

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)»

ФАКУЛЬТЕТ ИННОВАЦИЙ И ВЫСОКИХ ТЕХНОЛОГИЙ
КАФЕДРА АНАЛИЗА ДАННЫХ

Выпускная квалификационная работа
по направлению
01.03.02 "Прикладные математика и информатика"
НА ТЕМУ:

**ПОСЛОГОВОЕ АКУСТИЧЕСКОЕ МОДЕЛИРОВАНИЕ С ПОМОЩЬЮ
СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ**

Студент _____ Торунова А.С.

Научный руководитель _____ Федотов С. Н.

Зам. зав. кафедрой _____ Бунина Е.И.

Москва, 2017

Содержание

1 Введение	3
1.1 Архитектура системы распознавания речи	3
1.2 Обзор подходов к акустическому моделированию	5
2 Архитектуры нейронных сетей и функции потерь	7
2.1 Bidirectional LSTM	7
2.2 Connectionist Temporal Classification loss	10
2.3 Архитектуры сверточных сетей	11
3 Описание задачи	15
3.1 Постановка	15
3.2 Измерение качества	15
4 Эксперименты	17
4.1 Данные	17
4.2 Параметры декодирования	17
4.3 Слоговый парсер	18
4.4 Базовый эксперимент: буквенное СТС без сверток	19
4.5 Слогоное СТС без сверток	20
4.6 Буквенное СТС со свертками	21
4.7 Слогоное СТС со свертками	22
4.8 Буквенное СТС с Resnet-29	23
4.9 Слогоное СТС с Resnet-29	24
4.10 СТС со свертками для пар букв	25
4.11 СТС со свертками для ВРЕ	26
4.12 СТС со свертками для двухбуквенных и трехбуквенных слогов	27
4.13 Слогоное СТС с более широкими страйдами	28
4.14 Результаты экспериментов	30
5 Заключение	31
5.1 Итоги работы	31
5.2 Дальнейшие исследования	32

1 Введение

1.1 Архитектура системы распознавания речи

Распознавание речи - это задача преобразования звука в текст. Этую задачу пытаются решать с 50-х годов прошлого века, но статистический подход к распознаванию речи начал развиваться только с конца 70-х. Все современные системы распознавания речи используют статистический подход. Система распознавания речи получает на вход предобработанный звук и выдает наиболее вероятную последовательность слов. Звук предобрабатывают следующим образом: запись разбивают на участки длины по 25 мс, которые могут пересекаться (обычно по 10 мс), для каждого такого участка (фрейма) считают некоторое преобразование, например, спектrogramму или Mel Frequency Cepstral Coefficients (MFCC). Это не единственные преобразования, существует еще много других вариантов, например raw waveform. На выходе у преобразования получается вектор чисел. Таким образом звук представляется как последовательность векторов. Эта последовательность и подается на вход системе распознавания.

Архитектура любой системы распознавания речи содержит три основных компоненты: акустическую модель, языковую модель и декодер.

Акустическая модель позволяет по звуковому фрейму определить, какая фонема (в общем случае - токен некоторого фиксированного алфавита) была на нем произнесена. Для каждого временного отсчета звука, поданного на вход, она выдает распределение по фонемам, прозвучавшим в этот момент времени. Акустическая модель является в каком-то смысле самой основной частью системы распознавания речи, поскольку именно она описывает преобразование из звука в последовательность фонем. Если просто взять для каждого временного отсчета звука наиболее вероятную фонему, то уже получится читаемая фонетическая транскрипция исходной записи, а если потом воспользоваться фонетическим словарем, то получится уже привычная буквенная транскрипция. Однако эта транскрипция, будет не очень хорошего качества. Действительно, поскольку распределения, выдаваемые для данного временного отсчета

зависят от предыдущего контекста, то взятие argmax для каждого отсчета может дать в итоге транскрипцию с не самой большой вероятностью. Также в полученной транскрипции могут быть последовательности букв, не являющиеся словами. Чтобы решить эти две проблемы используется языковая модель и более сложный декодер.

Языковая модель описывает наиболее вероятные последовательности слов языка, то есть она предсказывает условную вероятность слова при условии контекста: $P(w|w_1, \dots, w_k)$. Один из способов построения такой модели - n-gram language models. Для каждого сочетания слов длины от 1 до n подсчитываются частоты встречаемости этих сочетаний. Еще один способ, появившийся с развитием нейросетей - моделирование языковой модели с помощью рекуррентной нейросети. Сеть обучается предсказывать следующее слово по предыдущему. Если подавать такой сети на вход слово, то она будет выдавать следующие возможные слова с их вероятностями.

Декодер объединяет информацию акустической и языковой моделей, чтобы по входному звуку предсказать наиболее вероятную фразу. Для этого используются Weighted Finite State Transducers(WFST) [1] - конечные автоматы, у которых есть входной алфавит, выходной алфавит, и на ребрах задана некоторая функция веса. На каждом ребре такого автомата написан символ входного алфавита, выходного алфавита и вес, поэтому путь в этом автомате от стартовой вершины до конечной можно интерпретировать как отображение из входного алфавита в выходной, поэтому это удобно использовать в распознавании речи, так как нужно построить отображение из фонем в буквы. В качестве весов в этом случае используются вероятности перехода, полученные из данных языковой и акустической моделей. Если использовать данные только акустической модели, то слова, в полученной фразе скорее всего будут написаны неправильно, из-за того, что акустическая модель "не знает" какие слова существуют в языке. Чтобы такого не происходило и транскрипции состояли из корректных слов используются языковые модели, которые можно тренировать как на отдельном датасете, так и на транскрипциях к звуковому датасету, если их достаточно много. Поиск наиболее вер-

ятной транскрипции по такому конечному автомату нельзя произвести напрямую из-за экспоненциально большого количества вариантов, однако можно воспользоваться эвристикой - алгоритмом beam-search. Этот алгоритм представляет собой поиск в ширину, только с одним ограничением: мы всегда обходим фиксированное число наиболее вероятных переходов из вершины. Это число называется beam size (ширина луча) по аналогии с шириной луча фонарика в темноте – чем он шире, тем больше объектов мы видим / можем обойти. Иногда вместо beam size для ограничения количества обходимых соседей используют некоторое пороговое значение beam. Если вероятность текущей наиболее вероятной последовательности отличается от вероятности последовательности, в которую мы хотим сделать переход, больше чем величина beam, то мы переход не делаем.

1.2 Обзор подходов к акустическому моделированию

С начала семидесятых до приблизительно 2000-х годов в качестве акустической модели в системах распознавания речи использовались скрытые марковские модели (Hidden Markov Models) [2] с Gaussian Mixture Models(GMMs) для предсказания вероятностей фонем при условии входа, начиная с 2000-х стали использоваться гибридные модели (DNN/HMM), в которых вместо GMM использовались нейронные сети. В описанных подходах требуется генерировать выравнивания между звуком и фонемами в транскрипции, которые затем используются для того, чтобы для каждого фрейма выдавать распределение по фонемам. В DNN/HMM подходе это приводит к тому, что нужно, чтобы между обновлениями нейросети происходили обновления выравниваний, генерируемых HMM, что ведет к итеративному алгоритму обучения. В 2010-х с развитием GPU стал набирать популярность так называемый end-to-end подход, который призван обойти эту проблему. В этом подходе для генерации распределения по фонемам используется одна модель, которой на вход подается предобработанный звук. На выходе у нее сразу распределение вероятностей по фонемам(или буквам) для каждого временного отсче-

та, без промежуточных шагов. Существуют различные варианты таких моделей: нейросети вида encoder-decoder, рекуррентные сети с функцией потерь специального вида, сверточные сети. Получается, что такая модель все равно неявно может выучить выравнивания между звуками и фонемами, но без прерывания тренировки для его обновления, как в подходе DNN/HMM, что влияет еще и на скорость обучения. Также в end-to-end подходах отсутствуют априорные знания о структуре речи, заложенные человеком, в отсутствие которых модель строит представление речи совсем иным способом, что возможно и помогает ей справляться с задачей лучше.

Еще одной особенностью классических систем распознавания речи является то, что в некоторых случаях они используют фонетическое представление речи, поэтому для того, чтобы переводить последовательности фонем в слова, нужен либо отдельный фонетический словарь, составление которого требует работы лингвистов, либо отдельно натренированная модель grapheme-to-phoneme, переводящая буквы в фонемы.

Одним из наиболее популярных end-to-end подходов является использование RNN с функцией потерь CTC-loss (Graves et al [3]), который мы будем использовать в данной работе. При использовании RNN с CTC-loss можно легко заменить распознаваемые символы на, например, буквы, в этом случае не нужен фонетический словарь. Для многих языков(например, русского) вполне естественным предположением будет разбиение речи не на фонемы или буквы, а на слоги. Целью данной работы является построение акустической модели на основе RNN с CTC-loss, выдающей распределение по слогам.

2 Архитектуры нейронных сетей и функции потерь

В этом разделе описываются архитектуры нейронных сетей и функции потерь, которые будут использоваться при проведении экспериментов.

2.1 Bidirectional LSTM

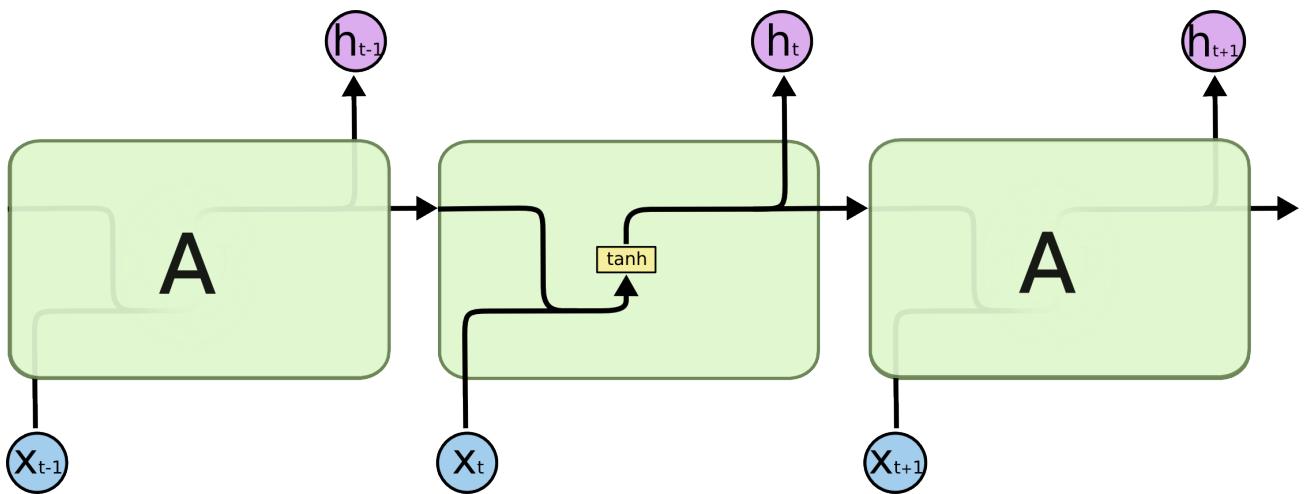


Рис. 1: Обычная RNN ([4])

В рекуррентных нейросетях скрытое состояние зависит от предыдущих, что позволяет обрабатывать данные, которые состоят из последовательностей. Если, например, нужно классифицировать по последовательности кадров в видео, что на них происходит, то скорее всего происходящее на одном кадре зависит от предыдущих, поэтому есть смысл использовать модели, которые запоминают информацию о предыдущих кадрах, такие как RNN. Однако у обычных RNN существует проблема длинных зависимостей, то есть если последовательность очень длинная, то RNN начинает "забывать" информацию о скрытых состояниях, которые были много шагов назад. Это связано с тем, что градиенты по очень старым состояниям начинают затухать при умножении.

Эту проблему решает особый тип RNN - Long Short-Term Memory (LSTM) [5]. У таких сетей повторяющийся блок устроен сложнее, и в них есть дополнительное скрытое состояние C_t , градиент по которому

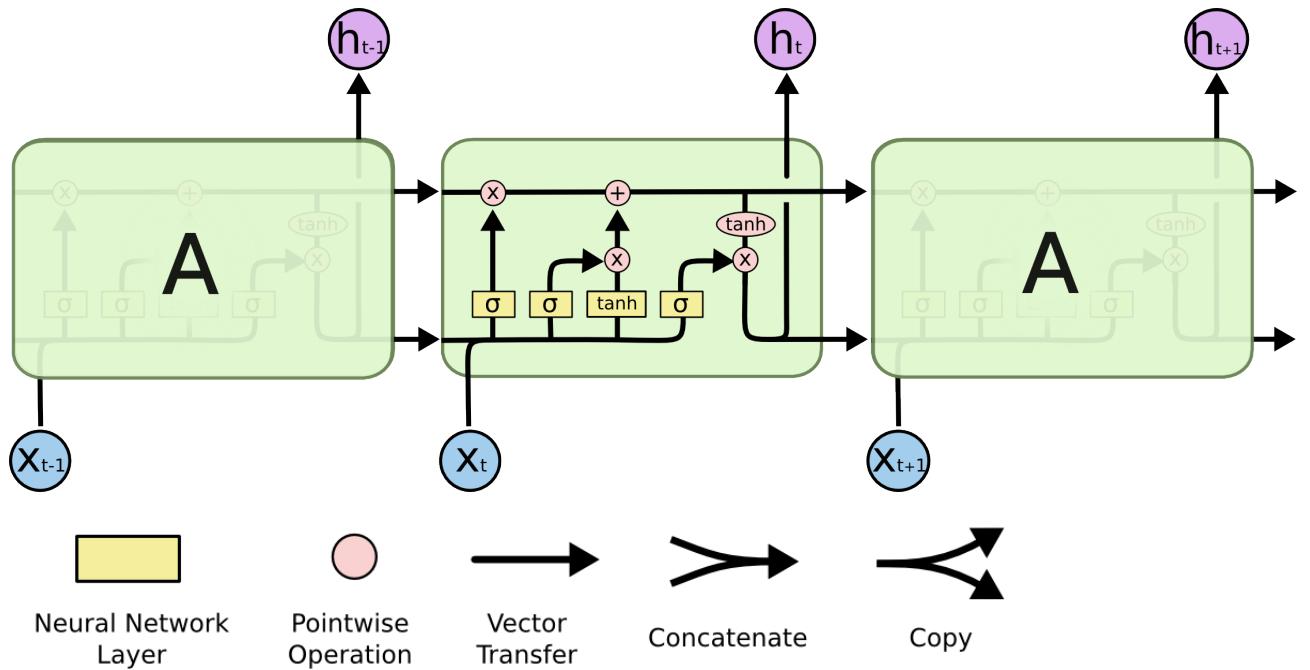


Рис. 2: LSTM ([4])

не уменьшается. В этом состоянии и хранится информация о длинных зависимостях, при этом есть возможность добавлять или удалять из него информацию. Именно из-за того, что в LSTM нет проблемы с длинными зависимостями, в большинстве случаев используется именно она. Еще существует модификация LSTM - Gated Recurrent Unit (GRU) [6], у которого принцип работы схож с LSTM, но у него одно внутреннее состояние. Отсюда преимущество GRU - у сети с GRU меньше параметров, чем у такой же сети с LSTM.

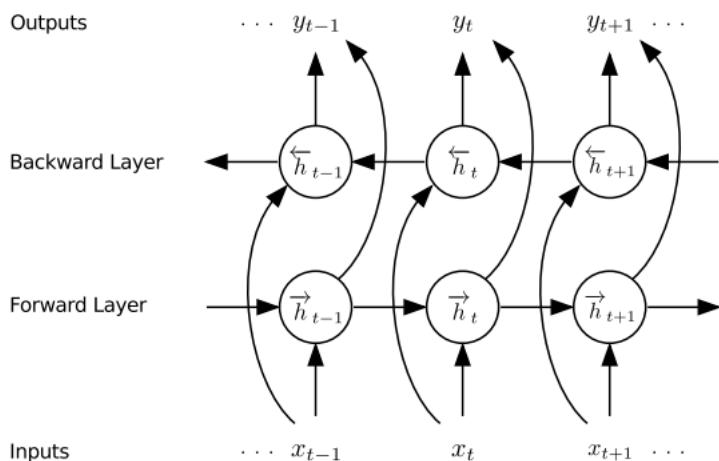


Рис. 3: Bidirectional LSTM (Graves et al [7])

Для нашей задачи распознавания речи тоже можно использовать (unidirectional) LSTM, однако особенность этой задачи в том, что мы транскрибуем сразу всю запись, поэтому нет особого смысла использовать только левый контекст. Отсюда возникает идея использовать два скрытых слоя: один будет обрабатывать запись в прямом порядке (forward layer), а второй в обратном (backward layer), а затем конкатенировать полученные скрытые состояния. В таком случае под слоем для удобства будем понимать два слоя: forward и backward. Таким образом трехслойная bidirectional LSTM на содержит 3 forward и 3 backward слоя. В наших экспериментах мы будем использовать пятислойную bidirectional LSTM, архитектура которой приведена на рисунке 4.

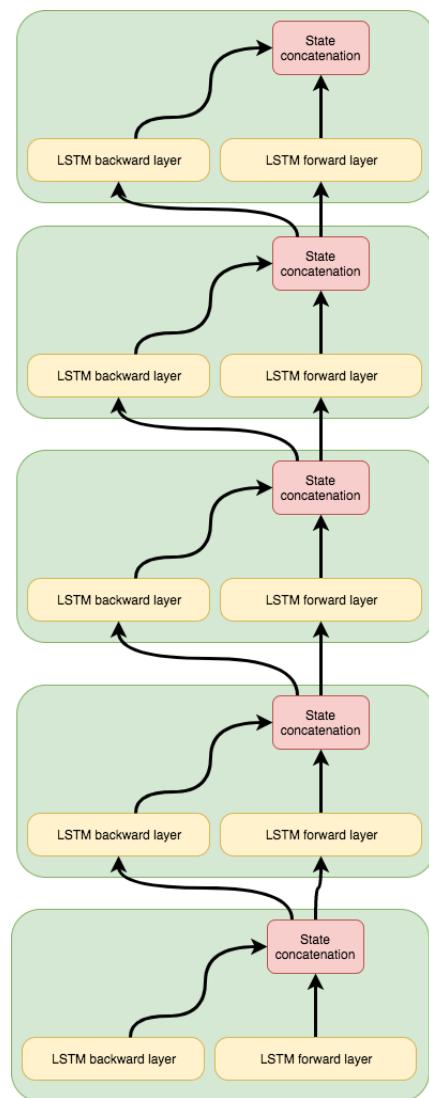


Рис. 4: 5-layer Bidirectional LSTM

2.2 Connectionist Temporal Classification loss

Теперь опишем функцию потерь, которую хотим минимизировать – СТС-loss. Рассмотрим RNN, на вход которой подается вектор x длины T , где T - количество временных отсчетов. На выходе у этого нейросети векторы $y_t, t \in 1 \dots T$ высоты N , где от 0 до N пронумерованы элементы транскрипции, которые мы хотим предсказывать(фонемы, буквы и т.д.) плюс специальный символ ' _ ', обозначающий пустоту(пропуск). Будем интерпретировать эти векторы следующим образом: вероятность того, что в момент времени t был произнесен k -ый элемент транскрипции будет равна

$$P(k, t|x) = \frac{\exp(y_t^k)}{\sum_l \exp(y_t^l)} = \text{Softmax}(y_t)[k]$$

Назовем a разметкой входа x , если это последовательность длины T , состоящая из элементов транскрипции и пропусков, соответствующих элементам x . Будем считать, что для любого входа x все элементы разметки этого входа на элементы транскрипции независимы. Тогда вероятность разметки при условии входа равна

$$P(a|x) = \prod_{t=1}^T P(a_t, t|x)$$

, где a_t - элемент разметки на месте t .

Заметим, что одной и той же транскрипции входа x могут соответствовать различные разметки a . Вспомним, что ' _ ' - пустой элемент разметки, тогда разметки $(c, _, a, t)$, $(c, _, a, _, t)$, $(c, _, _, a, t)$ соответствуют одной и той же транскрипции (c, a, t) . Также мы будем удалять повторяющиеся символы, то есть разметки (c, c, a, t) и (c, a, t, t) тоже соответствуют транскрипции (c, a, t) . Обозначим теперь за B оператор, который сначала удаляет в разметке повторяющиеся символы, а затем пропуски, то есть мы получаем транскрипцию, соответствующую разметке. Тогда вероятность транскрипции y при условии входа x это:

$$P(y|x) = \sum_{a \in B^{-1}(y)} P(a|x)$$

Получается, что если у нас есть для каждой аудиозаписи правильная транскрипция, то мы можем натренировать нейросеть так, чтобы она минимизировала функцию СТС (Connectionist Temporal Classification) function:

$$CTC(x) = -\log P(y^*|x)$$

где y^* - правильная транскрипция. То есть мы хотим таким образом максимизировать вероятность получения абсолютно правильной транскрипции(фонетической или буквенной).

2.3 Архитектуры сверточных сетей

Обычно в нейросетях, которые применяются для задачи sequence-to-sequence, которой является задача распознавания речи, есть рекуррентные слои. Это неудивительно, поскольку они по построению умеют запоминать зависимости в последовательностях. Однако это не единственный инструмент для решения этой задачи, поскольку можно использовать одномерные свертки по временной размерности. Действительно, такая операция свертки сохраняет информацию о нескольких предыдущих временных отсчетах, причем выполняется она очень быстро. Одномерные свертки уже применялись в статье Facebook Wav2Letter [8], где было отмечено, что при использовании сверток обучение ускоряется по сравнению с обычным LSTM. Однако Facebook использовал немного другую функцию потерь для обучения, мы же в наших экспериментах попробуем объединить свертки с bidirectional LSTM с CTC-loss.

Первая архитектура, которую мы будем использовать будет содержать 6 сверточных слоев и 3 рекуррентных слоя (bidirectional LSTM). Параметры сверточных слоев будут следующие: длина свертки будет 17 для всех слоев, количество фильтров будет 256, 256, 256, 512, 512, 512 и страйды будут длины 1, 1, 2, 1, 1, и 2 для слоев от первого до шестого соответственно. Размер скрытого слоя у bidirectional LSTM будет 512. Эта архитектура показана на рисунке 5.

Следующим шагом будет сделать более глубокую сверточную сеть, для этого мы будем использовать немного модифицированную архитек-

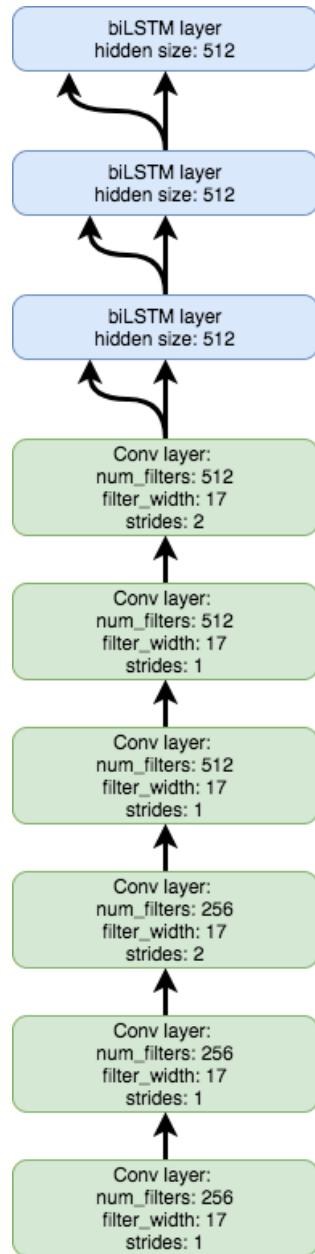


Рис. 5: LSTM with convolutions

туру Resnet-34 [9], у которой будет не 34, а 29 слоев. Мы будем ее использовать, поскольку она уже применялась для задачи распознавания речи Microsoft Research и показала state-of-the-art результаты. Кроме сверточных слоев еще будет 3 рекуррентных слоя (bidirectional LSTM). Особенностью этой архитектуры являются residual connections - дополнительные связи, которые передают информацию с нижних слоев на верхние благодаря тому, что градиент по слагаемому в сумме не затухает, отсюда возможности для обучения более глубоких архитектур. Длина свертки в нашей модели является постоянной для всех сверточных слоев и равна 3,

количество фильтров возрастает с номером слоя: для первых двух слоев 64 фильтра, для следующих семи – 128, для следующих восьми – 256, для последних пяти – 512. Также есть два слоя pooling(max и average) со страйдами 2. Pooling тоже выполняется по временной размерности. Слой upscale увеличивает длину по временной размерности в 2 раза, чтобы можно было тренировать буквенные модели с этой архитектурой. Без upscale слишком большие страйды делают невозможной тренировку буквенных моделей, так как длина входа по времени для LSTM становится меньше длины транскрипции в буквах, из-за чего нельзя посчитать CTC-loss. Однако для слоговых моделей можно убирать слой upscale, что является одним из их преимуществ по сравнению с буквенными. Описанная выше архитектура показана на рисунке 6

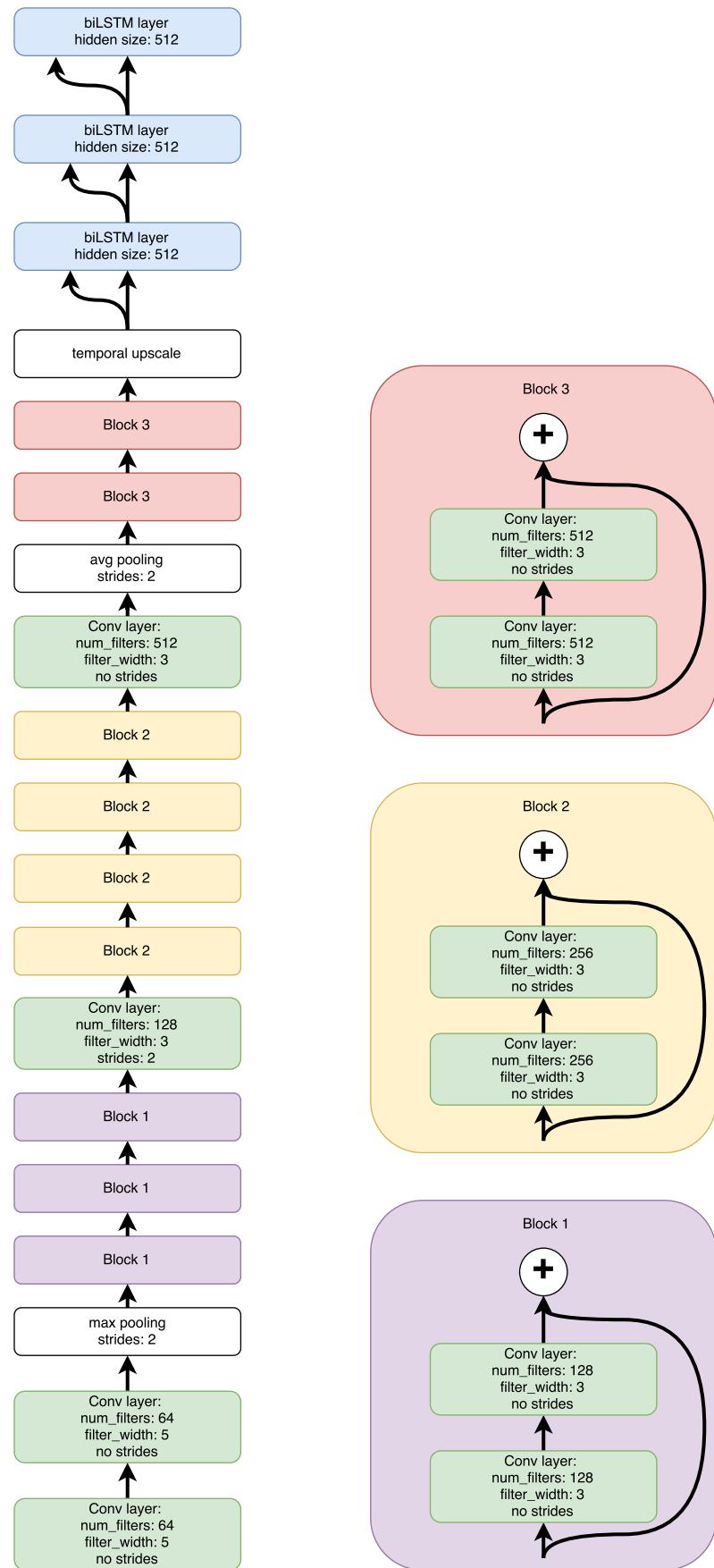


Рис. 6: LSTM with Resnet-29

3 Описание задачи

3.1 Постановка

Как уже было сказано ранее, целью данной работы является построение акустической модели на основе RNN с CTC-loss(Graves et al [3]), выдающей распределение по слогам.

Для обучения такой модели требуются данные, в которых слова разбиты на слоги. Эти данные можно получить, если научиться разбивать слова на слоги. Оказывается задача разбиения на слоги не такая тривиальная, как кажется, и набором правил разбиения обойтись нельзя. Поэтому мы поступим следующим образом: обучим LSTM предсказывать для каждой буквы в слове вероятность того, что после нее нужно сделать разбиение, а в качестве функции потерь возьмем Cross Entropy loss.

Научившись разбивать слова на слоги, мы разобьем все транскрипции в имеющемся звуковом корпусе на слоги, затем мы будем обучать наши акустические модели на основе рекуррентных сетей с CTC-loss выдавать распределение по слогам вместо фонем или букв.

Следующим этапом в данной работе будет исследование применения сверточных слоев, где свертка выполняется по временной размерности, вместе с рекуррентными в этой задаче.

Итогом работы будет сравнение качества и скорости обучения слоговых и буквенных акустических моделей. Также кроме слоговых акустических моделей будут исследованы другие способы разбиения слов в транскрипции на подслова и их применение в акустическом моделировании.

3.2 Измерение качества

Для того, чтобы измерить качество нашей акустической модели нам нужны оставшиеся 2 части системы: языковая модель и декодер.

В качестве декодера будет использоваться Eesen decoder [10], в репозитории проекта Eesen на Github есть скрипты для построения WFST

для декодирования, их мы и будем использовать. Это можно сделать для слов, построив правильный lexicon.txt, в котором вместо букв будут слоги, буквы или другие под слова.

Для получения языковой модели мы будем пользоваться SRILM toolkit [11], которым можно сгенерировать n-грамную языковую модель в ARPA формате.

В качестве метрики по которой мы будем сравнивать качество моделей будет WER(Word Error Rate):

$$WER(y, y^*) = \frac{edit_distance(y, y^*)}{len(y^*)}$$

где y^* - правильная последовательность слов, y - предсказанная, $edit_distance$ - расстояние Левенштейна для последовательностей слов, а $len(y^*)$ - количество слов в y^* .

Для примера возьмем в качестве y^* фразу "Мама мыла раму" , а в качестве y фразу "Мама мыла велосипед". Видно, что y^* получается из y заменой слова 'велосипед' на 'раму' , поэтому $edit_distance(y, y^*) = 1$ и $WER(y, y^*) = \frac{1}{3}$.

Еще иногда для примерной оценки качества модели во время тренировки используется LER(Label Error Rate):

$$LER(y, y^*) = \frac{charlevel_edit_distance(y, y^*)}{len(y^*)}$$

где y^* - правильная последовательность слов, y - предсказанная, $charlevel_edit_distance$ - обычное расстояние Левенштейна, а $len(y^*)$ - количество букв в y^* . В предыдущем примере $LER(y, y^*) = \frac{9}{14}$.

Стоит отметить, что с помощью LER невозможно сравнивать модели, которые предсказывают разные токены (например, буквы и слоги), поскольку для одного и того же слова количество токенов зависит от их типа.

4 Эксперименты

Все эксперименты были выполнены используя API Tensorflow [12] для Python. Код экспериментов на Github:

1. Код слогового парсера: penguin138/ctc_convolution
2. Основной код СТС с дополнениями для работы со слогами: standy66/py_asr

4.1 Данные

Поскольку для СТС нужен датасет объема 80-100 часов, то доступные открытые датасеты русской речи не подходят, так как слишком малы.

В качестве датасета использовался закрытый датасет звонков, который содержит ≥ 600 часов спонтанной зашумленной речи на финансовоую тематику. В качестве звуковых признаков использовались спектрограммы фреймов по 25 мс с пересечением по 10 мс. Данные для языковой модели были частично взяты из транскрипций звукового датасета, поскольку он достаточно объемен, еще туда были добавлены слова на общую тематику.

В качестве данных для обучения слогового парсера использовался русский Wiktionary, поскольку это единственный открытый словарь, в котором для слов есть разбиение на слоги.

4.2 Параметры декодирования

Для сравнения качества моделей использовался Eesen decoder. У него подбирались следующие параметры: acoustic scale – отношение веса вероятностей из акустической модели к весу вероятностей из языковой модели и beam – параметр, отвечающий за то, включается ли сосед вершины в список обхода или нет. Если логарифм вероятности гипотезы отличается от логарифма вероятности наилучшей на данный момент гипотезы больше, чем на величину beam, то эта гипотеза не включается в список обхода, иначе включается.

Для декодирования с помощью скрипта ngram-count из SRILM toolkit были сгенерированы две 4-граммные языковые модели: одна на основе частот на финансовую тематику и другая на основе транскрипций к используемому звуковому датасету. Однако, поскольку часть транскрипций, на основе которых была построена вторая модель, были в валидационной выборке для акустических моделей, то результаты декодирования со второй моделью лучше, чем с первой. Еще можно заметить, что у моделей, показывающих наилучший результат со второй моделью acoustic_scale как правило меньше, чем у моделей, показывающих наилучший результат с первой моделью. Поэтому правильнее смотреть на результаты именно с первой языковой моделью, но для сравнения мы приведем результаты обеих моделей.

4.3 Слоговый парсер

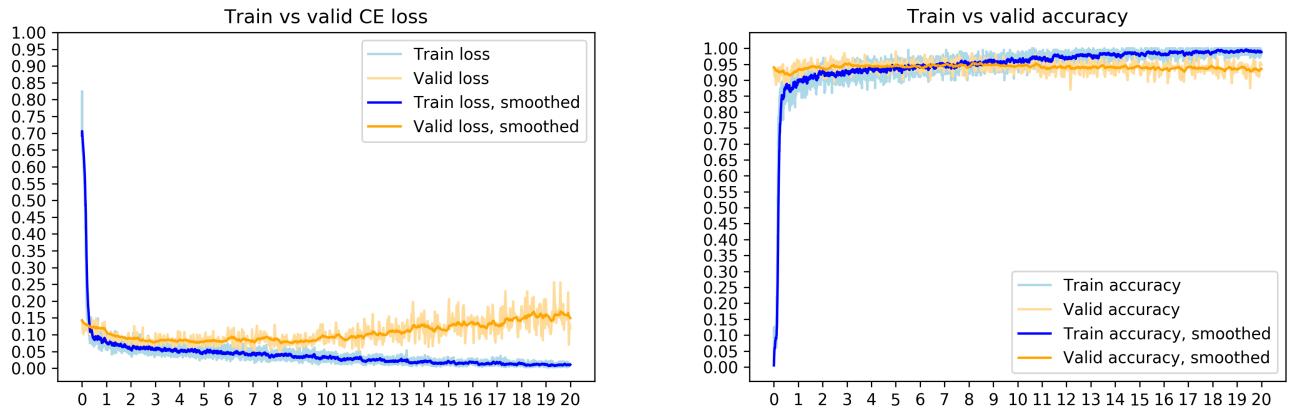


Рис. 7: Syllable parser training

Поскольку правила разделения на слоги не такие простые, как кажется на первый взгляд, было решено сделать собственный слоговый парсер, который выучивает правила разбиения из данных. Для этого была использована рекуррентная сеть, которая должна была для каждой буквы в слове предсказывать, нужно ли делать после нее разбиение.

Когда человек разбивает слово на слоги, то он при принятии решения разбить или не разбить в данном месте слово учитывает не только то, какие буквы были слева. Он учитывает и правый контекст тоже, по-

этому логично использовать для парсера двунаправленную рекуррентную сеть. Поэтому для разбиения на слоги использовалась пятислойная bidirectional LSTM с Cross-Entropy Loss. Размер скрытого слоя был 512. Тренировалась сеть с помощью AdamOptimizer [13].

В качестве обучающей выборки был использован русский Wiktionary. Всего слов в нем около 200000, но слов с корректным разбиением на слоги всего 65000, они и были использованы для обучения.

Получилось приемлемое качество разбиения на слоги: 94% accuracy, где accuracy тут это процент полностью правильно разбитых на слоги слов, но видно, что после 6-8 эпохи парсер переобучается. Всего различных слов получилось 8386, из которых 281 не являются настоящими слогами (содержат более 1 гласной), поэтому и не использовались при разбиении слов. Полученные слоги в дальнейшем были использованы как выходные классы для СТС. Для каждого временного отсчета сеть предсказывала распределение по этим классам (+ пустой символ).

4.4 Базовый эксперимент: буквенное СТС без сверточек

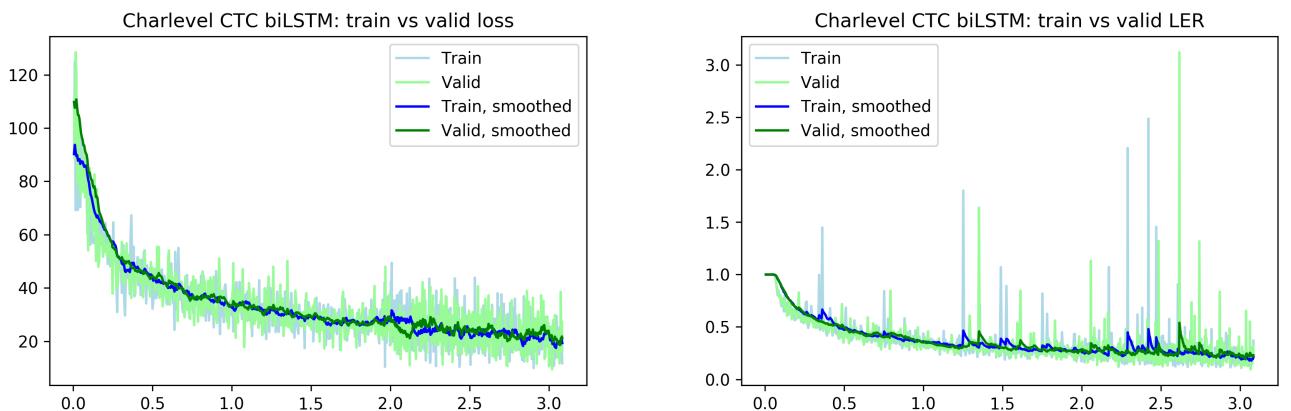


Рис. 8: Char CTC biLSTM training

В качестве базового эксперимента, с которым мы сравнивали нашу слоговую модель, выступала буквенная акустическая модель. Архитектура модели: пятислойная bidirectional LSTM с CTC-loss. Использовалась именно буквенная модель, так как целью данной работы было про-

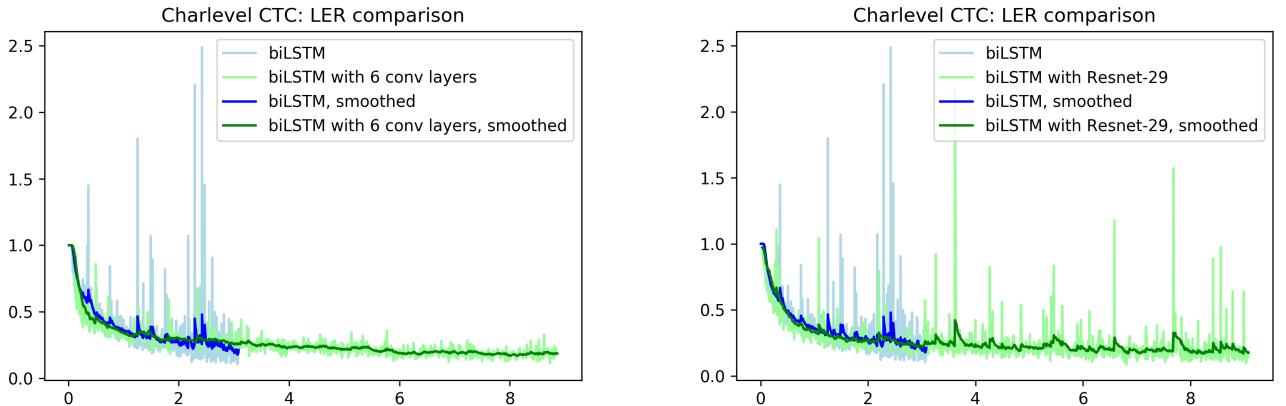


Рис. 9: Char CTC LER comparison

верить, как влияет изменение разбиения слов на меньшее количество подсловных элементов на качество распознавания. Также предполагалось, что разбиение на более сложные элементы позволит получить достаточно хорошее качество распознавания и без использования языковой модели на этапе декодирования.

Тренировалась сеть с помощью AdamOptimizer с $\text{learning_rate} = 10^{-4}$. Размер батча был равен 64.

После декодирования с Eesen decoder наилучший средний WER на 500 фразах был 25.98% с языковой моделью, основанной на чатах и 22.12% с моделью, основанной на транскрипциях.

Также нужно отметить, что чисто рекуррентные модели обучаются гораздо медленнее, чем сверточные, поэтому на графиках можно заметить, что рекуррентные обучались меньше количества эпох. На самом деле обучение рекуррентных сетей было даже дольше по времени, чем обучение сверточных, а слоговая версия рекуррентной модели была самой медленной.

4.5 Слогоное СТС без сверток

В этом эксперименте использовалась архитектура предыдущего, базового эксперимента, только для предсказания слогов.

Тренировалась сеть с помощью AdamOptimizer с $\text{learning_rate} = 10^{-4}$. Размер батча был равен 20.

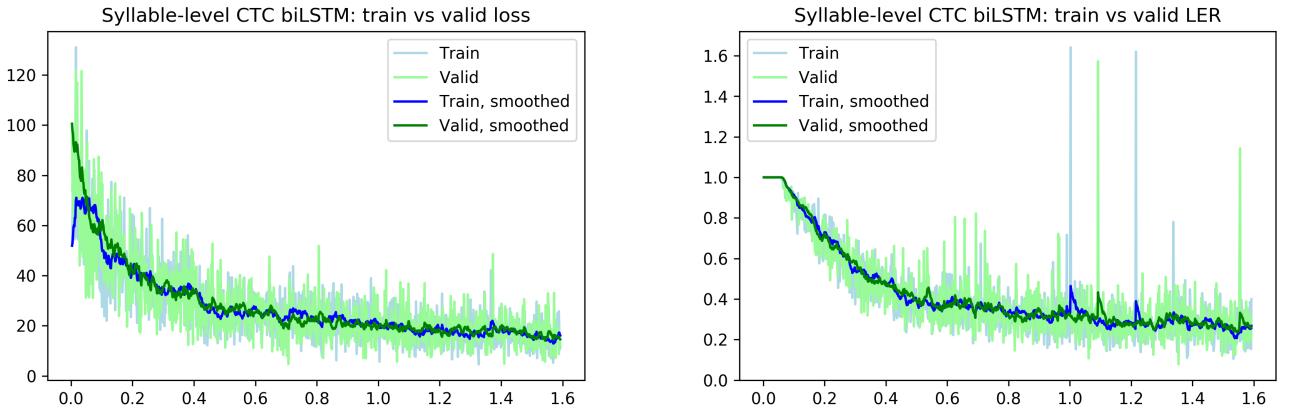


Рис. 10: Syllable CTC biLSTM training

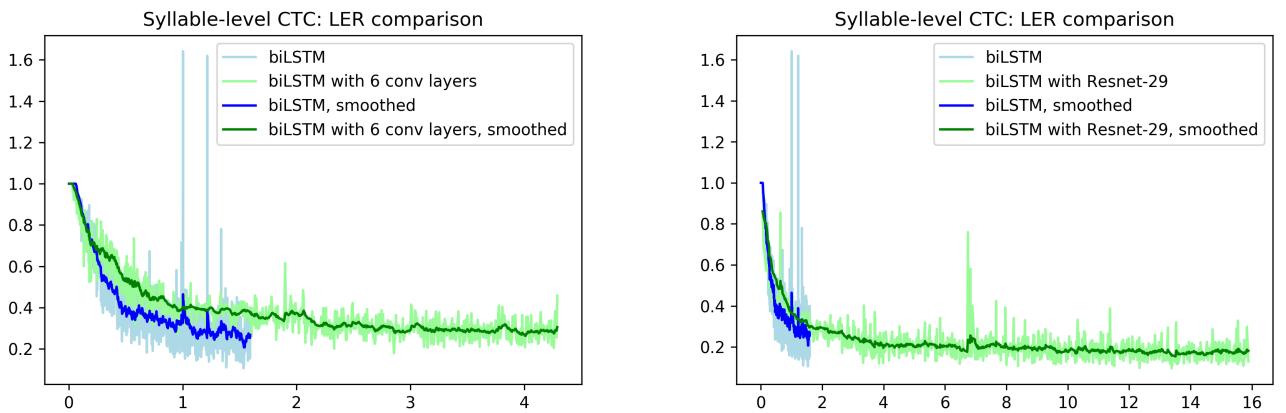


Рис. 11: Syllable CTC LER comparison

После декодирования с Eesen decoder наилучший средний *WER* на 500 фразах был 31% с языковой моделью, полученной из чатов, и 28.89% с языковой моделью, полученной из транскрипций.

4.6 Буквенное СТС со свертками

В этом эксперименте сеть состояла из 6 сверточных слоев и 3 слоев bidirectional LSTM. Параметры сверточных слоев: длина свертки была 17 для всех слоев, количество фильтров 256, 256, 256, 512, 512, 512 и страйды 1, 1, 2, 1, 1, 2 для слоев 1 - 6 соответственно. Размер скрытого слоя у LSTM был 512. Эта архитектура показана на рисунке 5

Тренировалась сеть с помощью AdamOptimizer с learning_rate 10^{-4} с уменьшением до 10^{-5} . Размер батча был равен 64.

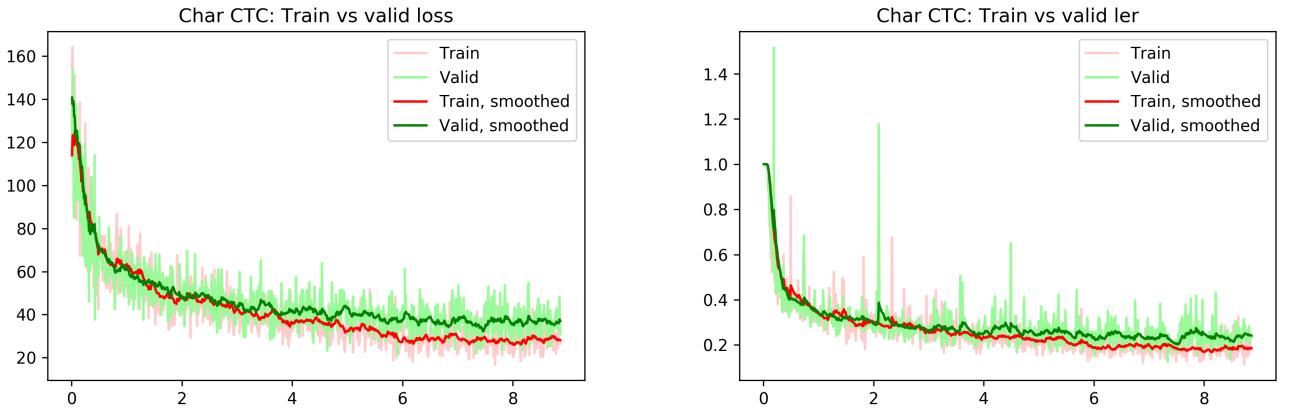


Рис. 12: Char CTC training

Для декодирования была сгенерирована bigram language model с помощью SRILM. После декодирования с Eesen decoder наилучший средний *WER* на 500 фразах был 30.5%.

4.7 Слогоное СТС со свертками

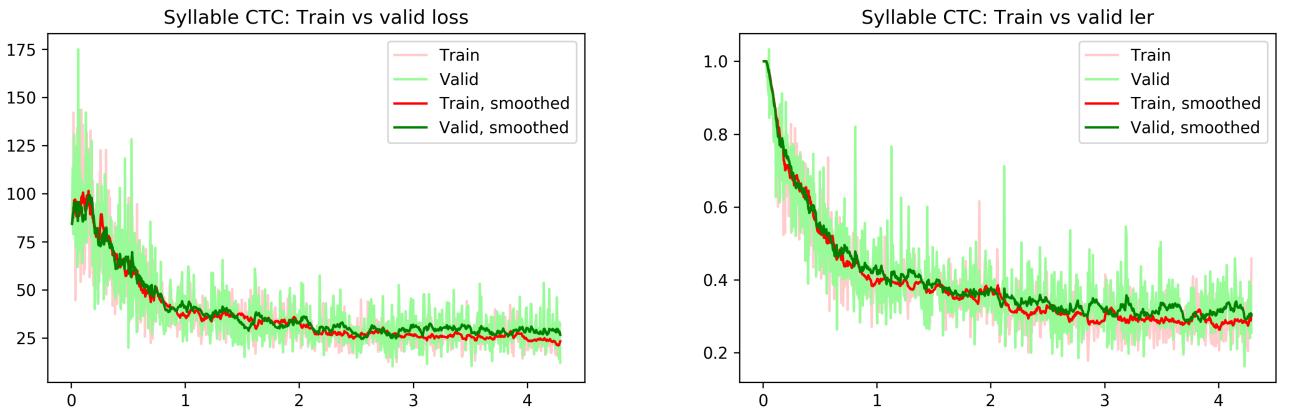


Рис. 13: Syllable CTC training

В этом эксперименте была произведена попытка применить архитектуру из предыдущего эксперимента для слогов. После декодирования с Eesen decoder наилучший средний *WER* на 500 фразах получился 41%. Однако по графикам тренировки (12 и 13) видно, что слоговая модель меньше переобучается, чем буквенная.

После этого эксперимента было замечено, что в датасете очень много ошибок в транскрипциях. После этого была проведена фильтрация

датасета от редко встречающихся слов (например, написанных с ошибками). Затем датасет был заново разбит на слоги. Их количество при этом уменьшилось до примерно 6000. В последующих экспериментах использовался уже очищенный датасет.

4.8 Буквенное CTC с Resnet-29

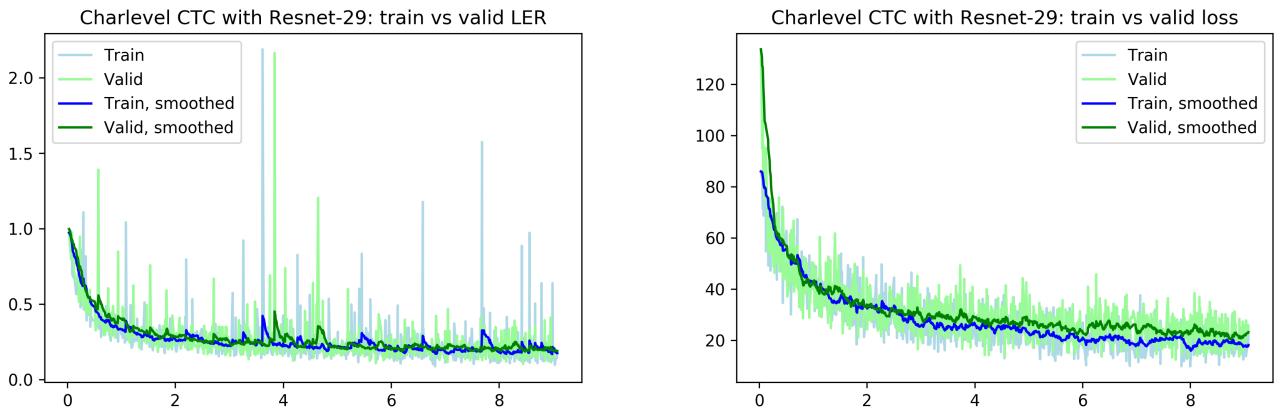


Рис. 14: Charlevel CTC with Resnet-29 training

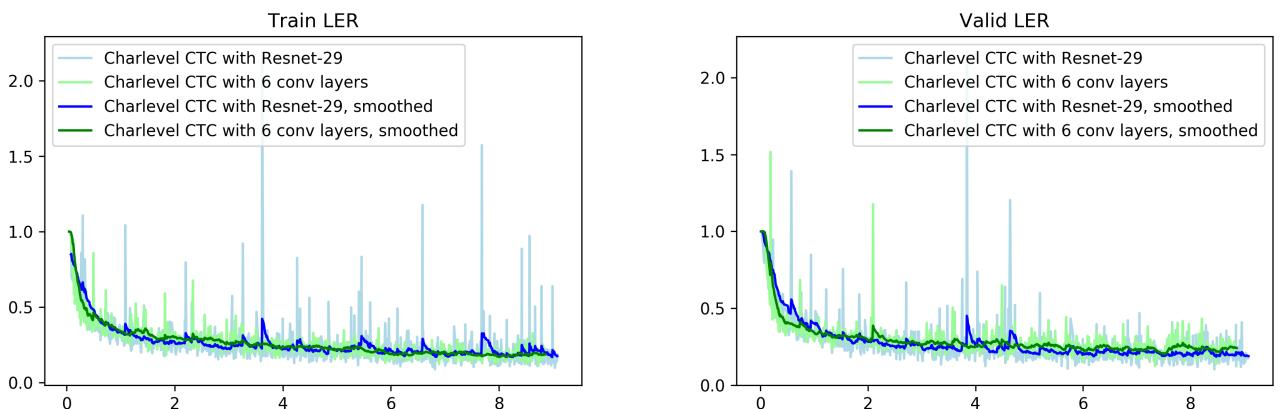


Рис. 15: Charlevel CTC with 6 conv layers vs Charlevel CTC with Resnet-29 comparison

В этом эксперименте была произведена попытка объединить трехслойный LSTM с размером скрытого слоя 512 с Resnet-29 [9] [14] на первых слоях. Эта архитектура показана на рисунке 6.

Тренировалась сеть с помощью AdamOptimizer с learning_rate 10^{-4} с уменьшением до $5 \cdot 10^{-5}$. Размер батча был равен 64.

После декодирования с Eesen decoder наилучший средний *WER* на 500 фразах был 25.6% с моделью, основанной на чатах, и 23.6 с моделью, основанной на транскрипциях.

4.9 Слоговое CTC с Resnet-29

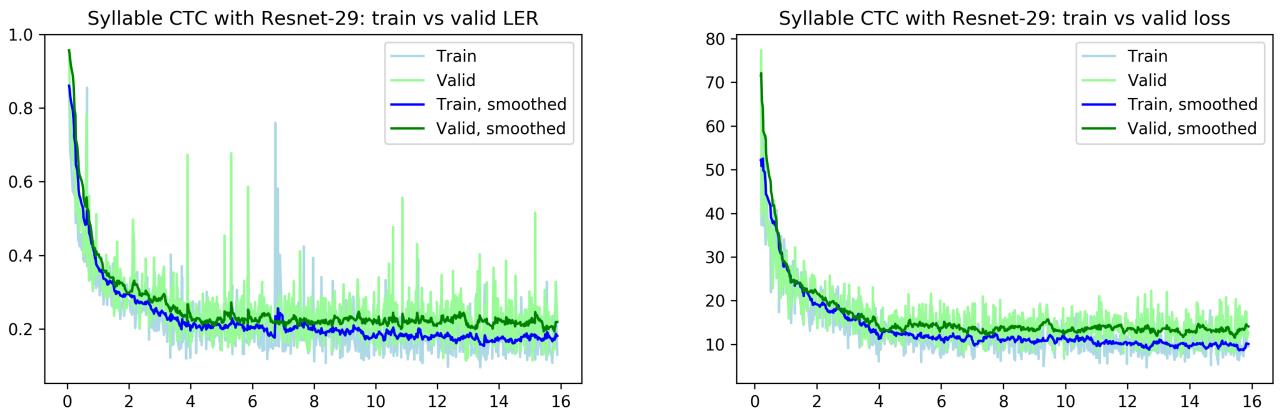


Рис. 16: Syllable CTC with Resnet-29 training

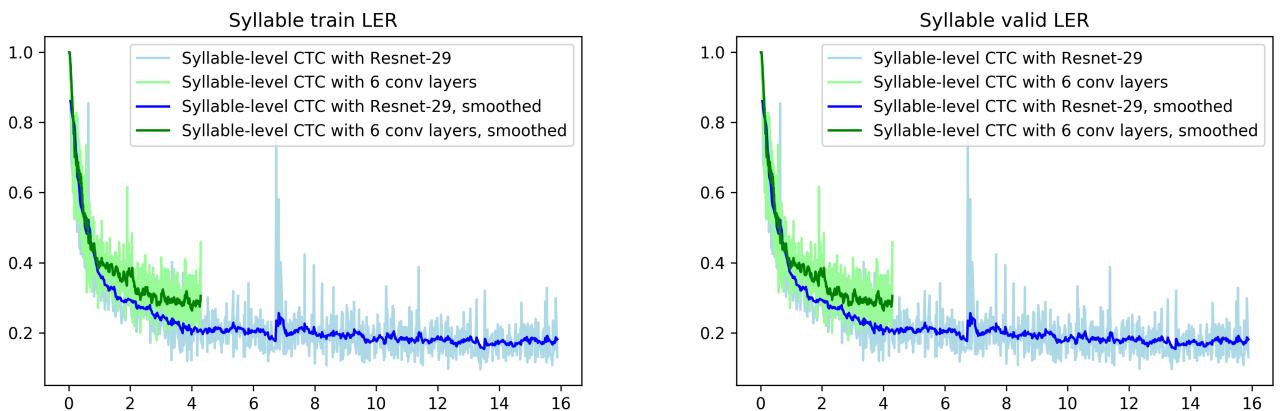


Рис. 17: Syllable CTC with 6 conv layers vs Syllable CTC with Resnet-29 comparison

Этот эксперимент был проведен для сравнения с буквенной моделью из предыдущего эксперимента. Была использована та же архитектура, что и в том эксперименте, только для слоговой модели.

Тренировалась сеть с помощью AdamOptimizer с learning_rate 10^{-4} с уменьшением до $5 \cdot 10^{-5}$. Размер батча был равен 64.

После декодирования с Eesen decoder наилучший средний *WER* на 500 фразах был 27.19% с использованием языковой модели, основанной на чатах и 26.38% с использованием модели, основанной на транскрипциях.

Можно заметить по графикам LER, что добавка Resnet дала большее уменьшение LER для слоговых моделей, чем для буквенных, по сравнению с моделями из экспериментов 5 и 4 соответственно.

4.10 СТС со свертками для пар букв

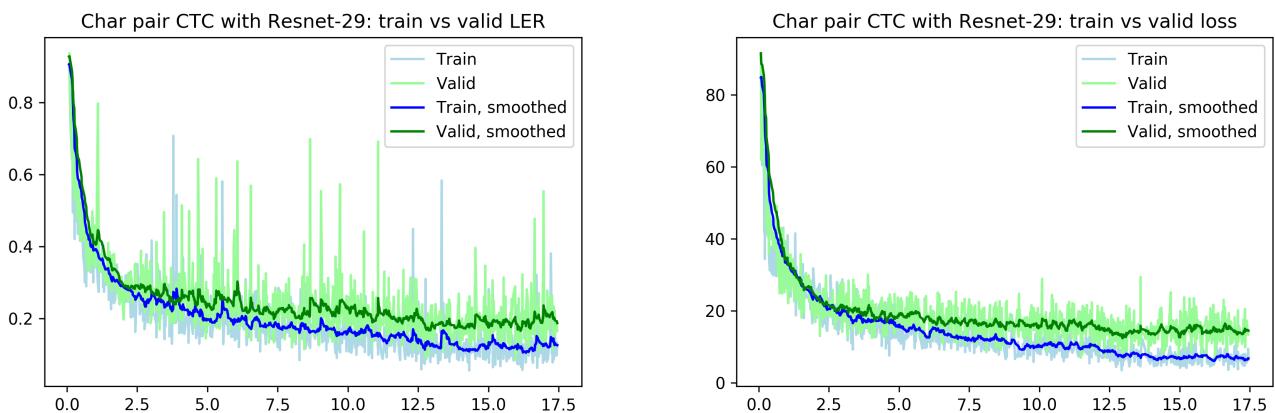


Рис. 18: Char pair CTC with Resnet-29 training

После экспериментов со слогами, которые не давали прироста качества по сравнению с буквенными моделями, было решено попробовать другие разбиения на подслова. В этом эксперименте вместо букв или слогов модель предсказывала пары букв или отдельные буквы(чтобы разбивать слова нечетной длины), похожим образом как описано в статье Deep Speech 2 [15]. Архитектура у сети была такая же, как в предыдущих двух экспериментах.

Тренировалась сеть с помощью AdamOptimizer с learning_rate 10^{-4} . Размер батча был равен 64.

После декодирования с Eesen decoder наилучший средний *WER* на 500 фразах был 24.68% с языковой моделью, использующей чаты, и 23.9% с моделью, использующей транскрипции.

WER при декодировании с первой языковой моделью в этой архитектуре оказался лучше, чем в такой же архитектуре для букв и слогов. Есть предположение, что это работает хорошо, потому что все основные слоги двухбуквенные или трехбуквенные. Поэтому нейросети легче выучить соответствие между кусочками записи и этими парами, чем сложными пятибуквенными слогами.

4.11 СТС со свертками для ВРЕ

После удачного эксперимента с парами букв возникла идея попробовать другие разбиения слов на подслова. В качестве такого разбиения было взято Byte Pair Encoding(BPE). ВРЕ изначально было придумано как алгоритм сжатия: самые часто встречающиеся пары символов объединялись и заменялись на 1 символ, затем находилась следующая по частоте встречаемости пара и так далее. Соответствия между парами и символами также записывалось в файл. Сейчас этот алгоритм уже не применяется для сжатия, но идея ВРЕ нашла свое применение в машинном переводе. Там она используется для того, чтобы переводить редкие слова, которых не было в обучающей выборке, путем разбиения их на подслова с помощью ВРЕ [16]. Для использования ВРЕ для разбиения транскрипций была взята готовая реализация с Github: rsennrich/subword-nmt.

Архитектура сети была такая же, как и в предыдущих трех экспериментах.

Тренировалась сеть с помощью AdamOptimizer с learning_rate 10^{-4} с уменьшением до $5 \cdot 10^{-5}$. Размер батча был равен 64.

После декодирования с Eesen decoder наилучший средний WER на 500 фразах был 53.02% с языковой моделью, основанной на чатах и 51.23% с языковой моделью, основанной на транскрипциях.

Как видно результат ВРЕ хуже по качеству, чем все проведенные эксперименты. Наверное это объясняется тем, что части, на которые делит слова ВРЕ отличаются от "естественных" частей, на которые мы делим слова при их произношении, поэтому это не совсем правильно использовать ВРЕ для разбиения транскрипций. К слову пары букв тоже не

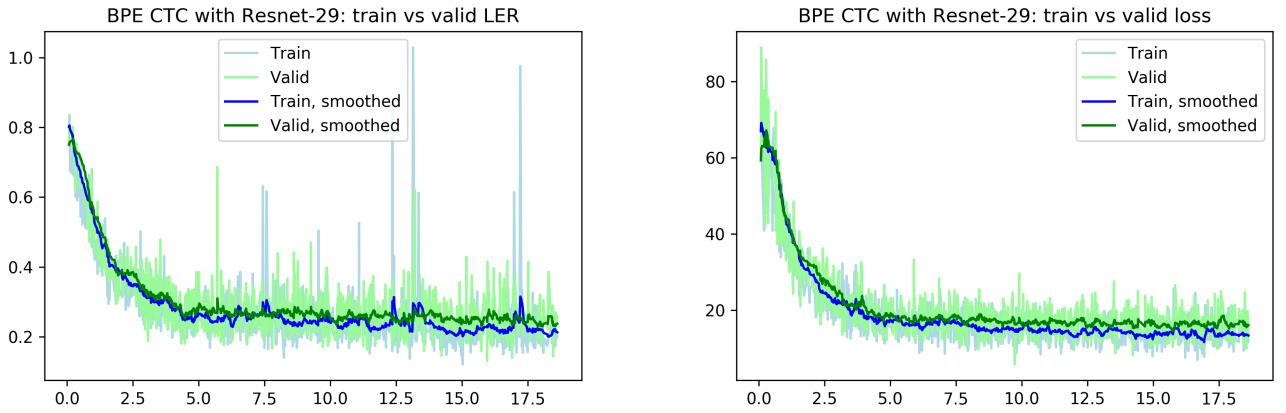


Рис. 19: BPE CTC with Resnet-29 training

совсем естественное разбиение слова, но оно больше похоже на слоги. Этому был посвящен следующий эксперимент.

4.12 СТС со свертками для двухбуквенных и трехбуквенных слогов

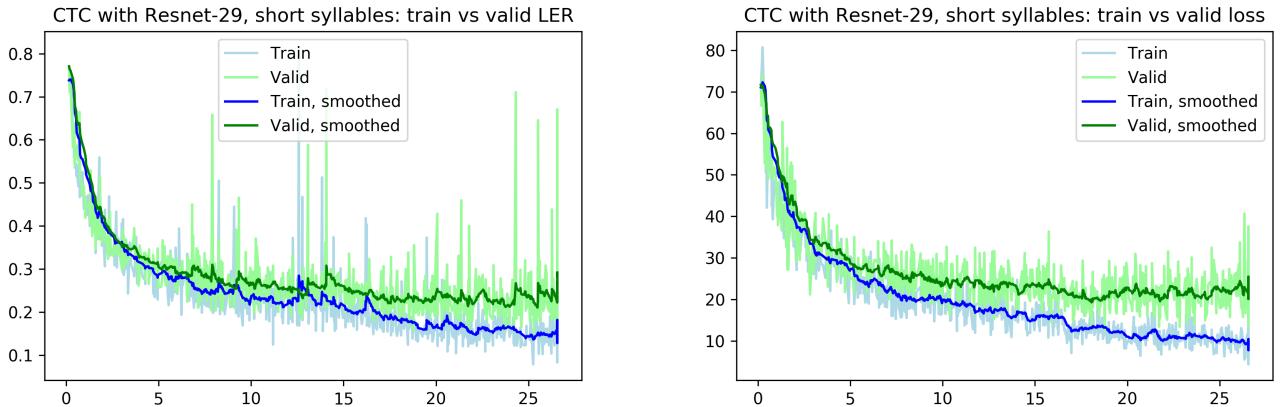


Рис. 20: CTC with Resnet-29 (short syllables) training

Вдохновившись успехом эксперимента с парами букв, было решено оставить среди слогов только короткие, то есть двухбуквенные и трехбуквенные, а более длинные слоги разбить на отдельные буквы, таким образом у нас слова будут разбиваться не только на слоги, но и на буквы.

Архитектура сети была такая же, как и в предыдущих трех экспериментах. Тренировалась сеть с помощью AdamOptimizer с learning_rate 10^{-4} с уменьшением до $5 \cdot 10^{-5}$. Размер батча был равен 64.

После декодирования с Eesen decoder средний *WER* на 500 фразах был 27.38% с языковой моделью, основанной на чатах и 26.87% с языковой моделью, основанной на транскрипциях.

4.13 Слогоное СТС с более широкими страйдами

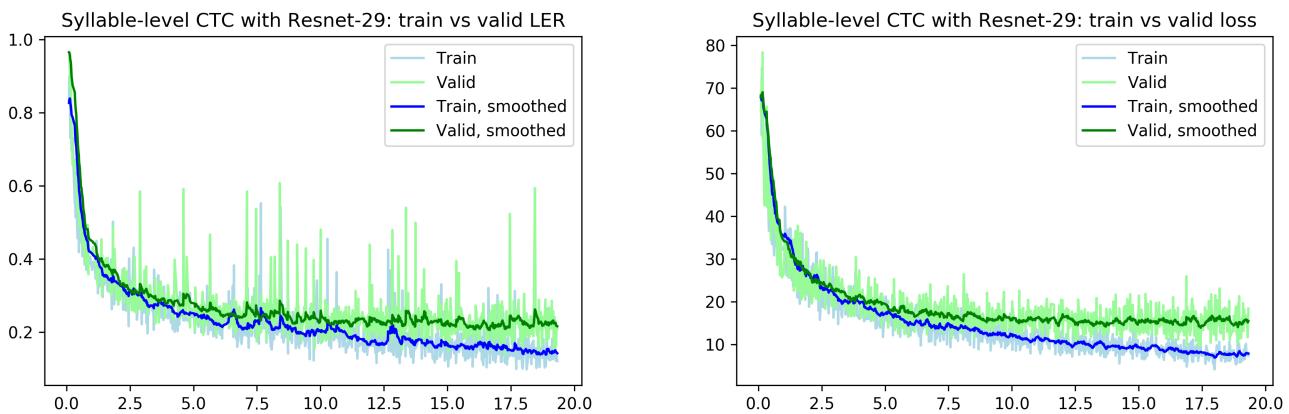


Рис. 21: Syllable CTC with Resnet-29 (big strides) training

После экспериментов со сверточными сетями было замечено, что они обучаются гораздо быстрее чистых LSTM, поэтому чтобы еще сильнее ускорить обучение было решено увеличить размер суммарных страйдов в модели с Resnet-29.

Если увеличивать страйды, то размер входа для LSTM становится меньше, меньше возможностей у градиентов чтобы взрываться, обучение становится стабильнее и быстрее. Также быстрее считается СТС-loss. Однако сильно увеличивать страйды для буквенных моделей не получается, поскольку тогда длина входной последовательности становится меньше, чем размер транскрипции в токенах, поэтому нельзя посчитать СТС-loss.

Слогоевые же модели позволяют сильнее увеличивать страйды, поскольку длина транскрипции в слогах сильно меньше, чем в буквах. Для того чтобы увеличить страйды нужно в модели Resnet-29, которая показана на рисунке 6 убрать шаг upscale, который увеличивает в 2 раза размер вектора. Тогда получается, что модель имеет суммарный страйд

8, по сравнению с предыдущими моделями, использующими эту архитектуру, которые имеют суммарный страйд 4.

Как уже было сказано, в этом эксперименте использовалась Resnet-29 модель без шага upscale. Тренировалась сеть с помощью AdamOptimizer с learning_rate 10^{-4} . Размер батча был равен 64. После декодирования с Eesen decoder средний *WER* на 500 фразах был 24.92% с языковой моделью, основанной на чатах и 23.75% с языковой моделью, основанной на транскрипциях. Данная модель превосходит все буквенные модели, натренированные до этого.

4.14 Результаты экспериментов

В таблице ниже приведены результаты всех проделанных экспериментов. Acoustic scale указан для наилучшего эксперимента, в котором и достигается указанный WER. Результат не сильно зависел от величины beam, в этих экспериментах она равна 20.

Признаки	Модель	Language Model	acoustic_scale	WER, %
Пары букв	resnet-29	чаты	3.25	24.68
Пары букв	resnet-29	транскрипции	2.0	23.9
Буквы	resnet-29	чаты	2.0	25.6
Буквы	resnet-29	транскрипции	1.5	23.6
Буквы	5 bilstm	чаты	2.0	25.98
Буквы	5 bilstm	транскрипции	1.0	22.12
Буквы	6 conv, 3 bilstm	транскрипции	1.0	30.5
Слоги	6 conv, 3 bilstm	транскрипции	1.0	40.1
Слоги	resnet-29	чаты	2.85	27.19
Слоги	resnet-29	транскрипции	2.5	26.38
Слоги	5 bilstm	чаты	3	31
Слоги	5 bilstm	транскрипции	2.5	28.89
Слоги	resnet-29 big strides	чаты	3.25	24.92
Слоги	resnet-29 big strides	транскрипции	2.0	23.75
Короткие слоги	resnet-29 big strides	чаты	2.5	27.38
Короткие слоги	resnet-29 big strides	транскрипции	2	26.87
BPE	resnet-29	чаты	3	53.02
BPE	resnet-29	транскрипции	3.5	51.23

5 Заключение

5.1 Итоги работы

В данной работе представлен способ построения слоговых акустических моделей на основе рекуррентных и сверточных сетей с СTC-loss.

Итак, подведем итог проделанной работы:

1. Была построена модель на основе рекуррентной нейросети, которая разбивает слова на слоги с достаточно хорошим качеством. Она была применена для получения слогового датасета из имеющихся транскрипций.
2. С использованием полученного слогового датасета были обучены акустические модели на основе рекуррентных сетей. Было проведено сравнение слоговых моделей с буквенными, которое выявило, что слоговые чисто рекуррентные модели работают хуже таких же буквенных.
3. Дальнейшие эксперименты со сверточными сетями показали что свертки дают больший прирост WER для слоговых моделей, чем для буквенных. В частности модель с широкими страйдами превзошла все тестировавшиеся буквенные модели. Также в случае слоговых моделей возможностей больше возможностей для ускорения из-за маленькой длины транскрипции
4. Были проведены эксперименты с акустическими моделями, в которых слова в транскрипции разбивались на пары букв и с помощью Byte Pair Encoding. По результатам этих экспериментов было выявлено, что лучше всего работают такие разбиения, которые больше всего похожи на "естественные" разбиения, которые человек делает в речи.

5.2 Дальнейшие исследования

Рассмотрим возможные варианты дальнейших исследований в рамках данной работы:

1. Попробовать другие архитектуры сверточных слоев, поскольку в данной работе было рассмотрено только две.
2. Исследовать влияние параметров декодирования: параметров построения языковой модели, beam size и beam search, etc.
3. Попробовать использовать curriculum learning - способ обучения, когда модель одну эпоху обучаю сначала на более простых примерах, затем постепенно усложняют их. Затем тренировка идет в обычном режиме с перемешиванием выборки. Применительно к данной задаче "более простыми" примерами могут считаться более короткие записи, поскольку кажется, что в них легче выучить выравнивания, чем в более длинных записях. Таким образом, предлагаются в первую эпоху отсортировать обучающую выборку по длине и проверить как это повлияет на качество модели.
4. Попробовать придумать другие разбиения слов на под слова. Изучить акустические модели на основе этих под слов.

6 Список литературы

1. M. Mohri, F. Pereira, and M. Riley, “Weighted finite-state transducers in speech recognition,” *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002. [Online]. Available: <http://www.cs.nyu.edu/~mohri/pub/csl01.pdf>
2. L. Rabiner and B. Juang, “An introduction to hidden markov models,” *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.957.202&rep=rep1&type=pdf>
3. A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks.” in *ICML*, vol. 14, 2014, pp. 1764–1772. [Online]. Available: <http://jmlr.org/proceedings/papers/v32/graves14.pdf>
4. C. Olah, “Understanding lstm networks,” 2015. [Online]. Available: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
5. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
6. K. Cho, B. van Merriënboer, Ç. Gülcəhre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: <http://arxiv.org/abs/1406.1078>
7. A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376. [Online]. Available: http://www.cs.toronto.edu/~graves/icml_2006.pdf
8. R. Collobert, C. Puhrsch, and G. Synnaeve, “Wav2letter: an end-to-end convnet-based speech recognition system,” *CoRR*, vol. abs/1609.03193, 2016. [Online]. Available: <http://arxiv.org/abs/1609.03193>

9. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: <http://arxiv.org/abs/1512.03385>
10. Y. Miao, M. Gowayyed, and F. Metze, “EESEN: end-to-end speech recognition using deep RNN models and wfst-based decoding,” *CoRR*, vol. abs/1507.08240, 2015. [Online]. Available: <http://arxiv.org/abs/1507.08240>
11. A. Stolcke *et al.*, “Srilm-an extensible language modeling toolkit.” [Online]. Available: <http://www-speech.sri.com/projects/srilm/papers/icslp2002-srilm.pdf>
12. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” *arXiv preprint arXiv:1603.04467*, 2016.
13. D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
14. W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, “The microsoft 2016 conversational speech recognition system,” *CoRR*, vol. abs/1609.03528, 2016. [Online]. Available: <http://arxiv.org/abs/1609.03528>
15. D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Y. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Y. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu, “Deep speech 2: End-to-end speech recognition in english and mandarin,” *CoRR*, vol. abs/1512.02595, 2015. [Online]. Available: <http://arxiv.org/abs/1512.02595>

16. R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” *CoRR*, vol. abs/1508.07909, 2015. [Online]. Available: <http://arxiv.org/abs/1508.07909>