

# Лекция : Автоматическое распознавание речи

Keywords: ASR, STFT, Mel, LAS, AED, CTC, RNN-T, CER, WER,  
MWER, SpecAugment.



# Содержание

Автоматическое распознавание речи. История.

Задача распознавания речи. Метрики качества.

Фичи для ASR. Преобразование Фурье.  
STFT. Log-Mel.

Модель Listen Attend and Spell (AED).

CTC-подход к задаче ASR(CTC).

RNN-transducer (RNN-T).

Твики для обучения.

Ссылки на материалы.

# Автоматическое распознавание речи. История.

• • • • •

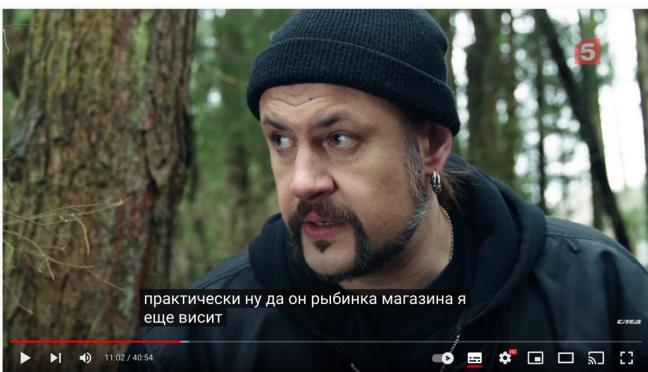
# Распознавания речи до машинного обучения.



- 1920 г. игрушка «Radio Rex» откликалась на свое имя и выходила из конуры, благодаря взаимодействия звуковой волны на механическую пружину, настроенную на 500 Hz(которая является 1-ой формантой звука [э']).
  - 1950-1960 г. IBM создает устройство под названием «Shoebox», способную распознавать числа и простые арифметические операции путем сравнения спектров приходящего аудио и заложенных в ее базу образцов.

# Распознавания речи в наши дни.

- Различные голосовые помощники: «Маруся», «Алиса», «Сбер», «Google Assistant», «Siri», «Alexa».
- Автоматическая транскрибация видеороликов.
- Автоматизированные операторы колл-центров: «Единый центр по борьбе с COVID-19», «Госуслуги», «Почта России».
- Персональные телефонные консьержи: «Алиса», «Олег».
- Системы доступа и идентификации: некоторые банки Китая, умные домофоны, и т.д.

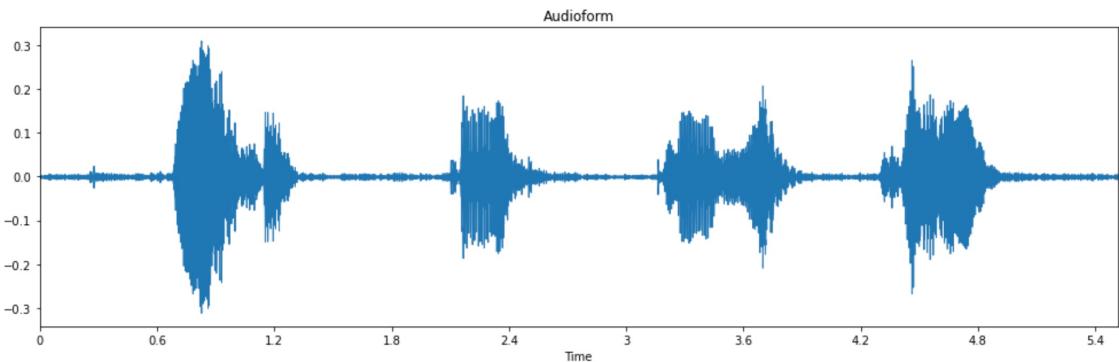


Google Ассистент

Задача  
распознавания  
речи. Метрики  
качества.

• • • • •

# Задача распознавания речи. Модальности.



ASR

1. Маруся, как твои дела ?
2. Маруся, где ты была ?
3. Маруся, куда ты ходила ?
4. Маруся, откуда ты

Размер используемого лексикона	Для кого предназначается речь диктора	Качество аудиосигнала		Особенности диктора			
Ограниченный лексикон. Примеры: распознавание цифр, ключевых фраз и т.д.	Открытым лексиконом. Примеры: разговор с голосовым ассистентом, транскрибация звонков и собраний.	Разговор с автоматизированной системой. В этом случае диктор, как правило, говорит с большими паузами, речь четкая.	Разговор между двумя людьми. Пример: транскрибация звонков, собраний, фильмов и т.д.	Аудиозаписи записанные в хороших помещениях с т.з. акустики с низким количеством эха и шумов.	Речь, записанная в ресторанах, автомобилях или шумных помещениях.	Диктор является носителем языка, отсутствуют проблемы с речью и произношением.	Диктор не является носителем языка или испытывает проблемы с речью, например: детская речь отличается от взрослой.

# Подзадачи распознавания речи.

Дополнительные подзадачи в распознавании речи:

- *Диаризация речи.* Разделение транскрипции между различными дикторами по принципу принадлежности.
- *Распознавание эмоций.* В этом смысле одни и те же слова в транскрипции могут иметь различные эмоциональные окрасы, из-за этого нести в себе разный смысл.
- *Распознавание голоса диктора.* Когда это представляется возможным.
- *Распознавание речи с проблемами произношения.* Как правило, данная задача выделяется в отдельный класс.
- *Аудио-визуальное распознавание речи.* Использование дополнительных сигналов для улучшения качества распознавания.

Так же различают модели распознавания речи по методу обработки входящего сигнала:

- **Оффлайн** – модели, которым для получения финальной транскрипции требуется полная аудиодорожка;
- **Потоковые** (streaming) – модели, которые могут обрабатывать речь «на-лету».

# Метрики качества в задаче распознавания речи.

CER (Character Error Rate):

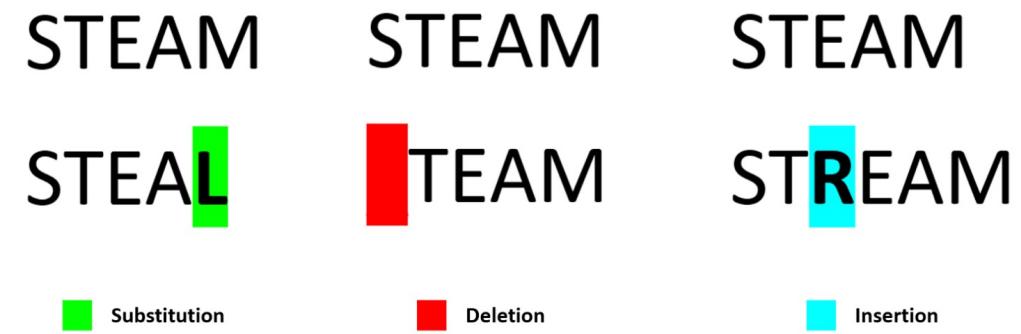
$$CER = \frac{S + D + I}{N}.$$

S (substitution) – кол-во «замен».

D (deletion) – кол-во «удалений».

I (insertion) – кол-во «вставок».

N – общее кол-во символов в правильной  
(ground truth) фразе.



Однако, проблема данной метрики в том, что она может превышать 1. Для устранения этого эффекта вводят нормализованную метрику (normalized CER):

$$nCER = \frac{S + D + I}{S + D + I + C}.$$

C (correct) – кол-во верных символов в транскрипции.

# Метрики качества в задаче распознавания речи.

WER (Word Error Rate), nWER (normalized Word Error Rate):

$$WER = \frac{S + D + I}{N}, \quad nWER = \frac{S + D + I}{S + D + I + C}.$$

S, D, I – кол-во «замен», «удалений», «вставок».

N – общее кол-во слов в правильной(ground truth) фразе.

normalized WER:

C (correct) – кол-во верных слов в транскрипции.

---

REF:	i *** ** UM the PHONE IS	i LEFT THE portable **** PHONE UPSTAIRS last night
HYP:	i GOT IT TO the ***** FULLEST i LOVE TO portable FORM OF STORES last night	
Eval:	I      I    S      D      S      S      S      I      S      S	

---

$$WER = \frac{6(S) + 3(I) + 1(D)}{13} = 0.769, \quad nWER = \frac{6 + 3 + 1}{6 + 3 + 1 + 6} = 0.625$$

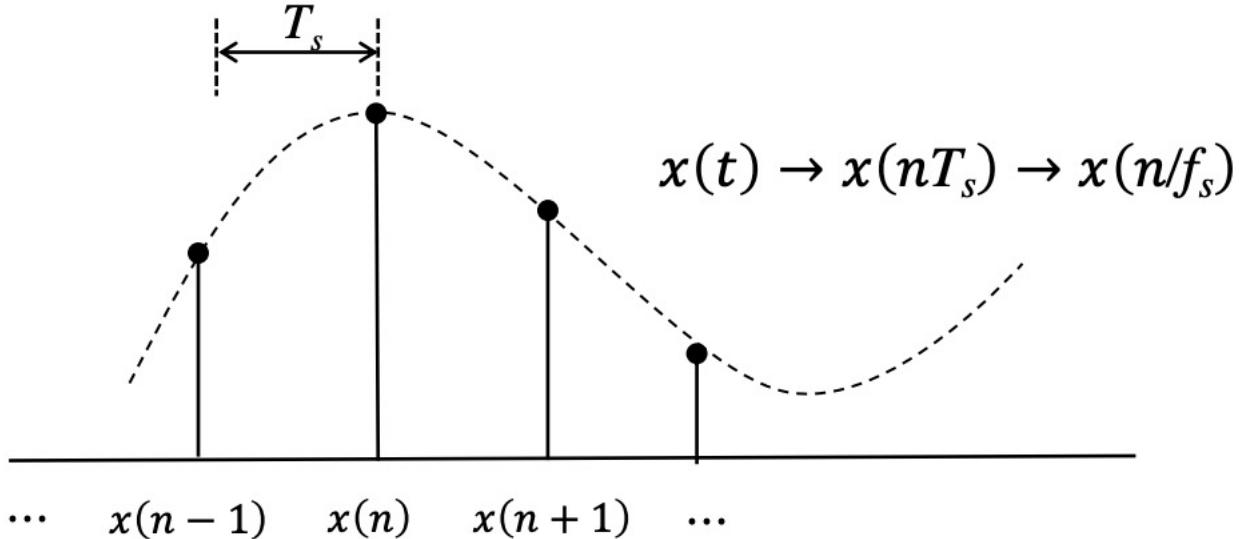
Фичи для ASR.  
Преобразование  
Фурье. STFT. Log-  
Mel.

• • • • •

# Дискретизация.

Дискретизация – процесс конвертации непрерывного сигнала в последовательность дискретных отсчетов путем выбора некоторых точек сигнала.

- Период сэмплирования( $T_s$ ) – кол-во времени между сэмлируемыми значениями.
- Частота сэмплирования( $f_s = \frac{1}{T_s}$ ) – количество сэмплируемых точек в секунду.

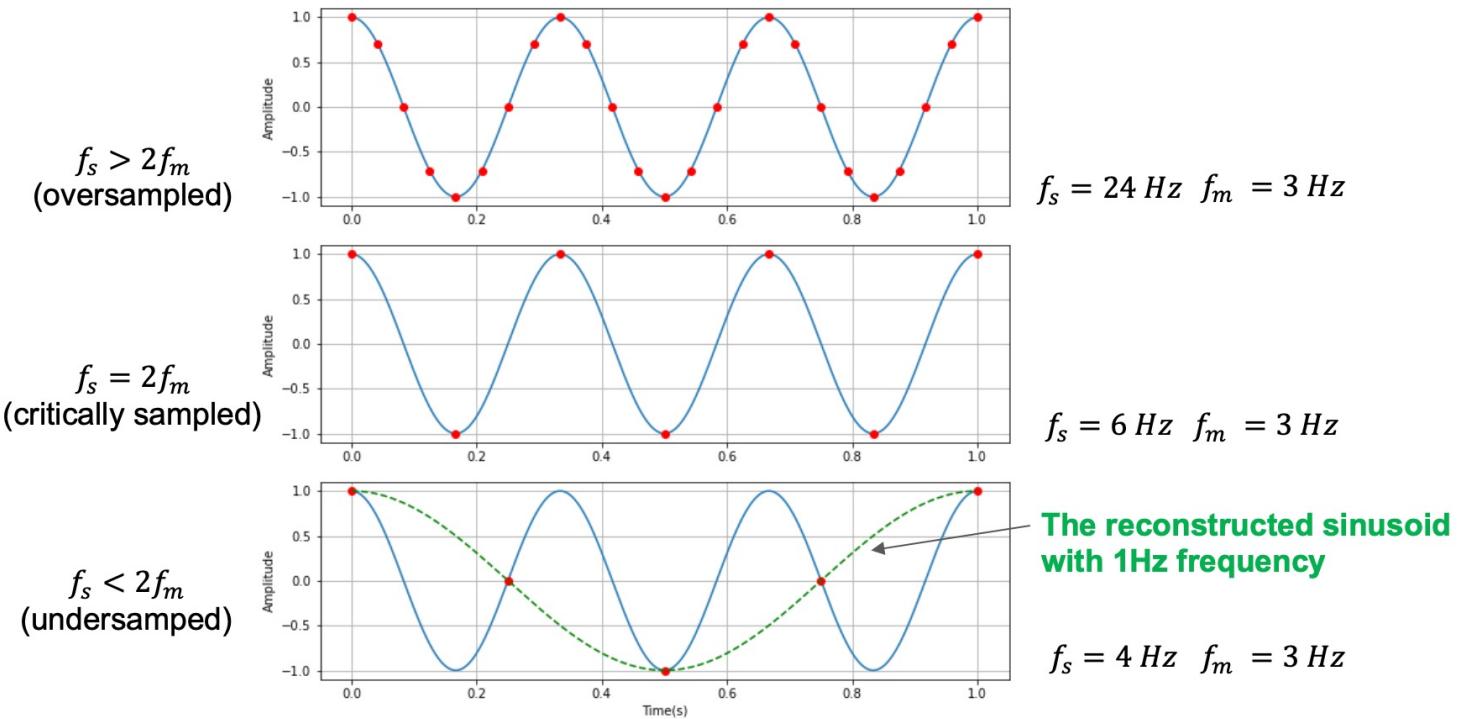


Как выбрать частоту сэмплирования?

- Слишком высокая – будем сэмплировать очень часто(много отсчетов, лишние данные).
- Слишком низкая – получим недостоверную информацию о сигнале.

# Дискретизация. Теорема отсчетов.

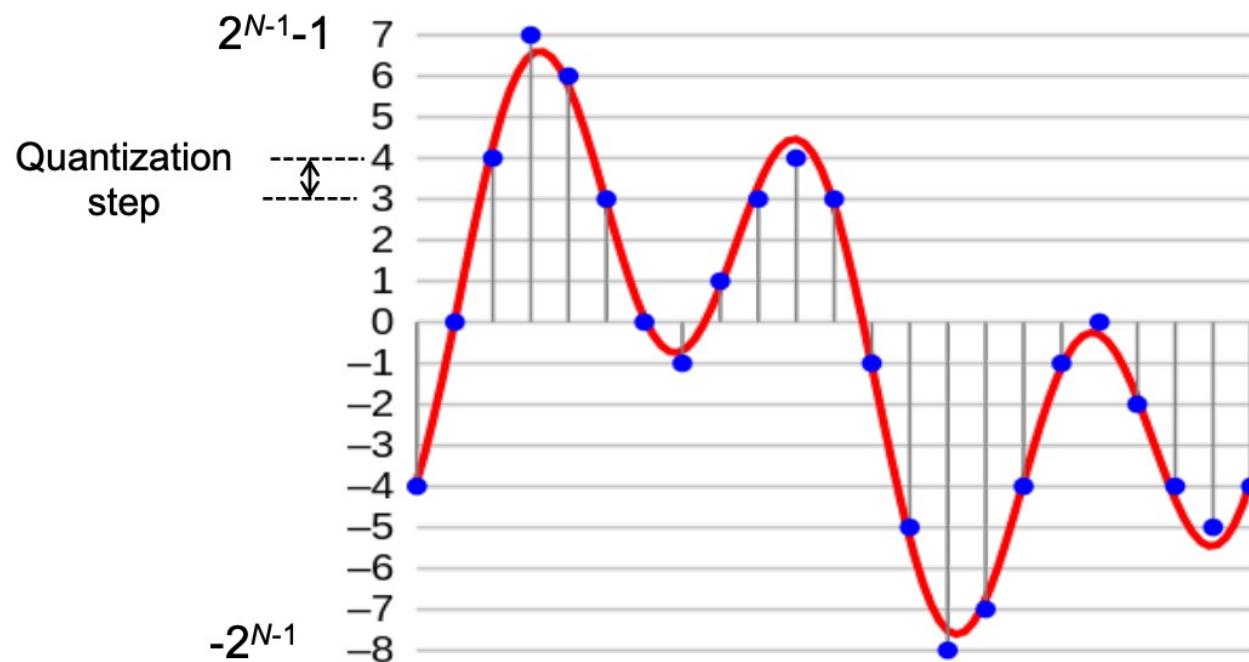
Теорема отсчетов (Котельникова-Найквиста-Шеннона) – для достоверного представления непрерывного сигнала, частота его дискретизации должна быть в два раза выше максимальной частоты имеющейся в сигнале:  $f_s > 2 * f_m$ .



# Квантизация.

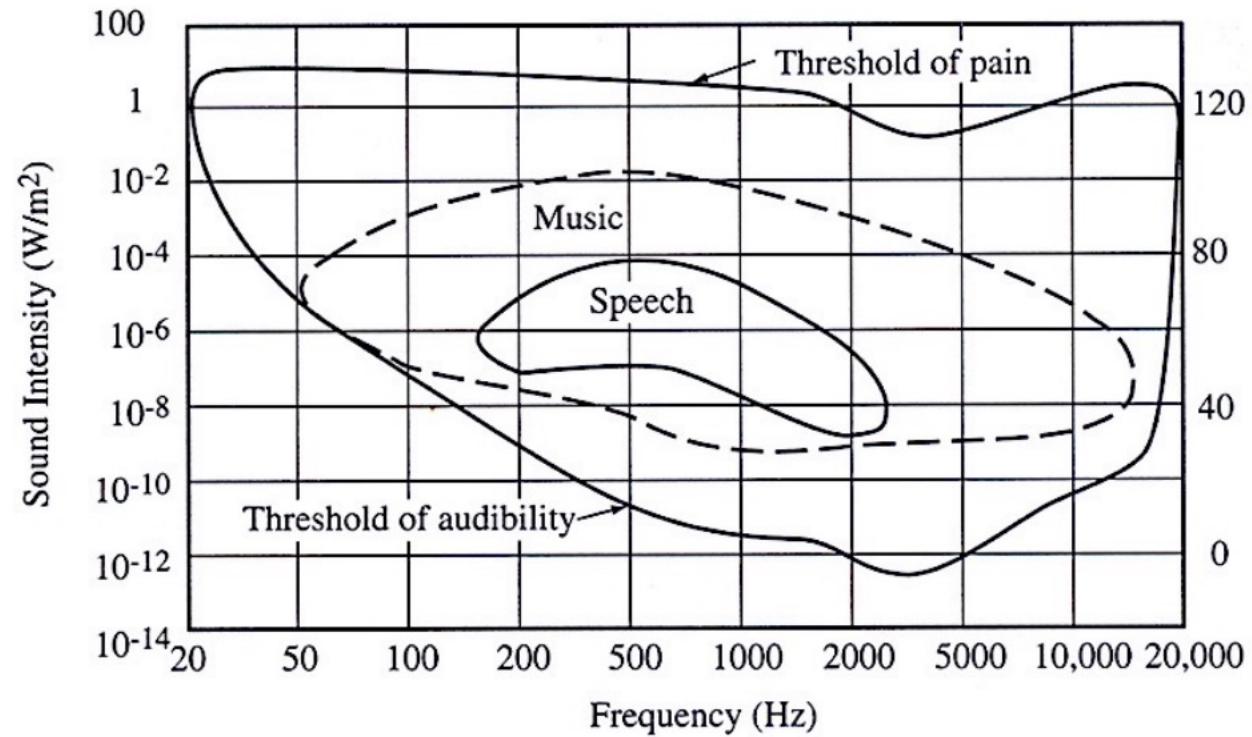
Квантизация – округление амплитуды непрерывного сигнала к ближайшим дискретным значениям.

- Количество дискретных значений определяется кол-вами бит типа данных.
- Тип данных, содержащий N бит, может представить значения от  $-2^{N-1}$  до  $2^{N-1} - 1$ : 8-битная величина от -128 до 127, 16-битная от -32767 до 32766.

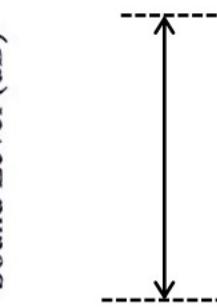


- Битовая «глубина» определяет динамический диапазон цифровых сигналов.
- Динамический диапазон:  
$$20 \log_{10} \left( \frac{\max\_value}{\min\_value} \right) \text{ db.}$$
  - 8 бит  $\approx 48 \text{ db}$ ;
  - 16 бит  $\approx 96 \text{ db}$ ;
  - N бит добавляет  $\sim 6N \text{ db}$ .

# Квантизация.



Sound Level (dB)



Music ≈ 80 dB

Covered by  
16 bits (96dB)

Speech ≈ 48 dB

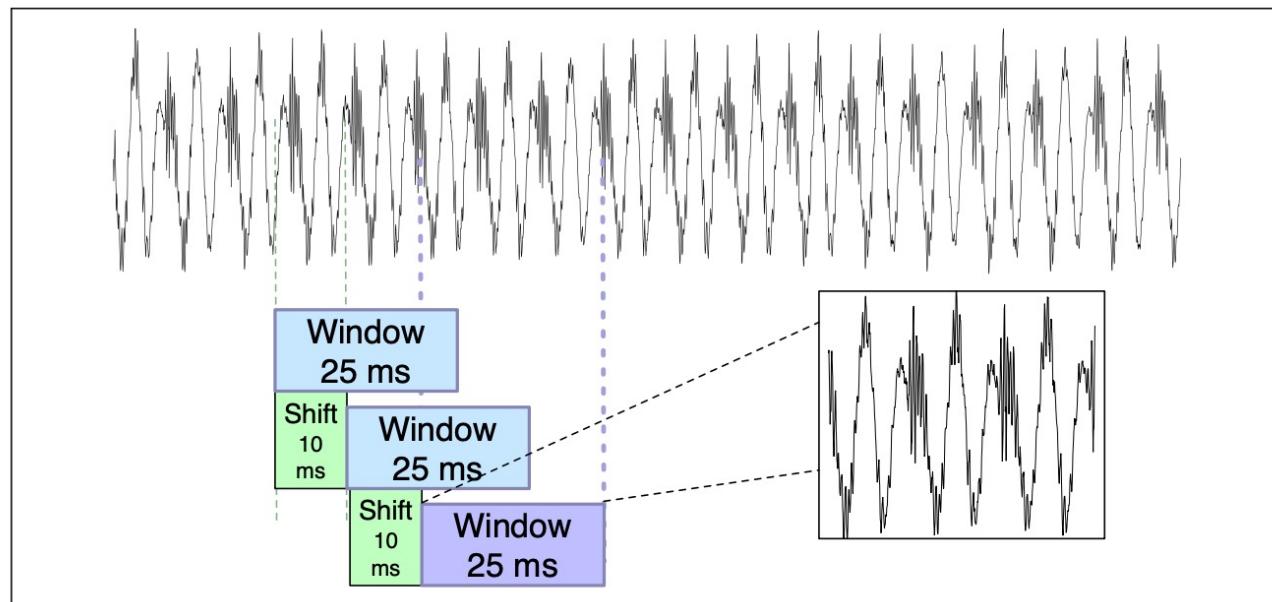
Covered by  
8 bits (48dB)

# Windowing.

Для удобства анализа дискретизированного, квантизованного сигнала удобно использовать не все аудио, а небольшие его части – **окна**. Поскольку небольшие части содержат полезную информацию о том, что в текущий момент «звучит». Так же в рамках небольшого окна мы можем считать, что сигнал является **стационарным**(т.е. статистические свойства сигнала являются постоянными). Речь, выделенная в каждом окне называется – (аудио)фреймом.

Основные параметры окон:

- Размер окна в отсчетах (**window size**, **window length**).
- Смещение между последовательными окнами (**offset**, **shift**, **hop length**).
- Форма сглаживающей функции окна(**window shape**).



**Figure 26.2** Windowing, showing a 25 ms rectangular window with a 10ms stride.

# Windowing.

Для получения фрейма аудио, необходимо умножить амплитуды сигнала внутри окна на значения сглаживающей оконной функции:  $y[n] = w[n]s[n]$ , где  $w[n]$ -сглаживающая ф-ия,  $s[n]$ -сигнал. В качестве сглаживающих оконных функций, как правило используют прямоугольные(*rectangular*), Хэмминга(*Hamming*) или Ханна(*Hann*).

Примеры окон:

$$\text{rectangular} \quad w[n] = \begin{cases} 1 & 0 \leq n \leq L-1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Hamming} \quad w[n] = \begin{cases} 0.54 - 0.46 \cos\left(\frac{2\pi n}{L}\right) & 0 \leq n \leq L-1 \\ 0 & \text{otherwise} \end{cases}$$

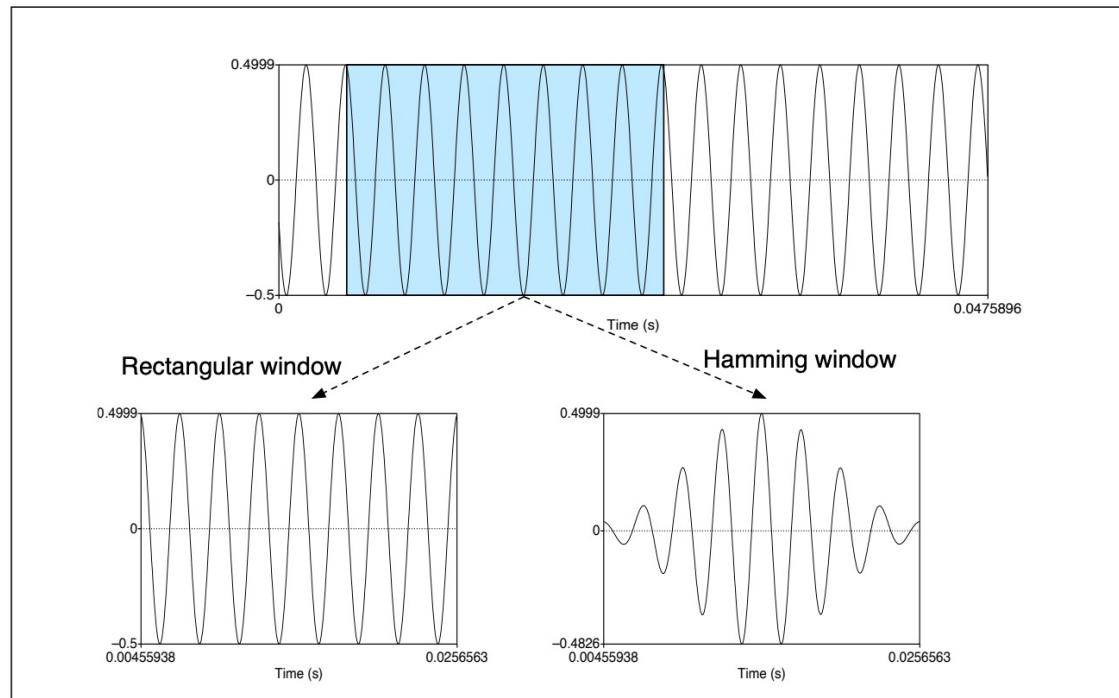
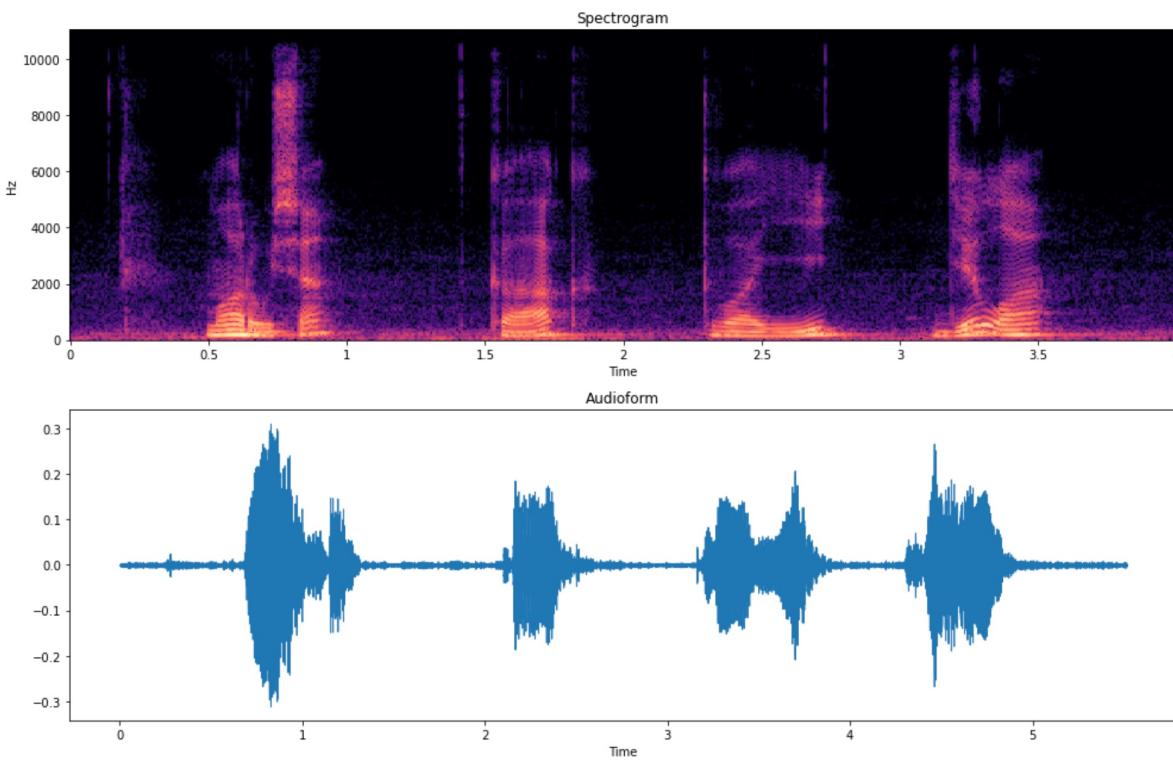


Figure 26.3 Windowing a sine wave with the rectangular or Hamming windows.

# Спектrogramма. STFT.

Для дальнейшего анализа фреймов аудио необходимо преобразовать временные отсчеты в соответствующие спектры при помощи DFT(Discrete Fourier Transform). Такая операция для всех окон носит название STFT(Short Time Fourier Transform). Спектrogramма, грубо говоря, отображает какое кол-во энергии приходится на каждый частотный интервал(бин).



Дискретное преобразование Фурье (DFT):

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-i\frac{2\pi}{N}kn} = X_{re}[k] + jX_{im}[k];$$

где  $k = (0, 1, \dots, N-1)$  – частотные бины.

Тогда амплитуда и фаза сигнала для  $k$ -го бина могут быть рассчитаны по формулам:

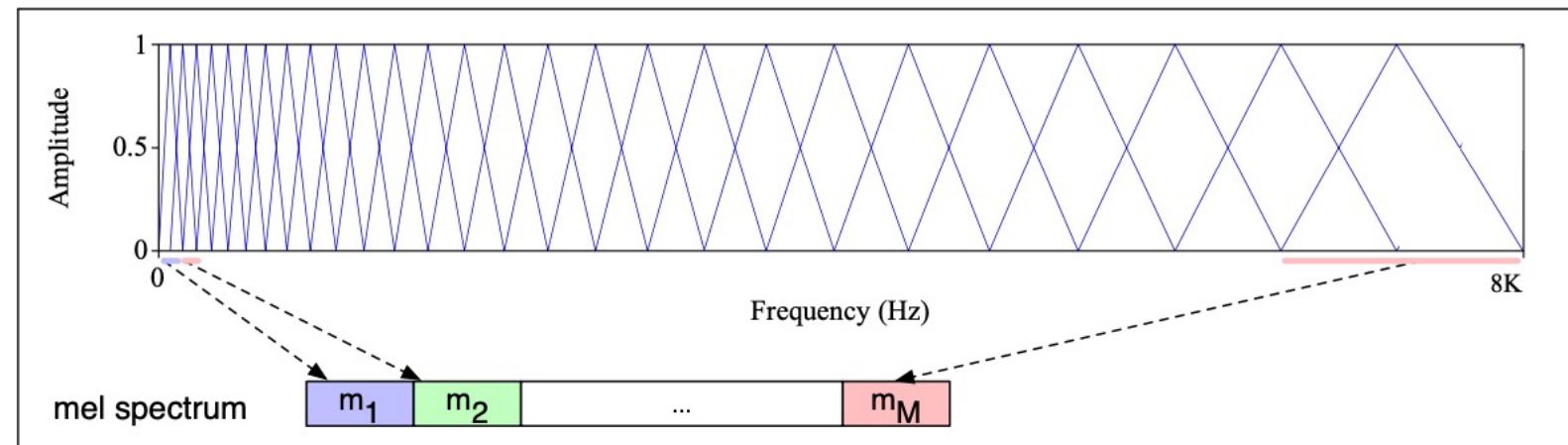
$$X_{mag}[k] = \sqrt{X_{re}^2[k] + X_{im}^2[k]};$$

$$X_{phase}[k] = \tan^{-1}\left(\frac{X_{im}[k]}{X_{re}[k]}\right).$$

# Log-Mel спектrogramma.

DFT дает нам знание об энергии аудиосигнала в каждом из частотных интервалов, однако человеческое ухо не одинаково чувствительно ко всем частотам. Оно имеет низкую чувствительность в области высоких частот, но более высокую в области низких. Для учета этой особенности слуха используется т.н. Mel-шкала (шкала слуховых частот). Мели рассчитываются по следующей формуле:  $mel(f) = 2595 \log_{10} \left( 1 + \frac{f}{700} \right)$ .

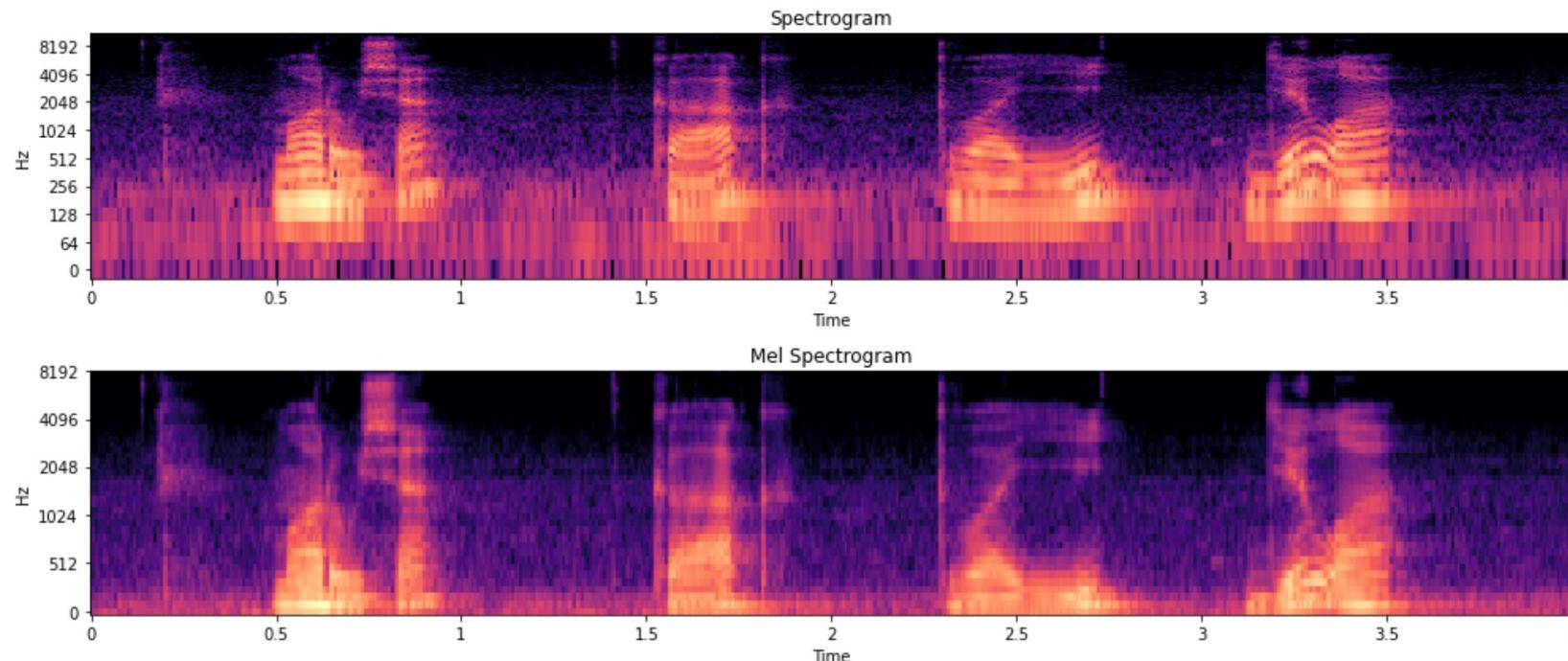
Для перевода спектrogramмы сигнала из Hz-области в область mel-ов, необходимо использовать т.н. Mel-filterbank – это по сути коэффициенты учета энергий сигнала в каждом из частотных бинов.



**Figure 26.5** The mel filter bank (Davis and Mermelstein, 1980). Each triangular filter, spaced logarithmically along the mel scale, collects energy from a given frequency range.

# Log-Mel спектrogramма.

Финально, получившуюся мел-спектрограмму мы логарифмируем(переводим в dB), поскольку человеческое ухо так же воспринимает амплитуду аудио-сигнала(как и его частоту) в логарифмической шкале. Такое преобразование позволяет нивелировать эффект изменения амплитуды сигнала, например, вследствие приближения/удаления диктора от микрофона.



Модель Listen  
Attend and Spell  
(AED).

.....

# Лексикон для построения модели ASR.

Одной из важнейших компонент при построении моделей ASR является лексикон на основе которого ведется распознавание. Его построение отталкивается от обучающего/тестового сета. В качестве лексикона мы можем использовать: простой(плоский) лексикон, BPE, Wordpiece, ULM, Sentencepiece, BPE-Dropout и т.д.

- **Простой лексикон:** включает в себя буквы, цифры, слова, SOS(start of sequence), EOS(end of sequence),  $\epsilon$ (пустой символ - blank), <UNK>(неизвестное слово) и т.д.
- **BPE – byte pair encoding.** Данный вид лексикона подразумевает объединение в пары, тройки и т.д. наиболее встречающихся символов(байт данных). Например:

<b>AABABCABBAABAC</b>	<b>ADDCDBADAC</b>	<b>EDCDBEAC</b>
<b>AA - 2</b>	<b>AD - 2</b>	<b>ED - 1</b>
<b>AB - 4</b>	<b>AD = E</b>	<b>DC - 1</b>
<b>BA - 3</b>		<b>CD - 1</b>
<b>BC - 1</b>		<b>DB - 1</b>
<b>CA - 1</b>		<b>DA - 1</b>
<b>BB - 1</b>		
<b>AC - 1</b>		<b>AC - 1</b>

# Лексикон для построения модели ASR.

- WordPiece – это BPE, в котором мы максимизируем правдоподобие, а не частоту встречаемости символов при слиянии:  $\max_{XY \in S} \frac{p(XY)}{p(X)*p(Y)}$ .
- ULM(Unigram Language Model) – метод основан на следующем подходе. Генерируем словарь большого размера, при помощи BPE, Wordpiece и т.д. Далее для каждого токена считаем на сколько уменьшится правдоподобие, если его выбросить из словаря. Далее выбрасываем все токены, которые не приводят к значительному ухудшению метрики.
- Sentencepiece – библиотека, реализующая метод ULM с дополнительными твиками и быстрым обучением словаря.
- BPE-Dropout – метод основанный на BPE, но при этом некоторые склейки символов пропускаются в процессе составления пар.

# Seq2Seq ASR. Listen Attend and Spell.

Рассмотрим модель для распознавания речи на основе подхода sequence-to-sequence.

Исходными данными для нашей модели является множество пар  $(X, Y)$ , где  $x_i$ -аудиозапись, представленная в виде отсчетов,  $y_i$ -соответствующая текстовая транскрипция.

Модель состоит из нескольких частей:

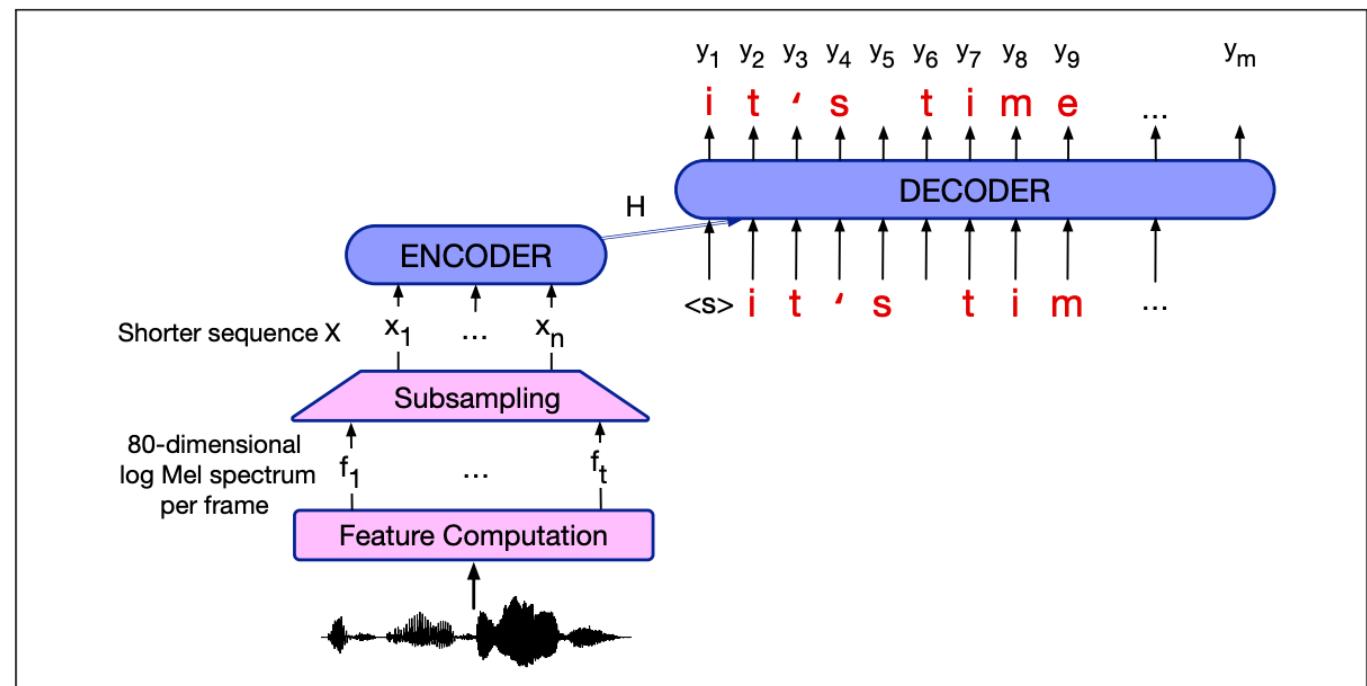
- Listener(Encoder);
- Speller(Decoder) + Attention.

Модель учится максимизировать правдоподобие правильной последовательности символов:

$$P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|\mathbf{x}, y_{<i})$$

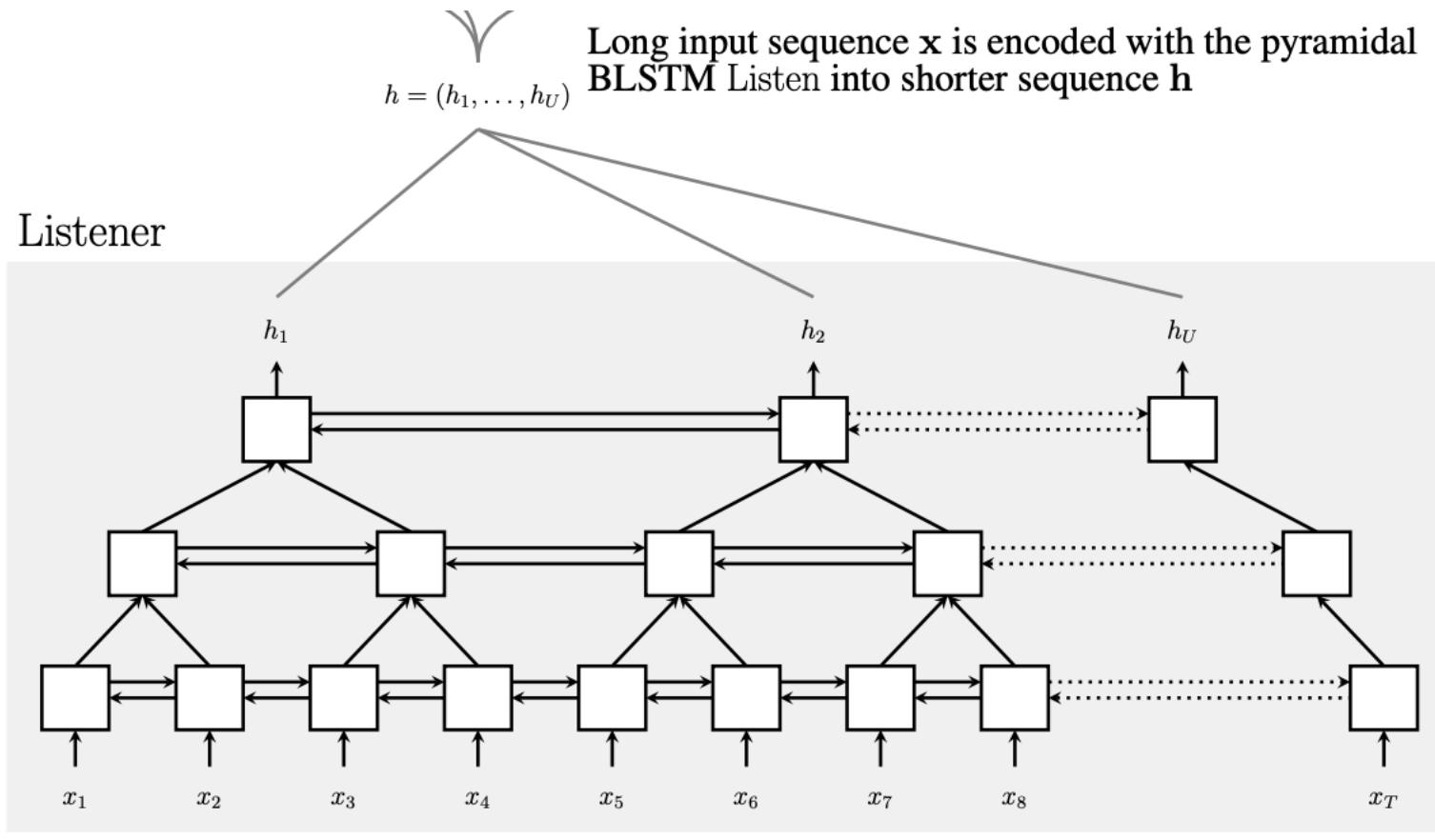
$\mathbf{h}$  = Listen( $\mathbf{x}$ )

$$P(\mathbf{y}|\mathbf{x}) = \text{AttendAndSpell}(\mathbf{h}, \mathbf{y})$$



**Figure 26.6** Schematic architecture for an encoder-decoder speech recognizer.

# Listen Attend and Spell. Listener.



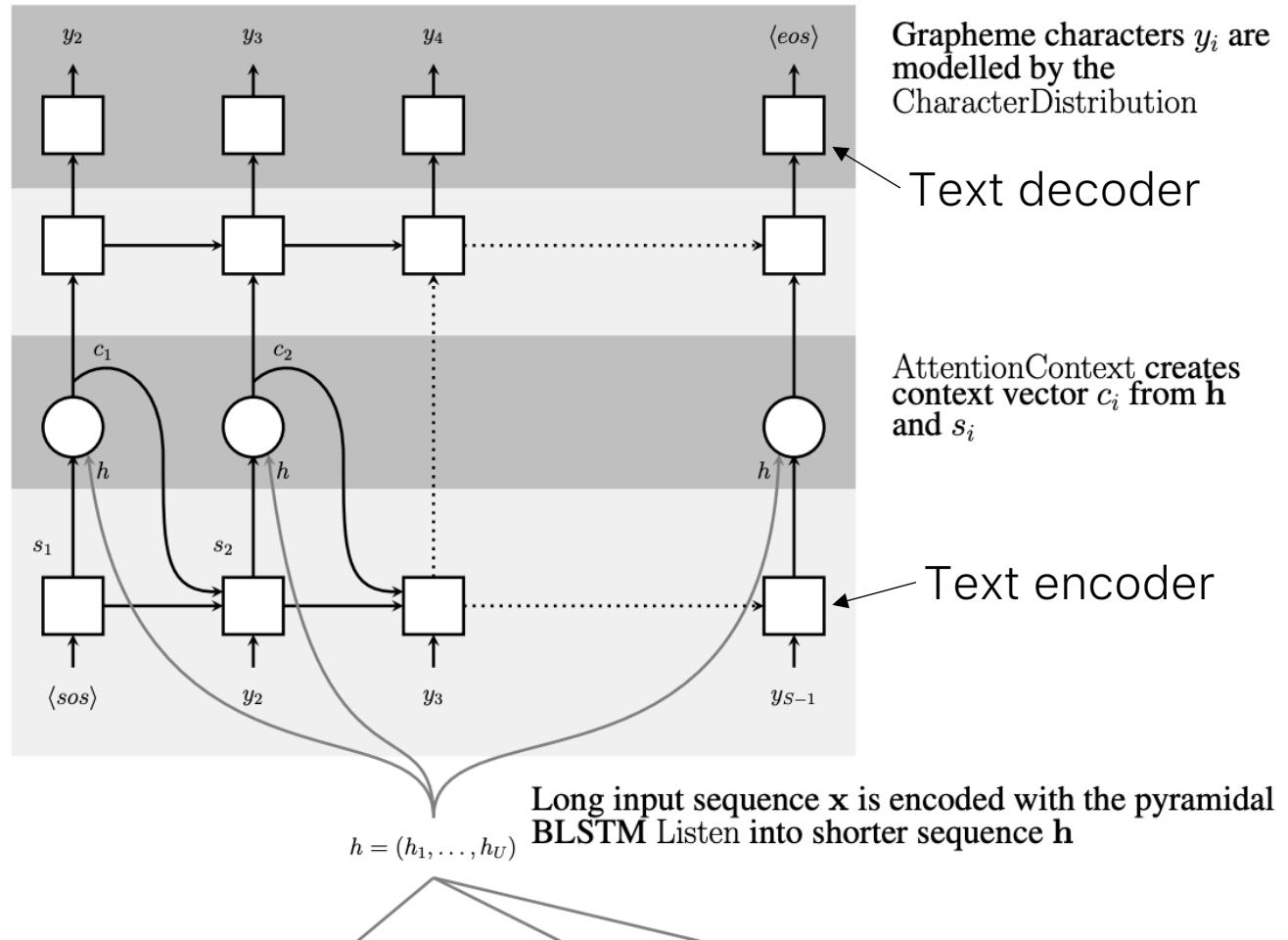
В качестве Listener-части модели используется двунаправленная многослойная RNN-сеть. На вход которой поступают фреймы(фичи) аудиодорожки  $x_T$ . На выходе получаем энкодинги аудио  $h_U$ .

Для уменьшения временной размерности используют склейку выходов более низких слоев RNN, т.о. уменьшая кол-во обрабатываемых данных во временной размерности в  $2^N$  раз, где  $N$ -глубина Listener.

$$h_i^j = \text{pBLSTM}(h_{i-1}^j, [h_{2i}^{j-1}, h_{2i+1}^{j-1}])$$

# Listen Attend and Spell. Speller + Attention.

Speller

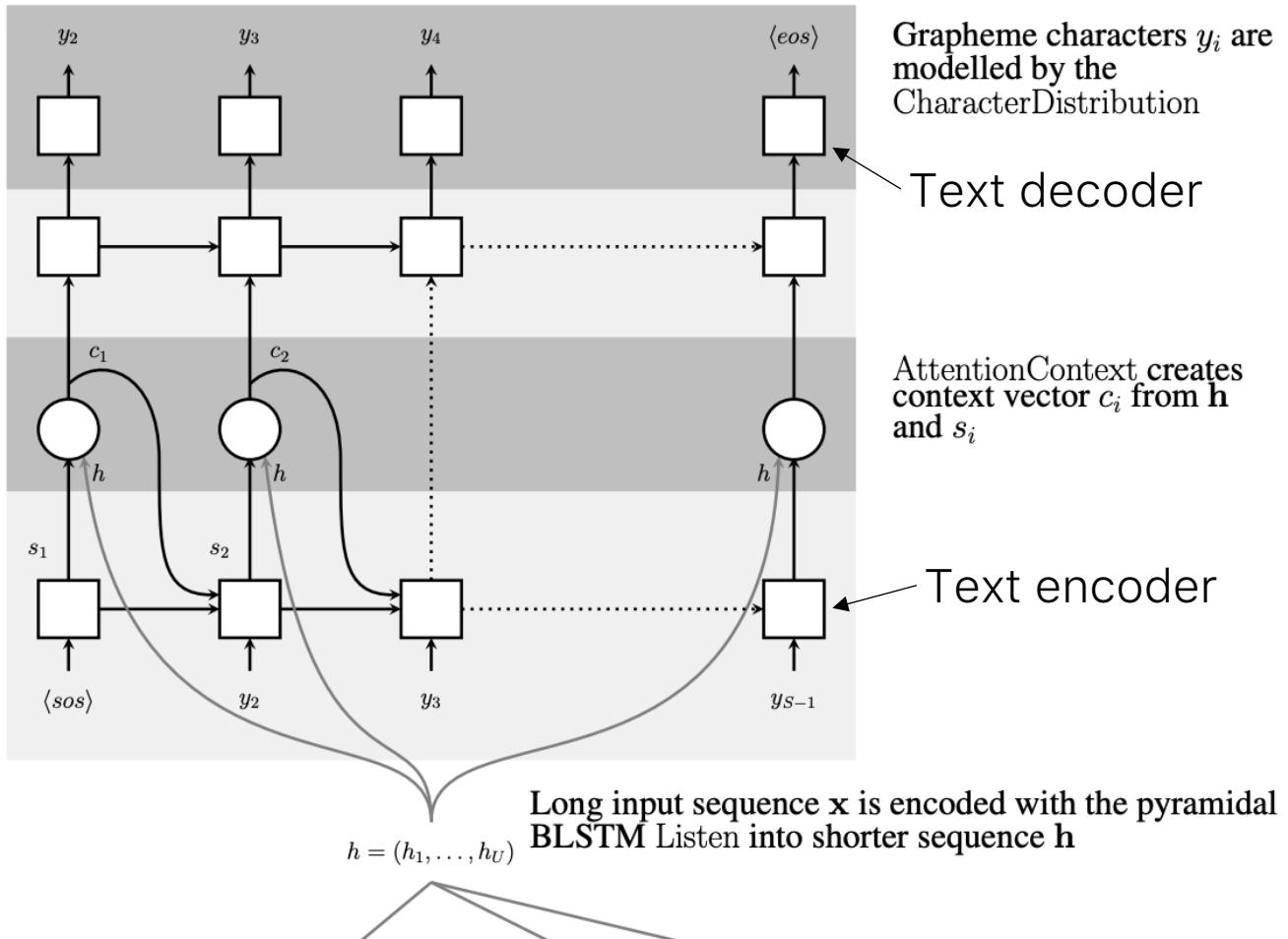


В качестве Speller-части модели используется несколько блоков:

- Text encoder – односторонняя RNN, на вход получает уже распознанные токены  $y_{i-1}$ , на выход выдает эмбеддинг текста  $s_{i-1}^{te}$ .
- AttentionContext – механизм внимания над текущим текстовым эмбеддингом  $s_{i-1}^{te}$  и энкодингами аудио  $h$ , на выход выдает контекст  $c_{i-1}$ .
- Text decoder – односторонняя RNN на основе своего предыдущего состояния,  $s_{i-1}^{td}$ , контекста  $c_{i-1}$  выдает распределение над словарем  $y_i$ .

# Listen Attend and Spell. Speller + Attention.

Speller



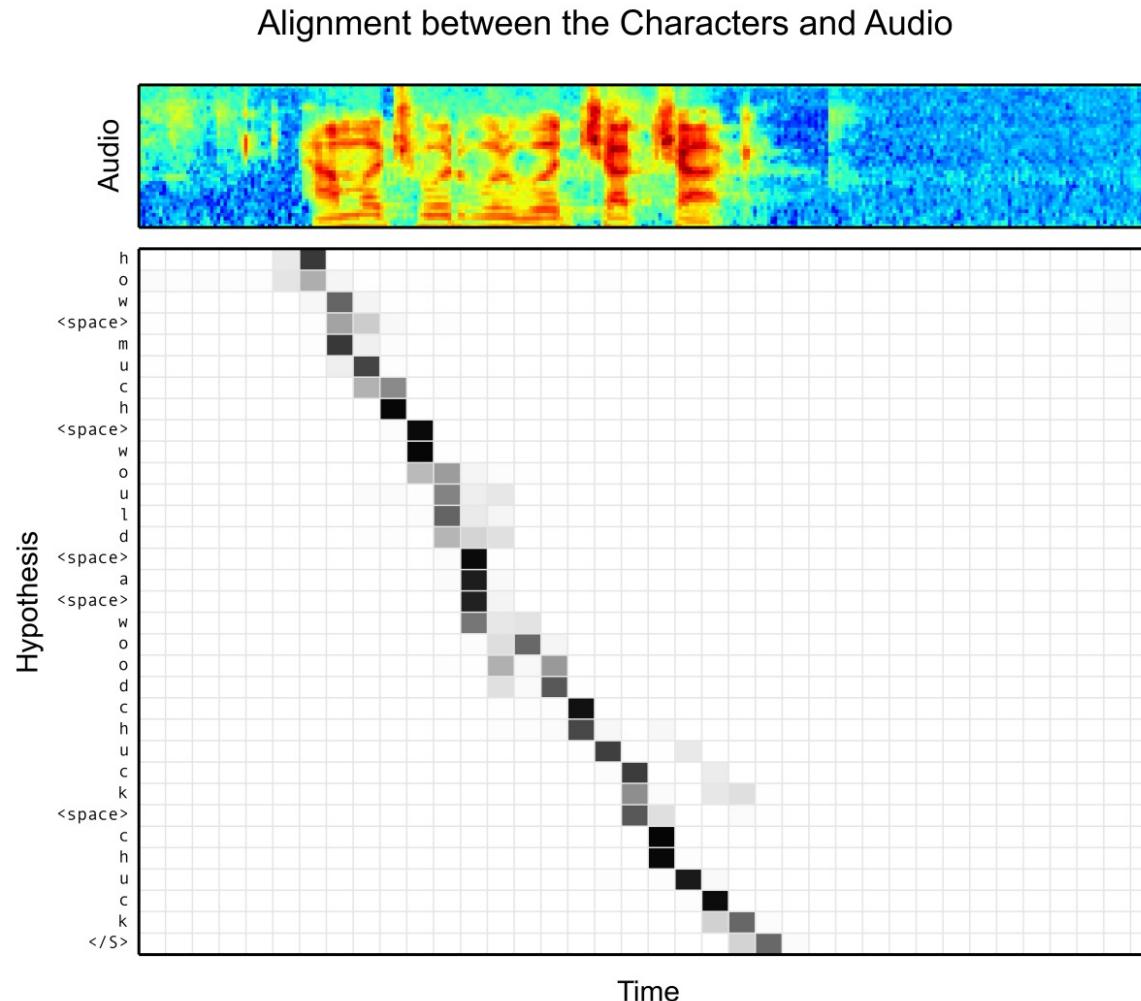
$$c_i = \text{AttentionContext}(s_i, h)$$

$$s_i = \text{RNN}(s_{i-1}, y_{i-1}, c_{i-1})$$

$$P(y_i | \mathbf{x}, y_{<i}) = \text{CharacterDistribution}(s_i, c_i)$$

По формулам, можно заметить, что не фиксируется конкретное кол-во слоев и блоков обработки входной информации(аудио/текста). Т.е. по своей сути LAS это некоторый фреймворк для построения систем распознавания речи.

# Listen Attend and Spell. Attention.



$$e_{i,u} = \langle \phi(s_i), \psi(h_u) \rangle$$

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_u \exp(e_{i,u})}$$

$$c_i = \sum_u \alpha_{i,u} h_u$$

Из формул видно, что нету жесткой привязки к тому, как предобрабатывать входы механизма внимания. В finale, как правило, применяют dot-product-attention. В добавок ко всему, количество голов аттеншена не фиксируют, на практике хорошо работает связка из 3-4 голов. Так же возможно регуляризовать маски аттеншена к диагональному виду как на картинке при помощи нормы Фробениуса.

# Listen Attend and Spell. Обучение и инференс.

- В качестве Loss-функции используем CrossEntropy-Loss:  $\max_{\theta} \sum_i \log P(y_i | \mathbf{x}, y_{<i}^*; \theta)$ , где  $y_{<i}^*$  – это ground truth токены фразы до  $i$ -го. Это выражается в подаче на вход текстовому энкодеру данных в режиме **Teacher forcing**.
- Проблема такого подхода, в том что во время инференса при возникновении ошибки генерации следующего токена модель не сможет восстановить правильную фразу. Для устранения этого эффекта, лосс модифицируют путем использования вместо некоторых ground truth токенов  $y_{<i}^*$ , токены  $\tilde{y}_i = \text{CharacterDistribution}(s_i, c_i)$  – сгенерированных моделью. Как правило, заменяемые токены выбирают случайно с некоторой вероятностью.
- Для инференса можно использовать самую простую схему – Argmax decoding. Ее суть в выборе на каждом шаге декодинга токена с наибольшей вероятностью:

$$\hat{y} = \operatorname{argmax}_y \log P(y | \mathbf{x})$$

# Listen Attend and Spell. Beamsearch.

- У argmax decoding есть существенная проблема – жадность алгоритма. Вследствие этого, получаемые гипотезы могут быть ошибочными.
- Данная проблема решается при помощи алгоритма **Beamsearch**:
  1. На начальном этапе имеем множество частичных гипотез  $\beta$ , состоящее из одной гипотезы  $\langle \text{sos} \rangle$ (start of sequence), а так же множество законченных гипотез  $B$ .
  2. На каждом шаге алгоритма к каждой из гипотез добавляются новые символы, тем самым расширяя beam до размера  $|\beta| * |\text{dict\_size}|$ . Так же можно расширять гипотезы только  $k$  наиболее вероятными токенами.
  3. После окончания шага 2 мы переносим гипотезы, у которых в кач-ве последнего символа выступает  $\langle \text{eos} \rangle$ (end of sequence) в мн-во  $B$ .
  4. Повторяем шаги 2-3 пока, не выполнится условие останова.

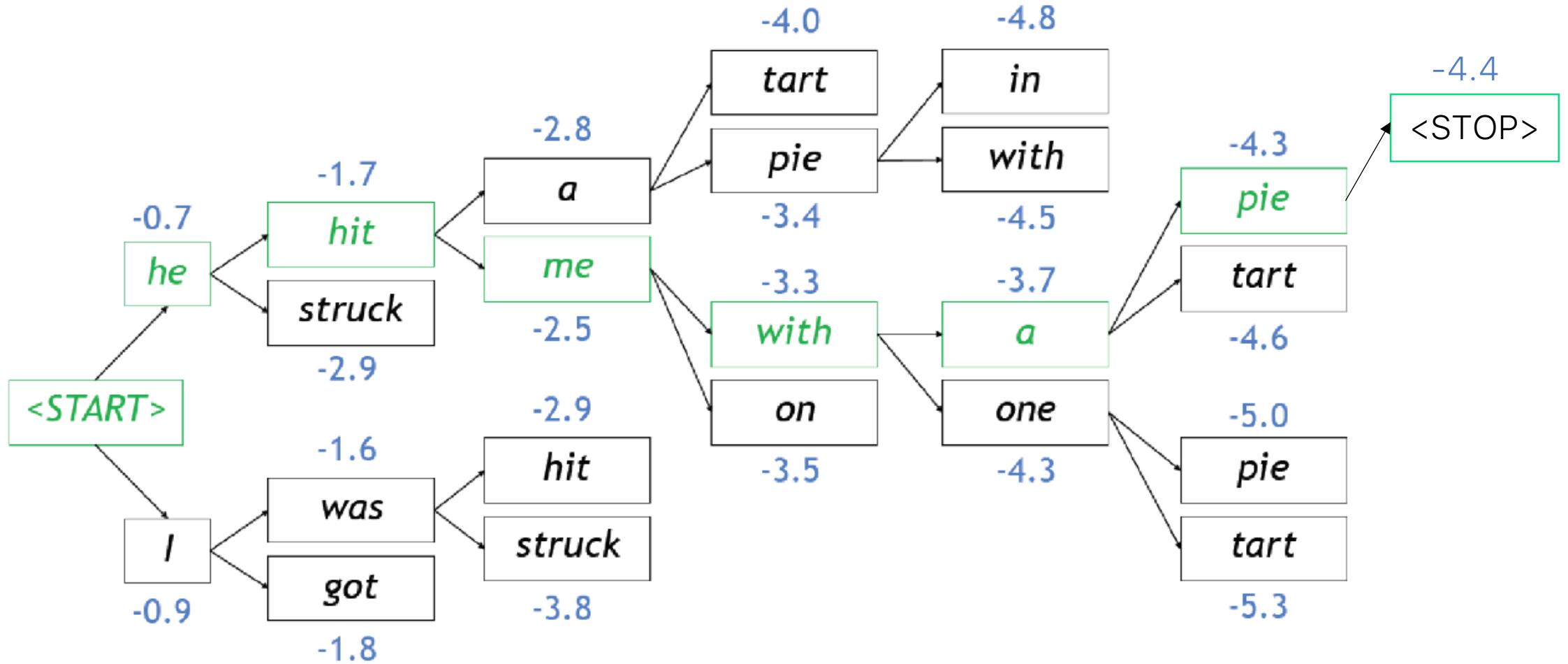
В качестве условия останова можно выбрать: размер мн-ва  $B$ , количество шагов декодинга, другие эвристики над гипотезами в  $B$ .

# Listen Attend and Spell. Beamsearch.

- У argmax decoding есть существенная проблема – жадность алгоритма. Вследствие этого, получаемые гипотезы могут быть ошибочными.
- Данная проблема решается при помощи алгоритма **Beamsearch**:
  1. На начальном этапе имеем множество частичных гипотез  $\beta$ , состоящее из одной гипотезы  $\langle \text{sos} \rangle$ (start of sequence), а так же множество законченных гипотез  $B$ .
  2. На каждом шаге алгоритма к каждой из гипотез добавляются новые символы, тем самым расширяя beam до размера  $|\beta| * |\text{dict\_size}|$ . Так же можно расширять гипотезы только  $k$  наиболее вероятными токенами.
  3. После окончания шага 2 мы переносим гипотезы, у которых в кач-ве последнего символа выступает  $\langle \text{eos} \rangle$ (end of sequence) в мн-во  $B$ .
  4. Повторяем шаги 2-3 пока, не выполнится условие останова.

В качестве условия останова можно выбрать: размер мн-ва  $B$ , количество шагов декодинга, другие эвристики над гипотезами в  $B$ .

# Listen Attend and Spell. Beamsearch.



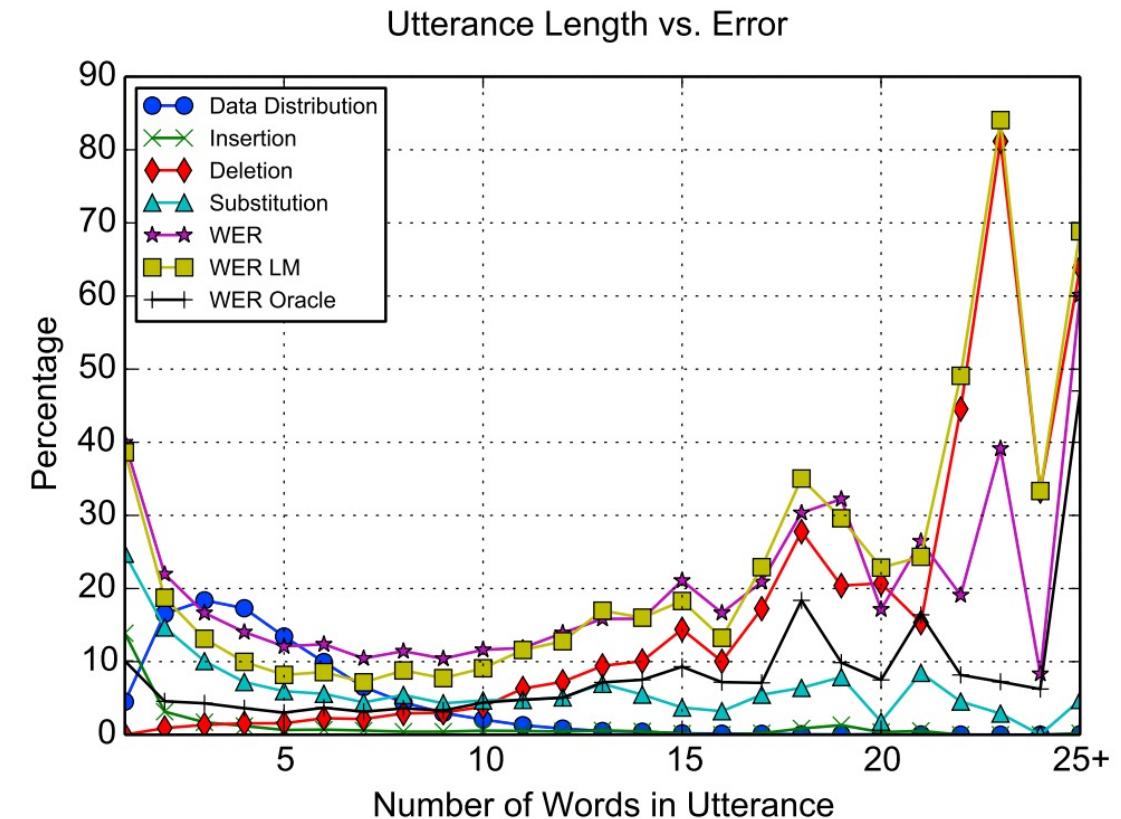
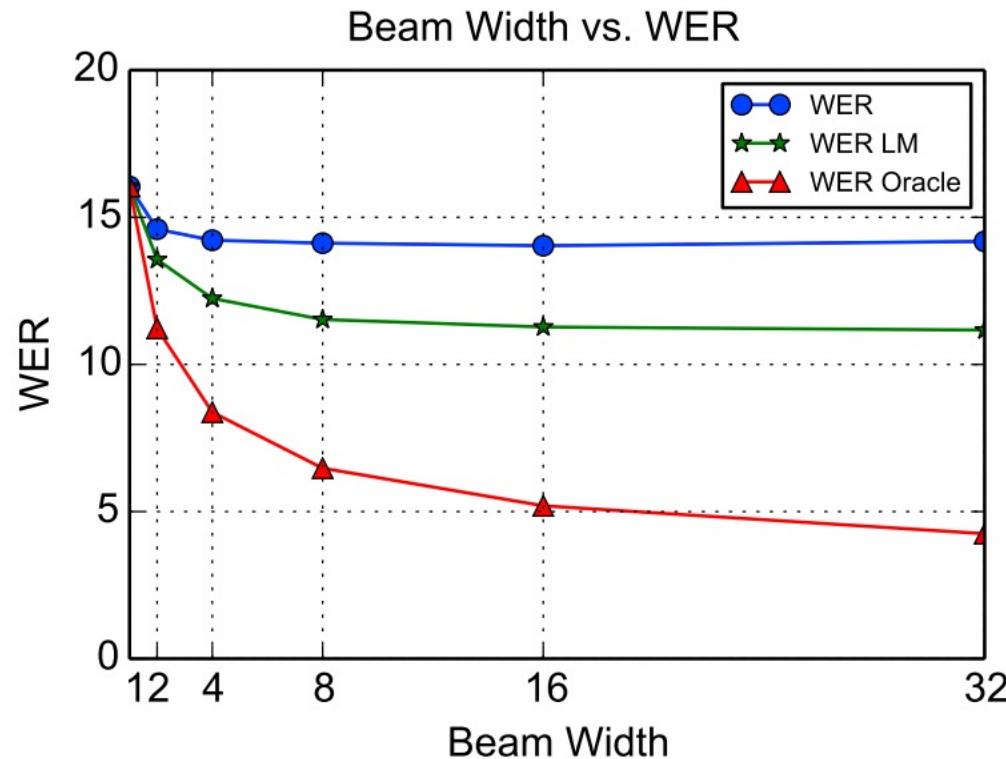
# Listen Attend and Spell. Beamsearch. LM.

- Поскольку модель LAS является seq2seq, ей присущи несколько проблем – генерация выдуманных фраз, слов.
- Для решения этой проблемы, а так же улучшения качества генерации часто прибегают к использованию языковых моделей(LM). В таком случае формула декодинга приобретает следующий вид:

$$s(\mathbf{y}|\mathbf{x}) = \frac{\log P_{ASR}(\mathbf{y}|\mathbf{x})}{|\mathbf{y}|} + \lambda \log P_{LM}(\mathbf{y}), \text{ где } |\mathbf{y}| - \text{длина гипотезы в токенах.}$$

- Можно заметить, что языковая модель применяется уже на этапе рескоринга к уже законченным гипотезам. Это не является существенным ограничением, языковую модель можно применять непосредственно на этапе генерации гипотезы.
- Так же важную роль играет нормировка на длину гипотезы, в противном случае функционал будет иметь тенденцию к генерации коротких фраз.

# Listen Attend and Spell. Beamsearch. LM.



- С некоторого момента качество перестает зависеть от размера beam.
- Качество LAS модели плохо обобщается на короткие и длинные записи. Во втором случае, если преимущественно в train были короткие и средние по длине записи.

# СТС подход к задаче ASR (СТС).

• • • • •

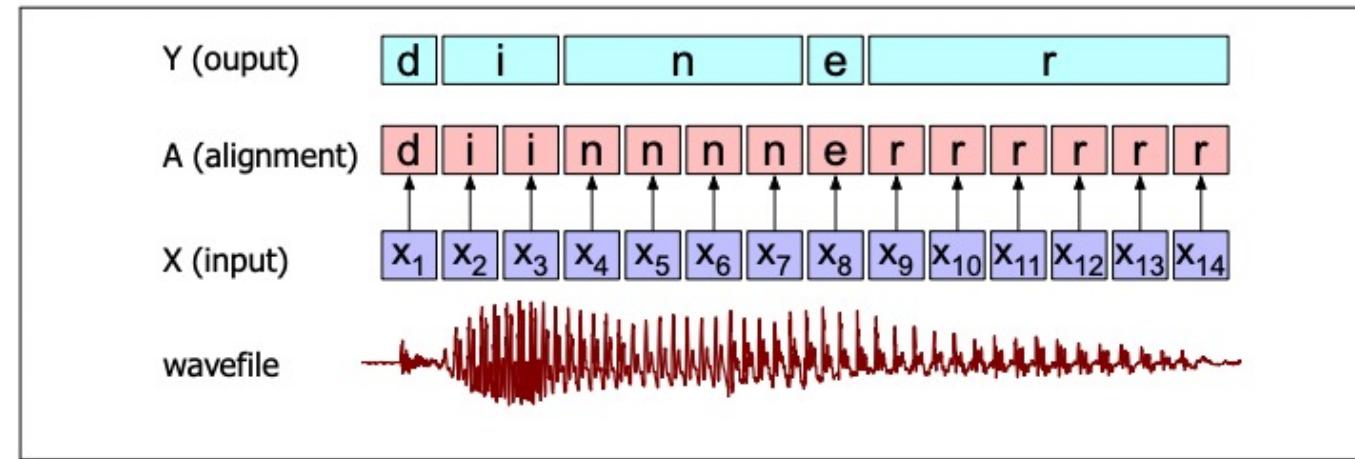
# Connectionist Temporal Classification.

- Недостатки у LAS-подхода к ASR:
  - Работает только с полной аудиодорожкой;
  - Не возможно сделать выравнивание между входом и выходом (alignment);
  - (Спорно) запоминает в весах так же языковую модель (ILM);
  - Чувствительна к изменению длины входа.

Для борьбы с этими недостатками, а так же отсутствием большой коллекции выровненных аудиодорожек для обучения задаче ASR, предложили использовать метод, называемый Connectionist Temporal Classification(CTC).

# Connectionist Temporal Classification.

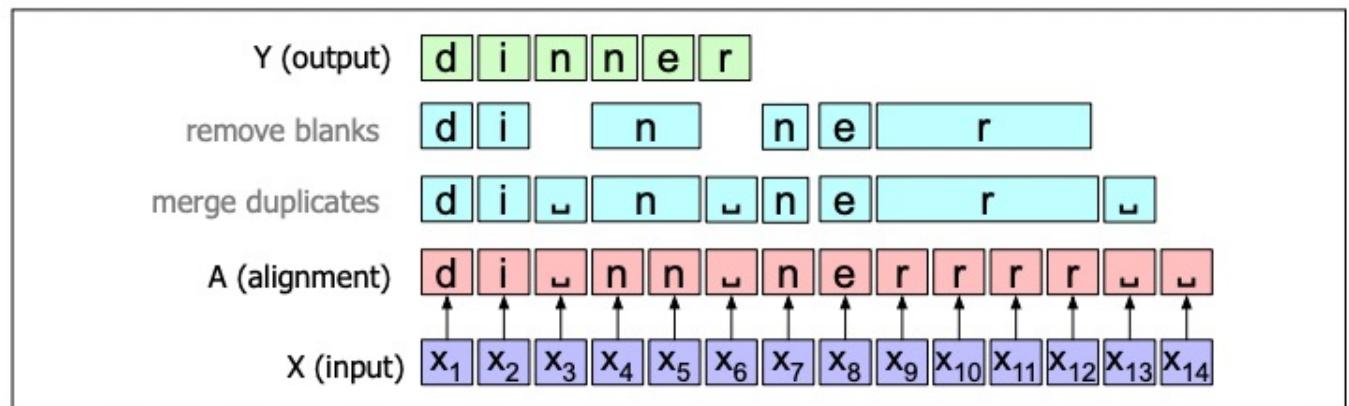
- Основная суть данного метода состоит в отказе от seq2seq архитектуры в пользу модели, которая для каждого входного аудио-фрейма  $x_t$  предсказывает выходной токен  $a_t$  из лексикона.



- В таком подходе возникает проблема – получения финальной транскрипции, поскольку:
  - Выходные токены будут повторяться и непонятно как учесть эти повторения ?
  - Какой токен выводить в случае, если в текущем аудиофрейме нету речи ?

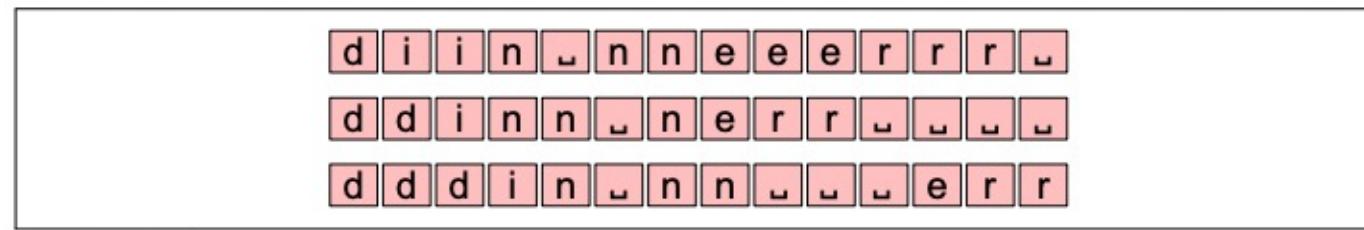
# Connectionist Temporal Classification.

- Для решения вышеупомянутых проблем были введены две «сущности»:
  1. Специальный пустой символ  $\epsilon$  (blank). Он используется для нескольких целей:
    - В качестве выводимого символа для тех фреймов, где никакой другой символ не подходит(тишина, неразборчивые звуки, шум и т.д.);
    - В качестве разделителя между подряд идущими одинаковыми токенами(программа, единственный и т.д.).
  2. Специальная функция обработки( $B: a \rightarrow y$ ) выходной пофреймовой транскрипции, состоящая из следующей последовательности действий:
    - Склеиваем все подряд идущие одинаковые токены, в т.ч. blank.
    - Удаляем все blank-символы.



# Connectionist Temporal Classification.

- Можно заметить, что одной и той же финальной транскрипции  $y$  могут соответствовать различные последовательности выходных токенов  $a$ .



- Данный факт нам пригодится в дальнейшем для построения loss-функции и проведения операции декодинга. Поэтому будем иметь ввиду, что существует такое мн-во  $A = B^{-1}(y)$ .

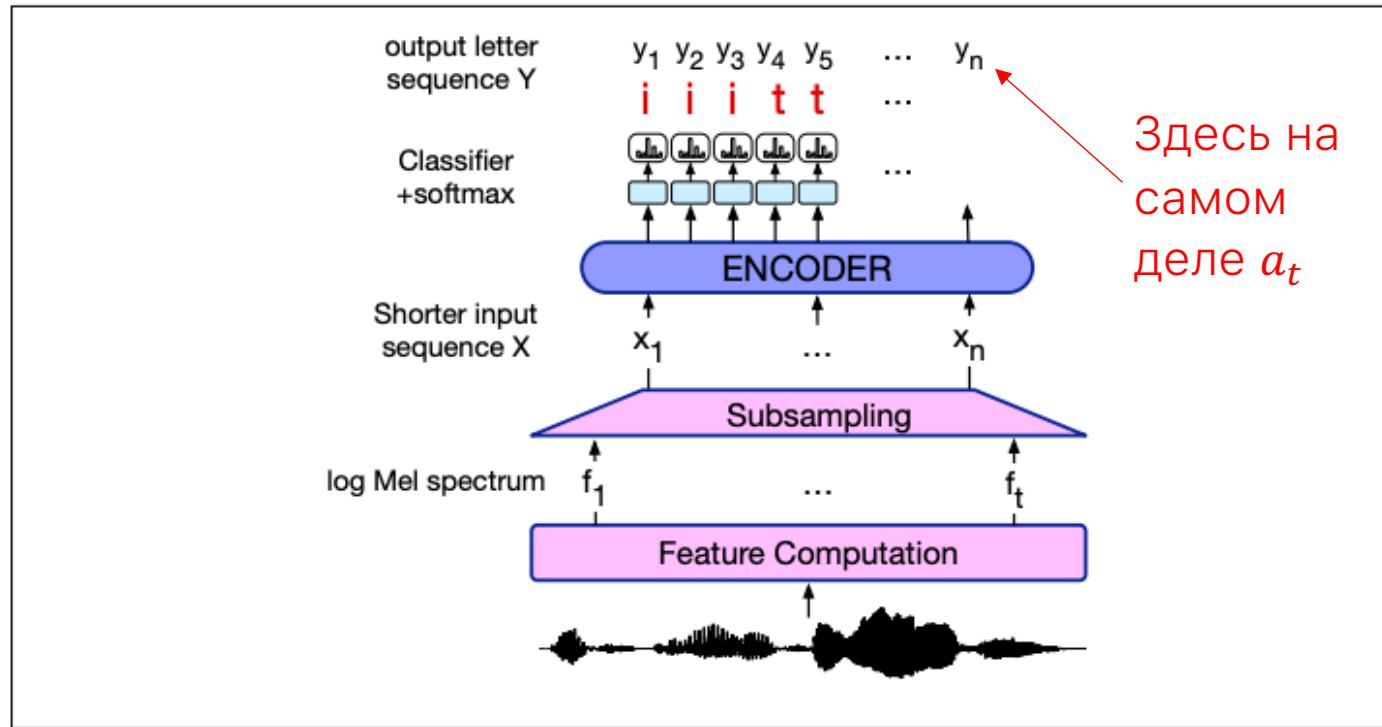
# СТС. Архитектура сети.

В отличие от LAS подхода, сеть на основе СТС состоит только из аудио-энкодера. Так же как и в seq2seq на вход сети подаются сэмплы аудио. На выходе модели имеем токены для каждого из входных фреймов  $x_t$ .

Модель учится максимизировать правдоподобие правильной транскрипции по всем возможным послед-ям выходных токенов:

$$P_{CTC}(y|x) = \sum_{a \in B^{-1}(y)} \prod_t P(a_t|x).$$

Обратите внимание, что вероятность каждого нового токена не зависит от предыдущих!



# СТС. Декодирование.

Аналогично LAS-подходу  
декодирование можно  
осуществлять несколькими  
способами:

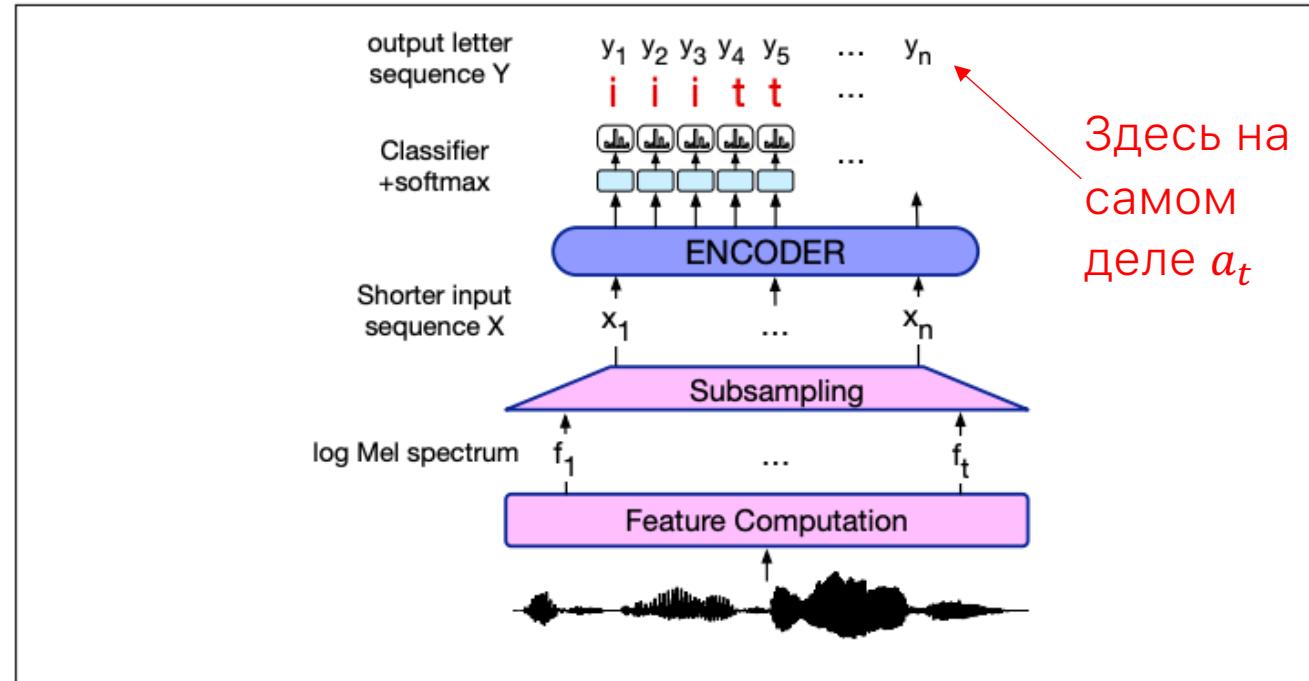
- Жадно:

$$\hat{a}_t = \underset{c \in \text{Lexicon}}{\operatorname{argmax}} P_t(c|x);$$

$$\hat{y} = B(\hat{a}).$$

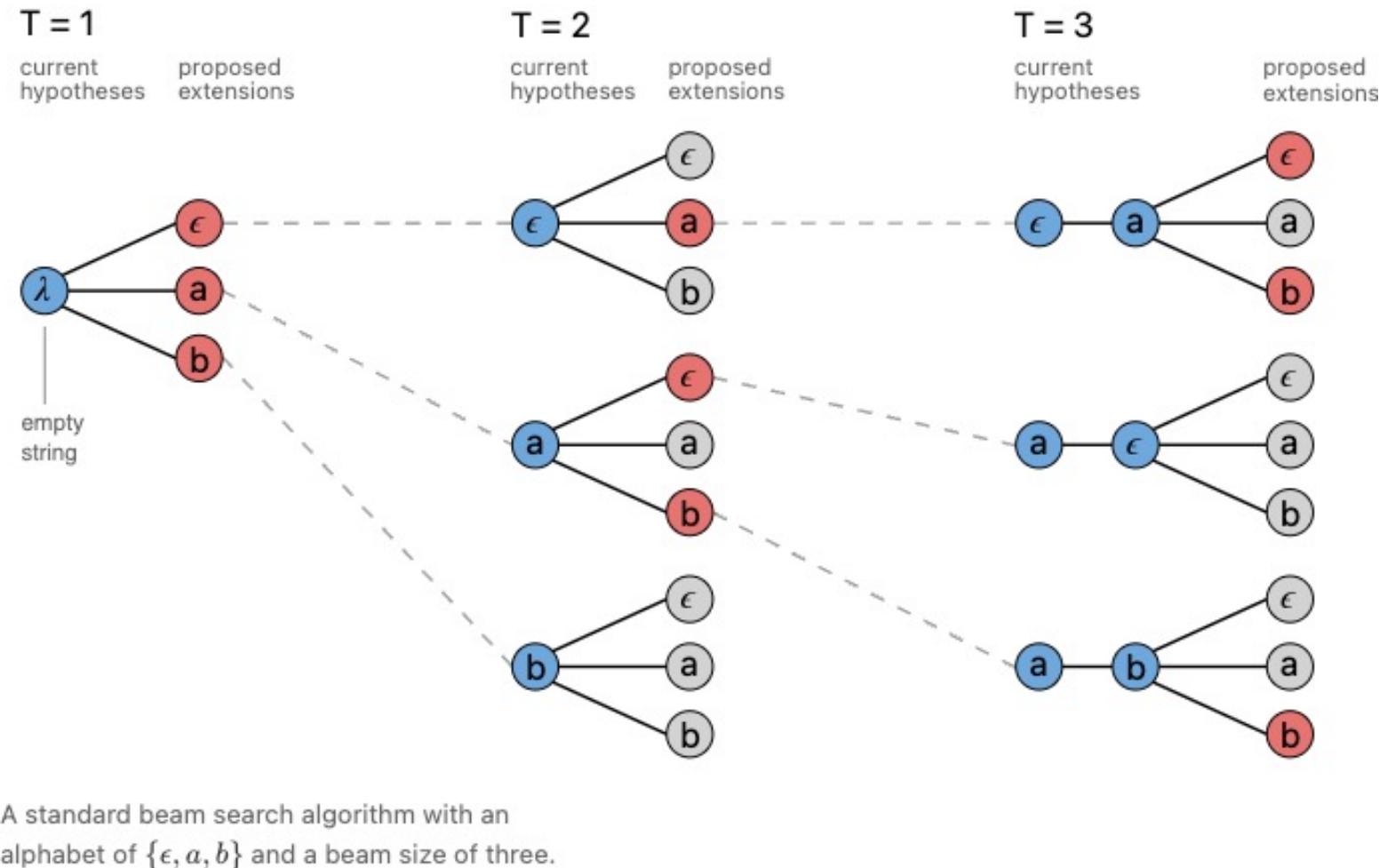
В данном подходе сохраняются  
все проблемы жадного  
декодирования.

- Prefix beamsearch.
- Prefix beamsearch + LM.



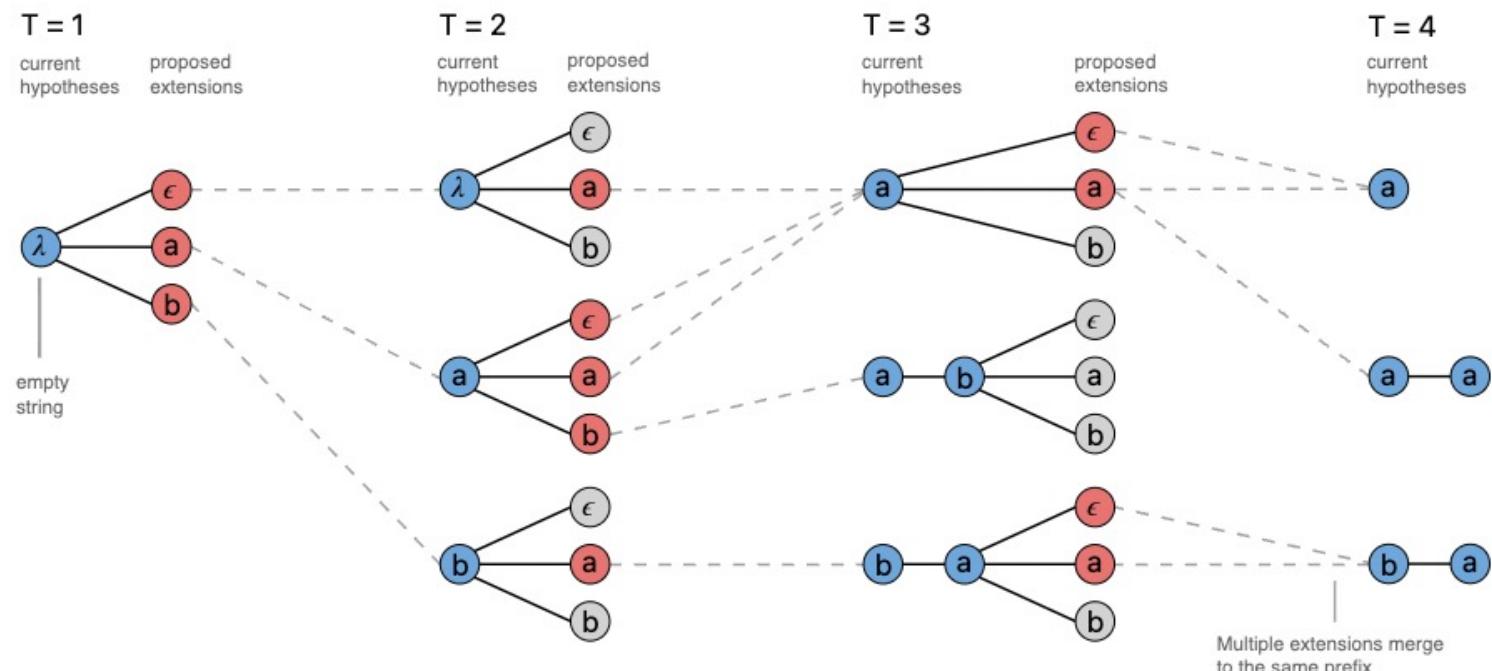
# СТС. Декодирование. Beamsearch.

- Рассмотрим «наивный» вариант beamsearch.
- В примере, изображенном на рисунке, первый и второй луч приводят к одному и тому же символу: «а».
- Такое наивное декодирование приведет к следующим недостаткам:
  - Вероятности одинаковых префиксов не будет учтены в виде суммы.
  - Получим повторяющиеся лучи в условиях ограничений на размер множества поддерживаемых гипотез при декодировании.
- Для устранения данных недостатков необходимо использовать модифицированную версию лучевого поиска – Prefix beamsearch.



# СТС. Декодирование. Prefix beamsearch.

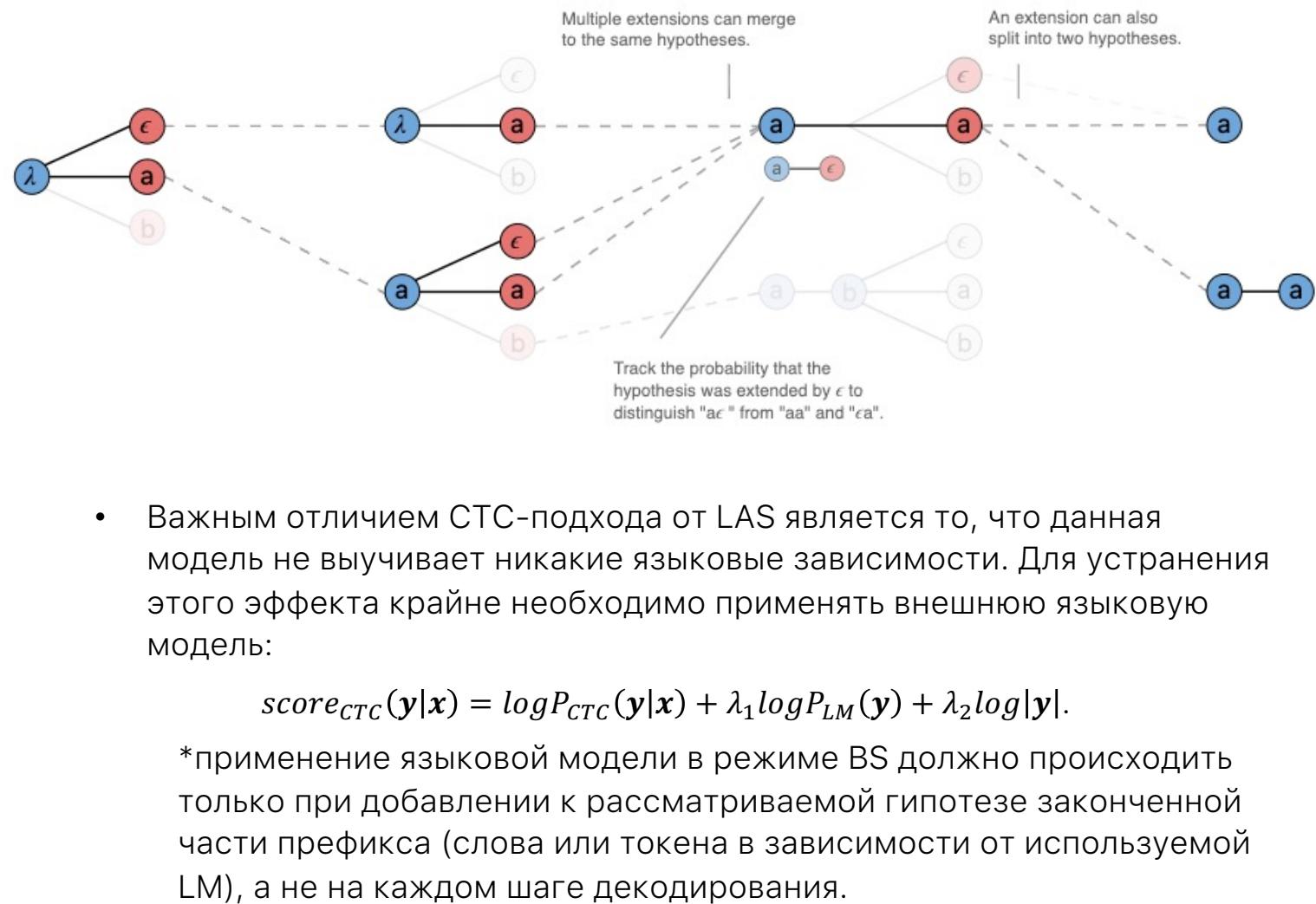
- В алгоритме (СТС)Prefix beamsearch предлагается заместо необработанных выходных последовательностей хранить в мн-ве текущих бимов префиксы, получающиеся после схлопывания повторяющихся и blank токенов.
- На каждом шаге мы аккумулируем вероятности префиксов, приводящих к одной и той же выходной последовательности токенов.
- На изображенном примере на шаге  $T=3$  мы схлопнули несколько префиксов, приводящих к одному и тому же символу « $a$ ».



The CTC beam search algorithm with an output alphabet  $\{\epsilon, a, b\}$  and a beam size of three.

# СТС. Декодирование. Prefix beamsearch. LM.

- Следуя правилу схлопывания выходных символов для одного и того же префикса нам будет необходимо поддерживать значения вероятностей для двух случаев:
  - Если данный префикс заканчивался символом blank.
  - Если данный префикс заканчивался тем же самым символом.
- Такой учет необходим для того, чтобы корректно обрабатывать ситуацию, когда мы встретим в процессе декодирования после символа blank еще один такой же символ, что был и до него: префикс полученный слиянием 3-х бимов: *blank+a*, *a+a*, *a+blank* распадется на 2 бима в случае его расширением символами *blank* и *a*.

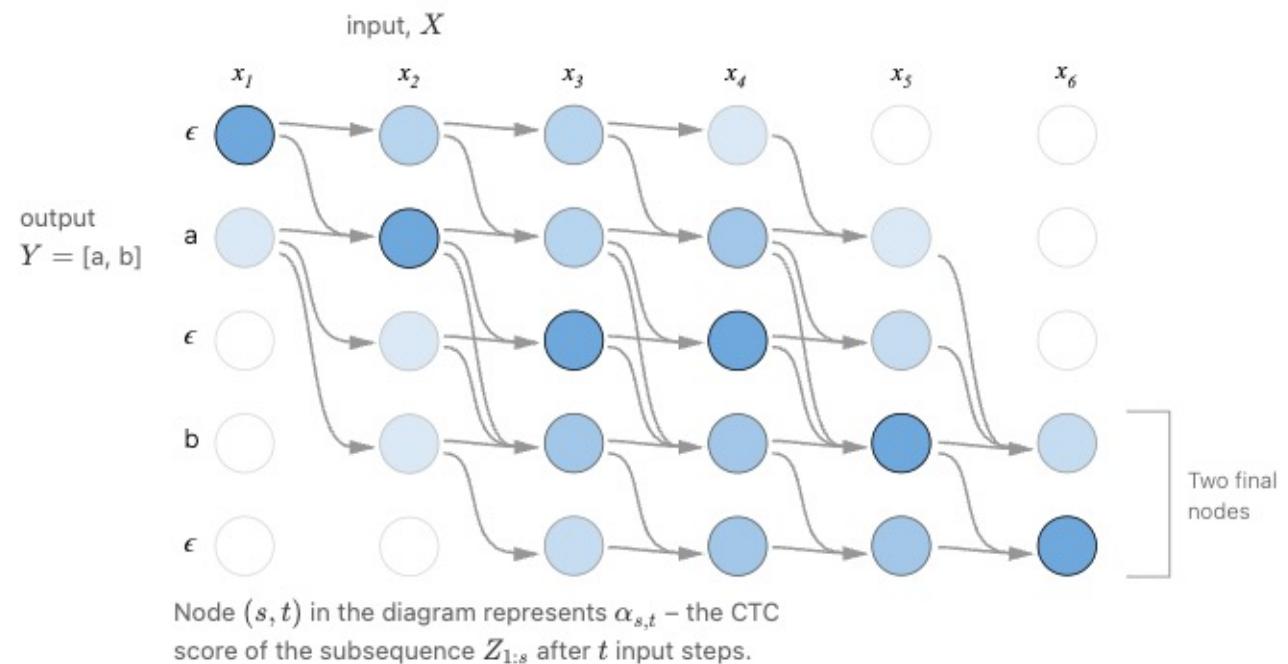


# СТС. Обучение. СТС-loss.

- В процессе обучения минимизируется логарифм правдоподобия со специальной СТС-loss функцией:  $L_{ctc} = \sum_{(\mathbf{x}, \mathbf{y}) \in D} -\log P_{ctc}(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta})$ .
- Как мы видели ранее, чтобы минимизировать данный функционал нам необходимо для каждой пары  $(\mathbf{x}, \mathbf{y})$  найти сумму вероятностей всех последовательностей выходных токенов, приводящих к  $\mathbf{y}$ :  $P_{ctc}(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}) = \sum_{\mathbf{a} \in B^{-1}(\mathbf{y})} \prod_t P(a_t|\mathbf{x}; \boldsymbol{\theta})$ .
- Если делать это просто перебором по всем возможным последовательностям, то даже для аудио, состоящего из 100 фреймов(около 3 секунд) и словаря из 33 букв + blank нам необходимо перебрать  $34^{100} \approx (1.4 * 10^{153})$  последовательностей.
- Необходимо использовать что-то более интеллектуальное. Выход есть – применим **динамическое программирование** и рассмотрим только те последовательности, которые реально приводят к правильной транскрипции.

# СТС. Обучение. СТС-loss.

- Как и в любом алгоритме из об-ти динамического программирования начнем с построения матрицы размерности  $|Z| \times |T|$ , где  $Z = [\epsilon, y_1, \epsilon, y_2, \epsilon, y_3, \dots, \epsilon, y_n, \epsilon]$ - последовательность токенов правильной транскрипции, разделенная символом *blank*,  $T$  - временная размерность входа (аудиозаписи).
- Обозначим через  $\alpha_{s,t}$  -  $P_{CTC}$  последовательности  $Z_{1:s}$  накопленную к моменту времени  $t$ . Далее мы увидим, что  $P_{CTC}(y|x) = \alpha_{s-1,t} + \alpha_{s,t}$  получается из значений  $\alpha$  на последних 2-ух шагах данного алгоритма.
- Инициализируем значения  $\alpha_{1,1} = P_{CTC}(\epsilon|x)$ ,  $\alpha_{2,1} = P_{CTC}(y_1|x)$ .
- Далее при помощи рекуррентного правила рассчитаем значения для остальных  $\alpha$ .



# СТС. Обучение. СТС-loss.

- Рассмотрим 2 случая:

- Когда для расчета  $\alpha_{s,t}$  мы не можем «перепрыгивать» через символ  $z_{s-1}$ . Первая причина(а) – если предыдущий символ был эл-ом посл-ти  $y$ , а мы не можем собрать верную транскрипцию без его учета, такая ситуация соответствует  $z_s = \epsilon$ . Вторая причина(б), если  $z_s = z_s \neq \epsilon$  – т.е. между повторяющимися символами обязательно идет *blank*.

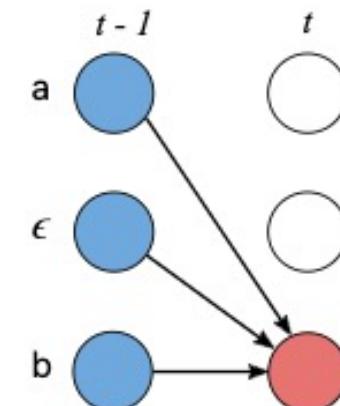
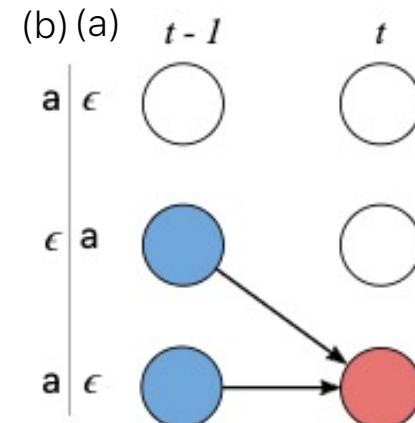
Правило расчета  $\alpha$  в таком случае будет:

$$\alpha_{s,t} = (\alpha_{s-1,t-1} + \alpha_{s,t-1}) * P_t(z_s | x)$$

- 
- Когда для расчета  $\alpha_{s,t}$  мы можем «перепрыгивать» через символ  $z_{s-1}$ . Этот тот случай, когда  $z_{s-1} = \epsilon$ , который расположен между двумя уникальными символами. Правило расчета  $\alpha$  в таком случае будет:

$$\alpha_{s,t} = (\alpha_{s-2,t-1} + \alpha_{s-1,t-1} + \alpha_{s,t-1}) * P_t(z_s | x)$$

\*Градиенты для СТС-Loss по параметрам сети рассчитываются схожим образом, только в обратном направлении.



# RNN-transducer (RNN-T).

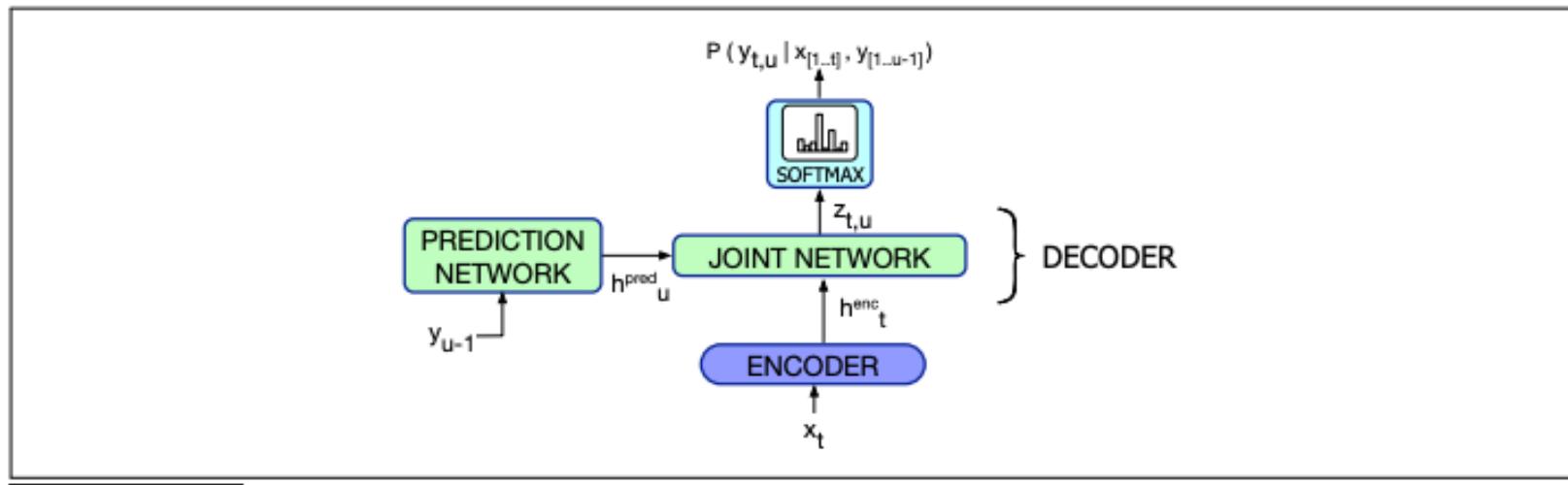
... ● ...

# RNN-Transducer.

- Мы помним, что в СТС-подходе к задаче ASR последовательность выходных токенов(а следовательно и транскрипций) не зависит от предыдущих сгенерированных токенов. Это приводит к тому, что в среднем качество транскрибации СТС-моделей хуже, чем у LAS. При этом, основное преимущество СТС в том, что они могут использоваться в потоковом режиме.
- Возникает потребность в том, чтобы совместить гибкость СТС-подхода и высокий уровень качества LAS-моделей. Для этих целей была разработана модель под названием RNN-Transducer (RNN-T).
- Модель RNN-T вносит в процесс генерации зависимость выходной последовательности от предыдущих сгенерированных токенов, но делает это не на этапе декодирования, а в виде отдельного блока модели.

# RNN-Transducer.

- RNN-T существенно повторяет СТС подход, тем не менее в нем есть отличия.
- Архитектура сети состоит из:
  - Prediction network – как правило это односторонняя RNN, на вход получает декодированные не blank(кроме шага 0) символы  $y_{u-1}$  и выдает эмбеддинг для обработанной последовательности  $h_u^{pred}$
  - Encoder – одно или двунаправленная RNN, которая на вход получает фреймы(спектрограммы) аудио  $x_t$  и на выход выдает эмбеддинги этих фреймов  $h_t^{enc}$ .
  - Joint network – для входной композиции эмбеддингов  $z_{t,u} = f(h_u^{pred}, h_t^{enc})$  строит распределение над выходным словарем –  $P(a_{t,u}|x_{[1..t]}, y_{[1..u-1]})$ .



- Отличия от СТС подхода:
  - В выходной лексикон так же добавляется символ *blank* символ  $\epsilon$ , но теперь он используется только для генерации отсутствия речи.
  - Функция  $B$  преобразования выходной последовательности Joint network теперь только удаляет символы *blank* из полученной транскрипции.
- Модель так же в процессе обучения максимизирует правдоподобие правильной транскрипции по всем возможным последовательностям выходных токенов:
$$P_{RNN-T}(y|x) = \sum_{a \in B^{-1}(y)} \prod_t P(a_t|x).$$
- Обратите внимание, что зависимость от истории декодирования скрыта внутри самой модели и явно не участвует в правдоподобии.

# RNN-Transducer. Декодирование.

- Алгоритм декодирования в подходе RNN-T так же отличается от СТС. Рассмотрим жадный вариант:
  1. Пропускаем через энкодер входную аудиозапись  $x$  получаем набор эмбеддингов  $h_{1..T}^{enc}$ . Подаем на вход Prediction network blank(можно и <SOS>) и получаем  $h_{u=0}^{pred}$ .
  2. Задаем  $u=0, t=1$ . Далее итерируемся по  $u, t$  следующим образом:
    - 1) Выбираем  $\hat{a} = \max_a P(a_{t,u} | x_{[1..t]}, y_{[1..u-1]})$  - из выхода Prediction сети.
    - 2) Если  $\hat{a}$  является blank символом, то заносим его в финальную транскрипцию и инкрементируем  $t$ .
    - 3) Если  $\hat{a}$  является символом исходного лексикона, то обновляем выход Prediction network  $h_u^{pred} \rightarrow h_{u+1}^{pred}$ , инкрементируем  $u$  и переходим к шагу 1).
    - 4) Если  $\hat{a}$  является символом конца последовательности <EOS> или же мы больше не можем инкрементироваться по  $t$  – заканчиваем декодирование.
  3. В finale применяем функцию  $B$  к получившейся последовательности токенов  $\hat{a}$ .
  4. Естественным образом данный алгоритм расширяется на Beamsearch, а так же Beamsearch с использованием LM. В данном случае есть необходимость правильно учесть blank символы при использовании внешней языковой модели.

# RNN-Transducer. Обучение

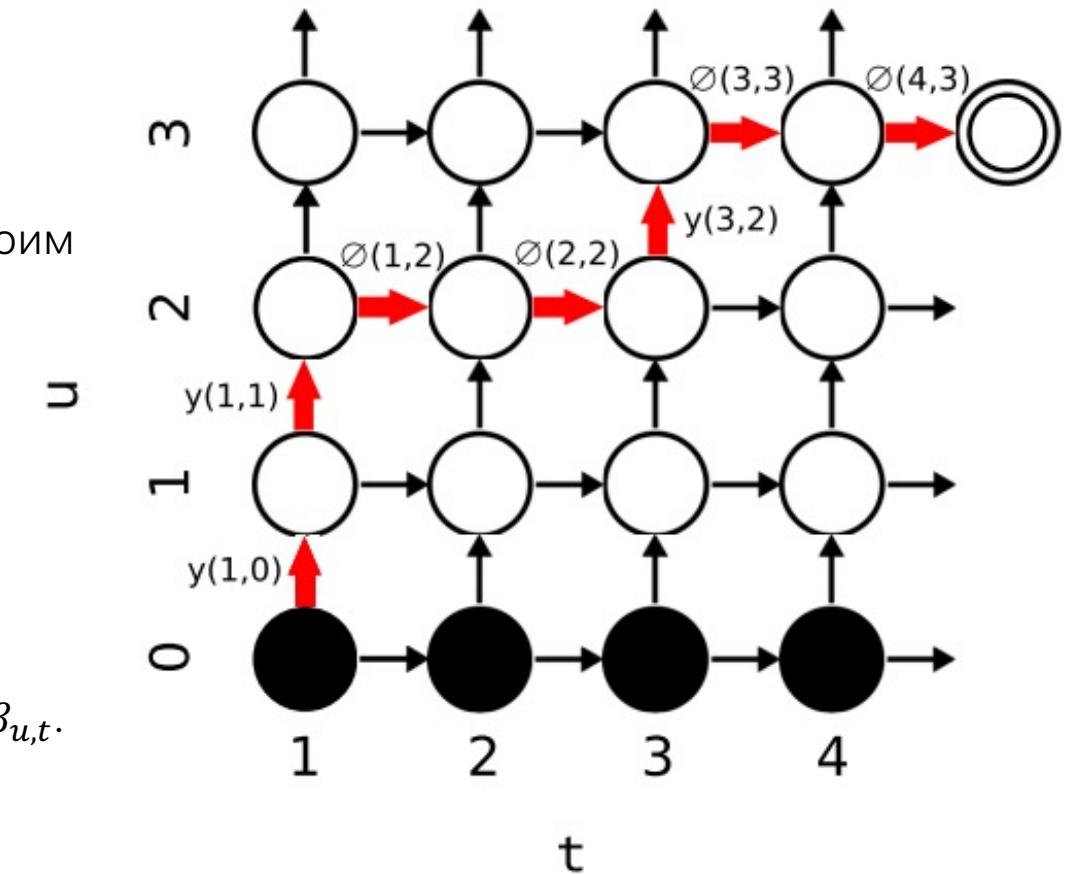
- В качестве лосса выступает отрицательный логарифм правдоподобия правильной транскрипции:

$$L_{RNN-T} = -\log P_{RNN-T}(\mathbf{y}^* | \mathbf{x}).$$

- Для его расчета все так же используем динамику. Строим матрицу размером  $|U| \times |T|$ , где размерность  $U$  соответствует последовательности символов  $[y_0, y_1, \dots, y_U]$ .
- Введем две переменные  $\alpha_{u,t}$ -суммарная вероятность получить на выходе  $\mathbf{y}_{1:u}$  обработав фреймы аудио  $\mathbf{x}_{1:t}$ ,  $\beta_{u,t}$ -суммарная вероятность получить на выходе  $\mathbf{y}_{u+1:U}$  обработав фреймы  $\mathbf{x}_{t:T}$ . Тогда можно заметить, что суммарная вероятность  $P_{RNN-T}(\mathbf{y}^* | \mathbf{x}) = \sum_{(t,u): t+u=n} \alpha_{u,t} * \beta_{u,t}$ .
- Инициализируем  $\alpha_{0,1} = 1$ ,  $\beta_{U,T} = P_{RNN-T}(\text{blank} | \mathbf{x}_T)$ .
- Воспользуемся рекуррентными формулами:

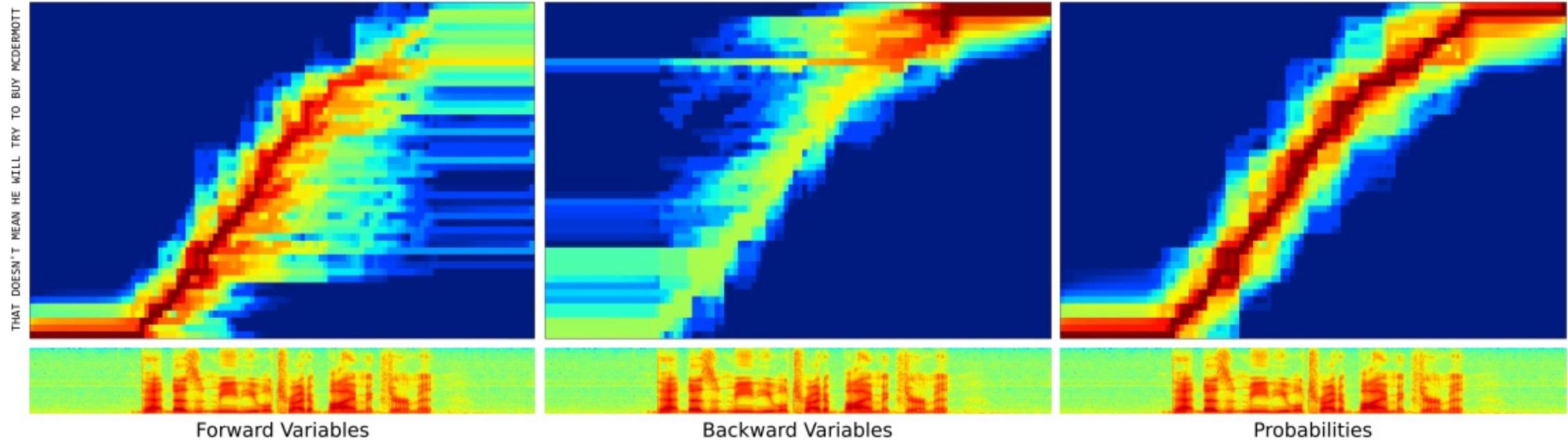
$$\alpha_{u,t} = \alpha_{u,t-1} P_{RNN-T}(\text{blank}_u | \mathbf{x}_{t-1}) + \alpha_{u-1,t} P_{RNN-T}(y_{u-1} | \mathbf{x}_t);$$

$$\beta_{u,t} = \beta_{u,t+1} P_{RNN-T}(\text{blank}_u | \mathbf{x}_t) + \beta_{u+1,t} P_{RNN-T}(y_u | \mathbf{x}_t).$$



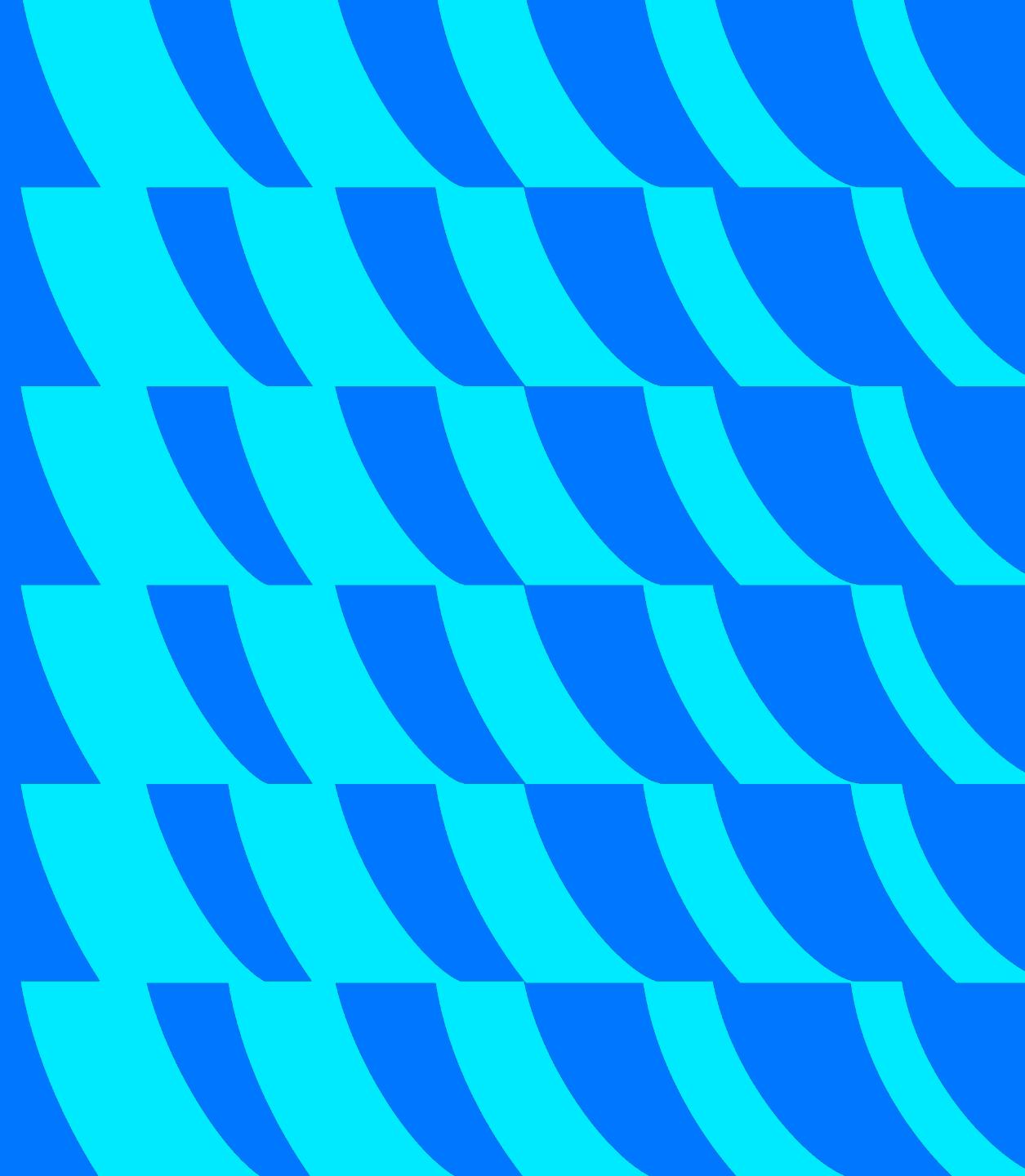
Важная remarque: во время обучения в Prediction Network подается последовательность  $\mathbf{y}$  в режиме Teacher forcing.

# RNN-Transducer. Обучение



**Figure 2. Forward-backward variables during a speech recognition task.** The image at the bottom is the input sequence: a spectrogram of an utterance. The three heat maps above that show the logarithms of the forward variables (top) backward variables (middle) and their product (bottom) across the output lattice. The text to the left is the target sequence.

# Твики для обучения.



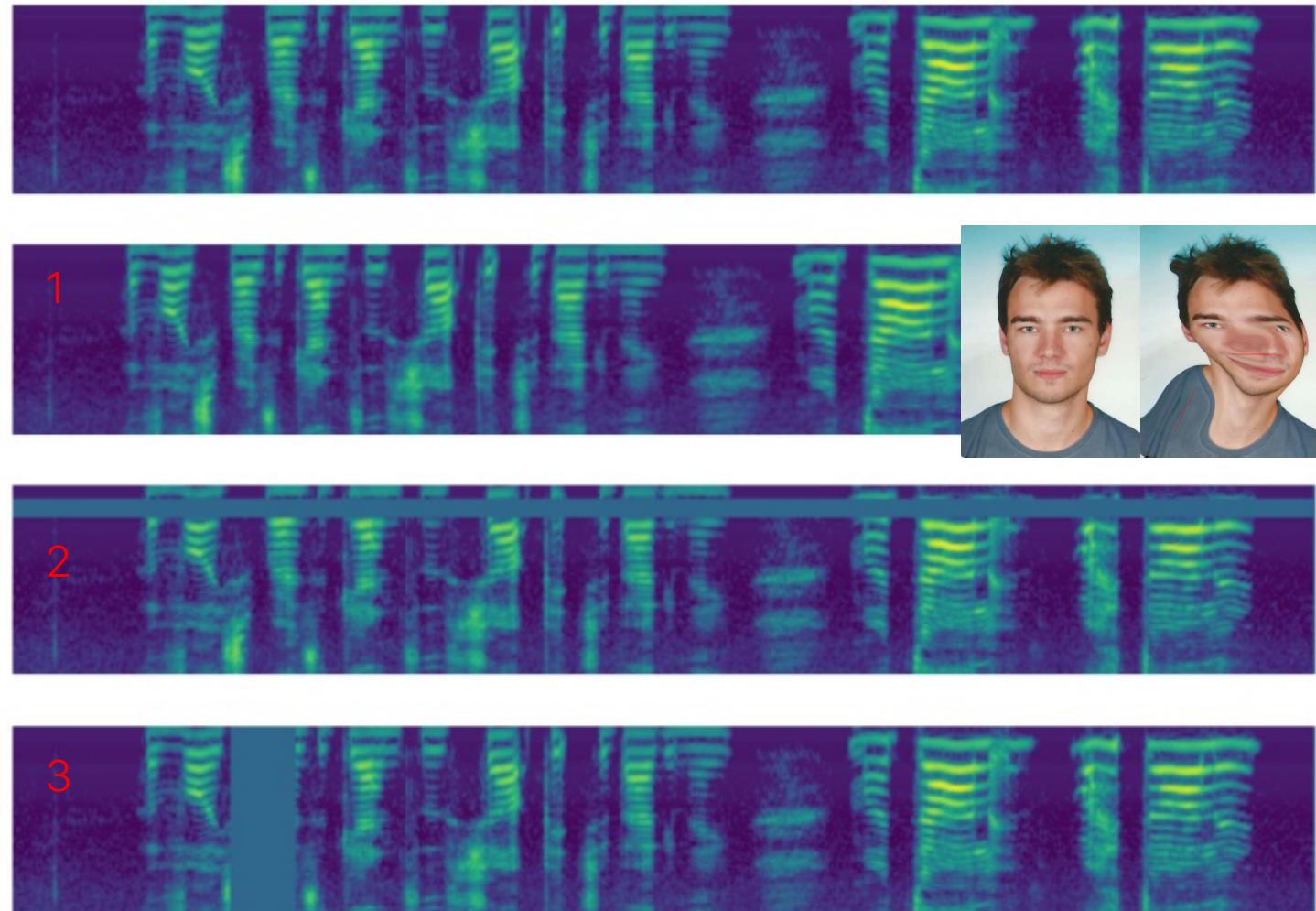
# Работа с данными. Аугментация аудио.

- Как и в других задачах обучения ASR, аугментация данных позволяет значительно повысить обобщающую способность модели, а следовательно улучшить ее качество.
- Аугментации, применяемые непосредственно к аудиозаписи можно отнести:
  1. Наложение шумов:
    1. Статистических: белый, красный и т.д
    2. Различные аудиозаписи – бытовой шум, музыка, фоновые голоса и т.д.Основным параметром для данного типа аугментаций является SNR(signal-to-noise ratio), т.е. насколько энергия сигнала выше/ниже энергии исходной аудиозаписи.
  2. Увеличение/уменьшение питча голоса диктора, т.е. его высоты.
  3. Изменение длины аудиозаписи – растяжение/сжатие, при этом нужно учитывать изменение питча, если вы используете механизм ресэмплинга.

# Работа с данными. Аугментация спектrogramм.

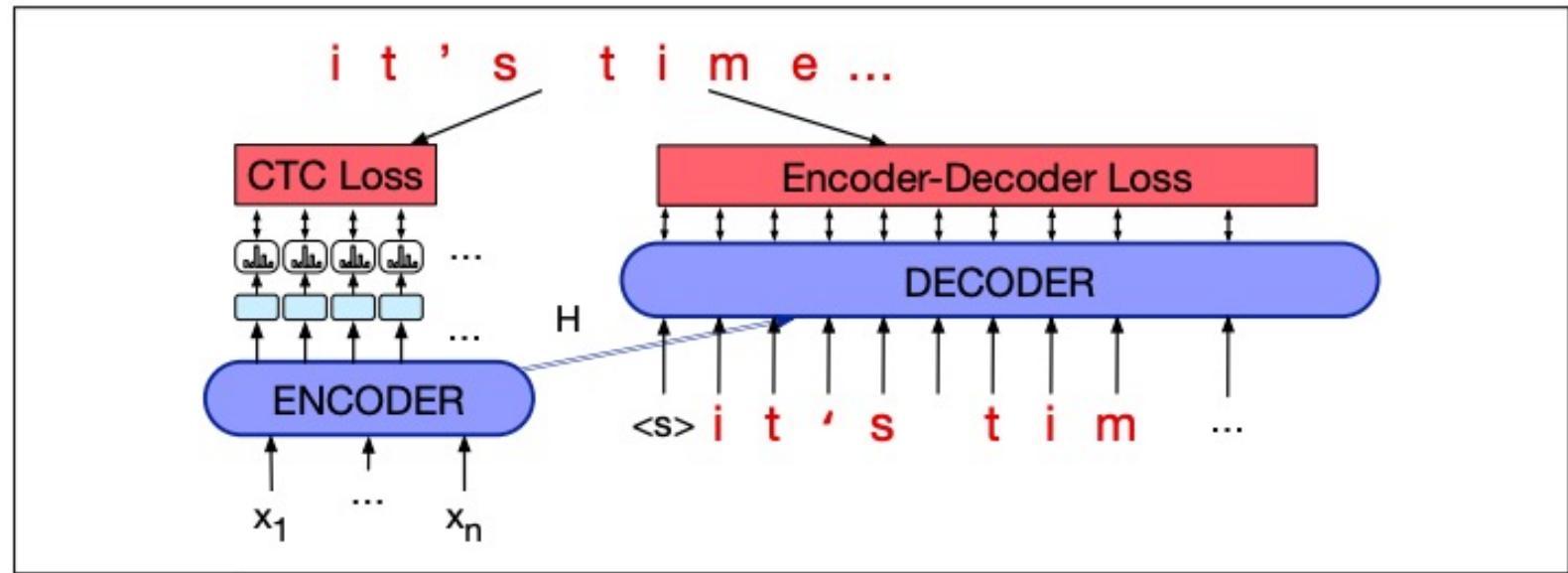
- Основным методом аугментации спектrogramм является **SpecAugment**.  
Данный метод представляет из себя 3 аугментации:
  1. Time warping(зачастую не используется). Деформация участка спектrogramмы по типу.  
Как это делается в изображениях.
  2. Маскирование одной и более частотных полос.
  3. Маскирование одного и более временных фреймов.

При этом, замаскированные фреймы/бины заполняются средним значением спектrogramмы.
- Все эти 3 техники можно применять как совместно, так и по отдельности.



# Комбинирование лосс-функций.

- Отличным подспорьем для получения качественной модели ASR является метод основанный на комбинировании подходов ASR.
- Особой популярностью пользуется связка CTC+LAS.
- Данный подход позволяет ускорить сходимость процесса обучения, зачастую за счет дополнительного сигнала для аудиоэнкодера.
- Так же помимо совместного обучения возможен режим рескоринга СТС-гипотез при помощи LAS-головы (каскадирование).



- Loss-функционал в таком случае выглядит:

$$L = -\lambda \log P_{LAS}(y|x) - (1 - \lambda) \log P_{CTC}(y|x).$$

- Функция скоринга:

$$\hat{y} = \max_y [\lambda_1 \log P_{LAS}(y|x) + \lambda_2 \log P_{CTC}(y|x) + \lambda_3 \log P_{LM}(y) + \lambda_4 \log |y|]$$

# Minimum Word Error Rate Loss.

- Идея MWER Loss достаточно простая – в качестве функционала для оптимизации использовать мат. ожидание кол-ва неправильно распознанных слов(WER):  $L_{MWER}(\mathbf{x}, \mathbf{y}^*) = E[W(\mathbf{y}, \mathbf{y}^*)] = \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) * W(\mathbf{y}, \mathbf{y}^*)$ , где  $W(\mathbf{y}, \mathbf{y}^*)$  - кол-во ошибок(в терминах WER) между гипотезой  $\mathbf{y}$  и правильной транскрипцией  $\mathbf{y}^*$ .
- Данный функционал не возможно оптимизировать напрямую, поскольку нужно просуммировать по всем возможным гипотезам  $\mathbf{y}$ .
- Поэтому выделяют 2 подхода к оценке данного функционала:
  - Оценка через сэмплирование. Случайно сэмплируем примеры транскрипций из модели, далее оцениваем loss-функцию:  $L_{MWER}(\mathbf{x}, \mathbf{y}^*) \approx L_{MWER}^{Sample}(\mathbf{x}, \mathbf{y}^*) = \frac{1}{N} \sum_{\mathbf{y}_i \sim P(\mathbf{y}|\mathbf{x})} W(\mathbf{y}_i, \mathbf{y}^*)$ . Проблема данного подхода в необходимости сэмплирования большого количества примеров для получения несмещенной оценки.
  - Оценка через N-лучших гипотез. Данный метод основывается на том предположении, что основная плотность вероятности концентрируется около N наиболее вероятных гипотез. Обозначим через  $Beam(\mathbf{x}, N) = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$  - мн-во N-лучших гипотез, полученных при помощи механизма beamsearch. Тогда мы сможем аппроксимировать  $L_{MWER}(\mathbf{x}, \mathbf{y}^*) \approx L_{MWER}^{N-best}(\mathbf{x}, \mathbf{y}^*) = \sum_{\mathbf{y}_i \in Beam(\mathbf{x}, N)} \hat{P}(\mathbf{y}_i|\mathbf{x}) [W(\mathbf{y}_i, \mathbf{y}^*) - \bar{W}]$ . Где  $\hat{P}(\mathbf{y}_i|\mathbf{x}) = \frac{P(\mathbf{y}_i|\mathbf{x})}{\sum_{\mathbf{y}_i \in Beam(\mathbf{x}, N)} P(\mathbf{y}_i|\mathbf{x})}$  - перенормированная вероятность по N-лучшим гипотезам, а  $\bar{W}$  - среднее кол-во ошибок по этим же гипотезам(данная величина помогает бороться с дисперсией лосса, при этом не смещает оценку градиентов).

Финальный лосс для обучения выглядит как:  $L(\mathbf{x}, \mathbf{y}^*) = \sum_{(\mathbf{x}, \mathbf{y}^*)} L_{MWER}^{N-best; Sample}(\mathbf{x}, \mathbf{y}^*) + \lambda L_{CE}$ .

# Ссылки на материалы.



# Ссылки на материалы лекции.

- LAS : <https://arxiv.org/pdf/1508.01211.pdf>
- Seq2Seq NN : <https://arxiv.org/pdf/1409.3215.pdf>
- CTC : [https://www.cs.toronto.edu/~graves/icml\\_2006.pdf](https://www.cs.toronto.edu/~graves/icml_2006.pdf)
- CTC DistillPub : <https://distill.pub/2017/ctc/>
- RNN-T : <https://arxiv.org/pdf/1211.3711.pdf>
- RNN-T Advances : <https://arxiv.org/pdf/2103.09935.pdf>
- Audio processing : <https://mac.kaist.ac.kr/~juhan/gct634/Slides/%5Bweek1-3%5D%20audio%20data%20representations.pdf>
- Audio features : <https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html>
- Melfilterbanks : <http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/>
- SpecAugment : <https://arxiv.org/abs/1904.08779>



Спасибо  
за внимание!

Не забудьте отметиться на портале и оставить отзыв!