САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Петрович Юрий Михайлович

Выпускная квалификационная работа

Архитектура приложения для передачи компактного представления видеоконтента

Уровень образования: магистратура
Направление 02.04.02 «Фундаментальная информатика и информационные технологии»
Основная образовательная программа «Технологии искусственного интеллекта и Big Data»

Научный руководитель:
Кандидат физико-математических наук,
доцент кафедры механики управляемого движения,
Потоцкая Ирина Юрьевна

Рецензент:
Научный сотрудник (Researcher) LG Electronics Inc,
Кетов Дмитрий Владимирович

Санкт-Петербург
2024

Содержание

Введение 3
Постановка задачи 5
Обзор литературы 6
Обзор существующих архитектурных подходов 15
1. Предлагаемая архитектура 22
2. Реализация прототипа 24
    2.1. Описание архитектуры использованной нейросети 24
    2.2. Описание Open source решения 27
    2.3. Описание реализованной системы 29
3. Анализ работы прототипа 34
Выводы 39
Заключение 41
Список использованных источников 42

Введение

Популярность видеоконференций для виртуальных рабочих встреч, дистанционного обучения или общения резко выросла с началом пандемии коронавируса. Некоторые эксперты предполагают, что зависимость от виртуальных собраний станет частью повседневной жизни во многих сферах. Если это так, то высокий спрос на пропускную способность сетей, который повсеместная видеоконференцсвязь навязывает Интернету, никуда не денется и станет существенной проблемой.

Даже при использовании современных видеокодеков видеоконференция может потребовать пропускной способности: от 1-го до 2-х Мбит / с на участника только для того, чтобы изображения с небольшим разрешением оставались на экране. Кроме того, появляется все больше свидетельств того, что с опытом пользователи становятся более критичными к качеству изображения, стремясь увидеть мельчайшие детали мимики, жесты и позы, которые несут так много информации при личной встрече. Эта тенденция ограничивает возможность приложений использовать более высокие коэффициенты сжатия для уменьшения потребностей в пропускной способности сети. Мелкие детали, которые убирает алгоритм сжатия, содержат как раз те подсказки, которые больше всего нужны опытному переговорщику.

В связи с этим сейчас появляется все больше работ, посвященных организации видеоконференций с уменьшенной нагрузкой на сеть. Для этих целей все чаще применяются технологии машинного обучения, в том числе нейронные сети.

Подобные системы могут оказаться очень полезными в сфере обучения, так как позволяют поддерживать большее количество пользователей с включенными камерами. Также такие инструменты будут полезны в местах, где связь затруднена, а скорость интернет соединения ограничена.

В данной выпускной работе будет предложена архитектура системы для организации видеоконференций с низкой нагрузкой на сеть, а также реализация прототипа такой системы.

Постановка задачи

Цель работы – разработать архитектуру приложения для передачи компактного представления видеоконтента, а именно человеческого лица, для использования в системах видеоконференцсвязи, а также реализовать прототип решения согласно предложенной архитектуре. Далее поставленные задачи описаны более подробно.

1. Разработать архитектуру приложения.
2. Изучить применение нейросетевых подходов к решению задачи компактного представления видеоконтента.
3. Реализовать функционирующий прототип решения.
4. Проанализировать работу созданной системы.

Обзор литературы

Стоит отметить, что интерес к данной сфере растет с каждым годом, чему свидетельствует увеличение количества статей по теме использования нейросетевого подхода для построения систем видеоконференцсвязи. Это проиллюстрировано на Рисунке 1.

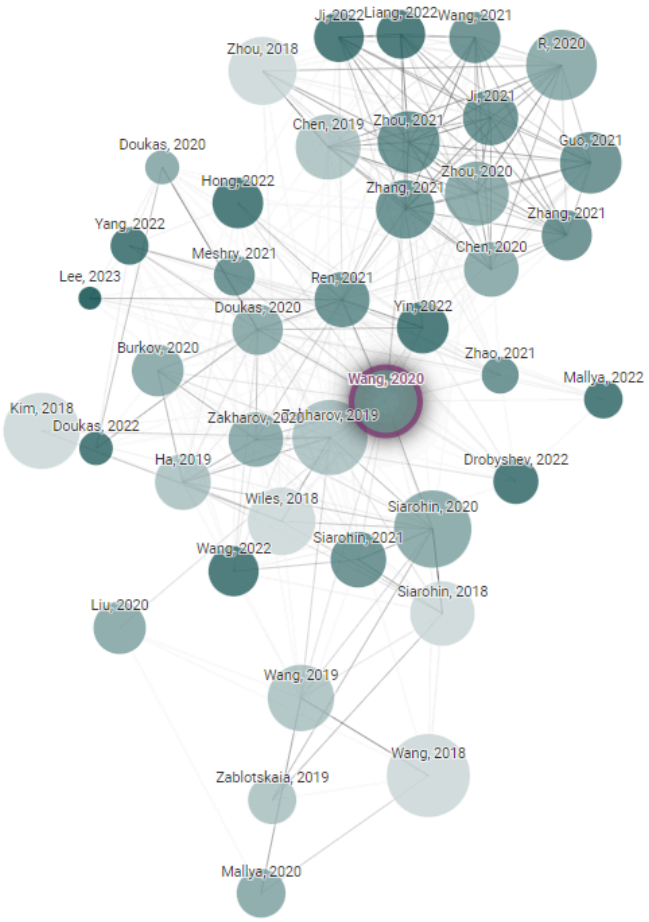

Рисунок 1 – Публикации, связанные с темой “One-Shot Free-View Neural Talking-Head Synthesis for Video Conferencing”

В приложенной литературе описываются различные нейросетевые подходы, которые можно использовать для задач сжатия кадров с лицами пользователей, а также различные архитектуры приложения для организации видеоконференцсвязи с низкой нагрузкой на сеть.

В приложении [3] предлагается архитектура приложения для организации видеосвязи при низкой пропускной способности сети. Авторы статьи предлагают кодировать кадры из видеопотока в наборы ASCII

символов, и, после дополнительного сжатия, передавать их другим пользователям. На Рисунке 2 представлена предлагаемая система.

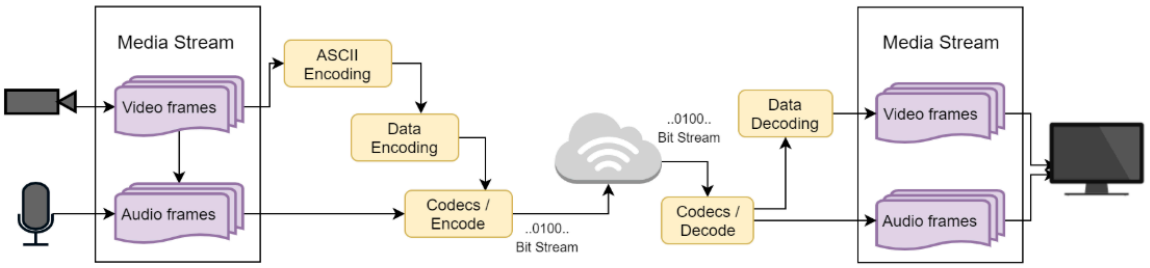

Рисунок 2 – Архитектура системы, основанной на кодировании видео в ASCII символы

Авторы статьи использовали преобразование битов изображения в ASCII символы, основывающееся на яркости пикселей. Всего предложено 15 градаций. В результате преобразования получается черно-белое изображение, состоящее из ASCII символов. Дальнейшее использование сжатия, основанного на количестве идущих подряд одинаковых символов позволило существенно сократить объем передаваемых данных. Результаты, полученные в статье, представлены на рисунке 3.

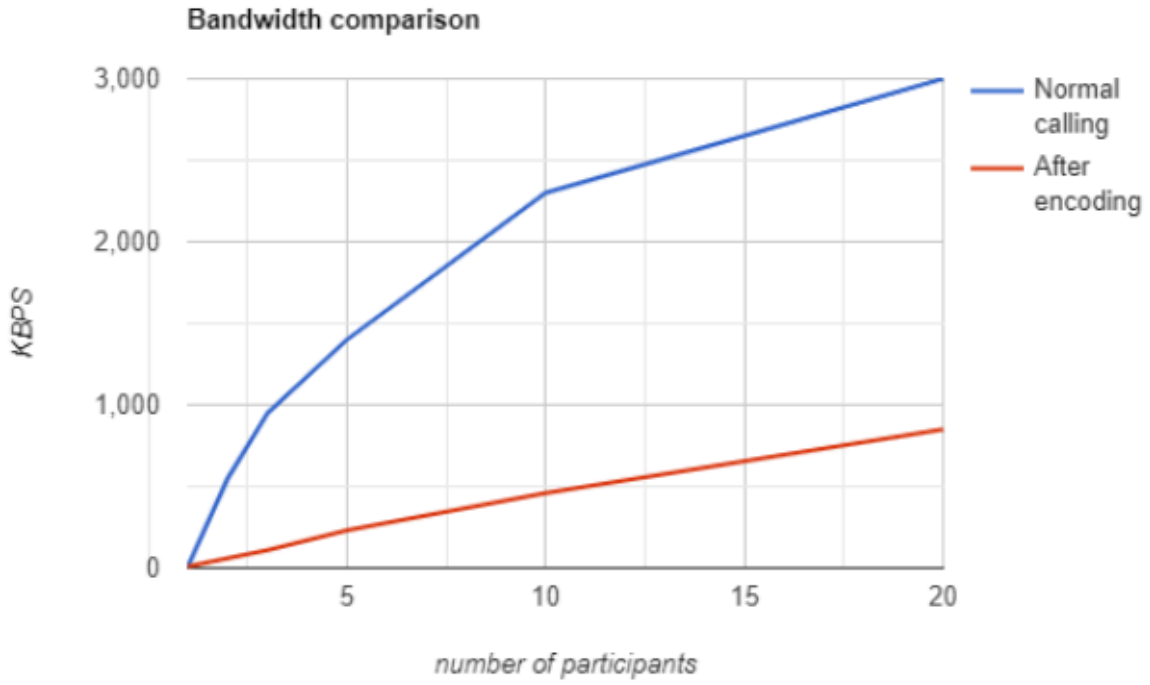

Рисунок 3 – Сравнение нагрузки. Красная линия показывает требуемую пропускную способность после кодирования [3]

Данная статья является отличным примером того, как, жертвуя качеством передаваемого видеопотока, можно существенно снизить нагрузку на сеть.

В приложении [4] предлагается подход, использующий нейросеть для сжатия видео, содержащего человеческое лицо, а также для обратного преобразования. Схема работы предлагаемого кодека представлена на Рисунке 4.

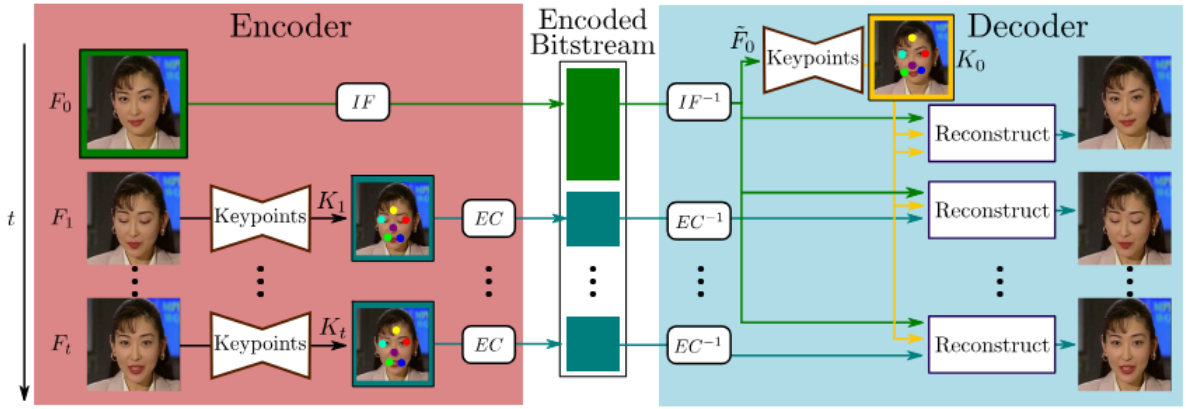

Рисунок 4 – Схема кодека, предложенного в [4]

Авторы статьи предлагают передавать исходный кадр, который в дальнейшем будет использоваться для декодирования, в декодер, и извлекать из него признаки, содержащие информацию о движении. Эти признаки называются ключевыми точками. В дальнейшем предлагается для каждого последующего кадра, называемого также “потоковым”, извлекать ключевые точки, сжимать их, передавать в декодер и использовать для восстановления каждого кадра. Сжатие информации о ключевых точках предлагается осуществлять посредством энтропийной кодировки.

Для восстановления потоковых кадров предлагается использовать нейросеть, описанную в [5]. Эта сеть состоит из двух частей. Первая сеть вычисляет оптический поток между ключевыми точками исходного и потокового кадров. Далее этот поток используется второй сетью для предсказания потокового кадра на основе исходного.

Авторы статьи также предлагают механизм адаптивного выбора исходного кадра, так как со временем корреляция между потоковыми кадрами и исходным будет ослабевать, и качество восстановления будет падать. Предлагается сохранять лучшие восстановленные кадры в буфер, и для последующего восстановления выбирать из буфера самый оптимальный “исходный” кадр. Для оптимизации предлагается восстановить кадр с использованием каждого, хранимого в буфере, и выбрать тот, для которого будет наименьший показатель PSNR. Если этот показатель больше задаваемого порога, то соответствующий кадр из буфера используется для восстановления. Если ниже, то используется самый первый кадр, а также запрашивается исходный потоковый, для добавления в буфер.

Таким образом, описываемый подход позволяет передавать по сети только сжатые ключевые точки, и, иногда, сжатые изображения. На Рисунке 5 представлены результаты экспертной оценки полученных изображений для предложенного подхода и кодека HEVC.

Bitrate (kbps)	No. of votes (Ours/HEVC)	PREFERENCE (%)
5	268 / 0	**100.00**
10	229 / 31	88.08
15	218 / 40	84.50
20	193 / 59	76.59
25	160 / 92	63.49
30	164 / 98	62.60

Рисунок 5. Результаты экспертной оценки метода, предложенного в [4], в сравнении с HEVC

Наконец, в приложении [9] приведено сравнение популярных приложений для организации видеоконференцсвязи по различным показателям качества и производительности.

Для сравнения авторы статьи выбрали Google Meet, MS Teams и Zoom. Измерения производились как на стороне отправителя, так и на стороне получателя видеопотока.

Производились следующие количественные измерения:

- объем исходящего трафика на стороне отправителя;
- объем входящего трафика на стороне получателя;
- IPAT – время между получением 2-х пакетов на стороне получателя, в качестве стандартного отклонения;
- нагрузка на центральный процессор;
- потребление оперативной памяти;
- потребление заряда батареи.

Измерения проводились в течение видеоконференций длительностью по 15 минут для каждого приложения. Всего произведено 12 измерений с различными настройками камеры и микрофона для каждого приложения. Также производилась оценка качества передаваемого видео по сравнению с исходным, для чего использовались метрики PSNR (Пиковое отношение сигнала к шуму) и SSIM (Индекс структурного сходства). Характеристики систем, использовавшихся при анализе указаны в Таблице 1.

роль участника	отправитель	получатель
CPU	Intel i5-8265U	Intel i5-8265U
RAM	16 Гб	16 Гб
OS	Windows 10	Windows 10
интернет соединение	WLAN 802.11ac через 150 Мб оптоволокно	WLAN 802.11ac через 150 Мб оптоволокно
батарея	41 Wh	41 Wh
браузер	Google Chrome	Google Chrome

Таблица 1 – Характеристики устройств, использованных в [9]

Для оценки пользовательского опыта использования приложений осуществлялся также качественный анализ работы. С этой целью был проведен опрос 15-ти участников для оценки качества работы приложения. Оценка производилась по следующим параметрам:

- качество видео;

- качество аудио;
- пропуск кадров;
- синхронизация видео и аудио;
- отставание.

Участникам опроса было предложено оценить каждый параметр по шкале от 1 до 5, где 1 – наихудший, а 5 - наилучший вариант.

Далее представлены результаты проведенного исследования.

На Рисунке 6 представлен график с объемом загружаемых данных.

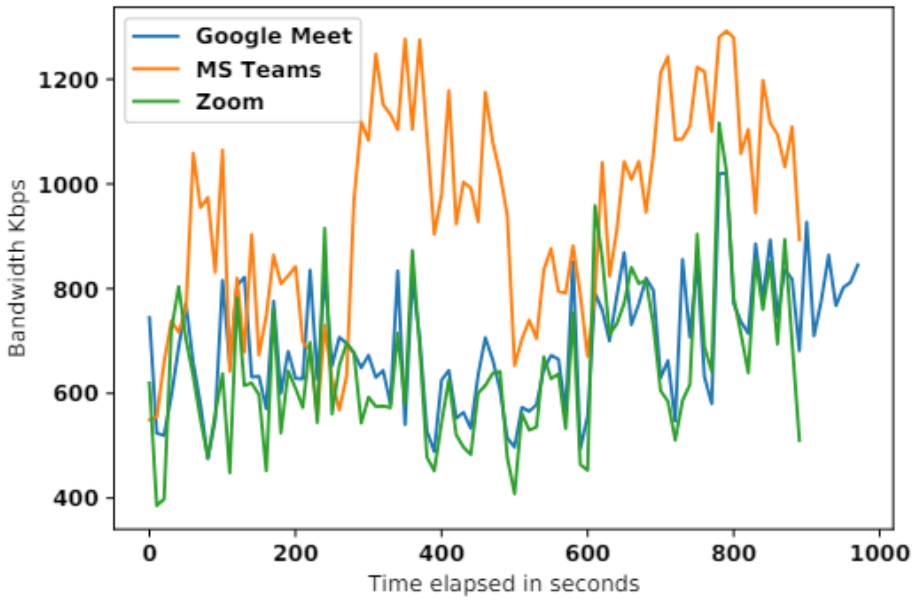

Рисунок 6 – Объем загружаемых данных. Измерения проводились каждые 10 секунд [9]

Как видно из графика, Google Meet и Zoom не только используют меньший трафик, но и более стабильны.

На рисунке 7 представлен график с объемом выгружаемых данных, совпадающий по показателям с представленным на Рисунке 6.

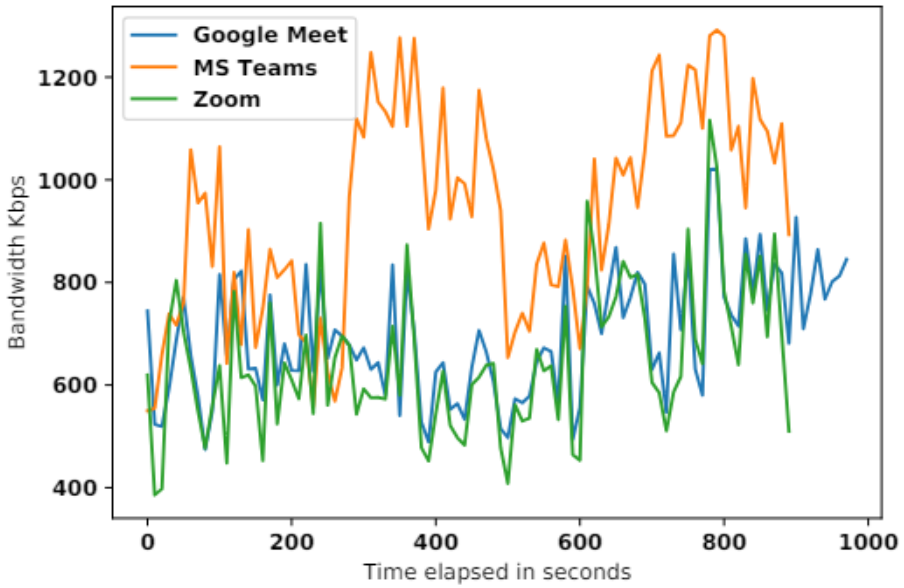

Рисунок 7 – Объем выгружаемых данных. Измерения проводились каждые 10 секунд [9]

Далее, на Рисунке 8, представлена таблица с результатами экспериментов при различных параметрах с показателями нагрузки на сеть, а также качества передаваемого видео.

Measurement Type	*App*	*Download Payload (MB)*	*Upload Payload (MB)*	*σ(IPAT) (ms)*	*PSNR (YUV)*	*SSIM (YUV)*
Mic OFF Cam OFF	Google Meet	54	58	16.64	33.98	0.97
	MS Teams	69	87	33.48	37.58	0.98
	Zoom	52	55	13.74	49.45	0.99
Mic ON Cam OFF	Google Meet	60	63	12.63	33.75	0.96
	MS Teams	76	88	12.13	37.21	0.98
	Zoom	61	63	9.43	49.15	0.99
Mic OFF Cam ON	Google Meet	74	77	8.11	32.11	0.88
	MS Teams	83	95	8.50	37.04	0.98
	Zoom	74	76	7.21	48.24	0.99
Mic ON Cam ON	Google Meet	79	82	8.07	31.26	0.85
	MS Teams	93	100	8.36	36.92	0.98
	Zoom	77	79	7.19	47.53	0.99

Рисунок 8 – Суммаризация данных, собранных по работе приложений. MT Teams потребляет больший объем трафика, чем остальные участники. Zoom показывает лучшие результаты по метрикам оценки качества видео. Также Google Meet и Zoom имеют меньшее отклонение IPAT, что обеспечивает лучший опыт во время просмотра

На Рисунках 9-10 представлены результаты измерений нагрузки на процессор, потребления оперативной памяти и заряда батареи для отправителя и получателя соответственно.

Test Type	*Platform*	*CPU Load (%)*	*Memory Consumption (MB)*	*Battery Consumption (%)*
Mic OFF Cam OFF	Google Meet	13.35	336.61	5.00
	MS Teams	26.58	294.68	8.00
	Zoom	15.91	355.23	6.00
Mic ON Cam OFF	Google Meet	22.68	388.71	6.00
	MS Teams	29.21	312.39	9.00
	Zoom	16.05	358.82	10.00
Mic OFF Cam ON	Google Meet	25.14	555.40	7.00
	Teams	30.89	360.77	10.00
	Zoom	20.54	370.17	9.00
Mic ON Cam ON	Google Meet	25.22	575.90	8.00
	MS Teams	31.74	372.20	10.00
	Zoom	22.88	382.85	10.00

Рисунок 9 – Потребление ресурсов на стороне отправителя. MS Teams потребляет существенно больше ресурсов центрального процессора. Google Meet потребляет существенно больше оперативной памяти пока камера включена

Test Type	*Platform*	*CPU Load (%)*	*Memory Consumption (MB)*	*Battery Consumption (%)*
Mic OFF Cam OFF	Google Meet	2.31	233.13	5.00
	MS Teams	4.64	209.33	7.00
	Zoom	9.93	232.85	6.00
Mic ON Cam OFF	Google Meet	6.07	269.09	6.00
	MS Teams	6.90	222.17	7.00
	Zoom	11.52	246.97	7.00
Mic OFF Cam ON	Meet	12.99	489.22	7.00
	Teams	9.11	275.53	8.00
	Zoom	13.81	258.24	8.00
Mic ON Cam ON	Google Meet	13.51	514.17	7.00
	MS Teams	9.98	291.55	8.00
	Zoom	14.30	263.09	8.00

Рисунок 10 – Потребление ресурсов на стороне получателя. MS Teams в среднем потреблчет меньшее количество ресурсов центрального процессора чем конкуренты. Google Meet потребляет существенно больше оперативной памяти пока камера включена

Наконец, на Рисунке 11 представлены результаты качественного анализа работы приложений.

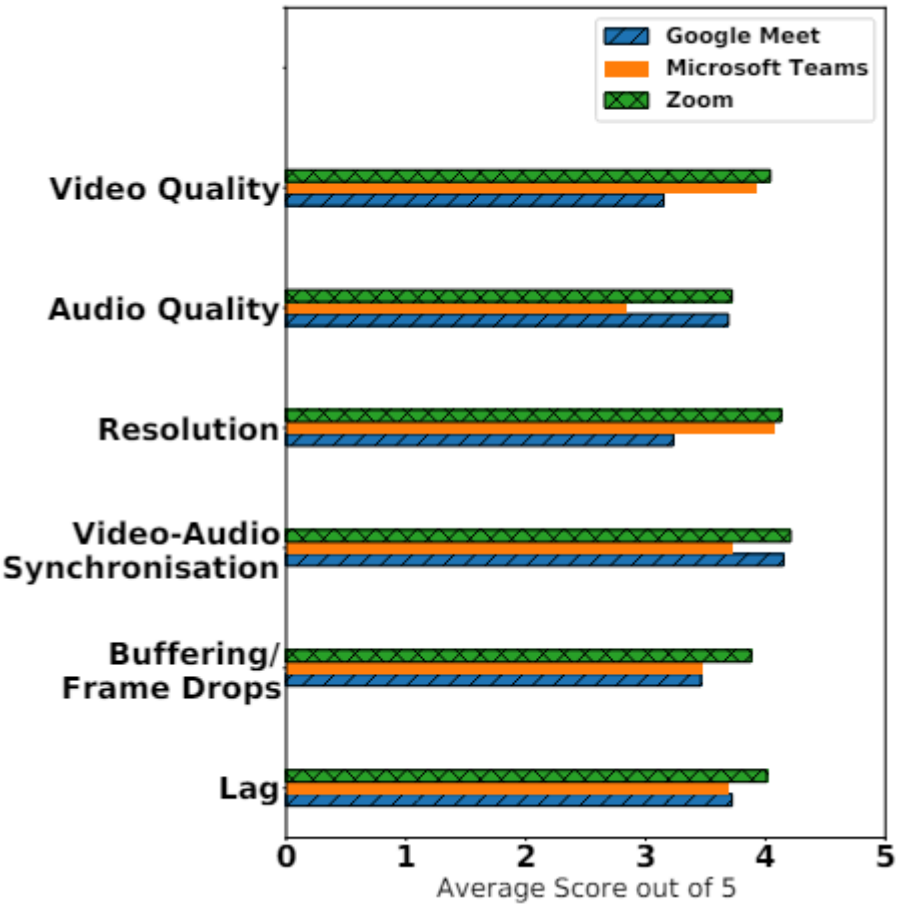

Рисунок 11 – Результаты опроса по качественным характеристикам передаваемого видео. Zoom обеспечивает наилучшее качество по всем оцениваемым параметром. MS Teams – второй выбор пользователей по качеству видео, но Google Meet и Zoom имеют лучшие показатели качества аудио и синхронизации видео с аудио [9]

В результате исследования можно сделать вывод о том, что различно организованные системы могут давать лучшие показатели качества или производительности, жертвуя при этом другими.

Обзор существующих архитектурных подходов

В данном разделе представлен обзор существующих архитектурных решений в сфере видеоконференцсвязи.

Несмотря на то, что WebRTC – это протокол, созданный для установления мультимедийных соединений с низкой задержкой и высоким уровнем безопасности, одна из его главных особенностей – гибкость. Его можно использовать как в формате peer to peer соединения, так и в связи с бэкенд сервером, а также различных других комбинациях.

У всех подходов есть свои плюсы и минусы, и использование каждого из них требует понимания целей разработки конкретной системы. Далее представлены основные существующие архитектуры.

I. Peer 2 Peer (P2P)

Первый подход – установка P2P соединения между всеми участниками конференции. В данном случае все стороны обмениваются данными напрямую друг с другом, а сервер играет лишь вспомогательную роль, храня адреса пользователей и помогая установить соединения. Схематично данная архитектура представлена на Рисунке 12.

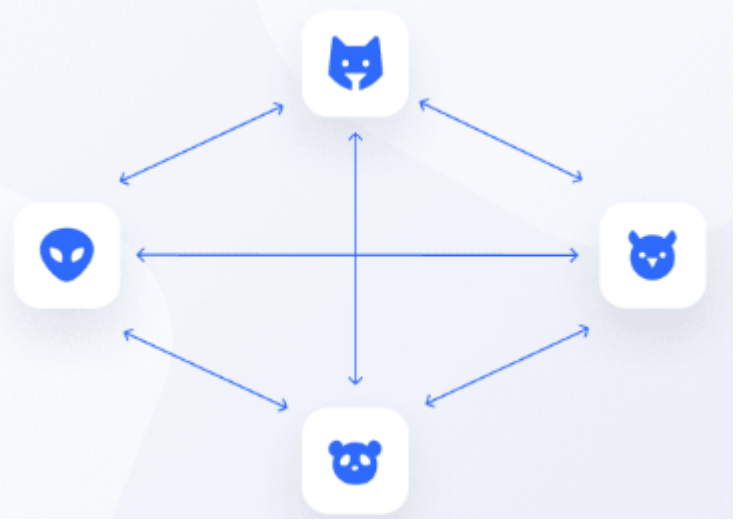

Рисунок 12 – P2P-архитектура для видеоконференций

Данный подход работает хорошо до тех пор, пока количество участников невелико (до 5 участников) и между всеми сторонами возможно установить стабильное соединение.

Ухудшение работы при большем количестве участников связано с ростом нагрузки на систему, так как для каждого соединения необходимо в реальном времени обрабатывать видео и аудиопотоки, что требует все больших ресурсов и количества соединений. Кроме того, в случае использования в корпоративных сетях, использующих NAT (Network Address Translation) или VPN, возможны проблемы при попытке установки соединения с узлами вне сети. Наконец, данный подход полностью исключает возможность ведения записи конференции на стороне сервера.

Примером приложения, использующего данный подход может послужить Google Meet.

II. Selective Forwarding Unit (SFU)

Второй подход использует медиа сервер для обмена данными между пользователями. Сервер принимает медиа потоки пользователей и рассылает их всем остальным участникам конференции. Схематично эта архитектура изображена на Рисунке 13.

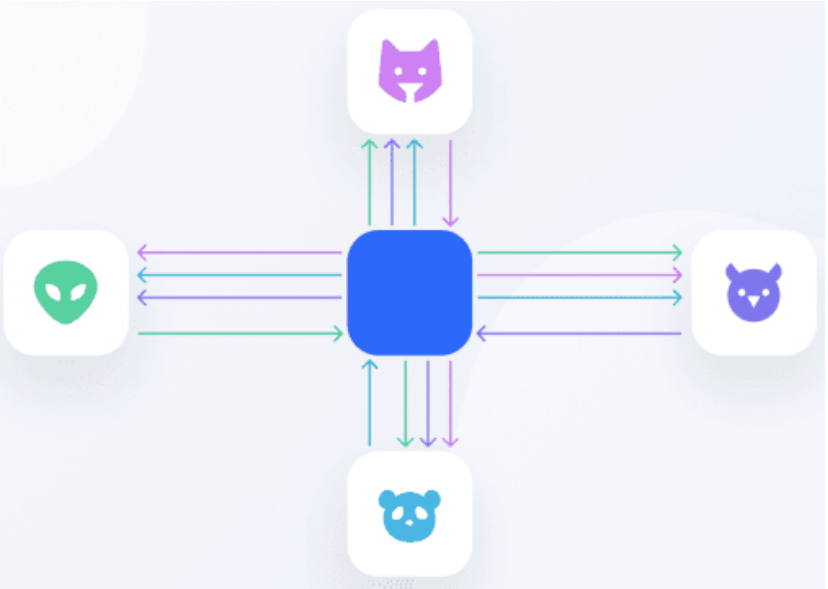

Рисунок 13 – SFU-архитектура для видеоконференций

Данный подход позволяет уменьшить количество соединений, необходимых для поддержания работы конференции на стороне клиента. Такой подход позволит каждому клиенту потреблять почти в 2 раза меньше ресурсов процессора, энергии и пропускной способности сети. Так, например, для конференции с десятью участниками, в SFU необходимо установить 10 соединений для каждого пользователя (1 исходящий медиапоток, и 9 входящих), в то время как для использования P2P потребовалось бы 18 соединений. Кроме того, прием медиа потоков других пользователей в виде отдельных потоков обеспечивает возможности адаптивного интерфейса, регуляции качества каждого отдельного потока и повышения стабильности конференции в условиях нестабильной сотовой сети.

Недостатки данного подхода дают о себе знать, когда количество пользователей приближается к 20. Эта архитектура работает на основе запросов, то есть посылает данные каждому клиенту по необходимости. Таким образом, раз WebRTC – P2P протокол, даже с использованием медиа сервера каждый поток требует отдельного соединения. Таким образом, собрание с десятью участниками потребует 100 соединений (10 для входящих потоков и 90 исходящих) на стороне сервера, что влечет нагрузку на потребление энергии, пропускной способности, и, в конечном итоге, рост стоимости эксплуатации.

Несмотря на описанные проблемы, системы, организованные подобным образом, допускают масштабирование. По мере роста нагрузки на медиа сервер часть потоков можно перенаправить на другие сервера, таким образом решая проблему чрезмерного числа участников конференции.

Примерами приложений, использующих данный подход, могут послужить: Skype, а также почти любой другой мобильный мессенджер с возможностью организации видеоконференций и записи видеозвонков.

III. MCU (Multipoint conferencing / Multipoint control unit)

Наконец, третий подход, так же, как и SFU, использует медиа сервер для обмена данными между клиентами.

При использовании данной архитектуры все потоки объединяются в один, после чего этот поток отправляется каждому участнику конференции. Схематично данный подход изображен на Рисунке 14.

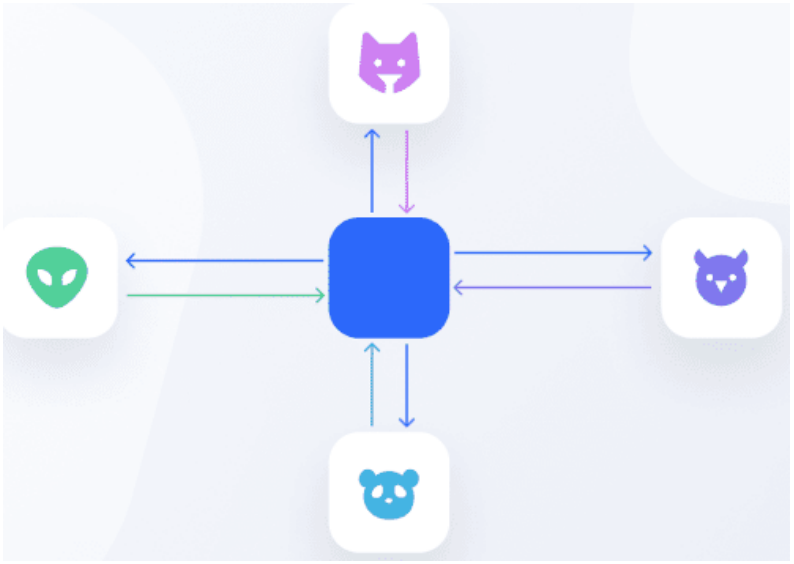

Рисунок 14 – MCU-архитектура для видеоконференций

Таким образом, MCU серверу требуется поддерживать лишь 20 соединений для конференции с десятью участниками (вместо 100 соединений в случае использования SFU). Благодаря такому подходу нагрузка на стороне клиента максимально снижается, позволяя даже маломощным устройствам поддерживать видеоконференции с большим количеством участников.

Недостатком же данного подхода является существенный рост требуемых мощностей на стороне сервера. Объединение нескольких видео и аудиопотоков в один гораздо сильнее снижает производительность, чем относительно простая пересылка потоков всем участников конференции. Если, к тому же, для каждого пользователя необходимо убрать его собственный аудиопоток из объединенного, то задача становится еще сложнее. Кроме того, еще одним недостатком, хоть и преодолимым, является то, что составленная “сетка” из видео является одинаковой для всех получателей, независимо от их разрешения экрана, соотношения сторон и

других параметров. Если требуется отправлять различные “сетки” для мобильных и стационарных устройств, то придется дважды генерировать единый поток.

При использовании WebRTC для видеоконференции, MCU, по сравнению с FCU, имеет гораздо меньший потенциал для масштабируемости: требование к объединению всех потоков в один с минимальной задержкой не позволит распределить нагрузку в рамках одной конференции. Тем не менее, возможно организовывать отдельный виртуальный медиа сервер для каждой конференции, или, для улучшения эффективности, прикреплять SFU сервер для рассылки объединенного медиа потока.

Примерами MCU-подобных систем являются Zoom и его альтернативы, предназначенные для организации видеоконференций с большим количеством участников.

Далее представлены преимущества, недостатки, а также сферы использования вышеописанных подходов.

1. P2P (~1-5 участников конференции)
    - Преимущества:
        - Минимальная задержка;
        - Легко масштабируется;
        - Потенциально самый безопасный подход.
    - Недостатки:
        - Для конференций с более чем 5-ю участниками качество может снижаться;
        - Наибольшее потребление пропускной способности сети;
        - невозможность записи на стороне сервера.
    - Сферы использования:
        - Личные / групповые звонки;
        - Онлайн поддержка / сфера продаж.

2. SFU (~5-20 участников конференции)
    - Преимущества:
        - Легко масштабируется по мере роста числа участников конференции;
        - Обеспечивает гибкость UX;
        - Может иметь избыточное количество узлов, обеспечивая отказоустойчивость.
    - Недостатки:
        - Средняя нагрузка на сеть и систему на стороне клиента;
        - Может потребовать использования MCU-подобного подхода для записи на стороне сервера.
    - Сферы использования:
        - Электронное обучение;
        - Корпоративные коммуникации.
3. MCU / MCU + FSU (20+ участников конференции)
    - Преимущества:
        - Наименьшая нагрузка на стороне клиента;
        - Подходит для обслуживания большого числа пользователей;
        - Позволяет легко вести запись конференции (как на стороне сервера, так и клиента).
    - Недостатки:
        - Наибольшие затраты на поддержку работы сервера;
        - Максимальное количество участников конференции ограничено производительностью медиа сервера;
        - Наименьшие возможности настройки выходящего видео.
    - Сферы использования:

- Потоковое вещание крупных событий;
- Социальные сети;
- Интернет-СМИ.

Таким образом, все три архитектурных подхода для организации видеоконференцсвязи с использованием WebRTC имеют свои преимущества и недостатки, что позволяет выбрать подходящий метод для решения каждой конкретной задачи.

1. Предлагаемая архитектура

В данной главе представлено описание предложенной архитектуры для организации видеоконференцсвязи.

После рассмотрения различных статей было принято решение строить архитектуру, подобную MCU, потому что она имеет следующие преимущества для решения задачи минимизации нагрузки на сеть со стороны клиента.

- Данный архитектурный подход уже предполагает наименьшую нагрузку на сеть со стороны клиента, дальнейшая оптимизация позволит снизить ее еще больше.
- Использование нейросетевых методов позволит снизить требования к производительности медиа сервера, но повысит требования к производительности каждого клиента.

Предлагается для передачи видеопотока вместо медиа сервера использовать обычный сервер для обмена данными. Этого можно достичь, если вместо видеопотока пересылать каждый отдельный кадр в сильно сжатом виде, в идеале – вектором чисел фиксированного размера, не зависящего от разрешения видео. Такое сжатие можно осуществить используя нейросетевые методы.

Для реализации предлагаемого подхода необходимо, чтобы нейросеть, используемая для сжатия видео, состояла из 2-х компонент: кодировщика и декодировщика. При этом сеть должна поддерживать сжатие отдельного кадра, а не целого видео.

Для осуществления обмена видео между клиентами, система должна состоять из 3-х основных компонент, а именно:

- клиент;
- локальный сервер;
- сервер для обмена данными.

На Рисунке 15, схематично изображена предлагаемая архитектура для обмена видео между пользователями конференции.

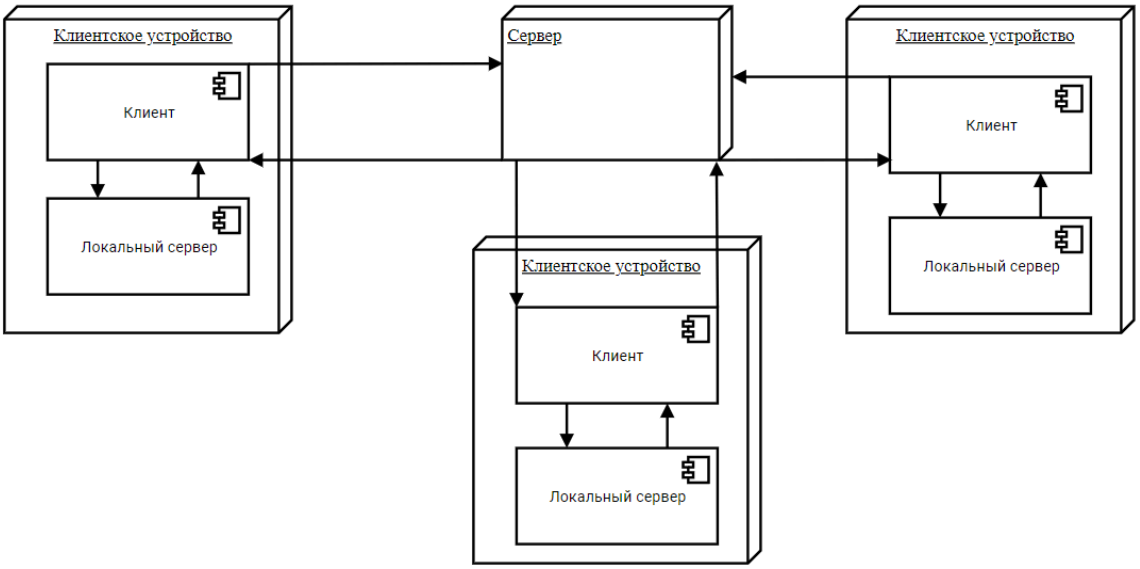

Рисунок 15 – Предлагаемая архитектура для обмена видео между клиентами

В предлагаемой архитектуре на каждом клиентском устройстве необходимо наличие локального сервера. Именно на нем будет работать модель, кодирующая и декодирующая кадры видео для обмена с другими клиентами. Видеопоток клиента передается для кодирования на локальный сервер, после чего в закодированном виде передается на сервер обмена данными, где рассылается остальным участникам конференции. Входящие закодированные сообщения передаются на локальный сервер для декодирования, после чего изображение передается клиенту для отображения.

Использование такого подхода позволит снизить нагрузку на сеть на стороне клиента соразмерно коэффициенту сжатия используемой модели. Кроме того, такой подход допускает использование различных моделей, в которых есть компоненты кодировщика и декодировщика.

Стоит отметить, что для обмена аудиопотоками между участниками конференции можно использовать любой из обозначенных в предыдущей главе методов. Для минимизации нагрузки на клиент рекомендуется

использовать MCU подход, что в сочетании с предложенной архитектурой обмена видео потоками позволит снизить нагрузку на медиа сервер, так как на его стороне не будет необходимости обрабатывать видеопотоки.

В следующем разделе представлено описание реализации предложенного метода для передачи видеоконтента, а также использованного для этого нейросетевого решения.

2. Реализация прототипа

В данном разделе приведено описание системы, реализованной по архитектурному подходу, предложенному в предыдущей главе. Стоит заметить, что целью реализации прототипа была оценка эффективности подхода для передачи именно видеоконтента. В следующем подразделе представлено описание использованной в реализации нейросети.

2.1. Описание архитектуры использованной нейросети

После изучения различных существующих нейросетевых решений для сжатия видео контента, содержащего человеческое лицо, было принято решение использовать реализацию архитектуры, описанной в статье [11] авторами из NVIDIA. Данная архитектура подходит для встраивания в систему по описанным далее причинам.

- Предполагает наличие кодировщика и декодировщика.
- Имеет возможность сжатия отдельных кадров видео в небольшой вектор значений, называемых ключевыми точками.
- Под эту архитектуру существует реализация с открытым исходным кодом, которая также будет описана далее в этом разделе.

Далее описывается общая структура метода, после чего представлено описание реализации.

Используются следующие обозначения:

- source – исходное изображение человеческого лица;

- $[d_1, d_2, ..., d_k]$ – входной набор кадров из видеопотока, содержащий человеческое лицо. Здесь d_i – i-й кадр, а k – общее количество кадров;
- $[y_1, y_2, ..., y_k]$ – набор восстановленных кадров видеопотока, где y_i – i-й кадр, сгенерированный на основе исходного изображения source, и вектора признаков, извлеченных из соответствующего кадра d_i.

В целом процесс описываемого метода состоит из 3 описанных ниже этапов.

1. Извлечение признаков (ключевых точек) исходного изображения.
2. Извлечение признаков потокового изображения.
3. Восстановление потокового изображения на основе исходного, а также наборов признаков, извлеченных на предыдущих шагах.

Схематично этот процесс можно увидеть на Рисунке 16.

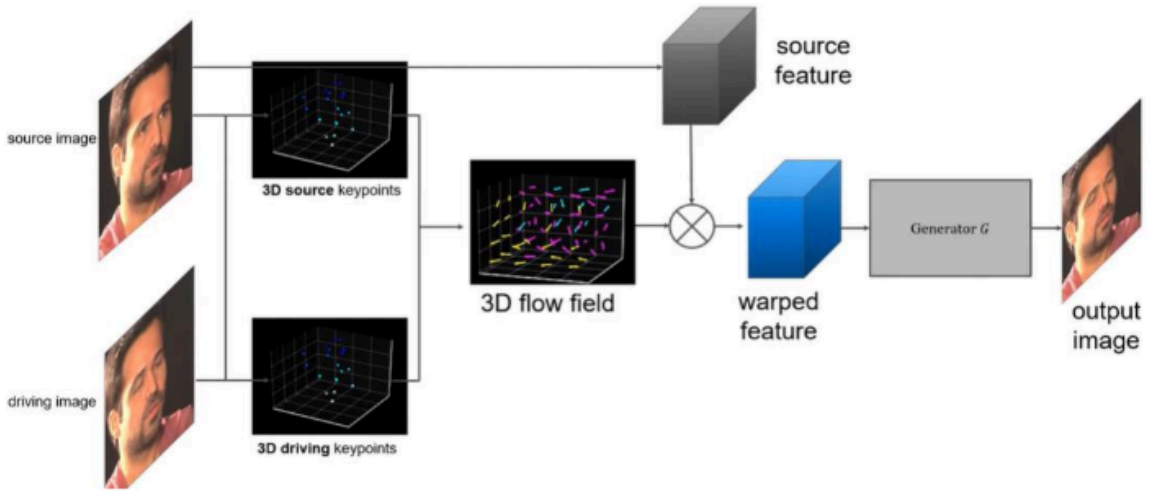

Рисунок 16 – Упрощенное представление архитектуры нейросети, предложенной в [11]. Из исходного и потокового кадров извлекаются ключевые точки, между ними ищется поток, который используется для восстановления потокового кадра по исходному в генераторе G

Исходное изображение необходимо для восстановления внешности человека. Информация об эмоциях, повороте и наклоне головы, а также о перемещении, содержится в ключевых точках изображений. Именно ключевые точки и представляют особый интерес в приложении к организации видеоконференций с низким потреблением интернет-трафика,

так как размер вектора ключевых точек намного меньше размера кадра из видеопотока.

Более подробно каждый шаг преобразований в нейросети можно увидеть на Рисунках 17-18.

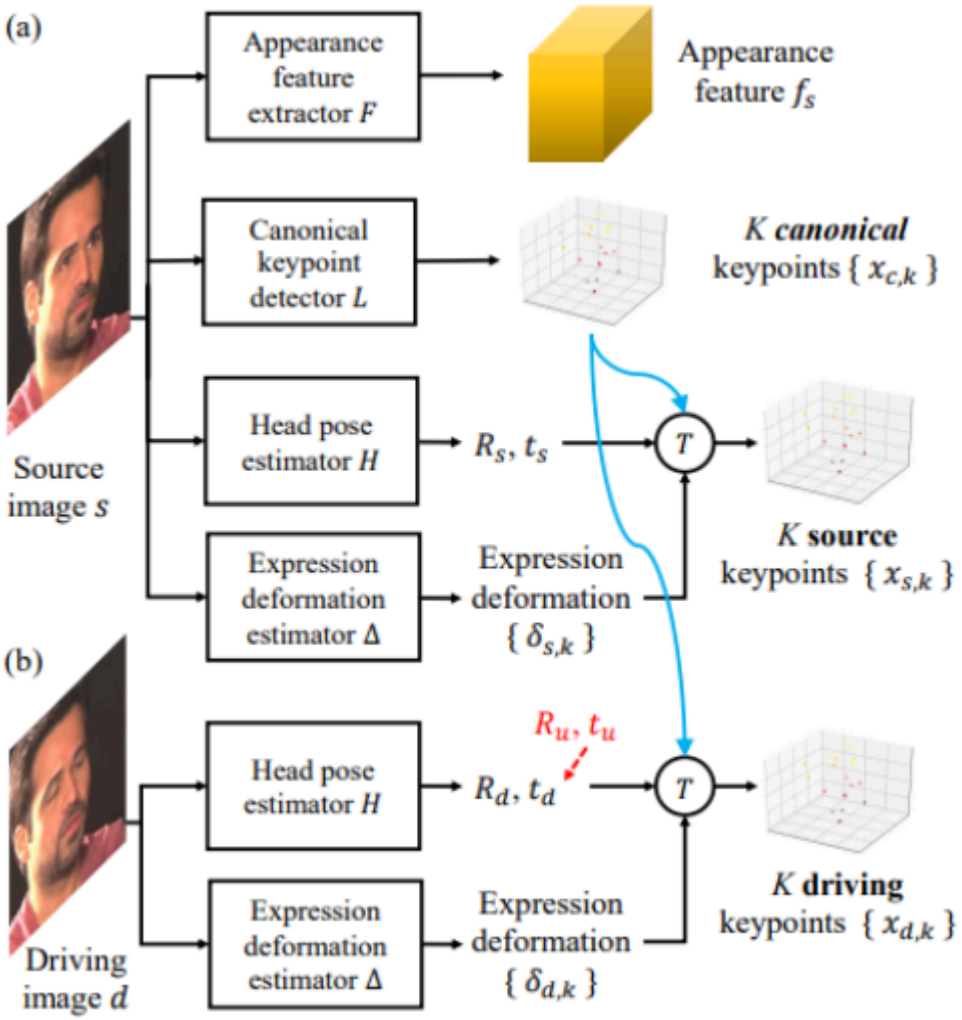

Рисунок 17 – Процесс извлечения признаков из изображений. Используется 4 нейронные сети: F (для извлечения признаков, характеризующих внешность), L (U-Net style encoder-decoder, извлекает ключевые точки исходного изображения, хранящие информацию о геометрии лица в стандартной позе при нейтральном выражении лица), H (ResNet, извлекает признаки, отвечающие за повороты, наклоны и перемещения головы), и Δ (Оценивает эмоции, извлекая разницу ключевых точек от нейтрального выражения) [11]

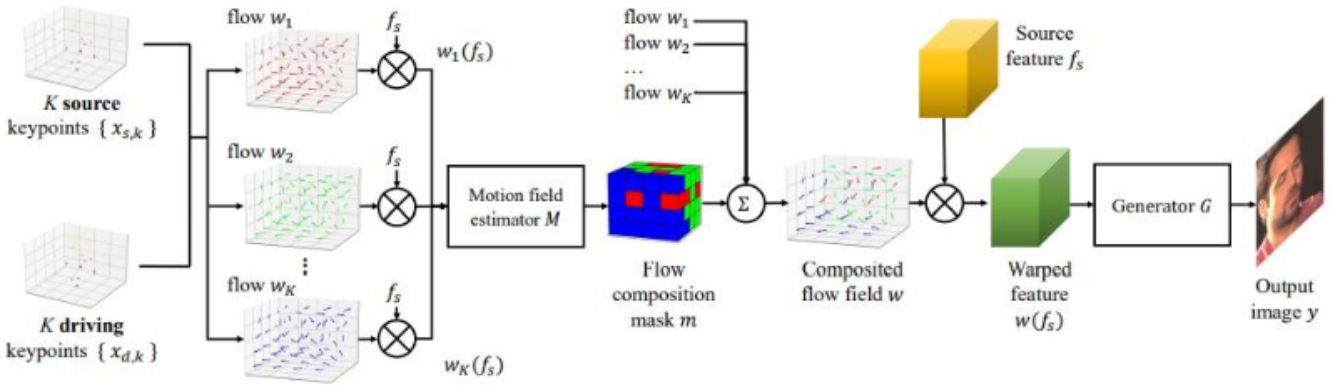

Рисунок 18 – Процесс генерации потокового кадра. В данном процессе используются две нейронные сети: M (Преобразует потоки в единую маску, используя метод, описанный в [12]) и G (Генеративная нейросеть, генерирующая выходной кадр) [11]

Стоит отметить, что архитектура данной сети позволяет заменять исходное изображение на любое другое, содержащее человеческое лицо. Эта особенность несет в себе ряд недостатков и угроз, но и ряд плюсов. Подробнее об этом описано в разделе “Выводы”.

Далее, на Рисунке 19, схематично представлен предложенный авторами [11] пример использования данной архитектуры в системах видеоконференцсвязи.

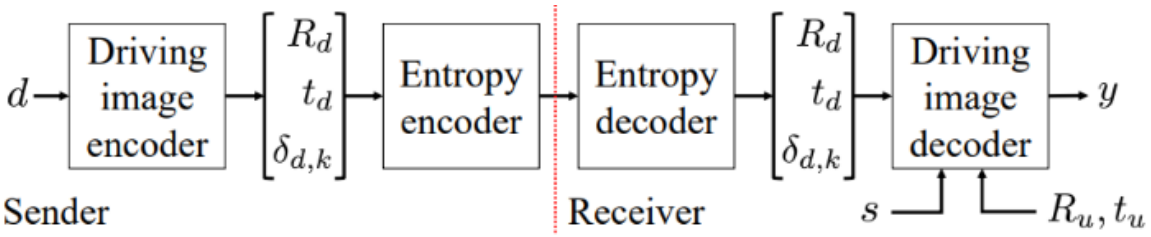

Рисунок 19 – Пример использования нейросети в системах конференцсвязи. Красной линией обозначено место, в котором данные пересылаются между различными участниками конференции

2.2. Описание Open source решения

В данном разделе представлено описание использованной в разработке прототипа модели, реализующей подход, описанный в статье [11]. Эта реализация находится в открытом доступе и написана на языке Python [13].

У данной реализации есть свои недостатки, а именно:

- отсутствие документации;

- отсутствие информации о необходимых зависимостях библиотек Python;
- модели предобучены, но по качеству существенно уступают коммерческим реализациям, вследствие большой ресурсоемкости процесса обучения.

Тем не менее, данная реализация подходит для создания прототипа, реализующего предложенную ранее архитектуру для обмена видео между пользователями видеоконференции.

Использованная в реализации модель занимает 2 Гигабайта памяти и обучена на датасете VoxCtltb.-v1 [14], содержащем более 100 тысяч лиц 1251 человека. На Рисунке 20 представлены диаграммы распределения данных внутри датасета.

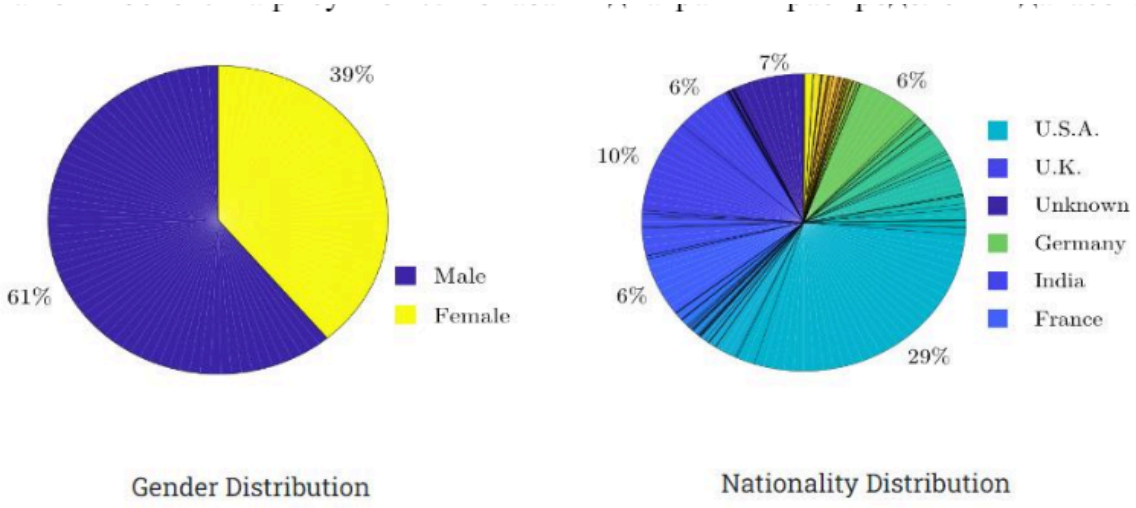

Рисунок 20 – Диаграммы распределения лиц в датасете VoxCeleb.-v1 [14]. В датасете преобладают мужчины, также видно преобладание жителей США

Взятая модель принимает на вход исходное (source) изображение, а также набор кадров или видео для преобразования. Размеры входных изображений приводятся к размеру 256x256 пикселей. В результате работы модель возвращает видео с разрешением 256x256 с восстановленным изображением лица.

Модель обучена на использование 15 ключевых точек для кодирования каждого кадра, таким образом для пересылки одного изображения необходимо отправить вектор, состоящий из 45 чисел типа float.

Кроме того, для ускорения работы модели присутствует возможность использования GPU, что, безусловно, является преимуществом данной реализации.

2.3. Описание реализованной системы

В данном разделе представлено описание реализованного прототипа системы, осуществляющей видеоконференцсвязь с использованием нейросетевого подхода.

Как уже говорилось ранее, система состоит из 3-х основных компонентов:

- клиент;
- локальный сервер;
- сервер для обмена данными.

Для обмена данными между компонентами было решено использовать протокол WebSocket. Основными преимуществами использования этого протокола являются:

- обмен данными в реальном времени;
- уменьшенная задержка (по сравнению с HTTP);
- лучшая масштабируемость (по сравнению с HTTP);
- эффективная передача данных (по сравнению с HTTP);
- простота интеграции.

Для полноты функционала также была реализована передача аудиопотоков между участниками конференции. Для простоты эта функция была реализована посредством технологии WebRTC с использованием архитектурного подхода P2P. Как было отмечено ранее, целью данной работы

является оценивание результатов использования более эффективного подхода к передаче именно видео контента.

В следующих разделах подробно описана реализация каждого из компонентов системы.

2.3.1. Описание реализации клиентской части

Начнем с описания реализации клиентской части системы. Клиент написан на языке TypeScript с использованием программной платформы NodeJs, а также следующих фреймворков:

- ReactJs – использовался для эффективной реализации компонент пользовательского интерфейса;
- Redux – использовался для эффективной реализации внутреннего хранилища клиента и реакции на различные события;
- React-Redux (необходим для совместной работы фреймворков);
- Socket-io – использовался для осуществления связи с серверами по протоколу WebSocket.

Функционал клиента можно условно разделить на 3 категории: взаимодействие с локальным сервером, взаимодействие с сервером обмена данными и внутренние функции.

Единственной функцией, не связанной напрямую с взаимодействием с серверами является осуществление снимков с камеры пользователя и сохранение их во внутреннем хранилище с настраиваемой периодичностью

Далее описан функционал, связанный с взаимодействием с серверами.

- Раз в настраиваемый промежуток времени изображение пользователя отправляется на локальный сервер для сжатия в вектор из 45 чисел посредством описанной ранее модели.
- При получении от локального сервера сообщения, содержащего ключевые точки изображения пользователя, оно пересылается на сервер обмена данными.

- При получении от сервера обмена данными сообщения, содержащего ключевые точки изображения пользователя (одного из других участников конференции), оно пересылается на локальный сервер для осуществления преобразования в изображения.
- При получении от локального сервера сообщения, содержащего изображение, а также метку (идентификатор) пользователя, которому оно принадлежит, клиент обновляет соответствующий элемент интерфейса, осуществляя тем самым покадровую смену изображений каждого конкретного пользователя.
- При подключении к конференции и далее раз в задаваемый период времени клиент передает на сервер изображение пользователя. Это необходимо для того, чтобы все собеседники получили или обновили исходное (source) изображение текущего пользователя и могли корректно восстанавливать дальнейшие кадры.
- При получении сообщения, содержащего изображение другого участника конференции, оно пересылается на локальный сервер для обновления хранимого исходного кадра соответствующего пользователя.

Таким образом, клиент реализует обмен видеокадров пользователей друг с другом, при этом пересылая сжатые на локальном сервере данные, что позволяет существенно снизить объем потребляемого трафика.

2.3.2. Описание реализации сервера обмена данными

Далее описана реализация сервера обмена данными. Он реализован на языке JavaScript, с использованием программной платформы NodeJs, а также следующих фреймворков:

- Express – использовался для создания API;

- Socket-io – использовался для реализации соединения по протоколу WebSocket.

Данный компонент системы отвечает за пересылку сообщений между клиентами. При получении какого-либо сообщения от одного из клиентов посредством WebSocket соединения, он пересылает его всем остальным участникам. Таким образом, при данной реализации, от сервера не требуется совершенно никакой вычислительной работы для передачи видеоконтента. Из этого следует, что использование архитектурного подхода MCU для передачи только аудиопотока позволит еще больше снизить нагрузку на сеть, уже не так сильно нагружая сервер, как при исходном варианте (с объединением видео и аудиопотоков в один).

2.3.3. Описание реализации локального сервера

Наконец, опишем реализацию последнего компонента системы – локального сервера. Он написан на языке Python, с использованием следующих фреймворков:

- Flask – для инициализации сервера и организации соединения с клиентом;
- Flask-SocketIO – для организации соединения с клиентом по протоколу WebSocket.

Локальный сервер реализует обмен сообщениями с клиентом и обработку данных моделью [13], взятой из открытых источников. Для эффективной работы с данными было реализовано 2 класса, один из которых отвечает за работу модели, а другой за описание структуры данных, хранящих информацию об исходных (source) изображениях пользователей. Эти классы представлены на Рисунке 21.

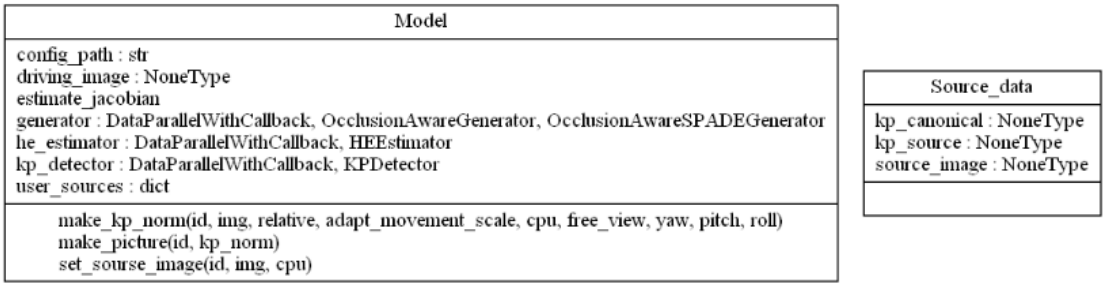

Рисунок 21 – Класс модели и данных, описывающих исходное (source) изображение пользователя

Класс Model отвечает за хранение данных, необходимых для работы модели, а также предоставляет доступ к функционалу этой модели. В полях класса хранятся:

- настройки модели;
- находящееся в обработке потоковое изображение;
- нейросети, составляющие модель;
- словарь и исходными (source) изображениями всех пользователей, участвующих в каждый момент времени в конференции.

Функции модели:

- сжатие полученного изображения пользователя в вектор ключевых точек (45 чисел типа float);
- восстановление по полученному вектору ключевых точек и идентификатору пользователя изображения этого пользователя (соответствующий source хранится в словаре внутри класса);
- обновление source изображения, хранимого внутри класса.

Класс Source_data используется для хранения информации о исходных (source) изображениях пользователей и их ключевых точек.

При получении соответствующих функциям модели запросов от клиента посредством WebSocket соединения, сервер осуществляет обработку данных и отсылает результат обратно на клиент. Кроме того, предусмотрена возможность обработки данных в нескольких потоках одновременно, но их число ограничено (но настраиваемо) с целью избежать перегрузки сервера.

3. Анализ работы прототипа

В данном разделе представлен анализ работы реализованного прототипа. Код проекта находится в репозитории на github: https://github.com/Kordebalet232/Low-bandwidth-videoconference-system.

Для анализа работы реализованной системы были проведены эксперименты, целью которых было измерение времени, затрачиваемого на кодирование и декодирование изображений, а также времени, требуемого для выгрузки результатов обработки из видеопамяти (что необходимо для отправки изображения с локального сервера на клиент).

Для проведения эксперимента использовалось видео с разрешением 256x256, содержащее человеческое лицо. Обрабатывалось 300 первых кадров видео.

Эксперимент проводился на двух устройствах с различными характеристиками. Характеристики использовавшихся устройств приведены в Таблице 2.

устройство	устройство 1	устройство 2
CPU	11th Gen Intel® Core™ i7-11800H 2.30 GHz	Gen Intel® Core™ i5-3470 3.2GHz
RAM	16 Гб	12 Гб
OS	Windows 10	Windows 10
GPU	NVIDIA GeForce RTX 3050 laptop gpu, 8GB	NVIDIA GeForce GTX 1060 3GB

Таблица 2 – Характеристики устройств, использовавшихся в эксперименте

На Рисунках 22-23 представлены результаты эксперимента, проведенного на устройстве 1.

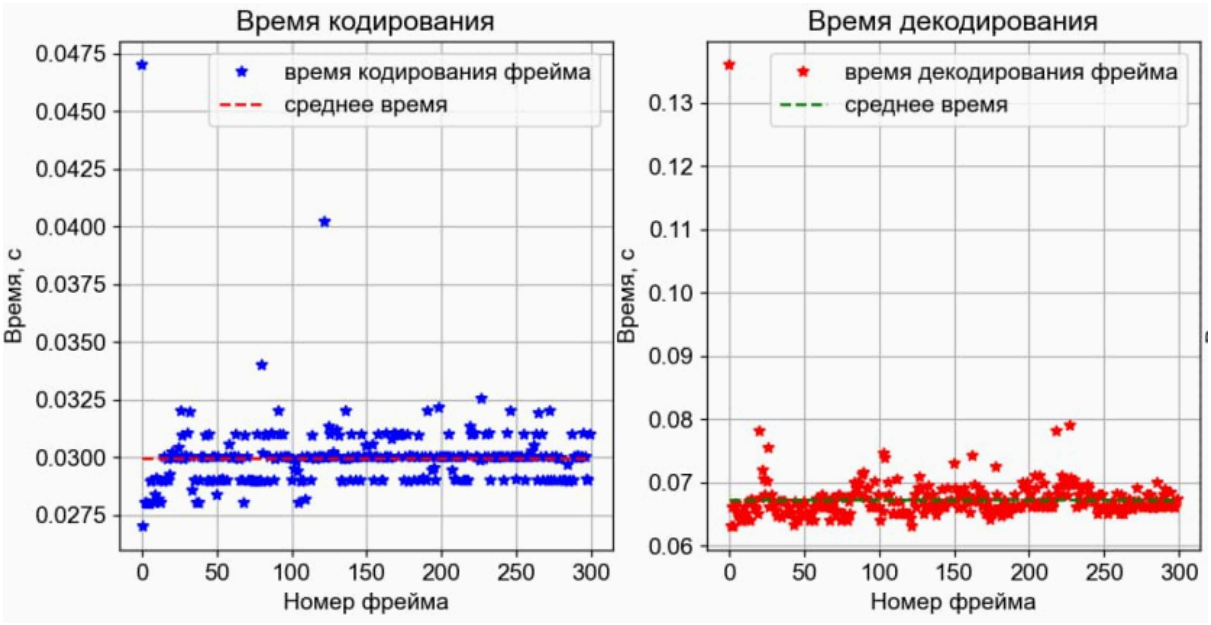

Рисунок 22 – На левом графике - время, затрачиваемое на кодирование одного кадра, на правом - время, затрачиваемое на декодирование одного кадра на устройстве 1. Среднее время кодирования - 0.03 секунды. Среднее время декодирования - 0.068 секунды

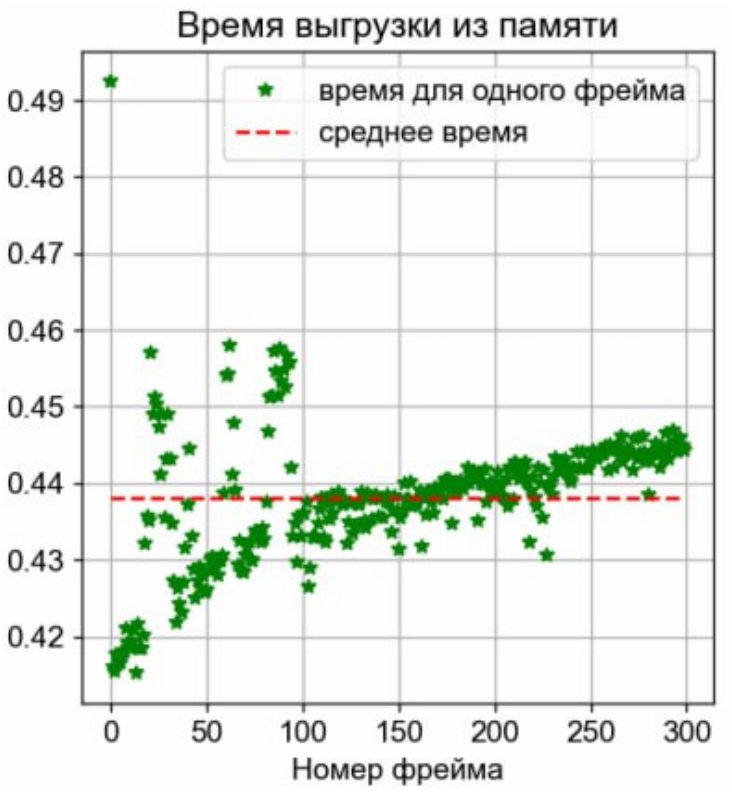

Рисунок 23 – Время, затрачиваемое на выгрузку восстановленного кадра на устройстве 1. Среднее время - 0.438 секунды

Далее, на Рисунках 24-25, представлены результаты эксперимента, проведенного на устройстве 2.

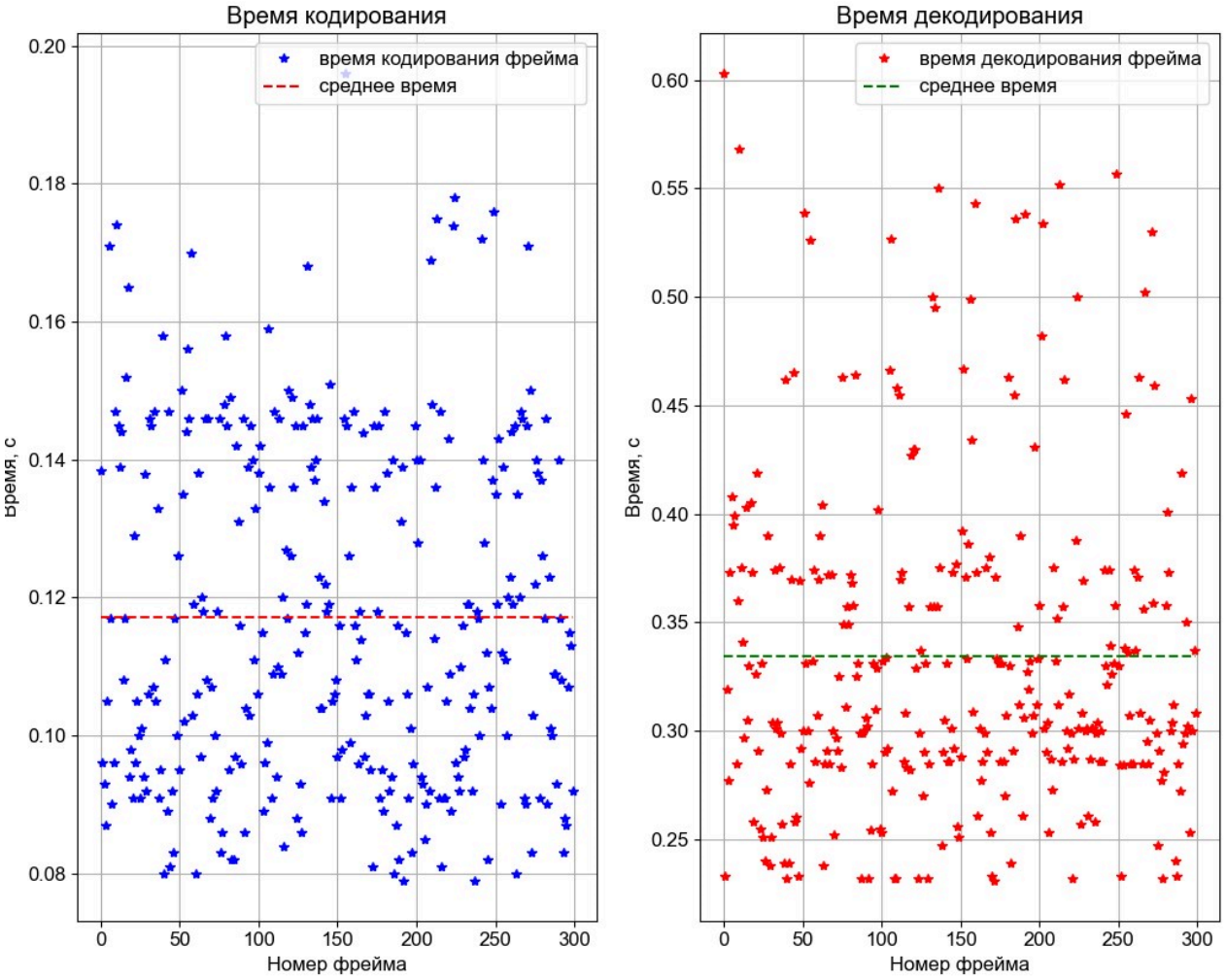

Рисунок 24 – На левом графике - время, затрачиваемое на кодирование одного кадра, на правом - время, затрачиваемое на декодирование одного кадра на устройстве 2. Среднее время кодирования - 0.117 секунды. Среднее время декодирования - 0.334 секунды

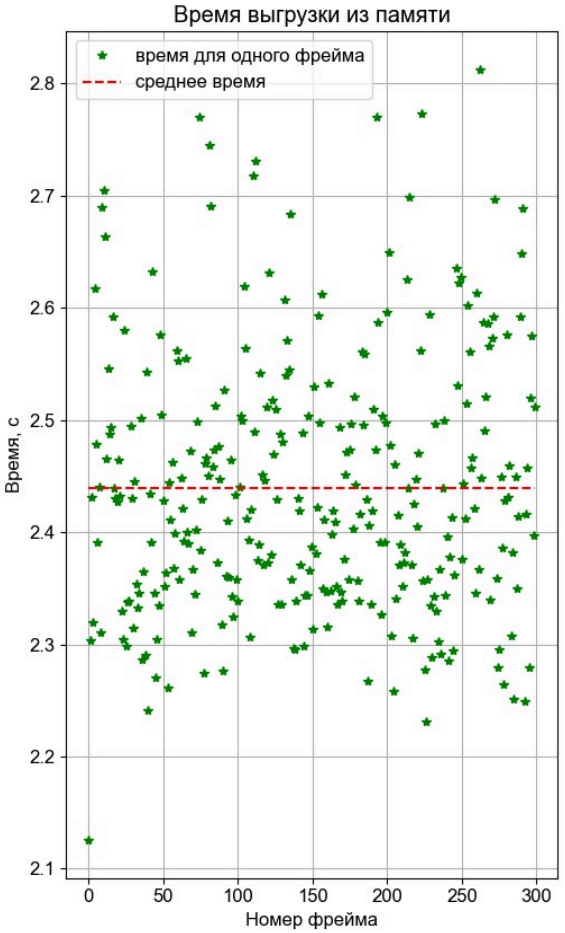

Рисунок 25 – Время, затрачиваемое на выгрузку восстановленного кадра на устройстве 2. Среднее время - 2.438 секунды

Как видно из результатов эксперимента, время, затрачиваемое на операции кодирования, декодирования, и выгрузки результатов обработки из видеопамяти, сильно отличается на различных устройствах. Это объясняется прежде всего различными графическими процессорами устройств, так как основные вычисления происходят именно на них. Так, устройство 2 кодирует кадры эффективнее устройства 1 в среднем в 3,9 раза, декодирует – в 4,9 раза, и выгружает из видеопамяти – в 5,56 раза.

Кроме того, можно заметить, что основное время тратится на выгрузку обработанного кадра из видеопамяти. Таким образом, если удастся реализовать отрисовку обработанного кадра напрямую из видеопамяти (без выгрузки), можно получить существенный прирост эффективности. Такое возможно при использовании низкоуровневых языков программирования или различных аппаратных решений. В реализованном прототипе системы

видеоконференцсвязи это позволило бы увеличить количество отображаемых кадров в секунду до 10 на устройстве 1, и до 2 на устройстве 2. Оптимизация модели, или использование иной, более эффективной, реализации позволит еще более существенно увеличить количество обрабатываемых кадров в секунду.

Таким образом, проведенный анализ эффективности работы используемой модели говорит о хорошем потенциале использования в системах видеоконференцсвязи.

Помимо анализа эффективности было произведено измерение объема передаваемых пакетов, содержащих ключевые точки изображений, между клиентом и сервером обмена данными. Для проведения этих измерений использовались следующие инструменты: Wireshark, и Npcap, предоставляющий адаптер для захвата локальных соединений.

Средний объем пакета, передающего ключевые точки – 1050 байт. Исходя из этого, можно вычислить, что для поддержки видеосвязи с частотой обновления кадров 10 в секунду, а также 20 участниками, в течение 5 минут, потребуется примерно 60Мб трафика на загрузку и 3Мб на выгрузку. Такой результат иллюстрирует огромный потенциал дальнейшего развития предложенного подхода.

Выводы

В результате проделанной работы по разработке архитектуры, реализации прототипа и анализу его работы можно сделать описанные далее выводы.

- Нейросетевые решения для сжатия изображений, содержащих человеческое лицо, имеют огромный потенциал для применения в системах видеоконференцсвязи.
- Предложенная архитектура позволит существенно снизить объем потребляемого для поддержки видеоконференции трафика за счет передачи сильно сжатых данных.

Также в результате анализа реализованного прототипа был выявлен ряд следующих проблем и предложены методы их решения.

- Выгрузка восстановленных изображений из видеопамяти занимает существенную часть времени, что сильно ограничивает функционал системы. Решение этой проблемы путем использования низкоуровневых языков программирования или аппаратного решения, позволит сильно увеличить качество работы системы.
- Результаты экспериментов показали, что требования к производительности клиентских устройств при предлагаемом подходе существенно возрастают. Тем не менее, использование более эффективных моделей и оптимизация работы системы в перспективе позволят ей функционировать на устройствах со средней производительностью и показывать удовлетворительный уровень качества.
- Также стоит отметить, что использование предложенного метода несет в себе опасность попыток подмены личности при участии в видеоконференции. Это происходит в следствие того, что модель в качестве исходного (source) изображения может принимать изображение одного человека, а воспроизводить при этом движения другого.

Таким образом, предложенную в данной работе архитектуру можно комбинировать с различными другими стандартными подходами к организации видеоконференций. Кроме того, систему можно дорабатывать, оптимизируя её работу и решая существующие проблемы.

В качестве продолжения исследования можно предложить оптимизацию работы с видеопамятью, разработку более эффективной и компактной модели для сжатия изображений с человеческими лицами, а

также работу по комбинированию данного подхода с методами, позволяющими минимизировать нагрузку на стороне клиента при передаче аудиопотоков.

Также, для улучшения качества выходных изображений пользователей, можно рассмотреть алгоритмы для восстановления лиц или повышения разрешения изображения.

Заключение

В ходе проведения исследования получены следующие результаты.

1. Проведен обзор существующих архитектурных подходов к реализации систем видеоконференцсвязи.
2. Проведен обзор различных нейросетевых подходов к сжатию изображений, содержащих человеческое лицо.
3. Предложена архитектура для передачи видео, содержащего человеческое лицо в рамках видеоконференции, позволяющая минимизировать нагрузку на сеть на стороне клиента.
4. Реализован прототип, следующий предложенной архитектуре и позволяющий легкое встраивание различных нейросетевых решений в качестве инструментов сжатия данных.
5. Проведен анализ работы прототипа, позволивший оценить перспективы развития данного направления, а также выявить преимущества и недостатки предложенного подхода.

В заключение отметим, что проведенная работа открывает новые перспективы в развитии систем видеоконференцсвязи и может служить основой для дальнейших экспериментов в этой области.

Список использованных источников

[1] One-Shot Free-View Neural Talking-Head Synthesis for Video Conferencing / Ting-Chun Wang, Arun Mallya, Ming-Yu Liu - 2020г.

[2] Comparison of Popular Video Conferencing Apps Using Client-side Measurements on Different Backhaul Networks / Rohan Kumar, Vinayak Naik, Dipanjan Chakraborty, Dhruv Nagpal - 2022г.

[3] Enabling video conferencing in low bandwidth / Muzzafer Ali, Suchetana Chakraborty. : Department of Computer Science and Engineering
Indian Institute of Technology Jodhpur, India - 2022г.

[4] Ultra-low bitrate video conferencing using deep image animation / Goluck Konuko, Giuseppe Valenzise, Stéphane Lathuilière. - 2021г.

[5] First order motion model for image animation / A. Siarohin, S. Lathuiliere, S. Tulyakov, E. Ricci, and ` N. Sebe. -2019г.

[6]Generative adversarial networks for extreme learned image compression / E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. V. Gool. - 2019г.

[7] Deepfovea: Neural reconstruction for foveated rendering and video compression using learned statistics of natural videos / A. S. Kaplanyan, A. Sochenov, T. Leimkuhler, ¨ M. Okunev, T. Goodall, and G. Rufo - 2019г.

[8] Region-of-interest-based rate control scheme for highefficiency video coding, / M. Meddeb, M. Cagnazzo, and B. Pesquet-Popescu - 2014г.

[9] Comparison of Popular Video Conferencing Apps Using Client-side Measurements on Different Backhaul Networks / Rohan Kumar, Vinayak Naik, Dipanjan Chakraborty, Dhruv Nagpal - 2022г.

[10] Video Conferencing Systems Architecture: P2P vs MCU vs SFU [Электронный ресурс]. URL: https://dev.to/forasoft/video-conferencing-systems-architecture-p2p-vs-mcu-vs-sfu-5elp (Дата обращения: 20.05.2024).

[11] One-Shot Free-View Neural Talking-Head Synthesis for Video Conferencing / Ting-Chun Wang, Arun Mallya, Ming-Yu Liu : IEEE/CVF Conference on Computer Vision and Pattern Recognition. - 2021г.

[12] First Order Motion Model for Image Animation / Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, Nicu Sebe : 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). - 2019г.

[13] Реализация метода, описанного в статье “One-Shot Free-View Neural Talking-Head Synthesis for Video Conferencing” [Электронный ресурс]. URL: https://github.com/zhanglonghao1992/One-Shot_Free-View_Neural_Talking_Head_Synthesis (Дата обращения: 20.05.2025).

[14] Датасет VoxCeleb-v1 [Электронный ресурс]. URL: https://www.robots.ox.ac.uk/~vgg/data/voxceleb/vox1.html (Дата обращения: 23.05.2024).

[15] WebSocket vs. Traditional HTTP: Choosing the Right Protocol for Your App [Электронный ресурс]. URL: https://appmaster.io/blog/websocket-vs-traditional-http#websocket (Дата обращения: 20.05.2024).