ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет Санкт-Петербургская школа физико-математических и компьютерных наук

Корепанов Алексей Романович

Внедрение модели автоматического распознавания речи - Конформер (Conformer) в продукт

Выпускная квалификационная работа
по направлению подготовки 01.04.02 Прикладная математика и информатика
образовательная программа «Машинное обучение и анализ данных»

Рецензент
Инженер искусственного интеллекта
_____
А.А. Абугалиев

Научный руководитель
к. т. н., старший преподаватель,
департамент информатики
_____
А.А. Шпильман
Консультант
Ведущий разработчик
машинного обучения, Контур
_____
В.Е. Канторов

Санкт-Петербург 2023

The Government of the Russian Federation
Federal State Autonomous Institution for Higher Education
National Research University Higher School of Economics
St. Petersburg Branch
St. Petersburg School of Physics, Mathematics and Computer Science

Korepanov Aleksei

Implementation of the Automatic Speech Recognition Conformer Model in Production

Master dissertation

Area of studies *01.04.02 «Applied Mathematics and Informatics»*

Master Program “Machine Learning and Data Analysis”

Reviewer	Research Supervisor
Artificial Intelligence	Lead Machine Learning
Engineer (Speech), Replika	Engineer, Kontur
_____	_____
Abugaliev Aleksandr	Vadim Kantorov

Saint Petersburg – 2023

Введение

1.1. Актуальность

В современном мире технологии распознавания речи получают все большее значение в различных сферах деятельности, таких как бизнес, образование, общение и многие другие. Это связано с тем, что автоматизация и оптимизация процессов в этих сферах может значительно повысить эффективность работы и улучшить качество услуг.

В области машинного обучения постоянно появляются новые архитектуры и технологии обучения, которые позволяют улучшать качество работы моделей и повышать их эффективность. Этот процесс неизбежно затрагивает продукты, которые используют машинное обучение в своей работе, и требует постоянного анализа и улучшения уже существующих решений, а также внедрения новых.

В данной работе рассматривается решение - модель Conformer [1], которая показала высокую точность и эффективность в задачах распознавания речи. Цель работы - исследовать возможности внедрения модели Conformer в продукт и по результатам исследования внедрить в продукт, сделав все необходимые оптимизации, которые требуются для качественной работы продукта. Для выполнения этой цели были выполнены следующие задачи:

1. Научиться стабильно обучать модель на данных компании
2. Добиться прироста качества
3. Поддержать работоспособность пословных временных меток
4. Оптимизировать модель, чтобы не было значительного ухудшения по скорости

Таким образом, данная работа является актуальной и имеет практическое значение, поскольку представляет способ улучшения качества

продукта за счет внедрения новой архитектуры нейронной сети, что может значительно повысить его эффективность.

1.2. Основные результаты

В рамках этой работы были получены следующие результаты:

1. Исследованы свойства модели Conformer
2. Решены проблемы сходимости Conformer за счет небольшого изменения архитектуры
3. Получено улучшение качества модели за счет смены архитектуры с Jasper на Conformer
4. Решены проблемы выравнивания полученной транскрипции по времени
5. Внедрение модели Conformer в продукт
6. Анализ скорости модели Conformer
7. Уменьшение времени, затрачиваемое моделью на предсказание.

2. Теоретические основы

2.1. Распознавание речи

Автоматическое распознавание речи (Automatic Speech Recognition, ASR) - это технология, которая позволяет компьютерным системам преобразовывать аудио сигналы, содержащие речь, в текстовую форму.

ASR является сложной и активно развивающейся областью и включает в себя использование различных алгоритмов и моделей машинного обучения, таких как рекуррентные нейронные сети (RNN), сверточные нейронные сети (CNN) и трансформеры. Для достижения высокой точности распознавания речи, требуется обширный набор обучающих данных и сложные алгоритмы обработки сигналов и языковых моделей.

Автоматическое распознавание речи находит широкое применение в различных сферах, включая телекоммуникации, медицину, транспорт, путешествия, образование, развлечения и многое другое. Оно значительно упрощает и улучшает взаимодействие человека с компьютерными системами и способствует автоматизации и оптимизации различных задач, связанных с обработкой и анализом речи.

2.2. Работа со звуком

Звук - это физический феномен, представляющий колебания акустических волн, которые воспринимаются нашим слухом. В контексте моделей машинного обучения, звук представляется в виде цифрового аудио сигнала. Цифровой звук представлен в виде последовательности числовых значений, которые отражают изменение амплитуды звуковых колебаний во времени. Одним из представлений звука являются спектрограммы и мел-спектрограммы, которые используются в этой работе.

Спектрограмма является графическим представлением спектра звука, где на оси X отображается время, на оси Y - частота, а цветовая шкала

отображает амплитуду звуковых частот. Она получается путем применения преобразования Фурье к аудио сигналу, что позволяет разложить его на составляющие частоты. Спектрограмма позволяет визуально представить изменения частотных компонент звука во времени и обычно используется для анализа акустических данных.

Мел-спектрограмма - это вариант спектрограммы, где частотная ось масштабирована по шкале мела. Шкала мела является психоакустической шкалой, основанной на восприятии звука человеческим ухом. Мел-спектрограмма широко используется в области распознавания речи и аудиообработки. Преобразование спектрограммы в мел-спектрограмму позволяет более эффективно представить аудиоинформацию, учитывая особенности восприятия звука человеком.

Мел-спектрограммы и спектрограммы являются важными представлениями звука для моделей машинного обучения, особенно в задачах, связанных с анализом речи и звука. Они предоставляют информацию о спектральных характеристиках звуковых сигналов, которая может быть использована для обучения моделей на задачи распознавания, классификации и генерации речи, аудиоаналитики и других задач, связанных с обработкой звука.

2.3. Функция потерь CTC

CTC (Connectionist Temporal Classification [2]) - это функция потерь, используемая для обучения модели распознавания речи без привязки к фонемам или другим заранее известным меткам. CTC позволяет модели сопоставлять входной аудиосигнал с последовательностью выходных меток, соответствующих транскрипции речи.

Процесс работы функции потерь CTC заключается в следующем:

1. На вход модели подается аудиосигнал, который проходит через кодировщик нейронной сети.

2. На выходе получается последовательность векторов, каждый из которых представляет вероятности для каждой из возможных меток речи на данном временном шаге.
3. Происходит процесс декодирования выходных векторов модели в соответствующую транскрипцию речи. Для этого используется алгоритм динамического программирования, который позволяет получить последовательность меток, соответствующих транскрипции, из выходных векторов модели.
4. Сравнивая полученную последовательность меток с эталонной транскрипцией, которая известна в процессе обучения, вычисляется функция потерь CTC. Эта функция позволяет сравнивать последовательности меток разной длины, учитывая возможные переходы между метками, которые не соответствуют фонемам.

Таким образом, функция потерь CTC позволяет обучать модели распознавания речи без заранее известных меток, что делает ее очень полезной для распознавания разговорной речи, где транскрипция может содержать множество вариантов и не всегда представляет собой четкую последовательность фонем.

2.4. Способы представления текста

В алгоритме распознавания речи, аудиосигнал используется как входные данные для модели, и в результате получается последовательность токенов, которые соответствуют различным временным отрезкам в аудио. Существует несколько методов для представления этих токенов. Один из наиболее простых подходов - использование алфавита языка, известный как графемное представление. Кроме того, можно использовать фонемное представление, где каждой букве в слове сопоставляется определенная фонема. Фонемное представление имеет преимущество в том, что одна и та

же буква может звучать по-разному в разных словах, и поэтому модель может научиться распознавать конкретные звуки речи.

Еще одним способом представления текста является метод BPE (Byte-Pair Encoding), который разделяет слова на подслова и позволяет модели предсказывать комбинации звуков или букв вместо отдельных символов. Это помогает модели более эффективно обрабатывать сложные слова и нестандартные комбинации звуков.

Выбор конкретного метода представления текста зависит от требований задачи и особенностей языка. Каждый из этих подходов имеет свои преимущества и может быть применен в моделях распознавания речи для достижения оптимальных результатов.

2.5. Схема работы моделей распознавания речи

Основная процедура работы модели распознавания речи включает следующие шаги:

1. Получение аудио на вход: Начальным этапом является получение аудиосигнала, который будет подвергаться обработке и распознаванию.
2. Преобразование в спектрограмму: Аудиосигнал преобразуется в спектрограмму или другие формы представления звука, такие как мел-спектрограмма. Спектрограмма представляет собой визуализацию зависимости амплитуды звука от времени и частоты, что позволяет извлекать характеристики и паттерны звука.
3. Кодирование спектрограммы: Спектрограмма передается в кодировщик, который генерирует звуковые представления для задачи распознавания речи. Рассматриваемые модели в данном контексте выступают в роли кодировщиков.
4. Передача кодированной информации в декодировщик: Полученные результаты от кодировщика передаются в декодировщик, который может быть обычным линейным слоем, LSTM-сетью или более сложными языковыми моделями. На выходе декодировщика получается

тензор размерности [vocab_size, time], который содержит вероятности присутствия каждого токена в каждом временном отрезке. Здесь vocab_size представляет размер словаря токенов для распознавания, а time - количество временных отрезков определенного размера в аудио.

5. Декодирование и получение последовательности токенов: Полученное распределение вероятностей подвергается декодированию, либо с использованием жадного алгоритма выбора наиболее вероятного токена, либо при помощи языковой модели, основанной на н-граммах. В результате получается наиболее вероятная последовательность токенов, которая может представлять буквы, фонемы или подслова, соответствующие различным временным отрезкам.
6. Постобработка последовательности токенов: На последнем этапе требуется провести постобработку последовательности токенов, чтобы преобразовать ее из формата "отрезок времени - токен" в удобочитаемый текст или другой соответствующий формат, который удовлетворяет конкретным потребностям задачи

Полученная модель может обучаться с помощью CTC функции потерь. В рамках этой работе именно так выглядит модель, где в качестве спектрограмм используются мел-спектрограммы, кодировщик в этой работе - Conformer, декодировщик - один линейный слой, декодирование вероятностей происходит при помощи жадного алгоритма.

2.6. Механизм внимания

Multi-head self-attention [3] (далее в работе будем называть обычным механизмом внимания) является ключевым компонентом в архитектуре трансформеров, используемых в моделях глубокого обучения. Он играет важную роль в обработке последовательностей данных, таких как тексты или аудио.

В основе механизма внимания идея взаимодействия разных "голов" или "каналов" внимания. Каждая голова внимания отвечает за выделение

различных аспектов контекста и вычисление весовых коэффициентов, которые отражают важность каждого элемента в контексте данного элемента.

В процессе работы механизма внимания, исходная последовательность данных разделяется на несколько подпоследовательностей, называемых "ключами", "значениями" и "запросами". Каждая голова внимания применяет процедуру внимания к этим ключам, значениям и запросам, чтобы получить весовые коэффициенты для каждого элемента.

Затем весовые коэффициенты умножаются на соответствующие значения, и результаты усредняются или конкатенируются, чтобы сформировать выходную последовательность, которая учитывает взаимодействие различных аспектов контекста.

Внимание позволяет моделировать долгосрочные зависимости и выделять важные элементы в контексте. Каждая голова внимания способна сфокусироваться на различных аспектах данных и выделять различные паттерны или аспекты в контексте задачи.

Этот механизм стал широко применяемым в области обработки естественного языка, машинного перевода, распознавания речи и других задач, где важно учитывать контекстуальные зависимости и выделять важные аспекты данных.

3. Обзор предметной области

3.1. Сверточные модели

Jasper [4], QuartzNet [5] и CitriNet [6] - это три широко используемые сверточные модели в области распознавания речи. Эти модели имеют схожую архитектуру, однако отличаются в некоторых деталях, таких как последовательность применения Dropout, типы конволюционных слоев (обычные, pointwise, depthwise) и их расположение. Каждая модель также имеет разное количество обучаемых параметров, что влияет на ее сложность и вычислительные требования. Однако все эти модели разработаны с целью достичь высокой точности и эффективности в задаче распознавания речи. В соответствующих статьях, посвященных этим моделям, представлены более подробные сведения о их архитектуре и параметрах.

Модель	LibriSpeech clean, WER	LibriSpeech other, WER
QuartzNet-15x5	3.90	11.28
Citrinet-1024	**2.52**	**6.22**
Jasper	3.86	11.95

Таблица 1. Сравнение метрик сверточных моделей.

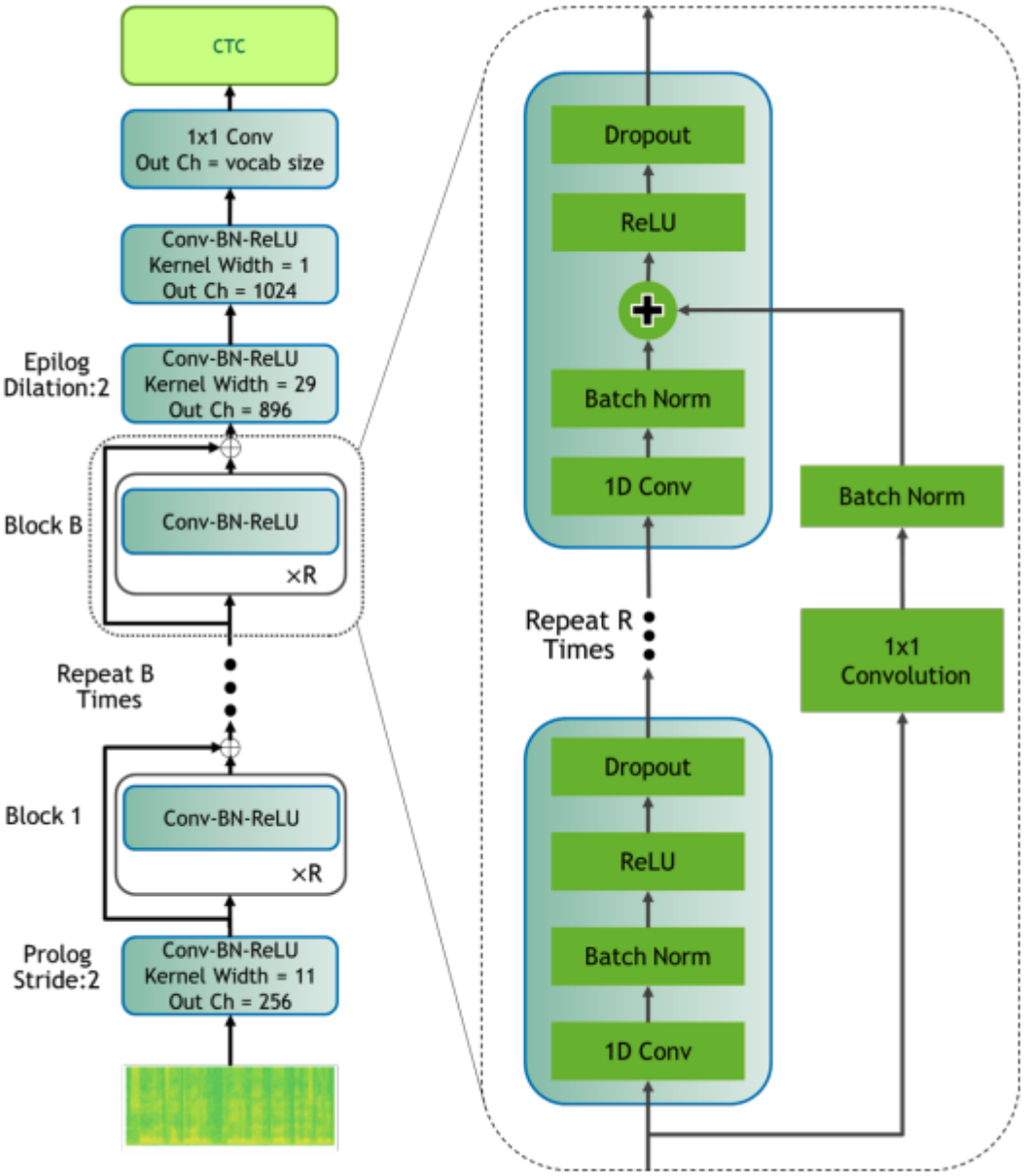

Иллюстрация 1. Архитектура модели Jasper.

3.2. Трансформерные модели

С распространением механизма внимания во множестве областей машинного обучения, его применение в задаче распознавания речи стало все более распространенным с целью улучшения качества результатов. Одной из самых простых реализаций механизма внимания стало использование трансформерного кодировщика.

Трансформерный кодировщик генерирует представления для каждого временного отрезка аудио, учитывая общий контекст аудио. Однако модель Conformer [1] (подробнее описанным в разделе 3.4) представляет собой улучшение этого метода путем добавления конволюционных слоев. Это позволяет модели уделять больше внимания локальному контексту, что является важным аспектом в задаче распознавания речи. Архитектура Conformer стала одной из наиболее широко применяемых архитектур в данной области настоящего времени.

С момента создания модели Conformer появилось множество научных статей, предлагающих улучшения для этой модели. Эти улучшения включают в себя как повышение качества распознавания модели, так и улучшение ее стабильности. Примеры таких статей с улучшениями включают SqueezeFormer [7], Efficient Conformer [8] и Conformer-1 [9, с. 1]. Эти работы способствуют постоянному развитию и совершенствованию методов распознавания речи на основе модели Conformer.

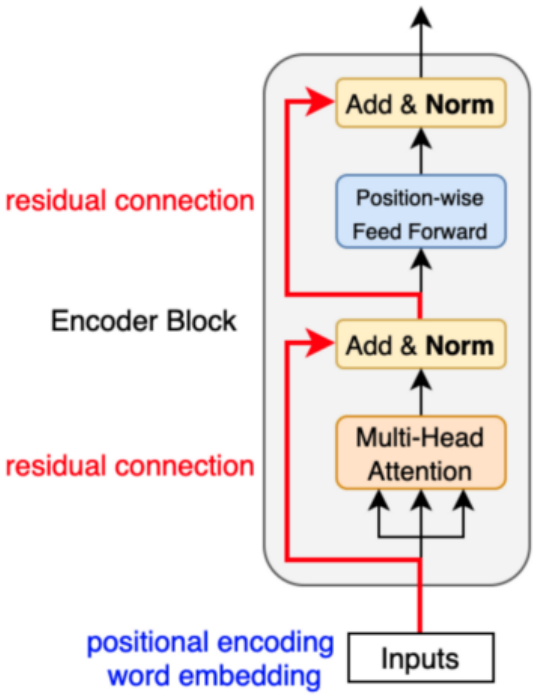

Иллюстрация 2. Архитектура кодировщика трансформера.

3.3. Conformer

Трансформерные модели хорошо улавливают глобальные взаимодействия на основе контента, в то время как сверточные нейронные сети эффективно используют локальные особенности. Conformer объединяет сверточные нейронные сети и трансформеры для моделирования как локальных, так и глобальных зависимостей в аудио последовательности с оптимальным использованием параметров. Модель Conformer значительно превосходит предыдущие трансформерные и сверточные модели, достигая лучших результатов по точности распознавания. Таким образом архитектура Conformer на данный момент является одной из лучших по качеству распознавания. По качеству ее обходят в основном только модели, обученные в self-supervised формате (подрбнее об этом в 3.5)

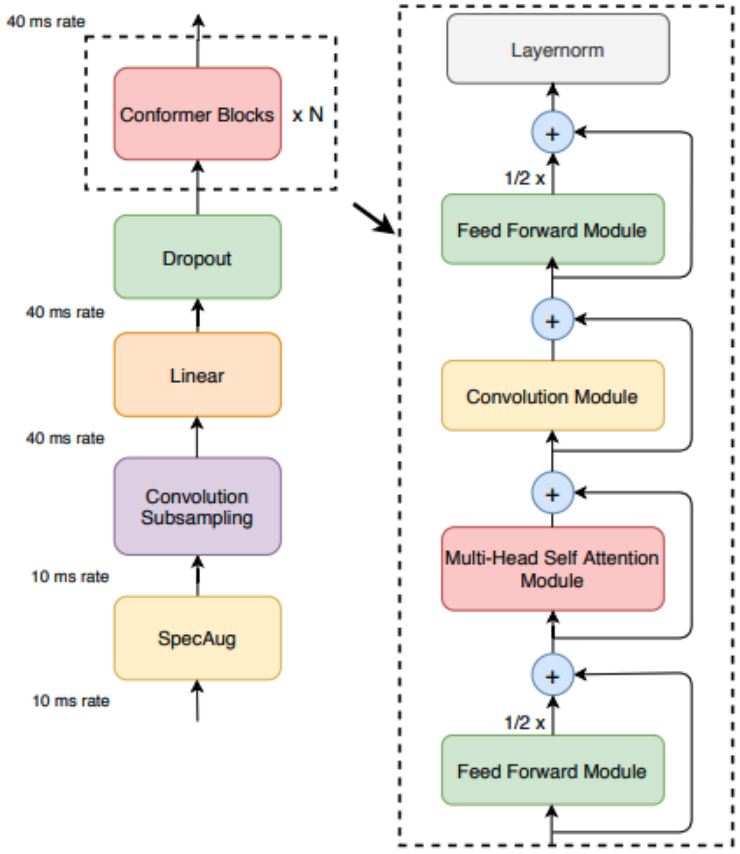

Иллюстрация 3. Архитектура модели Conformer.

Первые слои в Conformer - это subsampling, линейный слой и дропаут. Subsampling необходим для понижения размерности тензора для большей скорости работы и обучения, при этом не ухудшая качества, внутри этого блока находится несколько конволюций со страйдом 2, который уменьшает размерность по всем размерностям, кроме размера батча. Линейный слой нужен, чтобы привести размерность после subsampling к размерности, необходимой на входе Conformer блоков.

Модель Conformer состоит из множества Conformer блоков, например, в этой работе использовалось 18 блоков. Множество этих блоков помогает извлекать многоуровневую информацию из аудио. Один Conformer блок состоит из полносвязного модуля (Feed Forward Module), модуля внимания (Multi-Head Self Attention Module), конволюционного модуля (Convolution Module), второго полносвязного модуля и LayerNorm. Также поверх каждого модуля есть остаточное соединение, когда к выходу модуля прибавляется вход этого модуля.

Полносвязный модуль состоит из ряда линейных слоев, дропаутов, активаций и LayerNorm, также внутри модуля на первом линейном слое размерность каналов увеличивается вдвое, а потом уменьшается обратно на втором линейном слое.

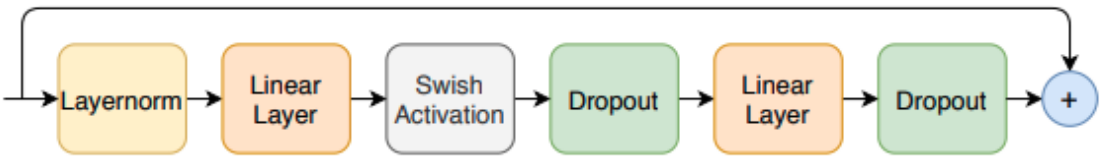

Иллюстрация 4. Архитектура полносвязного модуля Conformer.

В модуле внимание используется техника из модели Transformer-XL [10] - относительное синусоидальное позиционное кодирование. Относительное позиционное кодирование позволяет модулю внимания лучше обобщать информацию на разные длины входных данных, что делает кодировщик более устойчивым к вариации длины фразы. Также используется

preLayerNorm с остаточными связями и применяем метод дропаут, который помогает в обучении и регуляризации глубоких моделей.

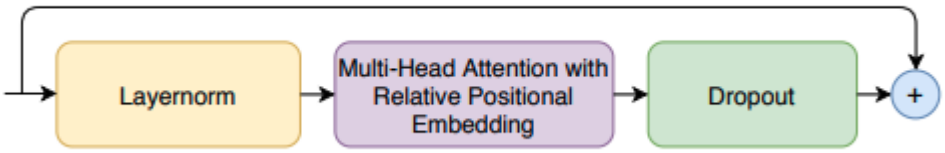

Иллюстрация 5. Архитектура модуля внимания Conformer.

Сверточный модуль начинается с механизма гейтинга, который включает в себя поэлементную свертку и гейтовый линейный блок (GLU). Затем следует один слой одномерной свертки по глубине. После свертки применяется слой нормализации по мини-батчу для улучшения обучения глубоких моделей.

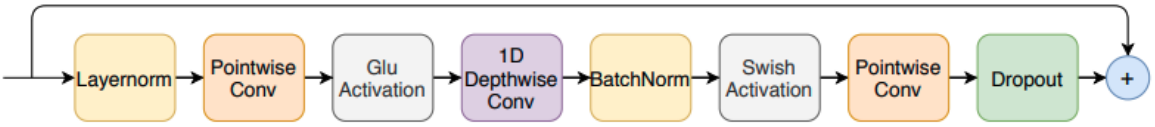

Иллюстрация 6. Архитектура сверточного моделя Conformer.

3.4. Методы обучения без учителя

Метод обучения без учителя в автоматическом распознавании речи играет важную роль, позволяя моделям извлекать и использовать скрытую структуру и зависимости в речевых данных без необходимости в размеченных обучающих данных. Этот метод позволяет моделям обучаться на неаннотированных аудиозаписях, что является особенно полезным, учитывая ограниченную доступность размеченных данных для обучения.

Обучение без учителя в автоматическом распознавании речи может быть использовано для решения нескольких задач. Во-первых, оно может помочь в изучении скрытых структур речи и языка, таких как фонетические или лингвистические признаки, что в свою очередь способствует более

глубокому пониманию речевых данных. Во-вторых, методы обучения без учителя позволяют изучать распределение речи в пространстве признаков и строить более компактные и информативные представления, что может улучшить производительность моделей при распознавании и классификации речи. Кроме того, обучение без учителя может быть использовано для создания предобученных моделей, которые затем могут быть дообучены на небольшом объеме размеченных данных, что способствует повышению качества распознавания речи.

Так одни из лучших методов для обучения модели без учителя является Wav2Vec 2.0 [11], HuBERT [12], WavLM [13]. Эти методы обладают небольшим архитектурным отличием от классических моделей, где из аудио извлекаются признаки при помощи мел-спектрограмм. Здесь вместо мел-спектрограмм используются обучаемые конволюционные слои.

Все эти методы обучаются на восстановление некоторого токена. То есть у нас аудио, часть которого маскируется, и нужно для этого замаскированного кусочка предсказать некоторую метку. Для wav2vec 2.0 - это метка в квантизованном пространстве аудио, для HuBert и WavLM - это некоторые кластеры, полученные после кластеризации данных.

В качестве базовой архитектуры кодировщика используется кодировщик трансформера, но можно и использовать любые другие архитектуры, например, Conformer.

Альтернативным способом обучения без учителя является псевдолейблинг, который используется в таким методах, как IPL [14], SlimIPL [15], Noisy Student [16]. Их суть заключается в том, чтобы при помощи уже обученной модели размечать данные без разметки, а потом дообучаться на этих данных. Например, в этой работе использовался метод IPL.

Таким образом методы обучения без учителя актуальны для задачи автоматического распознавания и показывают достаточно хорошие результаты, но в этой работе рассматривается именно изменение архитектуры, а не способ обучения.

4. Метод

4.1. Метрики оценки качества

Основными метриками качества распознавания речи являются CER (Char Error Rate) и WER (Word Error Rate). Эти метрики считают посимвольное и пословное расстояние Левинштейна от полученной транскрипции до идеальной. Считается как сумма удалений, замен и вставок, разделенная на длину идеальной транскрипции.

$$WER = \frac{S + D + I}{N}$$

Иллюстрация 7. Формула метрики Word Error Rate.

Чтобы посчитать CER или WER по всем данным, необходимо взять среднее по всем данным.

Альтернативным методом оценки качества на всем наборе данных является - взвешенное среднее, которое считается как

$$\frac{\sum_i S_i + D_i + I_i}{\sum_i N_i}$$

Иллюстрация 8. Формула метрики взвещенного Word Error Rate.

Таким образом мы учитываем длинные предложения больше, чем маленькие. В этой работе будет рассматриваться под WER по набору данных как именно взвешенный WER.

4.2. Метрики скорости модели

В этой работе используются две метрики скорости модели - это Inverse RTF, а также время длительности одного предсказания.

RTF - время распознавания одной секунды аудио, а Inverse RTF - это число обработанных секунд за секунду работы видеокарты. Так Inverse RTF вычисляется по формуле:

Inverse RTF = bs * audio_len / inference_time,

где bs - это число примеров в одном батче, audio_len - это длительность аудио в секундах, inference_time - это число секунд, которое потратилось на предсказания одного такого батча.

4.3. Наборы данных

В качестве данных мы будем использовать данные только на русском языке. В обучении участвуют данные, собранные из открытых источников, таких как Youtube, ЭхоМосквы, а также другие открытые наборы данных. Суммарная их длительность - более 5000 часов. Эти данные мы будем использовать для предобучения модели.

Для дообучения используются закрытые наборы данных, полученные с помощью асессоров, компании СКБ Контур, их размер где-то 1000 часов.

Для обучения в формате без учителя используются более 2000 часов доменных неразмеченных данных, которые размечаются специально дообученной моделью.

4.4. Способ обучения

Общий цикл состоит из четырех этапов и занимает суммарно 1.5 - 2 недели.

Первый этап - предобучение для получения базовой модели для дообучения. Предобучение длится ~6 дней на модели Conformer. Здесь используются данные из открытых источников, таких как ЭхоМосквы и Youtube.

Второй этап - первое дообучение, в котором мы обучаем базовую модель для того, чтобы разметить этим чекпоинтом неразмеченные данные.

Здесь используются данные, полученные с помощью асессоров, компании СКБ Контур.

Третий этап - обучение на псевдоразметки, где мы сначала размечаем неразмеченные данные чекпоинтом из второго этапа, а потом дообучаем на этих данных чекпоинт из базовой модели. Этот метод обучения без учителя можно назвать чем-то похожим на IPL.

Четвертый этап - второе дообучение, в котором мы дообучаем чекпоинт из третьего этапа на наших доменных данных. Этот этап аналогичен второму, но только начинается с другого чекпоинта.

4.5. Выбор модели для перехода

Ранее использовалась модель Jasper, качество которой хотелось улучшить. В качестве гипотезы по улучшению качества было принято решение выбрать новую, более актуальную, архитектуру.

Были выделены основные требования по выбору модели:

1. Должна иметь качество WER лучше, чем у Jasper на публичных наборах данных.
2. Не должна долго работать, допустимое ухудшение по скорости - медленнее в 5 раз.
3. Широко используется в различных продуктах с распознаванием речи.
4. Архитектура хорошо исследована, то есть существует множество исследований и публикаций, которые либо используют архитектуру, либо анализируют или улучшают эту архитектуру.
5. Есть множество имплементаций в открытых репозиториях, которые поддерживаются.
6. Привычный способ обучения без учителя.

Почему это важно?

1. Потому что в первую очередь мы хотим улучшить качество нашего продукта.

2. Потому что мы не хотим за счет улучшения и сильно увеличить время ожидания транкрипции для пользователя.
3. Потому что если это архитектура широко используется, то это говорит о ее качестве, стабильности и возможности базировать свое решение на этой архитектуре.
4. Потому что если активно исследуют, то она широко применяется, а также будет гораздо проще найти ответы на свои вопросы в процессе разработки и улучшения модели. Более того, если найдутся какие-либо проблемы, то можно будет найти статью, разбирающие эти проблемы.
5. Это важно, чтобы меньше писать код самому, а также это открывает возможно консультироваться с создателями репозиториев.
6. Это важно для уменьшения рисков, поскольку если брать метод обучения без учителя, как wav2vec 2 или HuBert, то тут накладывается двойной риск, что мы не только меняем архитектуру, но и полностью меняем способ получения претрейна. И потенциально это увеличивает необходимое число проделанной работы вдвое, так как может что-то сломаться в обеих частях.

Так посмотрим на текущий список лучших моделей с только supervised способом обучения:

Модель	**WER**	**Год**	**Число цитирований**
wav2vec 2.0	4.1	2020	2117
Transformer	4.6	2020	325
Squeezeformer-ML	6.05	2022	11
Citrinet 1024	6.22	2021	37
Conformer-CTC-L	6.55	2020	1434
Jasper	8.79	2019	205

Таблица 2. Сравнение метрик моделей на LibriSpeech [17] test-other.

Стоит упомянуть, что метрики здесь примерные, и в каждой статье способы и свойства обучения могут различаться. То есть, может быть использована разная языковая модель (нейронная или KenLM [18]), может различаться число данных для обучения. Также может отличаться и формат представления токенов - BPE или графемы. Поэтому метрики достаточно ориентировочные.

В целом также по опыту можно сказать, что Conformer и wav2vec 2 - это самые популярные модели в области распознавания речи на данный момент, и по ним больше всего материалов. Поэтому основываясь на этом, а также на наших требованиях, было принято решения внедрять Conformer, потому что в wav2vec 2 формат обучения без учителя и отличающийся кодировщик.

5. Предыдущие результаты

В качестве модели для сравнения используется Jasper, которая использовалась до перехода. Способ обучения модели упомянут в 4.3., было принято решение его не менять для уменьшений рисков.

	WER*
Предобучение	36.52
Дообучение 1	15.71
Обучение на псевдоразметке	15.23
Дообучение 2	14.76

Таблица 3. Метрики модели Jasper поэтапно на закрытом валидационном наборе данных.

Inverse RTF был равен 4789 с некоторой погрешностью.

Мы будем сравнивать Conformer с этими метриками. Соответственно мы хотим, чтобы метрика WER на этапе finetune_2 для Conformer была 13.25 или меньше, а Inverse RTF была бы больше, чем 1000.

6. Выполненные задачи

6.1. Научиться стабильно обучать модель

Первым этапом задачи стало - научиться стабильно обучать модель. Это необходимо для того, чтобы в будущих экспериментах не было проблем с обучением моделей. Более того, если продукт собирается переходить на новую архитектуру, то в будущем будут проходить новые эксперименты на этой архитектуре, например, перебор параметров, добавление новых датасетов, улучшения архитектуры для лучшего качество. Чтобы все будущие эксперименты на новой архитектуре проходили по плану и без дополнительных рисков, необходимо научиться стабильно обучать модель.

В ходе первичных экспериментов было выявлено, что модель обладает свойством расходиться. Это проявлялось в следующих формах:

1. Появление NaN в матричном умножении в блоке внимания
2. Резкое увеличение значения функции потерь при обучении

Эти проблемы появлялись с различными способами обучениями, стоит обратить внимание на параметры точности float16 и float32.

Также стоит обращать внимание на параметр learning rate при обучении, который сильно влияет на стабильность обучения. Поэтому стоит подбирать этот параметр тщательно. В ряде экспериментов было видно, что модель может расходится при увеличении learning rate.

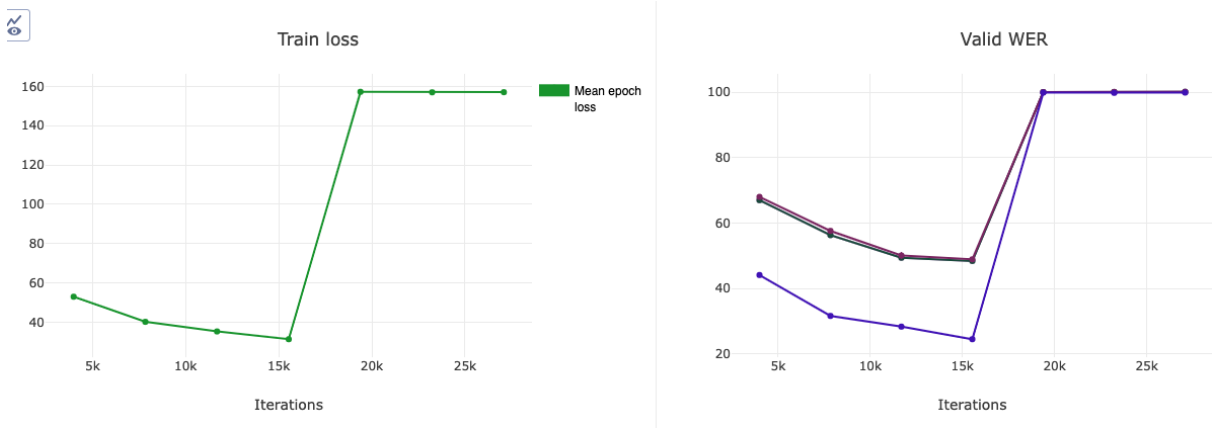

Иллюстрация 9. Графики с примером расхождения модели Conformer.

Режим обучения в float16 помогает значительно сократить время обучения и потребление памяти моделью, что особенно актуально в нашем случае, когда модель обладает большим числом параметров, а также обучение происходит на множестве тысячах часов данных.

При обучении в float16 модели могут обладать свойством переполнения, что и происходило в модели Conformer в матричном умножении в блоке внимания. Поэтому способ решения этой проблемы - это повысить точность, чтобы абсолютный максимум значений был значительно больше. В качестве решение можно обучать в типе bfloat16, диапазон значений которого лежит в промежутке [1e^-38, 3e^38], когда у float16 - [6e^-8, 65504]. Таким образом при умножении относительно больших чисел модель не будет выходить за пределы диапазона типа и не будут появляться NaN.

Другим способом, который помог улучшить стабильность модели Conformer, стала замена PreLayerNorm на нормирование всего тензора. Это изменение помогает, потому что при обучении модели Conformer с несколькими блоками Conformer возникает проблема, когда в каждый последующий блок, кроме первого, на вход подается дважды нормированный при помощи LayerNorm тензор. Эта методика была представлена в модели SqueezeFormer, которая помогала улучшить стабильность модели.

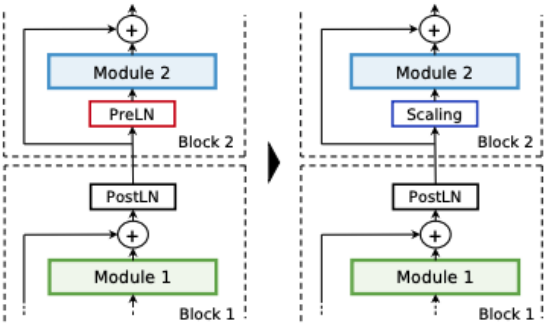

Иллюстрация 10. Сравнение решений с LayerNorm у Conformer (слева) и SqueezeFormer (справа).

Эти две методики действительно помогли сделать модель более стабильной и в дальнейших экспериментах не повторялись проблемы с расхождением модели.

6.2. Добиться прироста качества

Основной целью перехода на новую архитектуру стало улучшение качества модели распознавания речью. Поэтому чтобы перейти на новую архитектуру, нам необходимо показать улучшение качества.

С этой целью было запущено многоэтапное обучение, которое не отличается от способа, представленного в 4.4. Отсутствие отличий по данным необходимо, чтобы показать, что модель Conformer показывает лучшие результаты по-сравнению с Jasper, обучаясь на тех же данных.

В результате многоэтапного обучения были получены следующие метрики:

	Jasper	Conformer
Предобучение	36.52	36.01 (*-0.51*)
Дообучение 1	15.81	13.71 (*-1.9*)
Обучение на псевдоразметке	15.23	12.75 (*-2.48*)
Дообучение 2	14.76	12.58 (*-2.18*)

Таблица 4. Сравнение поэтапных метрик моделей Jasper и Conformer на закрытом валидационном наборе данных.

На этой таблице явно видно, что модель Conformer показывает явный прирост на каждом этапе обучения.

В итоге финальное улучшение составило 2.18 WER на последнем этапе, что соответствует нашим ожиданием, а также становится достаточным условием для перехода продукта на новую архитектуру.

6.3. Поддержать работоспособность пословных временных меток

Пословные временные метки - это часть интерфейса продукта, которые показывают в каком промежутке времени аудио было произнесено то или иное слово. Это необходимо, чтобы пользователь, во время прослушивания аудио и чтения транскрипции, мог визуально отслеживать слово в тексте, которое произносится в текущий момент.

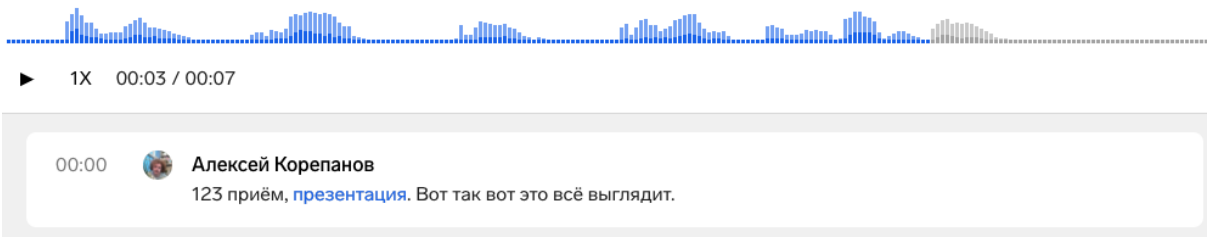

Иллюстрация 11. Пример работы временных меток в интерфейсе.

Чтобы уметь определять пословные временные метки, соответственно необходимо знать время начала и конца слова в аудио. Для определения начала и конца слова в модели Jasper использовался эвристический подход, описанный ниже:

1. На выходе модели Jasper получается строка размера "time" для каждого аудио. Здесь "time" - это количество временных отрезков, соответствующих предсказанию одной буквы. В модели Jasper каждое значение "time" соответствует 320 сэмплам аудио при частоте дискретизации 8 кГц. Таким образом, строка содержит предсказания буквы или специального символа для каждого значения "time".
2. Следующим шагом является определение значений начала и конца для каждого слова на основе значений "time". Слова могут разделяться специальным символом, обозначающим паузу без речи. Путем анализа последовательности значений "time" модель определяет места, где происходит разделение слов.
3. Затем значения начала и конца слова по "time" умножаются на 320, чтобы получить сэмплы оригинального аудио, соответствующие этому слову. Это позволяет определить соответствующие временные метки начала и конца слова в аудио.

Таким образом, описанный метод позволяет определить временные метки начала и конца слова в модели Jasper, основываясь на анализе выходных данных и переводе их в сэмплы оригинального аудио. Это важный шаг для точного определения пословных границ при распознавании речи.

Но в модели Conformer эта эвристика не работала. Основной причиной этому стало то, что для более оптимальной утилизации видеокарты, батч для предсказания, который подается на вход модели, формируется так, чтобы в нем было как можно меньше паддингов. Поэтому аудио конкатенируются, пока их суммарная длина не будет равна 30 секундам. Таким образом за один проход модель предсказывает набор из 16 аудио по 30 секунд, где каждое аудио - это конкатенация более маленьких аудио.

В модели Conformer это не работало, потому что в архитектуре присутствует блок внимания, который в качестве контекста смотрит на всю длину аудио дорожки и из-за этого нет строгого соответствия, что определенный отрезок оригинального аудио соответствует одному значению по “time”.

Эта проблема проявляется как при конкатенации аудио, так и при добавление нулей к аудио. То есть если, например, мы возьмем аудио с длительностью 1 секунда, добавим справа к этому аудио массив нулей, соответствующий 1 секунде аудио, сделаем предсказание для этого аудио, то получим некоторую строку, первая половина которой должна соответствовать оригинальному аудио, а вторая половина - добавленным нулям. Таким образом в первой половине должны быть слова, которые были произнесены в оригинальном аудио, а во второй половине не должно быть ничего. Так работало для модели Jasper. Но в Conformer строгое соответствие не соблюдалось и слова из оригинального аудио могли появляться и во второй половине.

Экспериментально удалось оценить эффект “расплывания” аудио. Получилось, что аудио может расходится максимум на одну секунды как в левую сторону, так и в правую. Поэтому было принято решение при предсказании и формировании батча добавлять к каждому аудио по одной секунде нулей справа и слева, чтобы дать ему максимально разойтись в добавленных нулях, но при это не давать расходится на следующее аудио.

Таким образом, чтобы предсказать одно аудио длительностью 1 секунды, надо предсказывать аудио размером 3 секунды - 1 секунды нулей слева, 1 секунда оригинального аудио и 1 секунда нулей справа. Но встает вопрос - как получить время начало и конца аудио, если его новая длительность не соответствует оригинальной. Для этого была разработана следующая эвристика:

1. Добавляем 1 секунду нулей к оригинальному аудио длины T сэмплов с двух сторон. Так получается новое аудио длины T + 16000 (для аудио с частотой дискретизации 8kHz)
2. Предсказываем моделью текст в этом аудио
3. Аудио “расплывается” в области нулей и получается K букв, которое было предсказано
4. Для получение длительности одной буквы мы считаем T / K

Тогда если, например, предсказание для оригинального аудио длиной 2240 состояло из 7 букв, после добавления нулей и предсказания моделью - предсказание текста стало состоять из 9 букв, чтобы получить длину одной буквы мы считаем 2240 / 9 = 248 сэмлов аудио. Такая эвристика проделывается для каждого оригинального аудио. И соответственно, зная место начала и конца каждого законкатенированного аудио, мы можем востановить время начала и конца каждого слова.

Таким образом удалось обеспечить модели Conformer стабильность пословных временных меток.

Небольшой проблемой стало уменьшение эффективности батча для предсказаний, потому что в них увеличилось число нулей, но экспериментально подтвердилось, что это незначительно уменьшает скорость предсказания.

6.4. Сравнение скорости модели Conformer с Jasper

В соответствии с требованиями, предъявленными к внедрению модели, возникла необходимость учесть некритическое ухудшение скорости работы модели. При этом было допущено, что ухудшение скорости модели в пределах 5 раз не считается критичным. Для измерения данного параметра использовалась метрика Inverse RTF, которая была подробно описана в разделе 4.2.

Для достоверного измерения метрики были проведены эксперименты в условиях, максимально приближенных к реальным условиям применения

модели. Это включало процесс конвертации аудио в необходимый формат, выделение отрезков аудио на основе голосовой активности, применение модели и всех соответствующих постобработок. Сравнение производилось при использовании размера батча, равного 16, и с длительностью одного примера в батче, составляющей 30 секунд.

В результате сравнения результатов нагрузочных тестов для двух моделей получились результаты: Jasper - 4789 Inverse RTF, Conformer - 2731 Inverse RTF.

В результате сравнительного анализа скорости работы моделей Conformer и Jasper было выявлено, что модель Conformer демонстрирует медленную производительность по сравнению с моделью Jasper. Конкретно, модель Conformer имеет увеличение времени работы в два раза по сравнению с моделью Jasper. Учитывая наши требования по некритичному ухудшению скорости модели, такое увеличение скорости в два раза является приемлемым.

На основании этого сравнения и соответствующих требований, мы рекомендуем внедрение модели Conformer в текущем виде в наш продукт. Данная модель удовлетворяет нашим требованиям в отношении некритичного ухудшения скорости, и ее использование будет способствовать достижению поставленных целей и ожидаемых результатов.

6.5. Анализ скорости модели Conformer

С целью выяснения причин, приведших к снижению производительности модели Conformer, необходимо провести анализ ее работы.

В соответствии с оригинальной статьей о модели Conformer, в блоке внимания был применен относительный метод кодирования позиций, основанный на работе Transformer XL. Это позволяет модели эффективно обрабатывать аудио произвольной длительности. Однако, при использовании абсолютного метода кодирования позиций, модель лишается этого свойства,

хотя она все равно будет функционировать. В данном случае, для упрощения эксперимента, мы будем анализировать вариант с использованием абсолютного метода кодирования позиций, который широко изучен и имеет установленные характеристики.

При проведении анализа производительности модели были получены следующие результаты:

- Время выполнения Subsampling составило 71 мс.
- Блок внимания занял 110 мс.
- Конволюционный блок занимал 83 мс.
- Полносвязный блок 1 занял 40 мс.
- Полносвязный блок 2 требовал 54 мс.

Необходимо отметить, что указанные цифры были получены для фиксированного размера входного тензора, который оставался неизменным в течение эксперимента. Кроме того, приведенные значения представляют собой суммарное время выполнения каждого блока, за исключением Subsampling, который применялся только один раз в течение одного прогнозирования модели.

Можно выявить, что больше всего времени занимает блок внимание, который применяется многократно, а также subsampling, который хоть и применяется всего один раз, но при этом занимает время сравнительно с другими блоками, которые применяются многократно. Так при оптимизации скорости в первую очередь стоит обратить внимание на эти два блока.

6.6. Ускорение модели Conformer

В рамках этой работы будет проведена работа по ускорению модели Conformer только для режима предсказания, обучение рассматриваться не будет.

Существует множество методов, таких как Flash Attention [19] и Memory Efficient Attention [20], предназначенных для улучшения производительности данного блока. Однако, на основании инженерных

соображений, было принято решение использовать именно метод Memory Efficient Attention.

Выбор данного метода обусловлен несколькими факторами. Во-первых, имплементация метода Memory Efficient Attention в пакете xformers обладает необходимой поддержкой и способностью обрабатывать большое количество масок. Во-вторых, этот метод работает на большем спектре окружений и видеокарт.

С учетом этих факторов, применение метода Memory Efficient Attention представляется наиболее подходящим для ускорения работы блока внимания в рамках данной работы.

На вход в блок внимания приходят тензоры в формате с паддингом, поэтому нужно в блок внимания также передавать маску. Подходящий формат масок из xformers [21] - это BlockDiagonalCausalWithOffsetPaddedKeysMask, который преобразует маску в блочно-диагональный формат, как BlockDiagonalMask, но при этом позволяет создавать пустые вставки между блоками, чтобы учитывать паддинг. Проблема этой маски в том, что на момент написания работы эта маска не поддерживает обучение, а работает только при предсказании. Поэтому для использования этой маски, необходимо проводить обучение с BlockDiagonalMask. Такой процесс замедляет обучение, потому что нужно приводить тензор к формату без паддинга, что занимает много времени на центральном процессоре. Это особенно критично, если обучение происходит с большим батчом данных. И таким образом хоть и происходит ускорение предсказания, но не происходит ускорение обучения. Улучшение этой маски предполагается проделать в будущей работе. Так или иначе, здесь приведено ускорение предсказания при помощи использования BlockDiagonalCausalWithOffsetPaddedKeysMask.

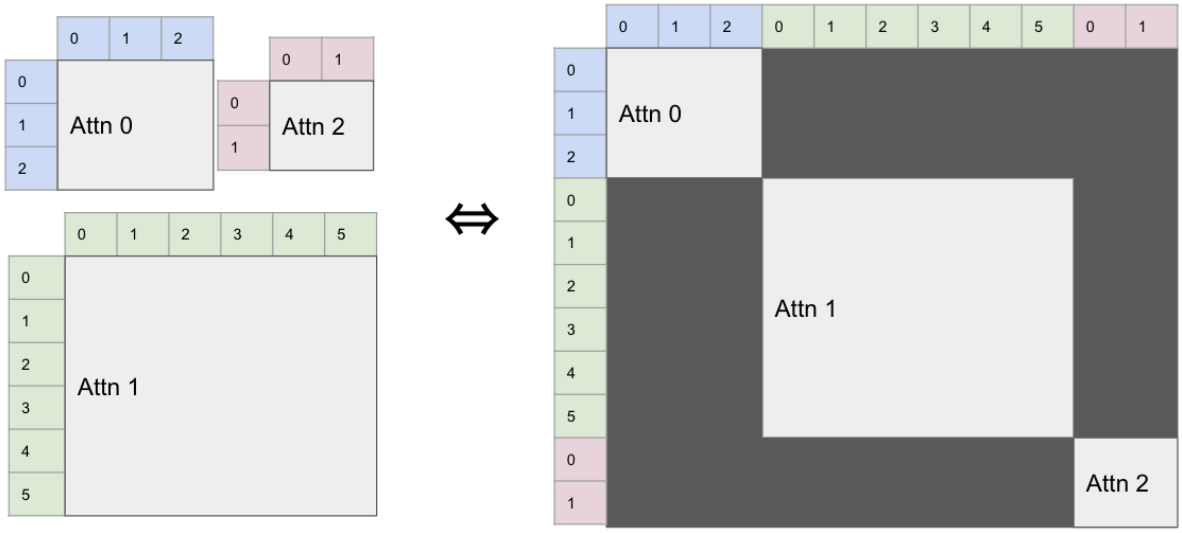

Иллюстрация 12. Схема работы BlockDiagonalMask для механизма внимания.

При замене классической реализации блока внимания с методом абсолютного кодирования позиций было достигнуто значительное ускорение данного блока. В результате, время выполнения блока внимания сократилось с 110 мс до 37 мс, что является приблизительным ускорением в 3 раза. Таким образом, блок внимания перестает быть самым медленно работающим блоком в модели и демонстрирует более эффективную производительность.

Для ускорения блока subsampling рекомендуется обратить внимание на метод Fast Conformer [22], который предлагает заменить текущую реализацию блока, состоящую из двумерных сверток, на Depthwise свертки. Это позволяет значительно ускорить данный блок в 7 раз - с 71 мс до 10 мс.

Кроме того, в рамках метода Fast Conformer можно рассмотреть увеличение subsampling factor с 4 до 8. Это приводит к уменьшению размерности тензоров внутри модели в 2 раза, что в свою очередь обеспечивает ускорение работы каждого блока модели. Важно отметить, что в исследовании указывается, что данные улучшения в скорости не оказывают никакого влияния на качество модели.

Таким образом, рассмотрение и применение указанных улучшений из метода Fast Conformer позволяет значительно ускорить работу блока subsampling без какого-либо снижения качества модели.

Еще одно улучшение, которое было рассмотрено - это оптимизация графа вычислений. Есть множество методов, которые делают это, но в этой работе рассматривается torch.compile [23]. Метод torch.compile представляет собой последнее средство для повышения производительности вашего кода на PyTorch. С помощью torch.compile возможно ускорение выполнения кода PyTorch путем JIT-компиляции кода PyTorch в оптимизированные ядра, при этом требуя минимального количества изменений в коде.

Применение метода torch.compile привело к ускорению модели в 1.5 раза. Однако, важно отметить, что в будущем предполагается более детальное исследование графа вычислений. Это связано с наличием множества операций, которые нарушают граф вычислений из-за отсутствия поддержки данных операций в PyTorch. Один из примеров такой операции - Memory Efficient Conformer, который использует специальное собственное ядро из пакета xformers.

После внедрения всех улучшений было достигнуто суммарное ускорение в 3.3 раза. Время выполнения сократилось с 376 мс до 112 мс.

Однако, в настоящее время эти улучшения еще не внедрены в продукт по ряду причин:

1. Необходимо провести ряд реальных экспериментов, чтобы подтвердить отсутствие ухудшения качества после применения методов ускорения модели.
2. На данный момент невозможно использовать метод абсолютного позиционного кодирования в продукте, так как длина аудио, подаваемого на предсказание, превышает длину аудио, использованную в процессе обучения модели. Поэтому необходимо провести работу по относительному позиционному кодированию.
3. В рамках будущих исследований также рассматривается ускорение процесса обучения. Однако для этого требуется реализация режима обучения для блока BlockDiagonalCausalWithOffsetPaddedKeysMask.

Таким образом, был успешно проанализирован способ ускорения модели с использованием абсолютного позиционного кодирования. Однако для внедрения этих методов в продукт требуется дальнейшая работа по улучшению и оптимизации.

Заключение

Таким образов в рамках этой работы были получены следующие результаты:

1. Модель Conformer была успешно интегрирована в существующую кодовую базу.
2. Архитектура Conformer была оптимизирована с целью обеспечения более высокой стабильности работы модели.
3. Путем применения модели Conformer удалось достичь улучшения метрик на 2 WER (Word Error Rate).
4. В ходе разработки модели были учтены пользовательские сценарии, включая поддержку временных меток и приемлемое ухудшение скорости работы модели.
5. Имплементация модели Conformer была оптимизирована с целью улучшения ее скорости работы. Было достигнуто ускорение в 3.3 раза по сравнению с исходной реализацией.
6. Модель Conformer успешно внедрена в продукт и в настоящее время активно используется.

Таким образом, результаты исследования и разработки модели Conformer свидетельствуют о ее успешной интеграции в продуктовую среду, а также о достигнутом улучшении метрик и оптимизации производительности модели.

Список литературы

[1] A. Gulati *и др.*, «Conformer: Convolution-augmented Transformer for Speech Recognition». arXiv, 16 май 2020 г. Просмотрено: 21 май 2023 г. [Онлайн]. Доступно на: http://arxiv.org/abs/2005.08100

[2] A. Graves, S. Fernandez, F. Gomez, и J. Schmidhuber, «Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks».

[3] A. Vaswani *и др.*, «Attention Is All You Need». arXiv, 5 декабрь 2017 г. doi: 10.48550/arXiv.1706.03762.

[4] J. Li *и др.*, «Jasper: An End-to-End Convolutional Neural Acoustic Model». arXiv, 26 август 2019 г. Просмотрено: 21 май 2023 г. [Онлайн]. Доступно на: http://arxiv.org/abs/1904.03288

[5] S. Kriman *и др.*, «QuartzNet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions». arXiv, 22 октябрь 2019 г. doi: 10.48550/arXiv.1910.10261.

[6] S. Majumdar, J. Balam, O. Hrinchuk, V. Lavrukhin, V. Noroozi, и B. Ginsburg, «Citrinet: Closing the Gap between Non-Autoregressive and Autoregressive End-to-End Models for Automatic Speech Recognition». arXiv, 4 апрель 2021 г. doi: 10.48550/arXiv.2104.01721.

[7] S. Kim *и др.*, «Squeezeformer: An Efficient Transformer for Automatic Speech Recognition». arXiv, 15 октябрь 2022 г. Просмотрено: 21 май 2023 г. [Онлайн]. Доступно на: http://arxiv.org/abs/2206.00888

[8] M. Burchi и V. Vielzeuf, «Efficient conformer: Progressive downsampling and grouped attention for automatic speech recognition». arXiv, 8 сентябрь 2021 г. doi: 10.48550/arXiv.2109.01163.

[9] «Conformer-1: a robust speech recognition model», *News, Tutorials, AI Research*, 15 март 2023 г. https://www.assemblyai.com/blog/conformer-1/ (просмотрено 21 май 2023 г.).

[10] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, и R. Salakhutdinov,

«Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context». arXiv, 2 июнь 2019 г. doi: 10.48550/arXiv.1901.02860.

[11] A. Baevski, H. Zhou, A. Mohamed, и M. Auli, «wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations». arXiv, 22 октябрь 2020 г. doi: 10.48550/arXiv.2006.11477.

[12] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, и A. Mohamed, «HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units». arXiv, 14 июнь 2021 г. doi: 10.48550/arXiv.2106.07447.

[13] S. Chen *и др.*, «WavLM: Large-Scale Self-Supervised Pre-Training for Full Stack Speech Processing», *IEEE J. Sel. Top. Signal Process.*, т. 16, вып. 6, сс. 1505–1518, окт. 2022, doi: 10.1109/JSTSP.2022.3188113.

[14] Q. Xu, T. Likhomanenko, J. Kahn, A. Hannun, G. Synnaeve, и R. Collobert, «Iterative Pseudo-Labeling for Speech Recognition». arXiv, 26 август 2020 г. doi: 10.48550/arXiv.2005.09267.

[15] T. Likhomanenko, Q. Xu, J. Kahn, G. Synnaeve, и R. Collobert, «SlimIPL: Language-Model-Free Iterative Pseudo-Labeling». arXiv, 29 август 2021 г. doi: 10.48550/arXiv.2010.11524.

[16] D. S. Park *и др.*, «Improved Noisy Student Training for Automatic Speech Recognition», в *Interspeech 2020*, окт. 2020, сс. 2817–2821. doi: 10.21437/Interspeech.2020-1470.

[17] V. Panayotov, G. Chen, D. Povey, и S. Khudanpur, «Librispeech: An ASR corpus based on public domain audio books», в *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, Queensland, Australia: IEEE, апр. 2015, сс. 5206–5210. doi: 10.1109/ICASSP.2015.7178964.

[18] K. Heafield, I. Pouzyrevsky, J. H. Clark, и P. Koehn, «Scalable Modified Kneser-Ney Language Model Estimation».

[19] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, и C. Ré, «FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness». arXiv, 23 июнь 2022

г. doi: 10.48550/arXiv.2205.14135.

[20] M. N. Rabe и C. Staats, «Self-attention Does Not Need $O(n^2)$ Memory». arXiv, 10 октябрь 2022 г. doi: 10.48550/arXiv.2112.05682.

[21] «facebookresearch/xformers: Hackable and optimized Transformers building blocks, supporting a composable construction.» https://github.com/facebookresearch/xformers (просмотрено 21 май 2023 г.).

[22] D. Rekesh *и др.*, «Fast Conformer with Linearly Scalable Attention for Efficient Speech Recognition». arXiv, 10 май 2023 г. doi: 10.48550/arXiv.2305.05084.

[23] «PyTorch 2.0». https://pytorch.org/get-started/pytorch-2.0/ (просмотрено 21 май 2023 г.).