бустрэпа.

Идея метода бэггинга заключается в том, что подбирается несколько независимых моделей и усредняются их прогнозы, чтобы получить модель с меньшей дисперсией. Однако на практике трудно подобрать полностью независимые модели, поскольку требуется большой объем данных. Поэтому используются бутстрэп выборки, которые можно рассматривать как «почти репрезентативные и независимые».

Пусть имеется обучающая выборка X , из которой генерируется несколько бутстрэп выборок X_1, \dots, X_B так, чтобы каждая новая бутстрэп выборка была еще одним почти независимым набором данных, взятого из истинного распределения. Затем на каждой выборке происходит обучение слабого эксперта $a_i(x)$.

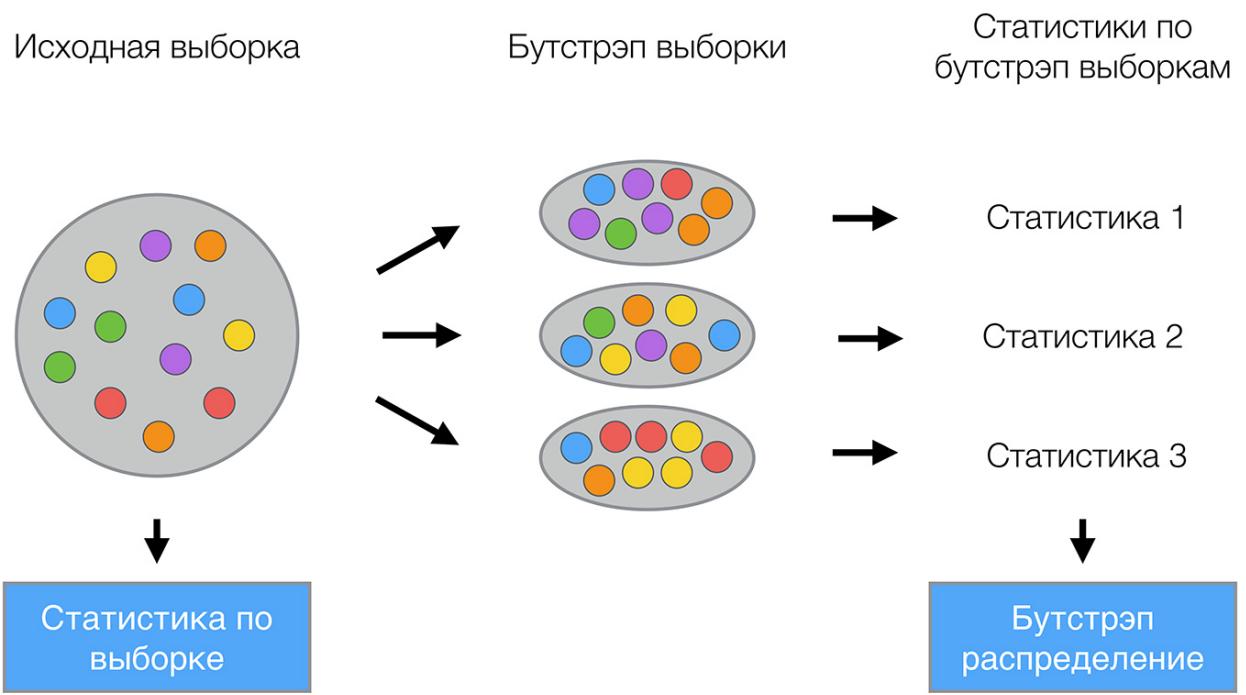


Рисунок 1.13 – Процесс бутсрэпа.

Сильный эксперт будет усреднять результаты полученные от слабых экспертов:

$$a(x) = \frac{1}{M} \sum_{i=1}^M a_i(x) \quad (2)$$

Усреднение результатов не изменяет ожидаемый ответ, но уменьшает его дисперсию. На рисунке 1.14 представлена схема метода бэггинга.

Существует несколько возможных способов объединения моделей, обученных параллельно. Например, для задачи классификации класс, предсываемый каждым слабым экспертом, можно рассматривать как голос, а класс, который получает большинство голосов, является результатом модели ансамбля (это называется мажоритарным голосованием). Также можно рассмотреть вероятности каждого класса, предсываемые всеми слабыми экспертами, усреднить эти вероятности, сохраняя класс с самой высокой средней вероятностью (это называется мягким голосованием или ранжированием).

Метод бэггинга может быть реализован с помощью алгорима случайного леса, где глубокие деревья (с глубиной в множество узлов), обученные на бут-

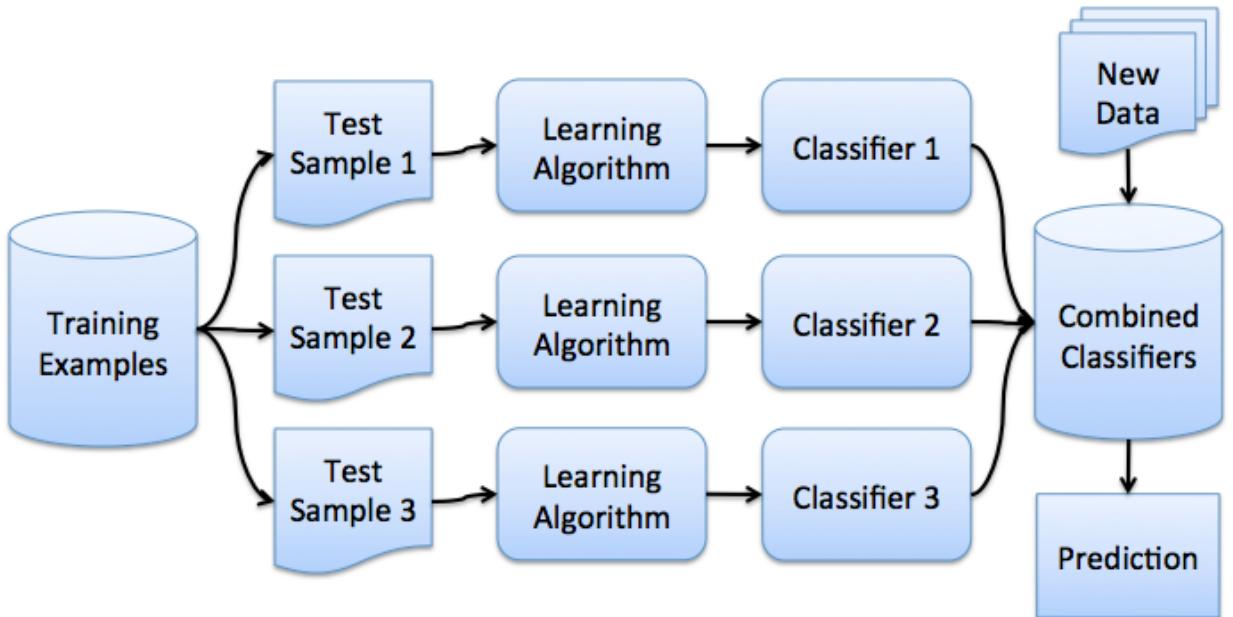


Рисунок 1.14 – Метод бэггинг.

стрэп выборках, объединяются для получения результата с более низкой дисперсией. Для того чтобы несколько обученных деревьев были менее зависимы друг от друга, используется следующий прием: при построении каждого дерева вместо выбора всех признаков из набора данных для генерации бутстрэпа, составляются выборки только с учетом случайного выбора подмножества признаков. Такая выборка приводит к тому, что все деревья не изучают одну и ту же информацию для принятия решения, что уменьшает зависимость между различными возвращаемыми выходными данными. Другое преимущество выборки по признакам заключается в том, что она делает процесс принятия решений более устойчивым к отсутствующим данным: значения наблюдения с отсутствующими данными можно восстановить с помощью регрессии или классификации на основе деревьев, которые учитывают только те признаки, где данные не отсутствуют. Таким образом, алгоритм случайного леса включает в себя идеи бэггинга и выбора подпространства случайных объектов для создания более устойчивых моделей.

1.7.2 Бустинг

В отличие от бэггинга, который направлен на уменьшение дисперсии, бустинг — это метод, идея которого заключается в использовании принципа последовательной адаптации нескольких слабых экспертов: каждая модель в последовательности подбирается так, чтобы она уделяла большее внимание тем объектам в наборе данных, которые плохо обрабатывались предыдущими моделями в последовательности. Каждая новая модель обращает внимание на самые сложные объекты выборки при обучении предыдущих моделей, чтобы получить в итоге сильного эксперта с более низким смещением (даже если бустинг будет при этом уменьшать дисперсию).

Слабые эксперты, которые рассматриваются для бустинга — это модели с низкой дисперсией, но с высоким смещением. Например, если в качестве слабых экспертов используются деревья решений, то рассматриваются неглубокие деревья (с глубиной в несколько узлов). Важная причина, по которой необходимо использовать модели с низкой дисперсией, но с высоким смещением, заключается в том, что они требуют меньших вычислительных ресурсов. Так как вычисления для подбора гиперпараметров к различным моделям не могут выполняться параллельно, в отличие от бэггинга, то это может привести к большим вычислительным затратам.

После того, как слабые эксперты выбраны, необходимо определить какие данные из предыдущих моделей нужно учитывать при подборе текущей модели. Для этого используется два ключевых алгоритма бустинга: адаптивный и градиентный. Они отличаются друг от друга тем, как они создают и объединяют слабых экспертов в ходе последовательного процесса обучения. Адаптивный бустинг обновляет веса, прикрепленные к каждому из объектов обучающего набора данных, в то время как градиентный бустинг обновляет значения этих объектов. Эта разница возникает потому что оба алгоритма решают задачу оптимизации, заключающуюся в поиске наилучшей модели, которая может быть

записана в виде взвешенной суммы слабых учащихся.

При адаптивном бустинге ансамблевая модель определяется, как взвешенная сумма L слабых экспертов:

$$s_L = \sum_{i=1}^L c_i \times w_i, \quad (3)$$

где c_i — коэффициенты, w_i — слабые эксперты.

Поиск лучшей модели ансамбля является сложной задачей оптимизации. Вместо того, чтобы пытаться решить ее за один раз аналитически (находя все коэффициенты и слабых экспертов, которые дают лучшую общую аддитивную модель), используется итеративный процесс оптимизации. На каждой итерации один за другим добавляется и просматривается слабый эксперт в поисках наилучшей возможной пары (коэффициент, слабый эксперт) для добавления к текущей модели ансамбля. Другими словами, s_l итеративно определяется как:

$$s_l = s_{l-1} + c_l \times w_l, \quad (4)$$

где c_l и w_l выбраны так, что s_l — это модель, которая наилучшим образом соответствует обучающим данным и лучше по сравнению с s_{l-1} .

Затем можно определить:

$$(c_l, w_l) = \arg \min_{c, w} E(s_{l-1} + c \times w) = \arg \min_{c, w} \sum_{n=1}^N e(y_n, s_{l-1} + c_l \times w(x_n)), \quad (5)$$

где E — ошибка подгонки данной модели, а e — функция потерь.

Таким образом, вместо глобальной оптимизации L-моделям в сумме, идет приближение к лучшему, оптимизируя локальное построение и добавляя слабого эксперта к сильному одного за другим.

При градиентном бустинге ансамблевая модель также определяется, как

взвешенная сумма L слабых экспертов:

$$s_L = \sum_{i=1}^L c_i \times w_i, \quad (6)$$

где c_i — коэффициенты, w_i — слабые эксперты.

Основное отличие от адаптивного бустинга заключается в определении процесса последовательной оптимизации: на каждой итерации слабый эксперт подгоняется к противоположному градиенту текущей ошибки подбора по отношению к текущей модели ансамбля. Процесс градиентного спуска по ансамблевой модели может быть записан как

$$s_l = s_{l-1} - c_l \times \nabla s_{l-1} E(s_{l-1}), \quad (7)$$

где E — ошибка подгонки данной модели, c_l — коэффициент, соответствующий размеру шага, и $-\nabla s_{l-1} E(s_{l-1})$ является противоположным градиентом ошибки подгонки относительно модели ансамбля на шаге $l-1$. Этот противоположный градиент является функцией, которая оценивается только для объектов в обучающей выборке. Такие оценки называются псевдо-остатками, прикрепленными к каждому объекту. Таким образом, нужно научить слабого эксперта псевдо-остаткам для каждого наблюдения. Коэффициент c_l вычисляется в соответствии с одномерным процессом оптимизации (линейный поиск для получения наилучшего размера шага c_l).

Градиентный бустинг можно рассматривать как обобщение адаптивного бустинга для произвольных дифференцируемых функций потерь.

1.7.3 Стекинг

Стекинг имеет два основных отличия от бэггинга и бустинга:

- данный метод учитывает разнородных слабых экспертов, т.е. объединяются разные алгоритмы и модели обучения, в то время как бустинг и бэггинг учитывают однородных слабых экспертов;
- данный метод объединяет базовые модели с использованием метамодели,

в то время как бустинг и бэггинг объединяют слабых экспертов с помощью детерминистических алгоритмов;

Для обучения стекового ансамбля, составленного из L слабых экспертов необходимо выполнить следующие шаги:

- 1) разделить обучающую выборку на две части;
- 2) выбрать L слабых экспертов и обучить их на первой части;
- 3) для каждого из L слабого эксперта сделать прогнозы для объектов из второй части;
- 4) обучить метамодель, используя в качестве входных данных прогнозы, сделанные слабыми экспертами.

На первом этапе обучающая выборка разделяется на две части, потому что данные, которые используются для обучения слабых экспертов, не должны иметь отношения к обучению метамодели. Очевидным недостатком такого разделения является то, что есть только половина данных для обучения базовых моделей и половина данных для обучения метамодели. Чтобы преодолеть это ограничение можно следовать подходу k -кратной перекрестной проверки (англ. k -fold cross-validation). Исходная выборка случайным образом разбивается на k подвыборок одинакового размера. Из k подвыборок одна подвыборка используется в качестве проверочных данных для тестирования и оценки модели, а остальные $k - 1$ подвыборки используются в качестве обучающих данных. Процесс перекрестной проверки повторяется k раз, причем каждая из k подвыборок используется ровно один раз в качестве данных проверки. Затем результаты k могут быть усреднены для получения единой оценки. Преимущество этого метода перед повторной случайной подвыборкой заключается в том, что все наблюдения используются как для обучения, так и для проверки, и каждое наблюдение используется для валидации ровно один раз. Число k является нефиксированным параметром, однако обычно используется 5-кратная перекрестная проверка. Таким образом, можно создать соответствующие прогнозы для каждого объекта исходной выборки, а затем обучить метамодель всем

полученным прогнозам.

1.8 Создание сфальсифицированных видео, основанных на замене лиц

На данный момент существует два подхода к созданию Deepfake:

- на основе автокодера;
- на основе генеративно-состязательной сети.

Для создания Deepfake-видео на основе автокодера необходимо из подлинных видео извлечь лицевые ориентиры, которые используются для выравнивания граней в соответствии со стандартной конфигурацией [26]. Выровненные лица обрезаются и передаются в автокодер [27] для синтеза Deepfake с теми же выражениями лица, что и у исходного видео.

Автокодер — сверточная нейронная сеть, которая имеет два входа, которые называются кодером и декодером. Кодер E преобразует лицо входной цели в вектор, известный как код. Декодер D , генерирует лицо соответствующего субъекта из кода. Затем синтезированные лица искажаются обратно до конфигурации исходной цели и обрезаются маской из ориентиров лица. Последний шаг включает в себя сглаживание границ между синтезированными областями и исходными видеокадрами. Весь процесс автоматизирован и выполняется практически без ручного вмешательства [28].

На рисунке 1.15 представлены процесс создания Deepfake на основе автокодера.

Исследователи из Samsung AI и Сколковского института науки и технологий создали систему, на основе генеративно-состязательных сетей, которая может создавать реалистичные поддельные видеоролики с говорящим человеком, имея всего несколько его изображений [29].

Предлагаемая структура состоит из трех модулей:

- сеть генератора (Generator);
- сеть встраивания (Embedder);
- сеть дискриминатора (Discriminator).

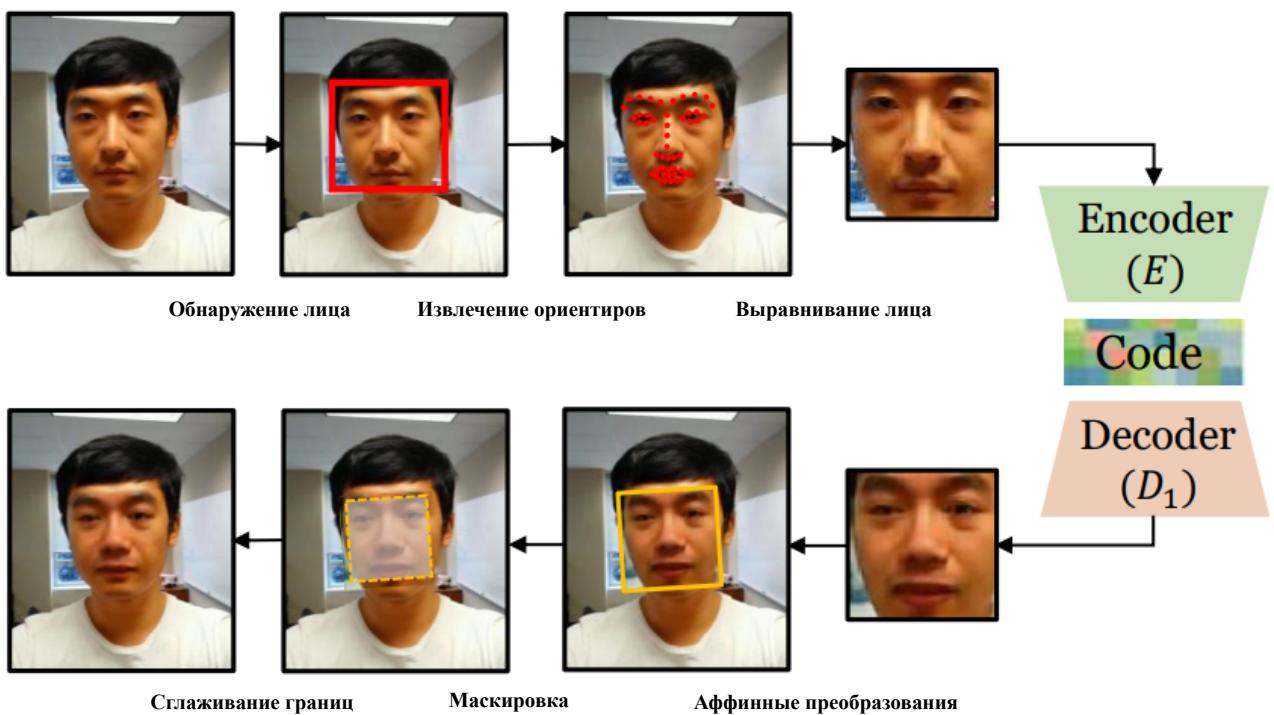


Рисунок 1.15 – Создание Deepfake на основе автокодера.

Структура была разработана таким образом, что она разделяет позу и черты лица человека и использует технику состязательного обучения для создания реалистичных видеороликов. На рисунке 1.16 представлена архитектура создания Deepfake на основе GAN.

Сеть встраивания — это модуль, который извлекает независимые от позы черты человека в данном видеокадре. Предполагается узнать личность человека и генерировать низкоразмерные вложения. Затем эти вложения передаются в сеть генератора как параметры AdaIN (адаптивная нормализация экземпляра). Это позволяет модулировать сверточные слои с помощью скрытых вложений, содержащих индивидуальную информацию.

Генераторная сеть принимает в качестве входных данных ориентиры лица (а также вложения и истинное изображение) и должна создавать образец синтетического изображения человека в виде видеокадра.

Наконец, сеть дискриминатора должна научиться различать распределения и заставлять генератор создавать выборки из реалистичного распределения.

Исследователи обучали систему под наблюдением, используя два набора

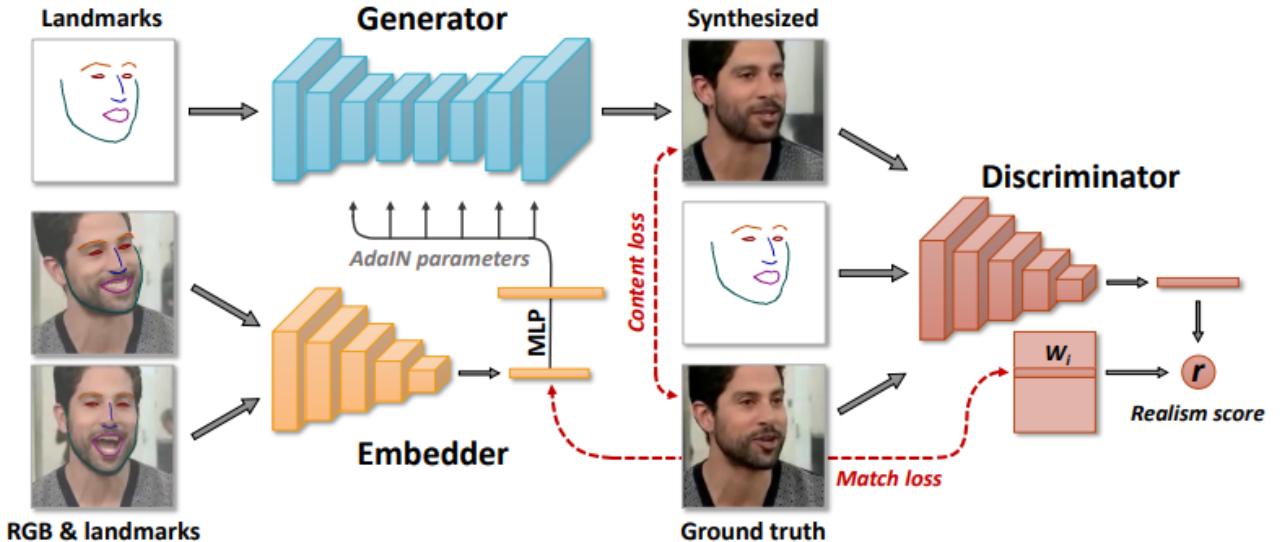


Рисунок 1.16 – Создание Deepfake на основе GAN.

данных видео с говорящими людьми: VoxCeleb1 [30] и VoxCeleb2 [31]. Оценка показала, что предложенный метод способен научиться генерировать видео с говорящей головой всего из одного образца изображения. Однако наилучшие результаты дает модель, обученная на 32 изображениях.

1.9 Существующие методов обнаружения сфальсифицированных видео, основанных на замене лиц

Методы обнаружения Deepfake-видео, можно классифицировать на:

- обнаружение с помощью физических характеристик;
- обнаружение с помощью методов, основанных на выявлении артефактов и манипуляций искажающих видео.

Идея методов обнаружения с помощью физических характеристик, заключается в обнаружении отсутствия физиологических признаков, присущих людям, которые плохо обрабатываются в синтезированных видео. Такие признаки могут включать спонтанные и непроизвольные физиологические действия, например, дыхание, пульс и движение глаз, которые часто упускаются из виду в процессе синтеза поддельных видео.

К методам обнаружения сфальсифицированных видео с помощью физических характеристик можно отнести:

- рекуррентную нейронную сеть, выявляющую частоту моргания глаз;
- сверточную нейронную сеть, выявляющую частоту сердечных сокращений;
- метод опорных векторов для оценки лицевых ориентиров.

К методам обнаружения сфальсифицированных видео, основанных на выявлении артефактов и манипуляций искажающих видео, можно отнести:

- сверточную нейронную сеть MesoNet;
- капсулную нейронную сеть.

В настоящее время злоумышленники научились создавать качественные Deepfake-видео с отсутствием визуальных дефектов [33]. Поэтому сравнение методов обнаружения Deepfake-видео будет проведено по следующим критериям:

- возможность обнаружения Deepfake-видео при отсутствии видимых визуальных дефектов;
- возможность работы с видео разного качества.

1.9.1 Метод обнаружения на основе оценки частоты моргания глаз

Моргание (англ. Eye Blinking) – открывание и закрывание глазных век, при котором передняя часть глазного яблока равномерно смачивается слезой [34]. Средняя частота моргания глаз человека в состоянии покоя составляет 17 морганий в минуту. Во время разговора количество морганий увеличивается до 26 и уменьшается во время чтения до 4.5 морганий в минуту [35]. Продолжительность моргания составляет 0.1 – 0.4 секунды [36].

На рисунке 1.17 представлен пример обнаружения Deepfake с помощью моргания глаз. В оригинальном видео оно происходит в течение 6 секунд, в то время как в поддельном это не так, что является неестественным с физиологической точки зрения.

Метод обнаружения DeepFake с помощью моргания глаз основан на обучении модели, объединяющую CNN с RNN для фиксации временных закономерностей в процессе моргания глаз [37]. CNN используются в качестве двоич-

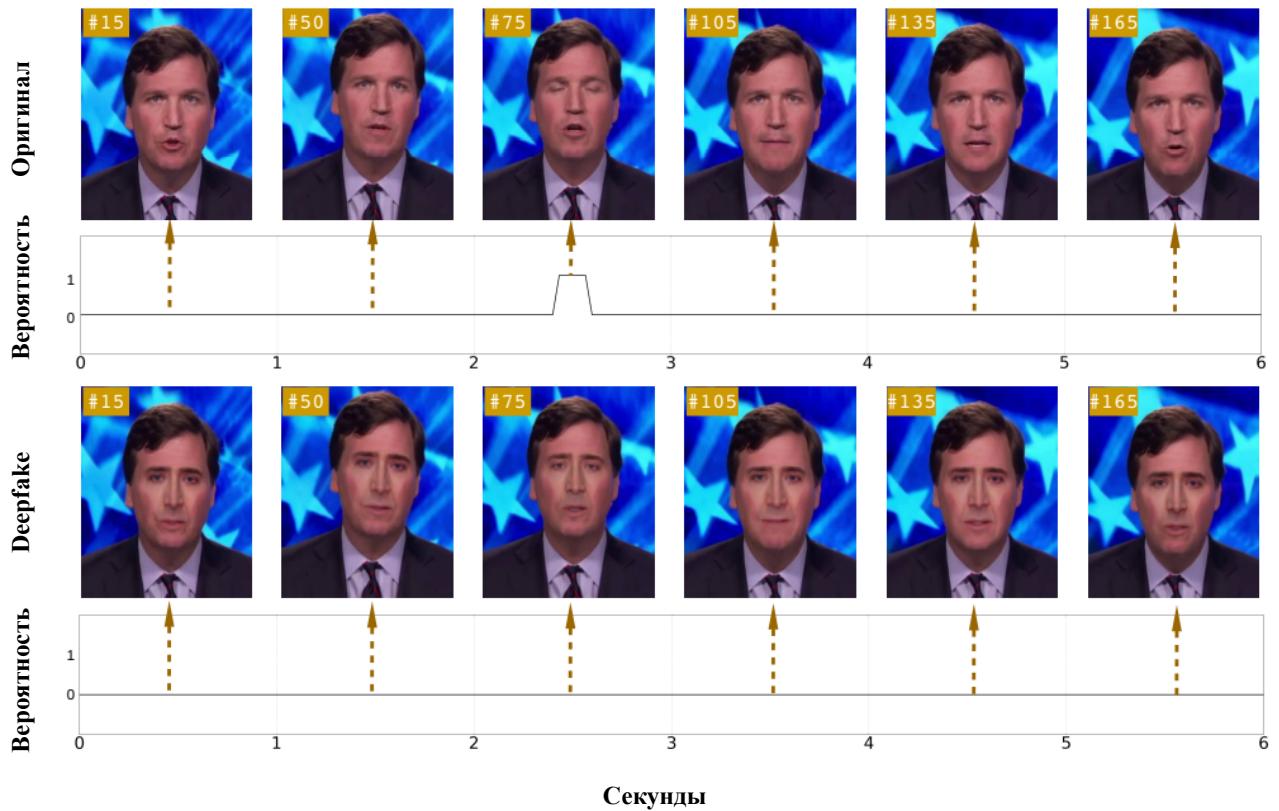


Рисунок 1.17 – Моргание в реальном и поддельном видео.

ного классификатора для различия состояний открытых и закрытых глаз каждого кадра видео. Поскольку моргание человеческого глаза имеет сильную временную зависимость с предыдущими состояниями, следует использовать LSTM [38].

На рисунке 1.18 представлена структура метода обнаружения поддельных видео с помощью моргания глаз. Сначала из каждого кадра в видео извлекаются области, соответствующие глазу. После предварительной обработки, моргание глаз определяется путем количественной оценки степени открытости глаза в каждом кадре с использованием LSTM.

Точность обнаружения сфальсифицированных видео с помощью данного метода уменьшается при работе с видео низкого качества и наличии визуальных дефектов. Стоит отметить, что моргание глаз является недостаточно эффективным способом для обнаружения, поскольку опытные фальсификаторы научились создавать реалистичные эффекты моргания с помощью последую-

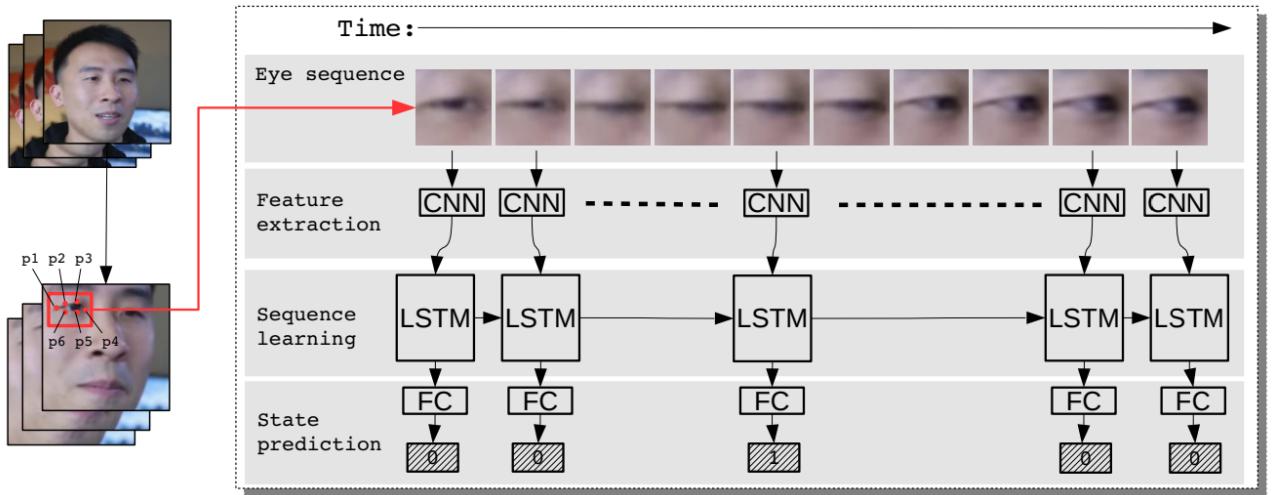


Рисунок 1.18 – Структура метода обнаружения основе оценки частоты моргания глаз.

щей обработки и более продвинутых моделей [37].

1.9.2 Метод обнаружения на основе оценки частоты сердечных сокращений

Удаленная фотоплетизмография (rPPG) – это метод бесконтактного измерения сердечной активности, основанный на анализе подробных деталей лица. Необработанные PPG-сигналы кодируются вместе с их спектральной плотностью в пространственно-временной блок, называемый ячейкой карты PPG. С помощью исследования биологических сигналов в картах PPG и обучении сверточной нейронной сети (CNN) можно обнаружить поддлинность видео [39]. Стоит отметить, что точность обнаружения сфальсифицированных видео с помощью данного метода снижается при ухудшении качества видео и наличии визуальных дефектов [40].

На рисунке 1.19 представлена архитектура метода обнаружения на основе оценки частоты сердечных сокращений.

1.9.3 Метод обнаружения на основе оценки лицевых ориентиров

В Deepfake-видео не гарантируется, что исходные и синтезированные лица будут иметь совпадающие ориентиры, например, глаза и уголки рта. Ошибки в их местоположении не всегда заметны, но могут быть выявлены по поло-

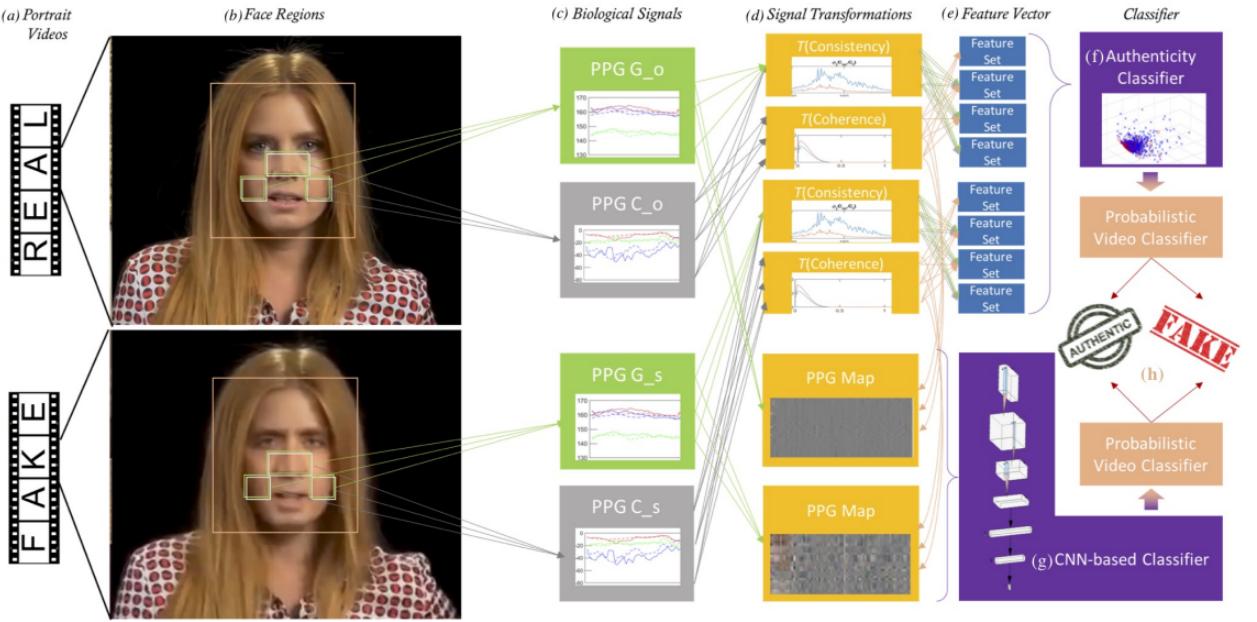


Рисунок 1.19 – Метод обнаружения на основе оценки частоты сердечных сокращений.

жению позы головы, лицевые ориентиры которых оценены в 2D-пространстве (рисунок 1.20). Используется разница в предполагаемой позе головы для обучения простого классификатора на основе SVM для определения подлинности видео [41].

Положению головы в 3D-пространстве соответствует преобразование мировых координат в соответствующие координаты камеры. Пусть $[U, V, W]^T$ – мировые координаты одного лицевого ориентира, $[X, Y, Z]^T$ – координаты камеры, $(x, y)^T$ – координаты его изображения, R – матрица поворота, \vec{t} – вектор перемещения. Преобразование между мировой и системой координат камеры приведено в формуле (8).

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \begin{bmatrix} U \\ V \\ W \end{bmatrix} + \vec{t} \quad (8)$$

Пусть f_x и f_y – фокусные расстояния в направлениях x и y , (c_x, c_y) – оптический центр, s – неизвестный коэффициент масштабирования. Преобразо-

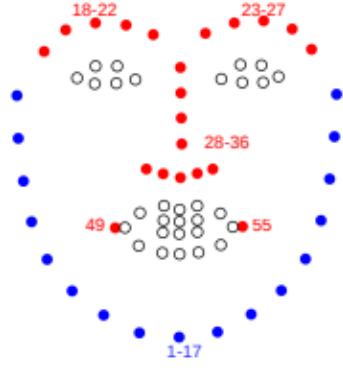


Рисунок 1.20 – Лицевые ориентиры.

вание между камерой и координатами изображения приведено в формуле (9).

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (9)$$

При оценке положения головы в 3D-пространстве необходимо решить обратную задачу, т.е. оценить s , R и \vec{t} , используя координаты 2D-изображения и 3D-координаты одного и того же набора ориентиров лица, полученных из стандартной модели. В частности, задача для набора из n лицевых ориентиров, может быть сформулирована, как задача оптимизации

$$\min_{R, \vec{t}, s} \sum_{i=1}^n \left\| s \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} - \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \left(R \begin{bmatrix} U_i \\ V_i \\ W_i \end{bmatrix} + \vec{t} \right) \right\|^2$$

, которую можно решить с помощью алгоритма Левенберга-Марквардта [42]. Оценочное значение R – положение камеры относительно мировых координат , а положение головы получается путем ее изменения в R^T (R - ортонормальная матрица).

На рисунке 1.21 представлен метод обнаружения на основе оценки оценки лицевых ориентиров.

Стоит отметить, что точность обнаружения с помощью данного метода

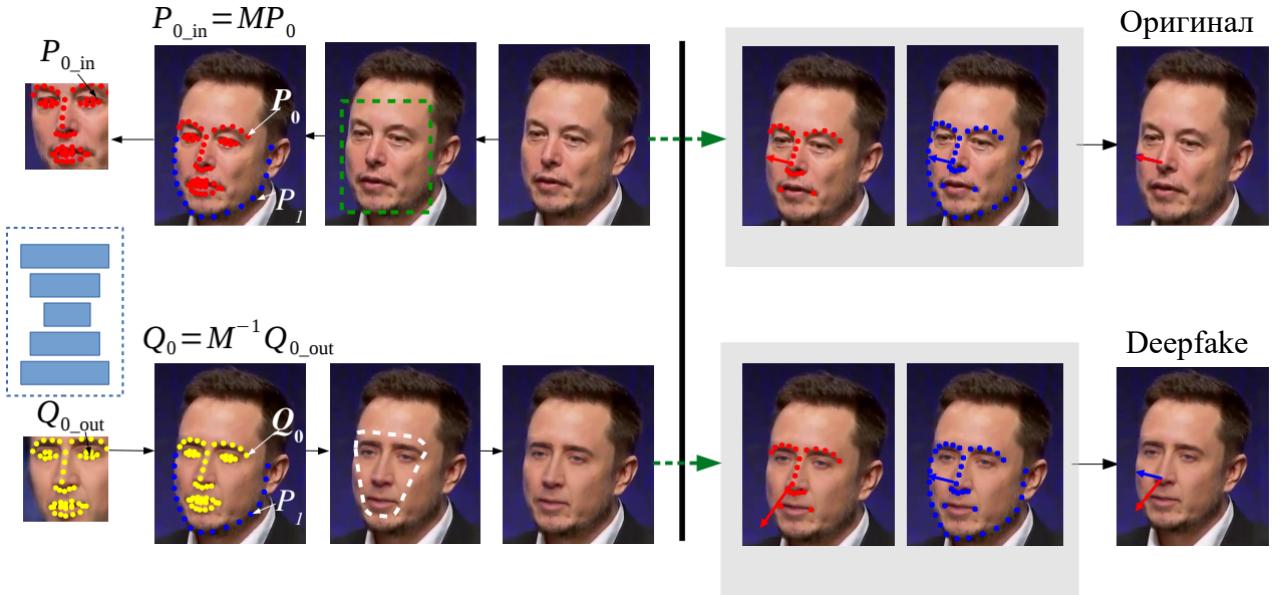


Рисунок 1.21 – Метод обнаружения на основе оценки лицевых ориентиров.

уменьшается при работе с Deepfake-видео высокого уровня качества.

1.9.4 Метод обнаружения с помощью сверточной сети MesoNet

Традиционные методы для микроскопического анализа изображений не подходят для видео из-за сжатия, которое сильно ухудшает качество данных. Аналогично, на более высоком семантическом уровне человеческий глаз с трудом различает поддельные изображения [43], особенно когда на них представлено человеческое лицо [44]. Поэтому предлагается использовать промежуточный подход с использованием сверточной нейронной сети с небольшим количеством слоев.

MesoNet – небольшая сверточная нейронная сеть, предназначенная для обнаружения Deepfake на мезоскопическом уровне анализа изображения. Наиболее эффективной является архитектура Meso4, которые основана на сетях для классификации изображений [45].

Перед началом работы видео разбивается на кадры. Затем с помощью алгоритмов распознавания лиц выделяются соответствующие области. В Meso4 существует требование к размеру входного изображения, оно должно быть 256×256 . Данный размер выбран не случайно: это четное число, которое упрощает обрезку и масштабирование области лица, и достаточно большое, чтобы предо-

ставить необходимую информацию для обнаружения поддельного контента.

На рисунке 1.22 представлена структура сети Meso4.

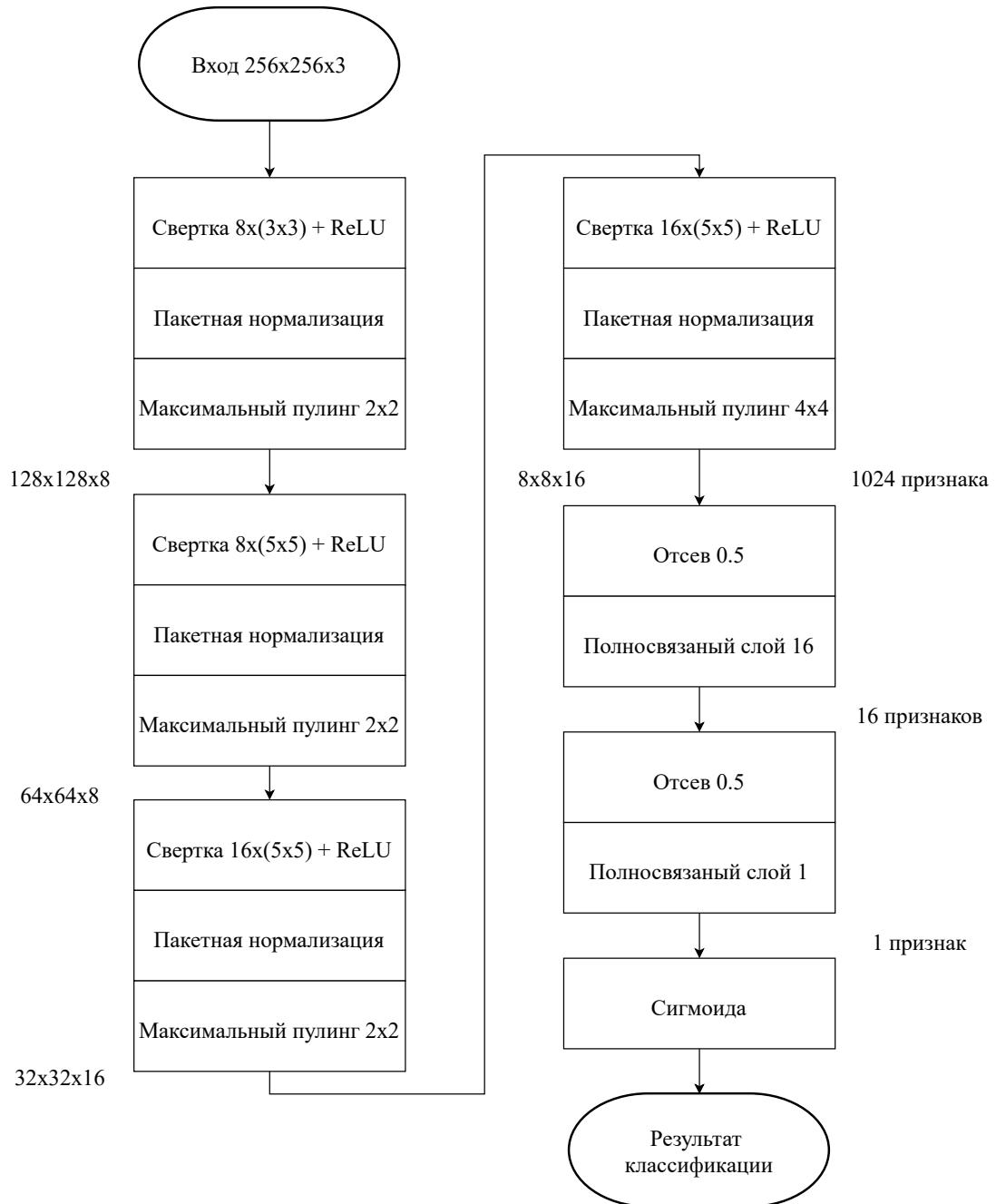


Рисунок 1.22 – Структура сети Meso4.

Meso4 начинается с четырех слоев последовательных сверток. Для улучшения обобщения сверточные слои используют функцию активации ReLU, которая вводит нелинейность и пакетную нормализацию [46] для упорядочения их вывода и предотвращения эффекта исчезновения градиента, а полносвязанные слои используют отсев [47] для упорядочения и повышения их надежно-

сти. На практике, чтобы стабилизировать процесс обучения коэффициент отсева должен быть равен 0.5.

MesoNet может быть применена для обнаружения сфальсифицированных видео высокого уровня, поскольку специализируется на выявлении манипуляции над изображением и не зависит от качества видео.

1.9.5 Метод обнаружения на основе капсулевых нейронных сетей

Капсулевые нейронные сети могут быть использованы для обнаружения сфальсифицированных видео в широком диапазоне сценариев, например, обнаружение повторных атак [49].

На рисунке 1.23 представлен метод обнаружения на основе капсулевых нейронных сетей.

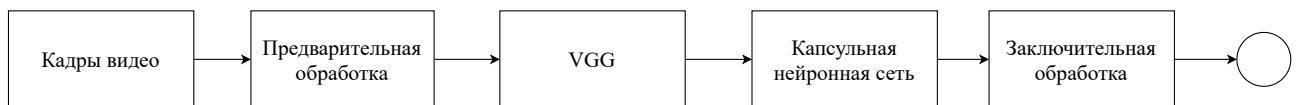


Рисунок 1.23 – Метод обнаружения на основе капсулевых нейронных сетей.

Перед началом работы видео разбивается на кадры. Затем с помощью алгоритмов распознавания лиц выделяются соответствующие области. Строгих требований к размеру входного изображения в CapsNet не существует. Обычно на практике используются следующие размеры: 100×100 , 128×128 , 256×256 . Однако стоит отметить, что предпочтительным размером изображения является 256×256 , поскольку это четное число, что упрощает обрезку и масштабирование области лица, и достаточно большое, чтобы предоставить необходимую информацию для обнаружения поддельного контента.

После этапа предварительной обработки изображения используется часть сети VGG-19 [48] для извлечения скрытых артефактов, которые являются входными данными для капсулевой сети.

CapsNet включает в себя несколько первичных капсул и две выходные капсулы (настоящая и поддельная). Количество первичных капсул должно быть не меньше трех. Стоит отметить, что достаточно большое количество первич-

ных капсул может повысить производительность сети, но за счет увеличения вычислительной мощности. Три капсулы обычно используются для простых сетей, которые требуют меньше памяти и вычислений. Например, десять капсул обычно используются для полных сетей, которые требуют больше памяти и вычислений, но обеспечивают лучшую производительность.

Каждая первичная капсула разделена на три части: 2D сверточная часть, слой статистического объединения и 1D сверточная часть. Слой статистического объединения помогает сделать сеть независимой от размера входного изображения. Это означает, что архитектура CapsNet может быть применена к задачам с различными размерами входных данных без необходимости перепроектирования сети. Среднее значение и дисперсия каждого фильтра вычисляются в слое статистического объединения.

Среднее значение рассчитывается по формуле:

$$\mu_k = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W I_{kij} \quad (10)$$

Дисперсия рассчитывается по формуле:

$$\sigma_k = \frac{1}{H \times W - 1} \sum_{i=1}^H \sum_{j=1}^W (I_{kij} - \mu_k)^2 \quad (11)$$

где k – индекс слоя, H и W – высота и ширина фильтра, I – двумерный массив фильтров.

Выходные данные статистического слоя подходят для одномерной свертки. После прохождения 1D сверточной части, выходные вектора с помощью алгоритма динамической маршрутизации отправляются в капсулы более высокого уровня. Конечный результат рассчитывается на основе активации выходных капсул.

На рисунке 1.24 представлена структура капсулной нейронной сети в методе обнаружения сфальсифицированных видео.

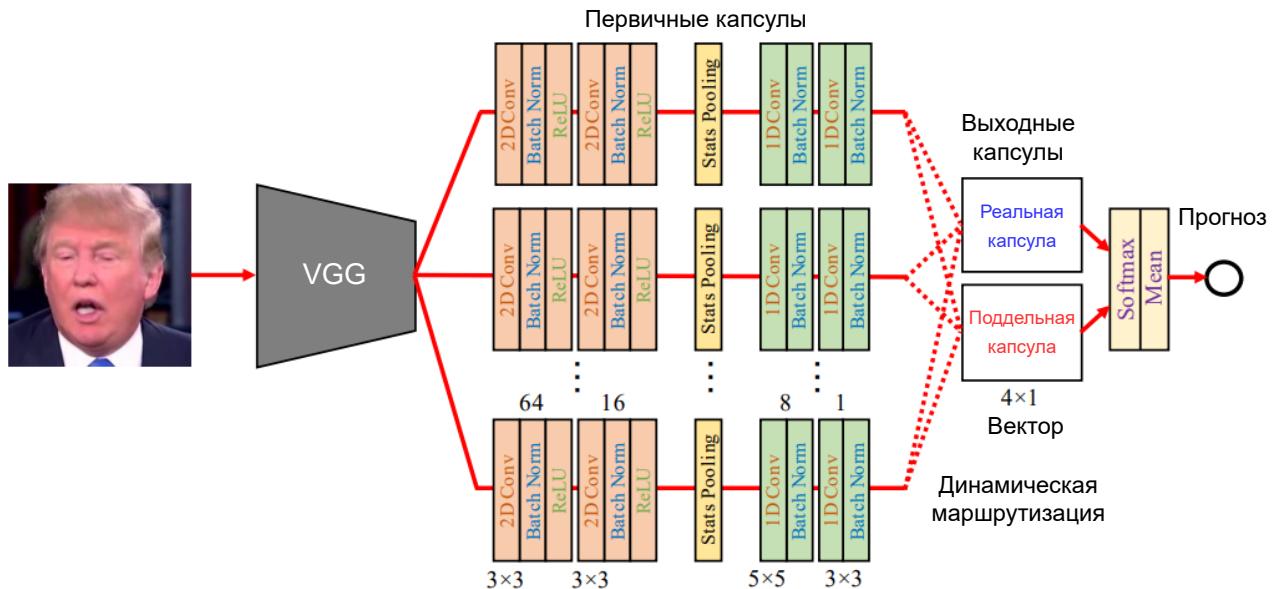


Рисунок 1.24 – Структура капсулльной нейронной сети в методе обнаружения сфальсифицированных видео.

Алгоритм динамической маршрутизации используется для вычисления соответствия между объектами, извлеченными первичными капсулами (подробнее можно ознакомиться в пункте 1.4.2). Согласование динамически вычисляется во время выполнения, и результаты направляются в соответствующую выходную капсулу (настоящую или поддельную). Пусть $u^{(i)}$ – выходной вектор каждой первичной капсулы, $V^{(1)}$ – вектор действительной капсулы, $V^{(2)}$ – вектор поддельной капсулы, $W^{(i,j)}$ – матрица, используемая для маршрутизации $u^{(i)}$ в $V^{(j)}$, r – количество итераций. Предлагается использовать модификацию алгоритма динамической маршрутизации путем введения двух регуляризаций: добавления случайного шума в матрицу маршрутизации и добавления операции отсева. Модификация используются только во время обучения, чтобы уменьшить переобучение. Кроме того, функция сжатия (squash), применяется к $u^{(i)}$ перед маршрутизацией для ее нормализации, что помогает стабилизировать обучающий процесс. Функция squash используется для масштабирования величины вектора до единичной длины.

На рисунке 1.25 представлен псевдокод модификации алгоритма динами-

ческой маршрутизации.

```

procedure ROUTING( $\mathbf{u}^{(i)}$ ,  $\mathbf{W}^{(i,j)}$ ,  $r$ )
     $\hat{\mathbf{W}}^{(i,j)} \leftarrow \mathbf{W}^{(i,j)} + \text{rand}(\text{size}(\mathbf{W}^{(i,j)}))$ 
     $\hat{\mathbf{u}}^{(i)} \leftarrow \hat{\mathbf{W}}^{(i,j)} \text{squash}(\mathbf{u}^{(i)})$ 
     $\hat{\mathbf{u}}^{(i)} \leftarrow \text{dropout}(\hat{\mathbf{u}}^{(i)})$ 
    for all input capsules  $i$  and all output capsules  $j$  do
         $b_{i,j} \leftarrow 0$ 
    for  $r$  iterations do
        for all input capsules  $i$  do  $\mathbf{c}_i \leftarrow \text{softmax}(\mathbf{b}_i)$ 
        for all output capsules  $j$  do  $\mathbf{s}_j \leftarrow \sum_i c_{i,j} \hat{\mathbf{u}}^{(i)}$ 
        for all output capsules  $j$  do  $\mathbf{v}^{(j)} \leftarrow \text{squash}(\mathbf{s}_j)$ 
        for all input capsules  $i$  and output capsules  $j$  do
             $b_{(i,j)} \leftarrow b_{i,j} + \hat{\mathbf{u}}^{(i)\top} \mathbf{v}^{(j)}$ 
    return  $\mathbf{v}^{(j)}$ 

```

Рисунок 1.25 – Модификация алгоритма динамической маршрутизации.

На практике, чтобы стабилизировать процесс обучения, случайный шум должен быть выбран из нормального распределения ($N(0, 0.01)$) и коэффициент отсева не должен превышать 5%. Две регуляризации используются вместе с инициализацией случайного веса для повышения уровня случайности, что помогает основным капсулам обучаться с различными параметрами.

На заключительном этапе полученные прогнозы для всех кадров усредняются. Этот усредненный прогноз является конечным результатом. Чтобы вычислить прогнозируемую метку \hat{y} , применяется функция softmax к каждому измерению векторов выходных капсул, чтобы добиться более сильной поляризации, а не просто использовать длину выходных капсул, как это делается в классических капсулальных нейронных сетях. Конечные результаты являются средними для всех результатов softmax:

$$\hat{y} = \frac{1}{m} \sum_i \text{softmax} \left(\begin{bmatrix} V^{(1)T} \\ V^{(2)T} \end{bmatrix}_{:,i} \right) \quad (12)$$

Стоит отметить, что поскольку нет необходимости в восстановлении изображения, то используется функция потери перекрестной энтропии (см. форму-

лу(13)) и оптимизатор Adam [50] для оптимизации сети.

$$L = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})), \quad (13)$$

где y – метка истинности, \hat{y} – предсказанная метка, а m – размер выходной капсулы v_j .

Капсулльная сеть может быть применена для обнаружения сфальсифицированных видео высокого уровня качества, поскольку специализируется на выявлении мелких деталей.

1.10 Сравнение методов обнаружения сфальсифицированных видео

В результате проведенного анализа рассмотренных методов обнаружения сфальсифицированных видео составлена таблица 2 по выбранным критериям из пункта 1.9.

Метод обнаружения	Зависимость видео от разного качества	Зависимость от визуальных дефектов
Реккурентная сеть, выявляющая частоту моргания	+	+
Сверточная сеть, выявляющая частоту сердечных сокращений	+	+
SVM для оценки лицевых ориентиров	+	+
Сверточная сеть MesoNet	+	+
Капсулльная сеть	-	-

Таблица 2 – Сравнение методов обнаружения сфальсифицированных видео.

Таким образом, методы обнаружения сфальсифицированных видео, основанных на выявлении артефактов и манипуляций искажающих видео, явля-

ются наиболее эффективными при работе с Deepfake-видео высокого уровня качества. Предлагается разработать и реализовать ансамбль сетей MesoNet и CapsNet на основе слабых экспертов. Так как данные модели являются разнородными, то при проектировании ансамбля необходимо следовать подходу стекинга.

1.11 Выводы

В данном разделе были проанализированы существующие программные решения, работа которых является неудовлетворительной, на основании чего было принято решение о разработке собственного приложения для обнаружения сфальсифицированных видео.

Сравнительный анализ существующих методов обнаружения сфальсифицированных видео показал, что методы обнаружения, основанные на выявлении артефактов и манипуляций искажающих видео, являются наиболее эффективными при работе с Deepfake-видео высокого уровня качества. Поэтому было предложено разработать и реализовать метод обнаружения сфальсифицированных видео на основе ансамблевой модели обучения сетей CapsNet и MesoNet, использующей подход стекинга.

2 Конструкторский раздел

2.1 Декомпозиция задачи

Разрабатываемый метод состоит из нескольких этапов, которые представлены на рисунке 2.26. Необходимо определить класс видео (реальное или поддельное).

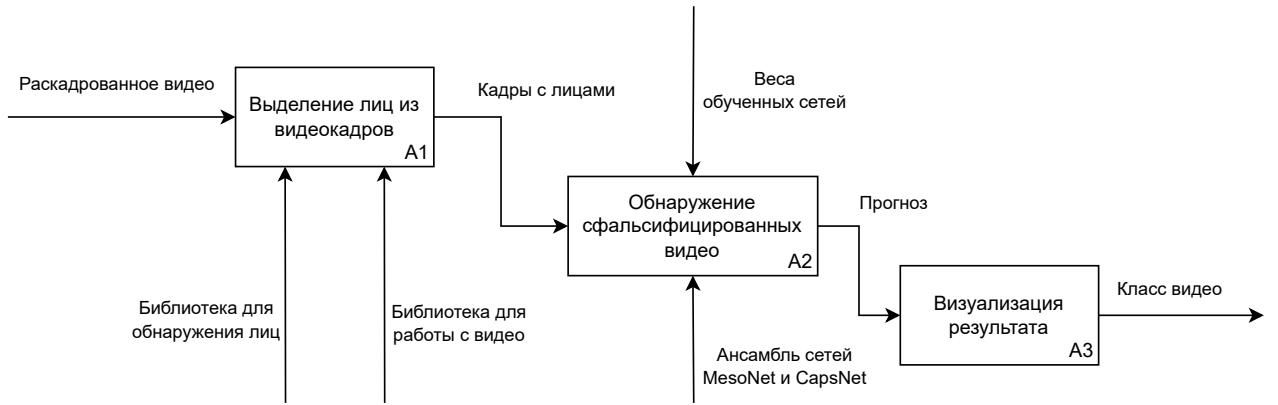


Рисунок 2.26 – Функциональная схема разрабатываемого метода.

2.2 Модифицированный алгоритм динамической маршрутизации

На вход алгоритма подается:

- $u^{(i)}$ — первичная капсула;
- $W^{(i,j)}$ — матрица, используемая для маршрутизации $u^{(i)}$ в $V^{(j)}$;
- r — количество итераций маршрутизации.

На выходе из алгоритма получается выходная капсула более высокого уровня $V^{(j)}$.

Предлагается использовать модификацию алгоритма динамической маршрутизации путем введения двух регуляризаций: добавления случайного шума в матрицу маршрутизации и добавления операции отсея. Модификация используется только во время обучения, чтобы уменьшить переобучение модели. Кроме того, применяется функция сжатия к входному вектору капсулы $u^{(i)}$ для нормализации перед маршрутизацией, что также помогает стабилизировать обучавший процесс.

Обновление весов выходной капсулы происходит путем суммирования предыдущего значения со скалярным произведением текущего выхода капсулы высокого уровня и входа в нее капсулы более низкого уровня.

На рисунке 2.27 представлена схема модифицированного алгоритма динамической маршрутизации.

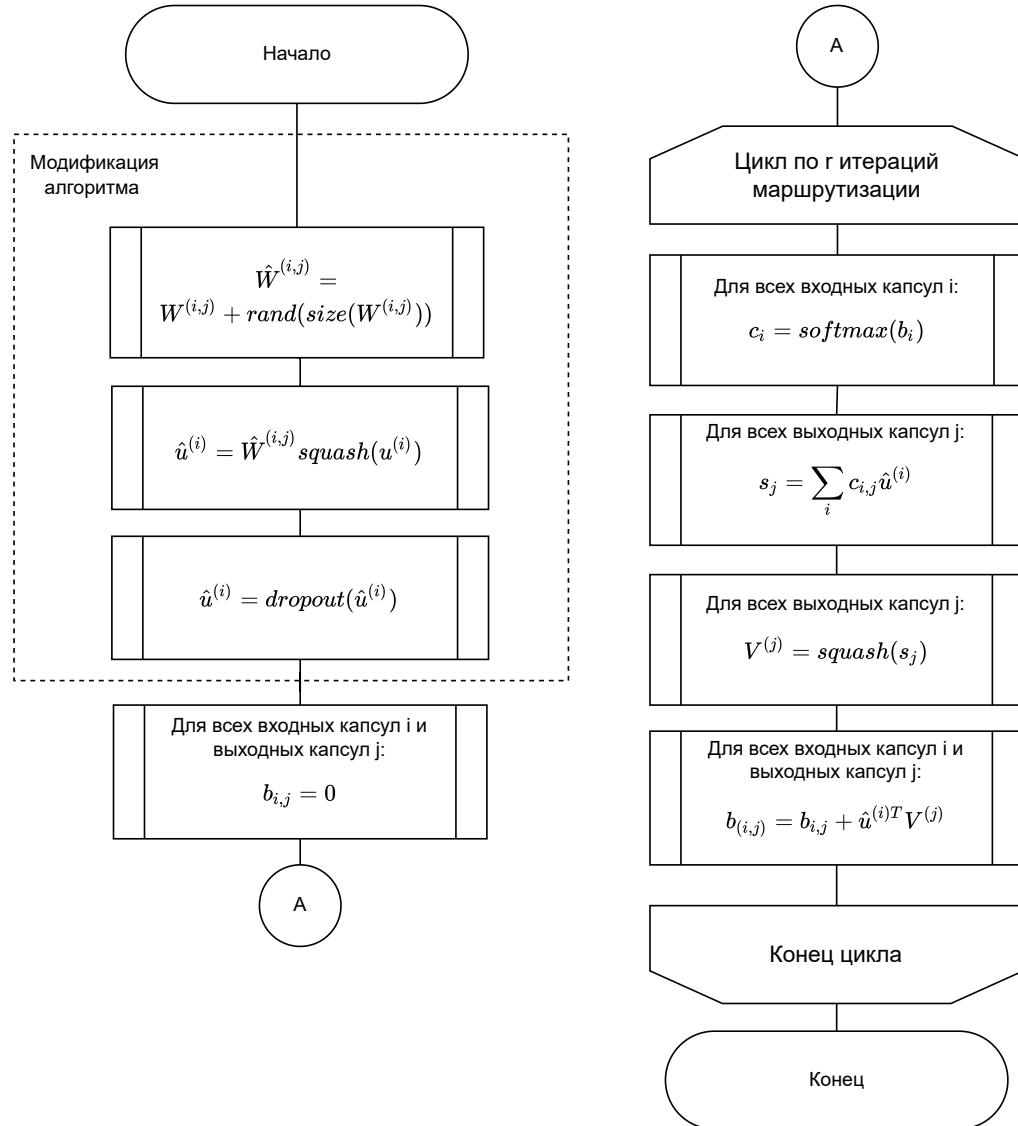


Рисунок 2.27 – Модифицированный алгоритм динамической маршрутизации.

2.3 Проектирование ансамбля нейронных сетей

Для построения ансамбля выбранных нейронных сетей (CapsNet и Mesonet, см. пункт 1.10), используется подход стекинга, идея которого состоит в том, чтобы обучить нескольких разнородных слабых экспертов и объединить их результат, обучив метамодель для вывода прогноза, основанного на множе-

ственных предсказаниях, возвращаемых этими слабыми экспертами.

В качестве метамодели используется логистическая регрессия, которая применяется в задачах классификации (прогнозирование метки класса).

Для подготовки обучающих данных для метамодели используется подход k-кратной перекрестной проверки базовых моделей, где прогнозы вне свертывания используются в качестве основы для обучения (подробнее рассмотрено в пункте 1.7.3).

На рисунке 2.28 представлена схема предложенной структуры ансамбля нейронных сетей.

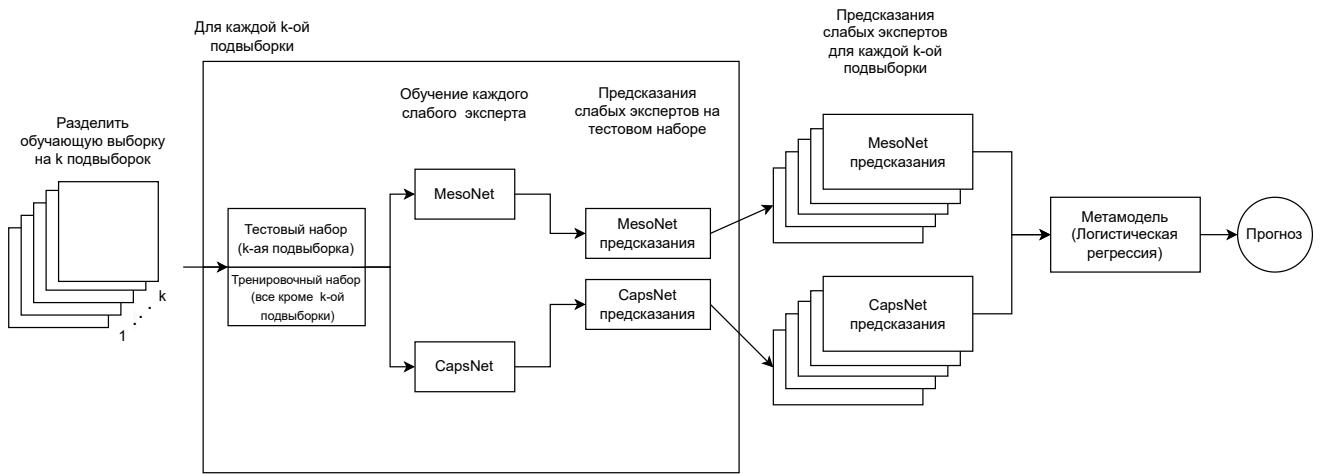


Рисунок 2.28 – Структура ансамбля нейронных сетей.

2.4 Проектирование архитектуры программного продукта

В 2022 году граждане Российской Федерации столкнулись с трудностями работы с магазинами приложений для смарт-устройств. Так, например, с 24 февраля из российского раздела App Store были удалены почти 7000 мобильных приложений [51]. Поэтому при построении архитектуры приложения был выбран подход к созданию клиент-серверного web-приложения, которое будет работать практически на любом устройстве (стационарный компьютер, ноутбук, планшет или смартфон).

В пользовательской части приложения (англ. frontend), визуализируется интерактивный и понятный интерфейс, выделяются сервисы, отвечающие за

обмен данных с серверной частью приложения.

За логику, работоспособность и правильное функционирование сайта отвечает серверная часть (англ. backend), которая скрыта от пользователя. Серверная часть включает в себя:

- модуль для обработки запросов от пользователя;
- модуль для предобработки видео;
- модуль для взаимодействия с обученной моделью обнаружения сфальсифицированных видео, называемой Deepfake-детектором.

На рисунке 2.29 представлена структура приложения.

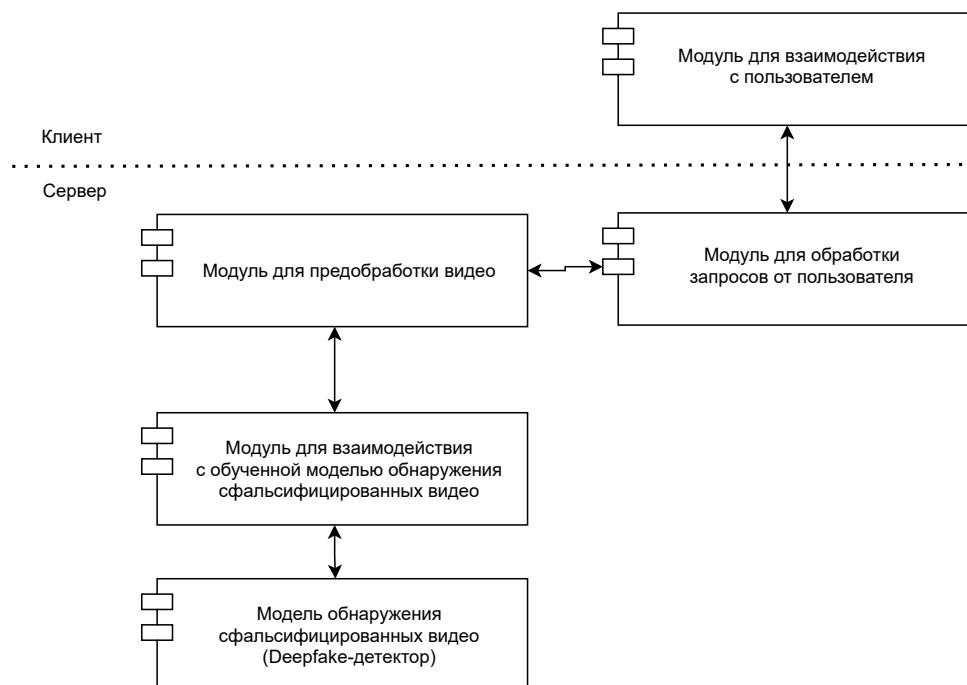


Рисунок 2.29 – Структура приложения.

2.5 Выводы

В данном разделе была проведена декомпозиция задачи, составлена ее IDEF0-диаграмма, описывающая ключевые этапы, необходимые для реализации метода обнаружения сфальсифицированных видео.

Представлена схема алгоритма динамической маршрутизации с модификацией, которая используется во время обучения капсулной нейронной сети.

Также рассмотрен принцип работы предлагаемой системы обнаружения

и приведена структура приложения, описаны основные программные компоненты и взаимодействия между ними.

3 Технологическая часть

3.1 Средства разработки для построения нейронных сетей

Для обработки и анализа данных, проектирования, разработки и обучения нейронных сетей использовался следующий стек технологий:

- Язык программирования Python — язык общего назначения, с помощью которого можно решать сложные задачи машинного обучения и быстро создавать прототипы для последующей их отладки. Язык гибкий и мультиплатформенный, имеет обширный набор библиотек для искусственного интеллекта. Python удобно использовать для обработки и подготовки обучающих данных.
- Pytorch — библиотека глубокого обучения с открытым исходным кодом, написанная на языке Python и созданная на базе Torch. Используется для решения различных задач машинного обучения: компьютерное зрение, создания и обучения нейронных сетей [52].
- Keras — библиотека глубокого обучения с открытым исходным кодом, написанная на языке Python [53].
- Pandas — библиотека для обработки и анализа данных [54].
- NumPy — библиотека с открытым исходным кодом, предназначенная для выполнения операций линейной алгебры и высокопроизводительных численных преобразований [55].
- OpenCV — библиотека с открытым исходным кодом, предназначенная для работы с видео [56].
- face-recognition — библиотека для распознавания лиц [57].

Вычисления происходили с использованием аппаратного ускорения на GPU Nvidia Tesla V100 SXM3 32GB.

3.2 Средства разработки для реализации web-приложения

Путем анализа современных трендов web-разработки было выявлено, что наиболее выигрышным является подход к созданию web-приложений, в виде

SPA (Single Page Application – одностраничное веб-приложение, которое загружается на одну HTML-страницу). Целесообразность использования SPA определяется следующими пунктами:

- Одностраничные веб-приложения работают значительно быстрее обычных сайтов. Скорость загрузки у них выше, соответственно, они удобнее пользователям.
- SPA лучше адаптировано под мультиплатформенность.

Динамическое обновление страницы, происходящее благодаря скриптам написанным на языке JavaScript, позволяет пользователю не перезагружать страницу при переходе к другому блоку приложения. В процессе работы пользователю может показаться, что он использует не веб-сайт, а десктопное приложение, так как оно мгновенно реагирует на все его действия, без задержек и «подвисаний». Добиться такого эффекта позволяет использование фреймворков для языка JavaScript, таких как: Angular, React, Vue.

Учитывая, что Angular предоставляет такую функциональность, как двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом, шаблоны, маршрутизацию [58], было принято решение использовать именно его.

Чтобы избежать ошибки, с которыми часто сталкиваются разработчики на JavaScript, используется дополнительное расширение JavaScript – TypeScript, которое позволяет работать со статической типизацией.

Для реализации серверной части был выбран Flask – это небольшой и легкий веб-фреймворк, написанный на языке Python, предлагающий полезные инструменты и функции для облегчения процесса создания веб-приложений с использованием Python [59].

3.3 Подготовка данных для обучения

На данный момент существует несколько различных общедоступных датасетов с сфальсифицированными видео для обучения нейронных сетей:

- The FaceForensics++ [60] — это набор данных, содержащий 1000 реаль-

ных видео, которые были сфальсифицированы с помощью различных алгоритмов генерации Deepfake и специальных приложений, например, таких, как Face2Face и FaceSwap. Полный размер датасета — 2ТБ.

- DFDC [16] — это набор данных для обнаружения сфальсифицированных видео, созданный для соревнования от IT-компаний AWS, Facebook, Microsoft. Датасет включает в себя более 5 тысяч поддельных видео, созданных с помощью различных алгоритмов генерации Deepfake. Создатели обеспечили разнообразие типов внешности людей: оттенок кожи, пол, возраст и другие внешние характеристики. Средняя продолжительность видеороликов составляет приблизительно 10 секунд при стандартной частоте кадров 30 кадров в секунду. Размер датасета — 471.84 ГБ.
- Celeb-DF [61] — это набор данных для обнаружения сфальсифицированных видео, который включает в себя 590 оригинальных видеороликов, собранных с YouTube, с людьми разного возраста, этнической группы и пола, а также 5639 соответствующих видеороликов DeepFake, средняя продолжительность которых составляет приблизительно 13 секунд при стандартной частоте кадров 30 кадров в секунду. Размер датасета — 9,26 ГБ.

На рисунке 3.30 представлены примеры Deepfake из доступных датасетов.



Рисунок 3.30 – Примеры Deepfake из общедоступных датасетов.

С целью рассмотрения возможности обнаружения Deepfake-видео, состав-

ленных разными алгоритмами, и в связи с ограниченностью доступной памяти предлагается составить собственный сбалансированный набор данных, содержащий в себе видео из доступных датасетов. Из каждого общедоступного датасета было взято около 1000 реальных и 1000 поддельных видеороликов.

Характеристики составленного датасета:

- состоит из 3075 реальных и 3075 сфальсифицированных видео;
- общее число кадров — 2316492;
- размер датасета — 20 ГБ.

3.4 Валидация данных

В классическом обучении датасет разделяется на два набора: тренировочный и тестовый. Тренировочная выборка (англ. *training sample*) — набор данных, по которому производится настройка внутренних параметров модели (весов и смещений). Тестовая выборка (англ. *test sample*) — набор данных, по которому оценивается качество построенной модели. Данные из тестовой выборки не используются моделью для корректировки значений внутренних параметров.

При таком подходе существует проблема переобучения: сеть начинает запоминать тренировочный датасет. Можно получить высокую точность на обучающей выборке, например 98%, и только 89% на тестовой выборке. Для избежания проблемы переобучения часто используется дополнительный набор данных для проверки, называемый проверочной (валидационной) выборкой.

Проверочная выборка (англ. *validation sample*) — выборка, по которой осуществляется выбор наилучшей модели из множества моделей, построенных по тренировочному набору данных. Валидационная выборка не используется для настройки внутренних параметров. Идея ее применения состоит в том, что после каждой эпохи проверяется состояние модели: вычисляется значение функции потерь на тренировочной и проверочной выборках. После определенной эпохи значение функции потерь на проверочной выборке начинает возрастать, а на тренировочной выборке, наоборот, продолжает уменьшаться. Это яв-

ляется сигналом переобучения нейронной сети. Проверочная выборка позволяет определить количество эпох, которые необходимо провести, чтобы сеть была более точной и не переобучалась.

Таким образом, при обучении CapsNet и Mesonet в работе используется три набора данных: для обучения, тестирования и валидации. Выборка разделена в следующем процентном соотношении: тренировочная — 60%, тестовая и валидационная по 20%.

3.5 Используемые метрики в задаче классификации

В задаче классификации для оценки качества моделей машинного обучения используются специальные показатели — метрики.

Для описания метрик используется матрица ошибок классификаций (англ. confusion matrix). Пусть есть два класса и метод, предсказывающий принадлежность каждого объекта одному из классов. В таблице 3 представлена матрица ошибок классификации.

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	True Negative (TN)	False Negative (FN)

Таблица 3 – Матрица ошибок классификации. y — истинный класс, \hat{y} — результат модели.

Простейшей метрикой является accuracy — доля правильных ответов:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (14)$$

Стоит отметить, что эта метрика может быть применима только на сбалансированном датасете.

Для оценки качества работы алгоритма на каждом из классов по отдель-

ности вводятся метрики точность (англ. precision) и полнота (англ. recall):

$$precision = \frac{TP}{TP + FP} \quad (15)$$

$$recall = \frac{TP}{TP + FN} \quad (16)$$

Precision — доля объектов, которые классификатор определил как положительные и при этом они действительно являются положительными, а recall — метрика, которая показывает, какую долю положительных объектов из всех объектов положительного класса обнаружила модель. Введение precision не позволяет определять все объекты в один класс, так как в этом случае получается рост FP. Recall показывает способность метода вообще обнаруживать данный класс, а precision — способность отличать этот класс от других. В отличие от accuracy, recall и precision не зависят, от соотношения классов и поэтому могут быть применимы на несбалансированном датасете.

Также существует несколько способов объединить recall и precision в объединенный критерий оценки. Например, F-мера (в общем случае F_β) — среднее гармоническое recall и precision:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \quad (17)$$

β в данном случае определяет вес точности в метрике.

F-мера достигает максимума при точности и полноте, равными 1, и близка к 0, если один из аргументов близок к 0.

3.6 Реализация капсулой нейронной сети

На основе архитектуры, описанной в пункте 1.9.5, с помощью библиотеки Pytorch была реализована модель сети CapsNet, которая включает в себя:

- десять первичных капсул;
- две выходные капсулы (настоящая и поддельная);
- алгоритм динамической маршрутизации для связи между капсулами.

В пунктах 4.1-4.3 был проведен подбор значений гиперпараметров модели сети CapsNet. Для обучения использованы значения гиперпараметров представленных в таблице 4.

Коэффициент скорости обучения	0.0001
Размер пакета для обучения	64
Количество эпох	10
Количество итераций алгоритма динамической маршрутизации	3

Таблица 4 – Гиперпараметры капсулльной нейронной сети.

3.7 Реализация нейронной сети MesoNet

На основе архитектуры, описанной в пункте 1.9.4, с помощью библиотеки Keras была реализована модель сети Meso4, которая включает в себя:

- четыре сверточных слоя, предназначенных для выделения признаков, с функцией активации ReLU;
- слои подвыборки для сокращения объема данных;
- слои исключения с коэффициентом отсева равным 0.5 (для стабилизации обучения);
- полносвязанные слои для распознавания паттернов.

В пунктах 4.1-4.3 был проведен подбор значений гиперпараметров модели сети Meso4. Для обучения использованы значения гиперпараметров представленных в таблице 5.

Коэффициент скорости обучения	0.001
Размер пакета для обучения	64
Количество эпох	10

Таблица 5 – Гиперпараметры сети Meso4.

3.8 Демонстрация работы программного обеспечения

Перед началом работы пользователю необходимо загрузить видео, затем он может проверить его на Deepfake, либо загрузить другое видео. На рисунках 3.31-3.32 представлен графический интерфейс пользователя.

Обнаружение Дипфейков



Выберите видео или перетащите файл сюда

Рисунок 3.31 – Первоначальная страница сайта.

Обнаружение Дипфейков

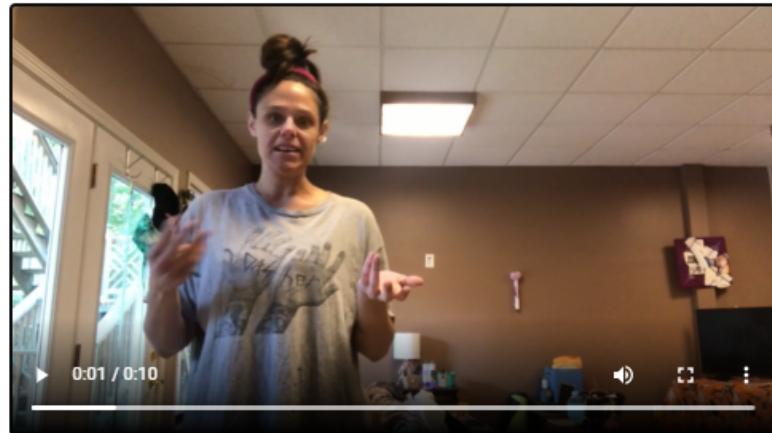
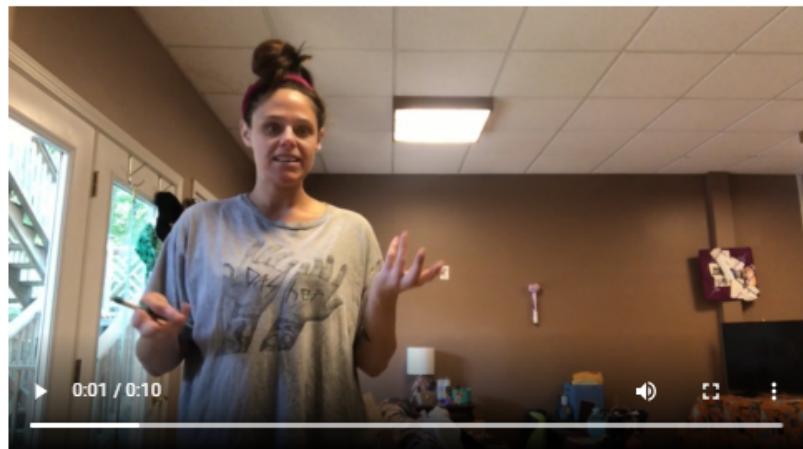


Рисунок 3.32 – Страница сайта после загрузки на него видео.

На рисунке 3.33 представлен пример проверки на Deepfake видео, которое не является поддельным.

Обнаружение Дипфейков



Результат прогноза видео:

Видео не является поддельным

Рисунок 3.33 – Результат проверки на Deepfake видео, которое не является поддельным.

На рисунке 3.34 представлен пример проверки на Deepfake видео, которое является поддельным.

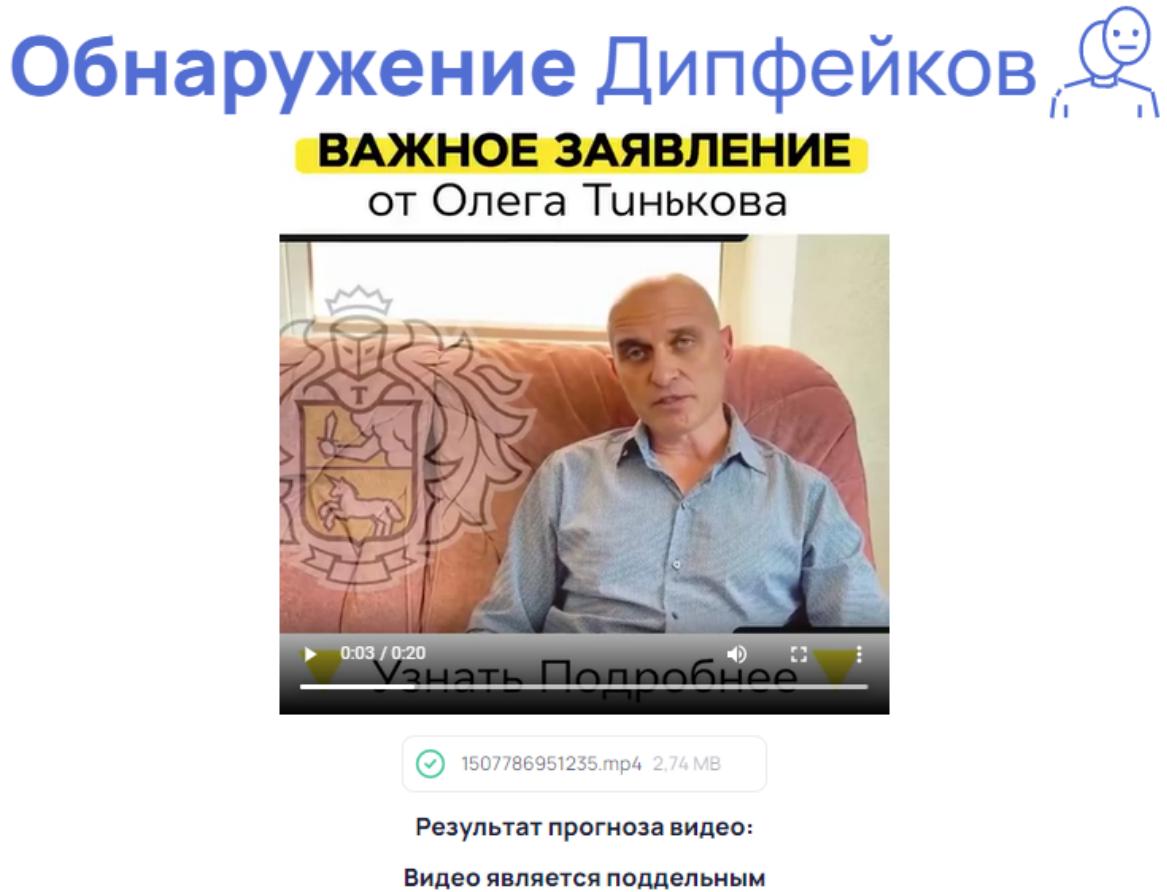


Рисунок 3.34 – Результат проверки на Deepfake поддельного видео.

На рисунке 3.35 представлен пример проверки на Deepfake видео, которое не содержит лиц.

Обнаружение Дипфейков



Initial D nascar.mp4 5,11 MB

Результат прогноза видео:

На видео не обнаружены лица

Рисунок 3.35 – Результат проверки на Deepfake видео, которое не содержит лиц.

3.9 Выводы

В данном разделе были описаны используемые средства разработки. Для реализации разрабатываемого метода выбран язык Python в качестве основного языка программирования, Typescript — как вспомогательный для реализации интерфейса.

Проведен анализ доступных датасетов сфальсифицированных видео, по результатам которого принято решение составить собственный сбалансированный датасет, на основе имеющихся данных. Описано, как контролируется процесс обучения с помощью проверочной выборки. Приведены используемые метрики для оценки точности и качества моделей.

Кроме того, приведен и описан интерфейс программного обеспечения, а также продемонстрирована его работа.

4 Исследовательский часть

4.1 Определение коэффициента скорости обучения

Цель исследования: определить значение коэффициента скорости обучения сетей CapsNet и MesoNet, построенной на архитектуре Meso4.

Скорость обучения - это гиперпараметр, с помощью которого корректируются веса в сети относительно снижения градиента функции потерь. Значение коэффициента скорости обучения выбирается в диапазоне от 0 до 1. Если скорость обучения низкая, то градиентный спуск делает небольшие шаги, несмотря на то, что он может быть большим. Это замедляет процесс обучения. Кроме того, низкие значения коэффициентов скорости обучения могут не позволить добиться реального прогресса, и даже после обучения, нейронная сеть будет далека от оптимальных результатов. Если скорость обучения высокая, то градиентный спуск делает большие шаги. Это может привести к тому, что модель не достигнет локального минимума.

Для определения скорости обучения было проведено сравнение процесса обучения сетей CapsNet и MesoNet на разных коэффициентах. На рисунке 4.36 представлены полученные результаты.

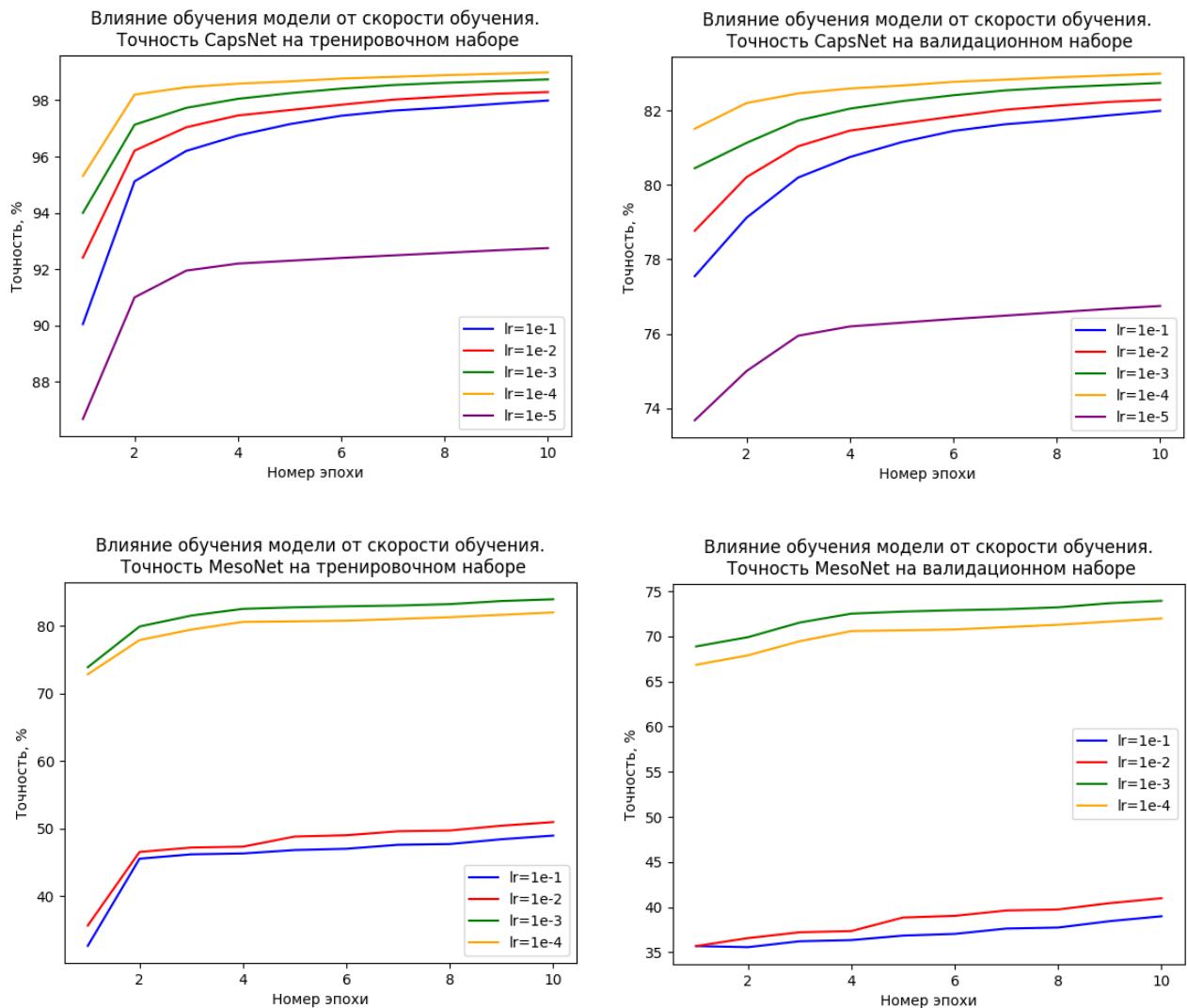


Рисунок 4.36 – Сравнение процесса обучения на разных коэффициентах скорости обучения (10 эпох).

Таким образом, по результатам проведенного исследования видно, что для капсулной нейронной сети лучше всего подходит коэффициент скорости обучения равный 0.0001. На коэффициентах 0.1 и 0.01 сеть MesoNet практически не обучается. На двух других коэффициентах сеть может обучаться, причем обучение с коэффициентом 0.001 происходит быстрее и эффективнее, чем с 0.0001. Таким образом, значение скорости обучения 0.001 для MesoNet является наиболее предпочтительным.

4.2 Определение размера пакета

Цель исследования: определить значение размера пакета для обучения сетей CapsNet и MesoNet, построенной на архитектуре Meso4.

Обучение нейронной сети происходит с большими объемами данных, которые требуют соответствующих затрат ресурсов и времени. Зачастую нет возможности загрузить сразу все данные в обработку, поэтому для преодоления этой проблемы их делят на части меньшего размера, называемые пакеты (англ. batch), которые загружаются по очереди. Затем в конце каждого шага корректируются веса нейронной сети. Размер пакета значительно влияет на обучение. Когда в сеть загружается пакет, происходит усреднение градиента. Использование слишком большого размера пакета может отрицательно повлиять на точность сети во время обучения, поскольку это уменьшает стохастичность градиентного спуска.

Другое преимущество пакетной обработки — вычисления на GPU. Стоит отметить, что число исполнительных блоков графического процессора часто является степенью двойки поэтому, использование размера пакета отличного от степени двойки приводит к плохой производительности.

Было проведено сравнение процесса обучения сетей CapsNet и MesoNet на разных значениях размера пакета: 16, 32, 64, 128. Более высокие размеры не удалось протестировать из-за ограничения видеопамяти GPU. На рисунке 4.37 представлены полученные результаты.

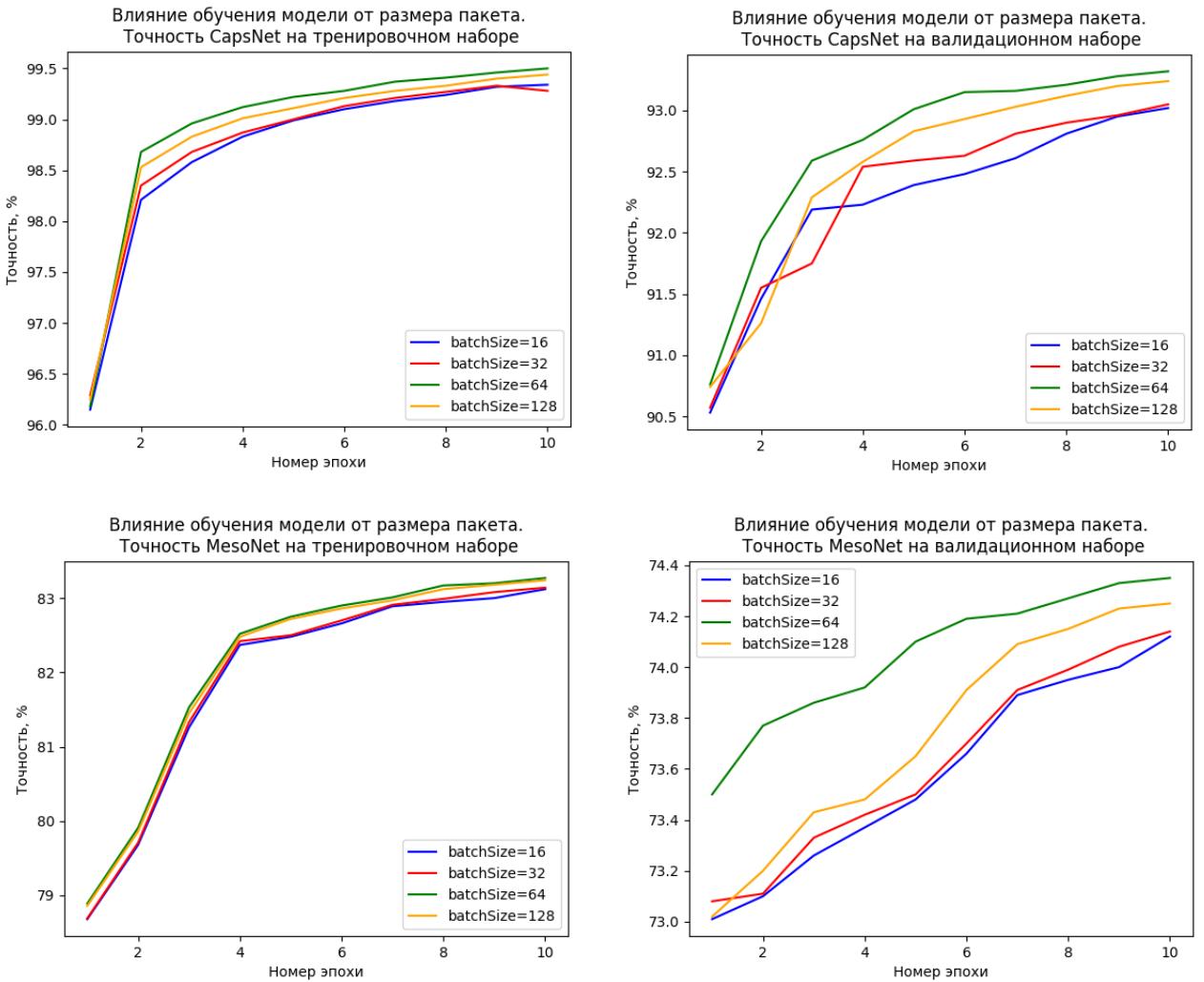


Рисунок 4.37 – Сравнение процесса обучения на разных размерах пакетов (10 эпох).

Таким образом, по результатам проведенного исследования видно, что для сетей CapsNet и MesoNet наиболее предпочтительным является размер пакета равным 64.

4.3 Сравнение реализованных методов обнаружения

Для сравнения реализованных методов обнаружения сфальсифицированных видео использовалась тестовая выборка размером 460606 кадров. На рисунке 4.38 представлены полученные результаты.

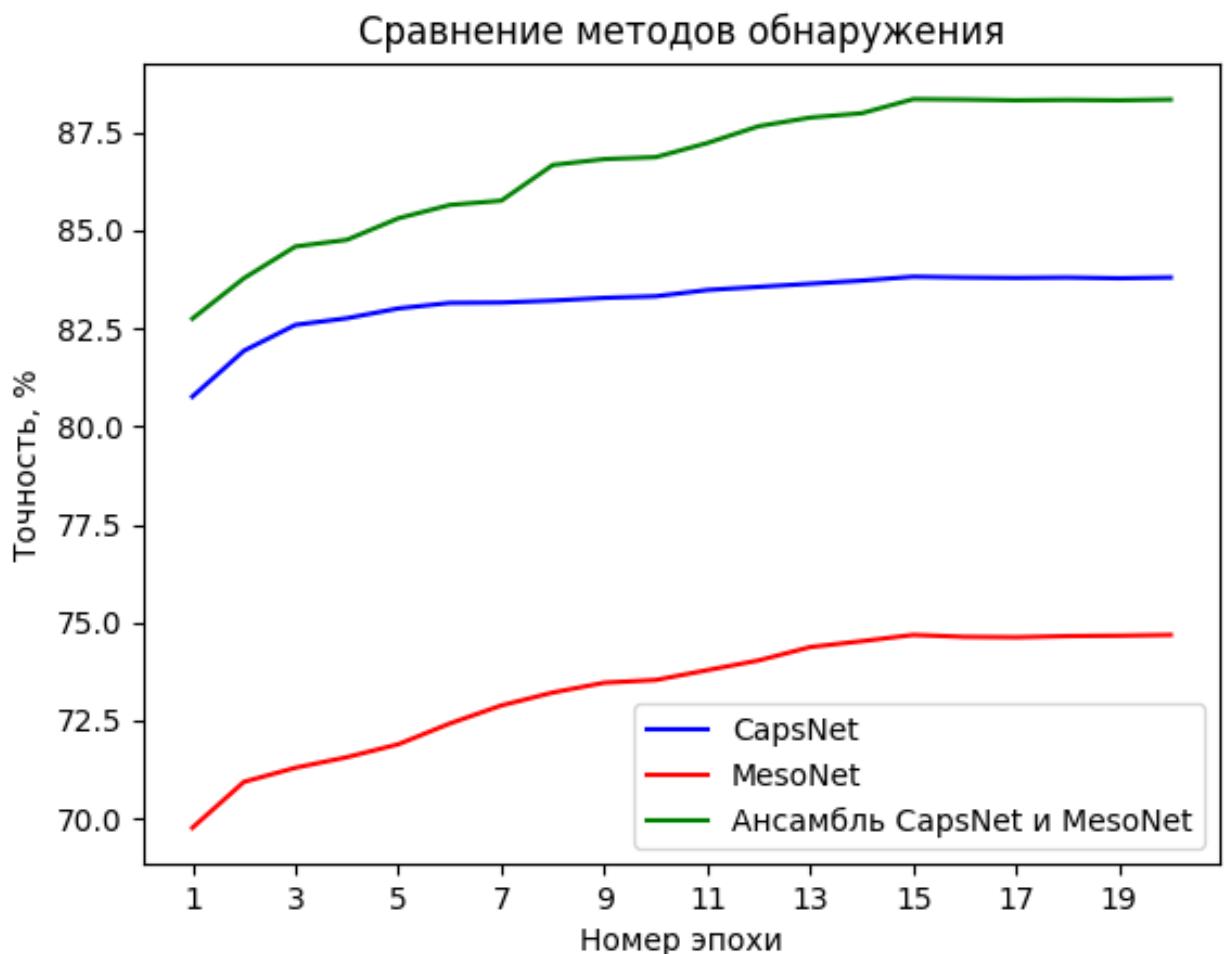


Рисунок 4.38 – Сравнение реализованных методов обнаружения.

Таким образом, для обучения моделей достаточно 15 эпох, при этом точность CapsNet составляет 83.80%, MesoNet — 74.68%, ансамбль этих сетей — 88.35%. Малое количество эпох говорит о том, что используется небольшая тренировочная выборка и сеть быстро обучается. Достигнутая точность обнаружения 88.35% подтверждает эффективность использования представленной архитектуры ансамбля и целесообразности его применения для задач обнаружения сфальсифицированных видео.

4.4 Выводы

Проведены исследования капсулной сети и Mesonet, которые показывают высокую чувствительность архитектур данных сетей к значениям используемых гиперпараметров. Определены значения коэффициента скорости обучения, размера пакета и количество эпох.

Продемонстрирована эффективность использования модели ансамбля по сравнению с обычными методами.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы был разработан метод обнаружения сфальсифицированных видео на основе ансамблевой модели обучения сетей CapsNet и MesoNet, использующей подход стекинга. Формализованы входные и выходные данные.

Были решены все поставленные задачи:

- Проанализирована предметная область и проведено сравнение существующих программных решений, работа которых является неудовлетворительной.
- Проанализированы существующие методы обнаружения поддельных видео на основе нейронных сетей. В ходе проведенного анализа, было выявлено, что методы обнаружения сфальсифицированных видео, основанных на выявлении артефактов и манипуляций искажающих видео, являются наиболее эффективными при работе с Deepfake-видео высокого уровня качества.
- В результате полученных во время анализа данных был предложен и разработан метод обнаружения сфальсифицированных видео на основе ансамблевой модели обучения сетей CapsNet и MesoNet, использующей подход стекинга. Спроектирована архитектура программного обеспечения, описаны основные модули.
- Построена капсулальная нейронная сеть, Mesonet и их ансамбль, а также выбраны гиперпараметры для обучения.
- Реализован разработанный метод в программном продукте.
- Проведено исследование работоспособности реализованного метода обнаружения сфальсифицированных видео. Достигнутая точность обнаружения 88.35% подтверждает эффективность использования представленной архитектуры ансамбля и целесообразности его применения для задач распознавания сфальсифицированных видео.

Разработанное программное обеспечение имеет перспективу дальнейшего развития. В целях экономии времени обучения моделей и доступной памяти на имеющихся вычислительных ресурсах был использован ограниченный набор данных для обучения. Поэтому для повышения точности обнаружения, а также для распознавания большего числа алгоритмов создания Deepfake-видео возможно обучение моделей на большем объеме данных или на других выборках. Систему можно сделать универсальной путем добавления метода обнаружения сфальсифицированных видео без лиц. Улучшение программного продукта также возможно с точки зрения добавления дополнительной функциональности и оформления графического интерфейса.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Искусственный интеллект [Электронный ресурс]. — Режим доступа <https://www.ibm.com/ru-ru/cloud/learn/what-is-artificial-intelligence>, свободный — (10.11.2021).
2. Машинное обучение [Электронный ресурс]. — Режим доступа: <https://www.ibm.com/ru-ru/cloud/learn/machine-learning>, свободный — (10.11.2021).
3. Компьютерное зрение [Электронный ресурс]. — Режим доступа: <https://www.ibm.com/topics/computer-vision>, свободный — (10.11.2021).
4. Нейронная сеть [Электронный ресурс]. — Режим доступа: <https://www.ibm.com/cloud/learn/neural-networks>, свободный — (10.11.2021).
5. Датасет [Электронный ресурс]. — Режим доступа: <https://www.bigdataschool.ru/blog/dataset-data-preparation.html>, свободный — (10.11.2021).
6. Thanh Thi Nguyen, Quoc Viet Hung Nguyen, Cuong M. Nguyen, Dung Nguyen, Duc Thanh Nguyen, Saeid Nahavandi. Deep Learning for Deepfakes Creation and Detection: A Survey, 2021.
7. Владислава Галкина оживили ради продолжения «Диверсанта». [Электронный ресурс]. — Режим доступа: <https://www.kp.ru/daily/27385/4580073/>, свободный — (30.04.2022).
8. Реальная подделка: в чем польза и вред дипфейков для бизнеса. [Электронный ресурс]. — Режим доступа <https://sber.pro/publication/realnaia-poddelka-v-chem-polza-i-vred-dipfeikov-dlia-biznesa>, свободный — (30.04.2022).
9. Технология deepfake как повод для начала войны [Электронный ресурс]. —

Режим доступа:<https://topwar.ru/165799-tehnologija-deep-fake-kak-povod-dlya-nachala-vojny.html>, свободный — (10.11.2021).

10. О.Л. Фиговский, О.Г. Пенский. Фейки и дипфейки в интернете: борьба по принципу айкидо [Электронный ресурс]. — Режим доступа: <http://www.proatom.ru/modules>, свободный — (10.11.2021).
11. Мошенники создали дипфейк с Олегом Тиньковым для рекламы поддельной страницы «Тинькофф Инвестиций» [Электронный ресурс]. — Режим доступа: <https://www.tinkoff.ru/invest/news/667153/>, свободный — (30.04.2022).
12. А. Панасенко. Технологии Deepfake как угроза информационной безопасности [Электронный ресурс]. — Режим доступа: <https://www.anti-malware.ru/analytics/ThreatsAnalysis/Deepfakes-as-a-information-security-threat>, свободный — (10.11.2021).
13. Около 1,5 млн фейков появилось с начала операции на Украине [Электронный ресурс]. — Режим доступа: <https://iz.ru/1299779/2022-03-03/okolo-15-mln-feikov-poiavilos-s-nachala-operatcii-na-ukraine>, свободный — (30.04.2022).
14. Facebook removes deepfake of Ukrainian President Zelenskyy [Электронный ресурс]. — Режим доступа: <https://www.theverge.com/2022/3/16/22981806/facebook-removes-deepfake-ukraine-zelenskyy-meta-instagram>, свободный — (30.04.2022).
15. Алгоритмы нейросетей против дипфейков, Seagate Russia [Электронный ресурс]. — Режим доступа: <https://vc.ru/seagaterussia/253186-algoritmy-neyrosetey-protiv-dipfeykov>, свободный — (10.11.2021).
16. Depfake Detection Challenge [Электронный ресурс]. — Режим доступа: <https://www.kaggle.com/competitions/deepfake-detection-challenge>, свободный — (10.11.2021).

17. Контракт на выполнение научно-исследовательской работы «Исследование возможных способов выявления признаков внутрикадрового монтажа видеоизображений, выполненного с помощью нейронных сетей» (в рамках ГОЗ) [Электронный ресурс]. — Режим доступа: <https://zakupki.gov.ru/epz/order/notice/ok504/view/event-journal.html?regNumber=0373100088721000002>, свободный — (10.11.2021).
18. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. — М.: ДМК Пресс, 2015. – 400 с.
19. Springenberg, Jost Tobias, Dosovitskiy Alexey, Brox, Thomas and Riedmiller Martin. Striving for Simplicity: The All Convolutional Net, 2014.
20. Geoffrey E. Hinton. Dynamic Routing Between Capsules, 2017.
21. Рекуррентные нейронные сети [Электронный ресурс]. — Режим доступа: <https://www.ibm.com/cloud/learn/recurrent-neural-networks>, свободный — (10.11.2021).
22. E. Xi, S. Bing, and Y. Jin. Capsule network performance on complex data, 2017.
23. C. Xiang, L. Zhang, Y. Tang, W. Zou, and C. Xu. Mscapsnet: A novel multi-scale capsule network, 2018.
24. M. T. Bahadori. Spectral capsule networks, 2018.
25. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Networks, 2014.
26. Vahid Kazemi, Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In CVPR, 2014.
27. Diederik Kingma, Max Welling. Auto-encoding variational bayes. In ICLR, 2014.

28. Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, Siwei Lyu. Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics, 2020.
29. Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, Victor Lempitsky, Samsung AI Center, Moscow Skolkovo Institute of Science and Technology. Few-Shot Adversarial Learning of Realistic Neural Talking Head Models, 2019.
30. A. Nagrani, J. S. Chung, and A. Zisserman, Voxceleb: a large-scale speaker identification dataset, 2017.
31. Joon Son Chung, Arsha Nagrani, Andrew Zisserman. VoxCeleb2: Deep Speaker Recognition, 2018.
32. Joseph Rocca. Ensemble methods: bagging, boosting and stacking, 2019.
33. DeepFake [Электронный ресурс]. — Режим доступа: <https://www.tadviser.ru/index.php/DeepFake>, свободный – (20.11.2021).
34. Большой толковый медицинский словарь (Oxford) Concise medical dictionary. / Под ред. Г. Л. Билича. — М. : Вече : ACT, 2001.
35. A. Bentivoglio, S. B. Bressman, E. Cassetta, D. Carretta, P. Tonali, A. Albanese. Analysis of blink rate patterns in normal subjects, 1997.
36. Schiffman, H.R., Average duration of a single eye blink. Sensation and Perception. An Integrated Approach, New York: John Wiley and Sons, Inc., 2001.
37. Yuezun Li, Ming-Ching Chang, Siwei Lyu, In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking, 2018.
38. J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venu-gopalan, K. Saenko, T. Darrell. Long-term recurrent convolutional networks for visual recognition and description, in CVPR, 2015, pp. 2625–2634.

39. Umur Aybars, Ilke Demir, Lijun Yin. How Do the Hearts of Deep Fakes Beat? Deep Fake Source Detection via Interpreting Residuals with Biological Signals, 2020.
40. Javier Hernandez-Ortega, Ruben Tolosana, Julian Fierrez, Aythami Morales. DeepFakesON-Phys: DeepFakes Detection based on Heart Rate Estimation, 2020.
41. Xin Yang, Yuezun Li, and Siwei Lyu. Exposing deep fakes using inconsistent head poses. In International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019.
42. G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools, 2000.
43. V. Schetinger, M. M. Oliveira, R. da Silva, and T. J. Carvalho. Humans are easily fooled by digital images, 2015.
44. S. Fan, R. Wang, T.-T. Ng, C. Y.-C. Tan, J. S. Herberg, and B. L. Koenig. Human perception of visual realism for photo and computer-generated face images. ACM Transactions on Applied Perception (TAP), 2014.
45. D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen. MesoNet: a Compact Facial Video Forgery Detection Network. In 2018 IEEE Workshop on Information Forensics and Security (WIFS), 2018.
46. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.
47. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 2014.
48. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

49. Huy H. Nguyen, Junichi Yamagishi, Isao Echizen. Capsule-forensics: using capsule network to detect forged images and videos, 2018.
50. D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2015.
51. Sarah Perez. Russia's App Store lost nearly 7K apps since its invasion of Ukraine, but some Big Tech apps remain [Электронный ресурс]. — Режим доступа: <https://techcrunch.com/2022/03/15/russias-app-store-lost-nearly-7k-apps-since-its-invasion-of-ukraine-but-some-big-tech-apps-remain/>, свободный — (15.05.2022).
52. Официальный сайт Pytorch, документация. [Электронный ресурс] — Режим доступа: <https://pytorch.org/>, свободный — (дата обращения: 24.05.22).
53. Официальный сайт Keras, документация. [Электронный ресурс] — Режим доступа: <https://keras.io/>, свободный — (дата обращения: 24.05.22).
54. Официальный сайт Pandas, документация. [Электронный ресурс] — Режим доступа: <https://pandas.pydata.org/>, свободный — (дата обращения: 24.05.22).
55. Официальный сайт NumPy, документация. [Электронный ресурс] — Режим доступа: <https://numpy.org/>, свободный — (дата обращения: 24.05.22).
56. Официальный сайт OpenCV, документация. [Электронный ресурс] — Режим доступа: <https://docs.opencv.org/>, свободный — (дата обращения: 24.05.22).
57. Исходный код библиотеки face-recognition, документация. [Электронный ресурс] — Режим доступа: https://github.com/ageitgey/face_recognition, свободный — (дата обращения: 24.05.22).
58. Официальный сайт Angular, документация. [Электронный ресурс] — Режим доступа: <https://angular.io/>, свободный — (дата обращения: 24.05.22).

59. Официальный сайт Flask, документация. [Электронный ресурс] – Режим доступа: <https://flask.palletsprojects.com/en/2.1.x/>, свободный — (дата обращения: 24.05.22).
60. Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. FaceForensics++: Learning to detect manipulated facial images. InICCV, 2019.
61. Yuezun Li. Celeb-DF: A New Dataset for DeepFake Forensics, 2019.