

Обработка речевых сигналов

Блок 2. Автоматическое распознавание речи

Максим Корневский
Старший научный сотрудник ООО «ЦРТ»,
к.ф.-м.н.



Настоящий блок лекций подготовлен при
поддержке «ЦРТ | Группа компаний»



Блок 2. Автоматическое распознавание речи (Automatic Speech Recognition, ASR)



Часть 5. End-to-end системы распознавания речи



План лекции

- Недостатки современных гибридных систем
- Connectionist Temporal Classification (CTC)
- RNN-Transducer (RNN-T)
- Encoder-Decoder-системы с механизмом внимания (AED)
- Комбинации end-to-end-подходов

План лекции

- Недостатки современных гибридных систем
- Connectionist Temporal Classification (CTC)
- RNN-Transducer (RNN-T)
- Encoder-Decoder-системы с механизмом внимания (AED)
- Комбинации end-to-end-подходов
- Прочее

Недостатки современных гибридных систем

Основных недостатков два:

- Очень длинный и сложно устроенный пайплайн обучения
 - Обучить фонемные (монофонные) GMM-HMM модели
 - Склонировать фонемы в аллофоны, обучить аллофонные GMM-HMM модели, сделать связывание состояний
 - Разметить всю обучающую выборку на связанные состояния аллофонов (сеноны)
 - Обучить кроссэнтропийную DNN-HMM модель
 - Доучить DNN-HMM модель по последовательно-дискриминативному критерию (либо обучить с нуля по LF-MMI)
 - Обучить языковую модель
 - Построить граф распознавания
- Акустическая и языковая модели учатся **отдельно** – это приводит к субоптимальным результатам

Недостатки современных гибридных систем

На что хотелось бы заменить эту процедуру:

- Есть всего **ОДНА модель**, которая инкапсулирует в себе свойства акустической модели, языковой модели
- Обучение модели происходит **за один этап** (возможно, длительный)
- Для обучения модели не требуется глубоких знаний фонетики и прочих свойств языка. Все, что требуется для обучения – это **фонограммы** и соответствующие им **текстовые расшифровки**
- Для распознавания желательно просто «пропустить» звук через модель и на выходе получить распознанный текст (т.е. в идеале модель должна заменять и декодер!)
- Системы, обладающие этими свойствами (всеми или некоторыми), называются **end-to-end (E2E)**

Вероятностная постановка задачи распознавания речи

- Произнесена последовательность слов $W = (w_1, w_2, \dots, w_n)$
- По ней получена последовательность наблюдений $O = (o_1, o_2, \dots, o_T)$
- Как, зная O , найти W ?

$$W = \arg \max_W P(W|O) = \arg \max_W \frac{p(O|W)P(W)}{p(O)} = \arg \max_W p(O|W)P(W)$$

План лекции

- Недостатки современных гибридных систем
- **Connectionist Temporal Classification (CTC)**
- RNN-Transducer (RNN-T)
- Encoder-Decoder-системы с механизмом внимания (AED)
- Комбинации end-to-end-подходов
- Прочее

Connectionist Temporal Classification (CTC)

Alex Graves, etal. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, 2006

- Пусть у нас есть нейросеть, которая должна выдавать последовательность l «символов» (токенов) из некоторого алфавита A (слова, слоги, буквы и т.д.), принимая на вход последовательно акустических наблюдений O длиной T
- Как можно было бы оценить вероятность $P(l|O)$?
- Есть две проблемы:
 - Последовательность l , как правило, значительно короче последовательности O
 - Мы не знаем, когда конкретно наша сеть должна выдавать тот или иной символ (т.е. последовательности НЕ ВЫРОВНЕННЫ относительно друг друга)

Connectionist Temporal Classification (CTC)

Отказ от явного выравнивания

- Добавим к сети один дополнительный выход, соответствующий специальному «пустому» (blank) символу ϵ , и потребуем, чтобы один из символов расширенного алфавита $A' = A \cup \{\epsilon\}$ выдавался на каждом кадре входной последовательности
- Такие последовательности выходных символов π длиной T назовем «путями»
- Каждому «пути» π сопоставим выходную последовательность $l = F(\pi)$ по правилу:
 - Сначала все последовательности одинаковых символов заменить одним
 - Потом удалить все blank-символы
 - Пример: $\pi = \epsilon\epsilon\text{л}\text{л}\text{л}\text{л}\epsilon\epsilon\epsilon\epsilon\epsilon\epsilon\epsilon\text{м}\text{м}\epsilon\epsilon\epsilon\epsilon\text{м}\text{м}\text{м}\text{м}\text{м}\text{м}\epsilon\epsilon\epsilon\text{а}\epsilon\epsilon \Rightarrow \epsilon\text{л}\epsilon\epsilon\epsilon\text{м}\epsilon\text{а}\epsilon \Rightarrow \text{лемма} = l$
- Одной и той же последовательности l может соответствовать множество путей π . Тогда

$$P(l|O) = \sum_{\pi: F(\pi)=l} P(\pi|O).$$

Connectionist Temporal Classification (CTC)

Отказ от явного выравнивания

- Предположим, что токены, выдаваемые на различных фреймах, **независимы при фиксированных параметрах нашей сети**. Тогда

$$P(\pi|O) = \prod_{t=1}^T P(\pi_t|O) = \prod_{t=1}^T y_{\pi_t}(t),$$

где $y(t)$ – вектор выходов нашей нейронной сети на фрейме t

- Следовательно,

$$P(l|O) = \sum_{\pi: F(\pi)=l} \prod_{t=1}^T y_{\pi_t}(t).$$

- Различных путей **экспоненциально** много. Как искать эту вероятность на практике?

Connectionist Temporal Classification (CTC)

























































































Forward алгоритм для CTC

- Вместо l будем рассматривать вспомогательные последовательности токенов l' , получаемые из l добавлением blank между всеми токенами l и с краев
- Введем **forward-вероятность** $\alpha(t, u)$ как суммарную вероятность всех путей длины t , которые отображаются в префикс l длины $\lfloor u/2 \rfloor$, т.е.

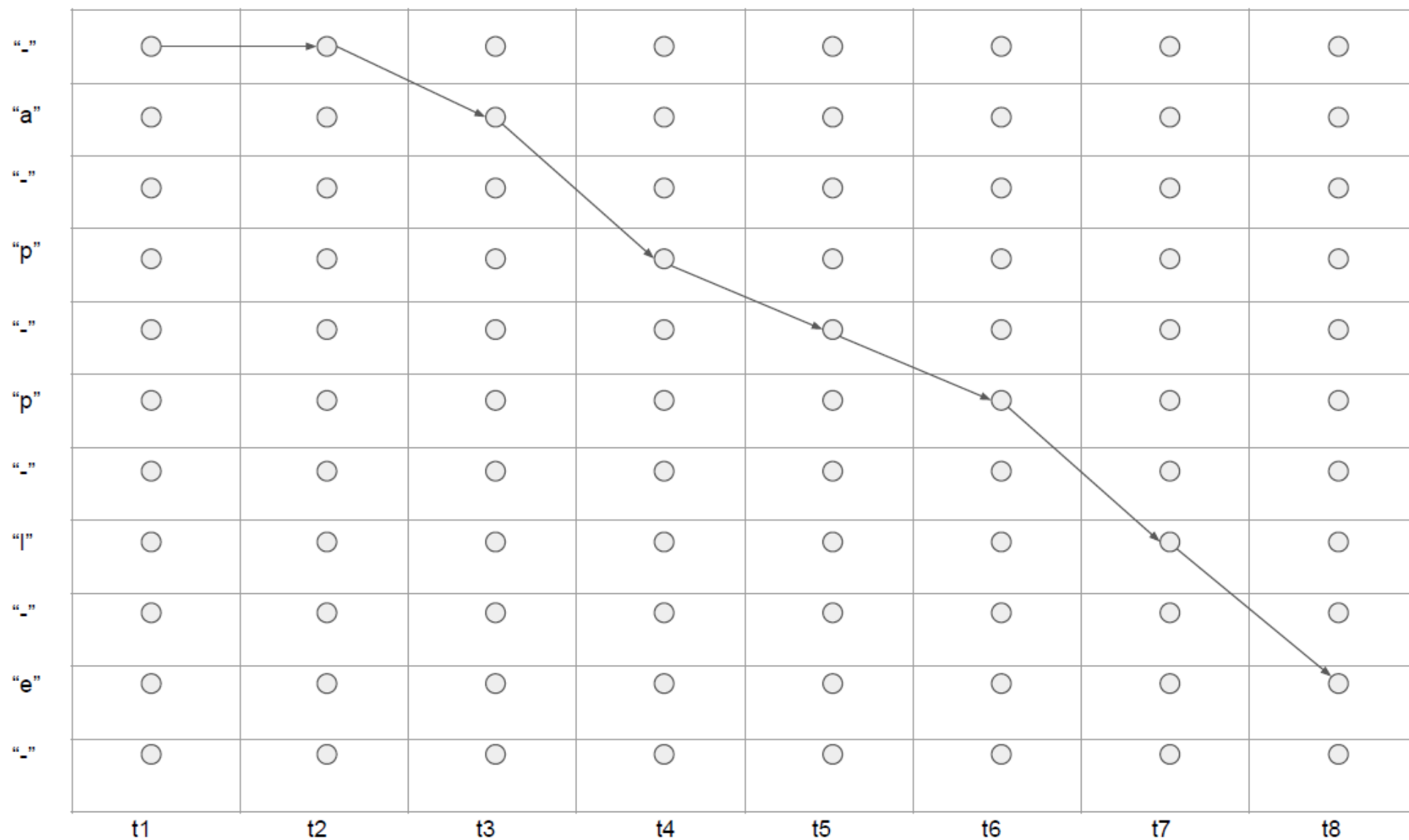
$$\alpha(t, u) = \sum_{\pi: F(\pi)=l_{1:\lfloor u/2 \rfloor}, \pi_t=l'_u} \prod_{i=1}^t y_{\pi_i}(i)$$

- Можно понять, что эту величину тоже можно пересчитывать рекуррентно
- В отличие от Forward-Backward-алгоритма для HMM, тут необходимо учитывать, что добавление к пути blank-токена ϵ не изменяет префикса l !

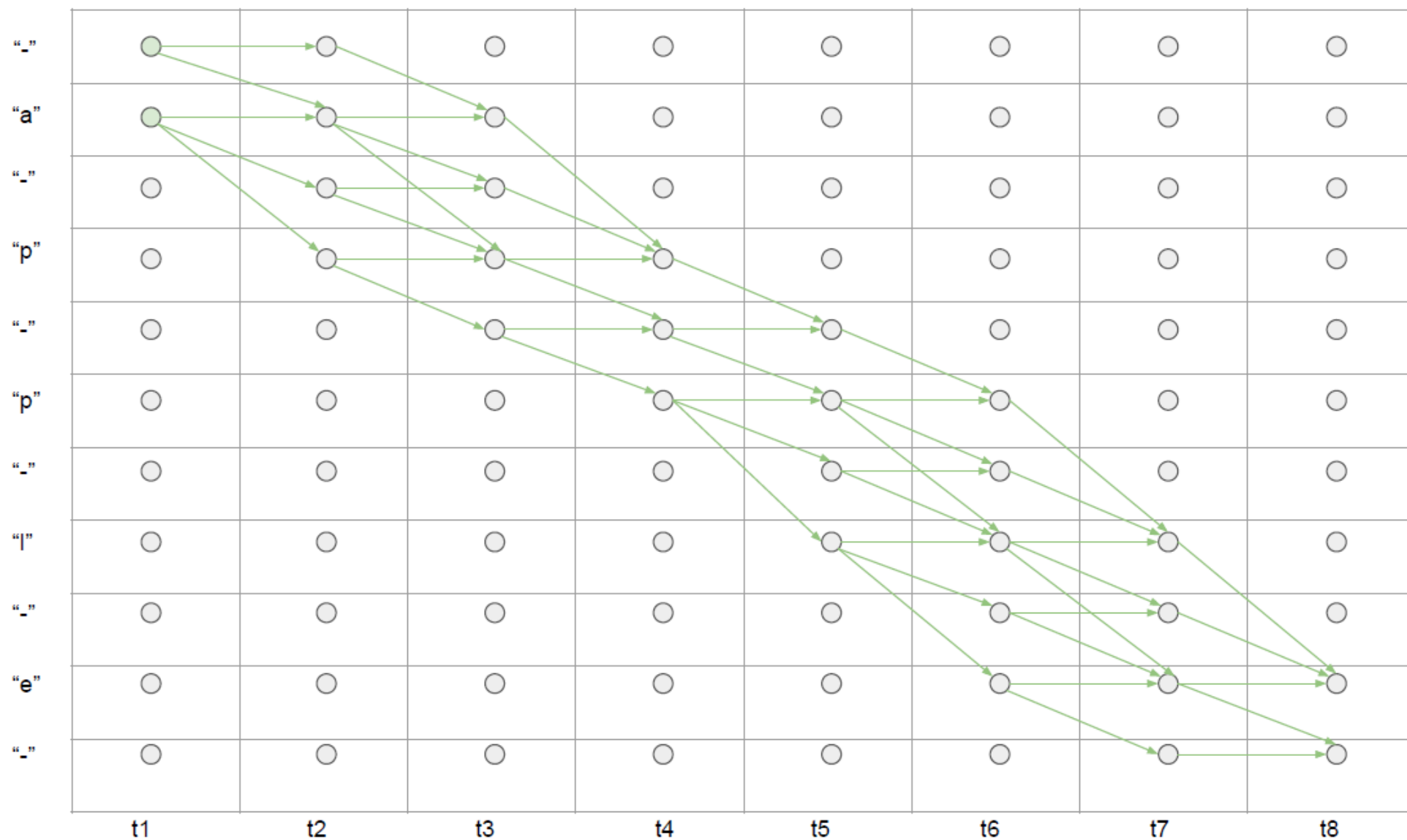
Connectionist Temporal Classification (CTC)

" - "								
"a"								
" - "								
"p"								
" - "								
"p"								
" - "								
"l"								
" - "								
"e"								
" - "								
	t1	t2	t3	t4	t5	t6	t7	t8

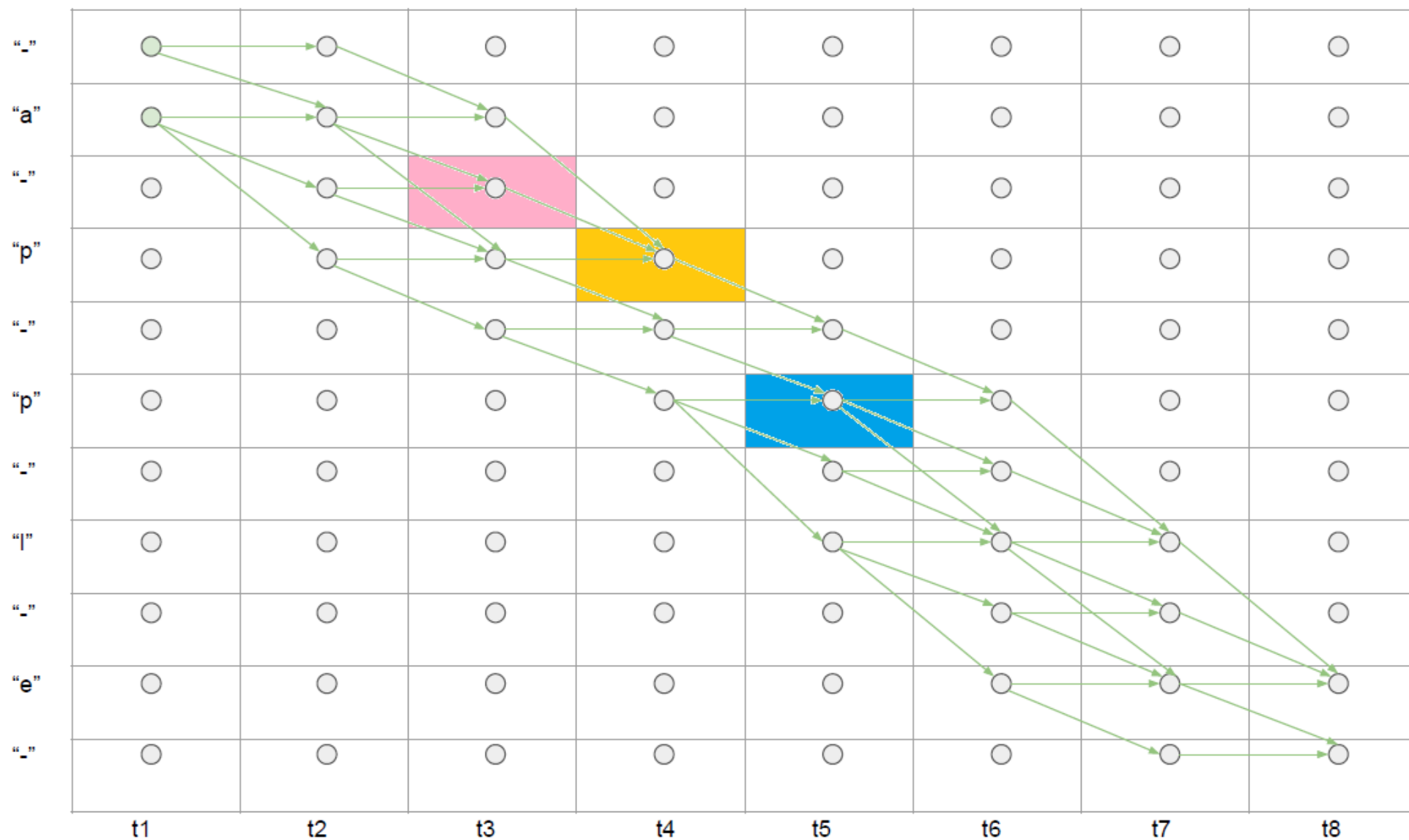
Connectionist Temporal Classification (CTC)



Connectionist Temporal Classification (CTC)



Connectionist Temporal Classification (CTC)



Connectionist Temporal Classification (CTC)

Forward алгоритм для CTC

- В результате рассмотрения всех возможных вариантов, получаем такую формулу:

$$\alpha(t, u) = y_{l'_u}(t) \sum_{i=f(u)}^u \alpha(t-1, i)$$

где

$$f(u) = \begin{cases} u-1, & \text{если } l'_u = \epsilon \text{ или } l'_{u-2} = l'_u \\ u-2, & \text{в противном случае} \end{cases}$$

с начальными условиями

$$\alpha(1,1) = y_{\epsilon}(1), \quad \alpha(1,2) = y_{l_1}(1), \quad \alpha(1,u) = 0 \text{ при } u > 2, \quad \alpha(t,0) = 0 \text{ для всех } t.$$

- Окончательно,

$$P(\mathbf{l}|\mathbf{O}) = \alpha(T, 2L) + \alpha(T, 2L+1).$$

Connectionist Temporal Classification (CTC)

Backward алгоритм для CTC

- Аналогично вводится **backward-вероятность**, как сумма вероятностей всех путей, дополняющих пути из forward-вероятности до полной длительности. Для нее

$$\beta(t, u) = \sum_{i=u}^{g(u)} \beta(t+1, i) y_{l'_i}(t+1)$$

где

$$g(u) = \begin{cases} u+1, & \text{если } l'_u = \epsilon \text{ или } l'_{u+2} = l'_u \\ u+2, & \text{в противном случае} \end{cases}$$

с начальными условиями

$$\beta(T, 2L+1) = \beta(T, 2L) = 1, \quad \beta(T, u) = 0 \text{ при } u < 2L, \quad \beta(t, 2L+2) = 0 \text{ для всех } t.$$

- Если все $\beta(t, u)$ вычислены, то

$$P(\mathbf{l}|\mathbf{O}) = \beta(1,1)y_{\epsilon}(1) + \beta(1,2)y_{l_1}(1) = \beta(0,1).$$

Connectionist Temporal Classification (CTC)

Forward-Backward алгоритм для CTC

- Если для данных t, u вычислены forward и backward-вероятности, то их произведение дает суммарную вероятность всех полных путей, на которых в момент t генерируется токен l'_u :

$$\alpha(t, u)\beta(t, u) = \sum_{\pi: F(\pi)=l, \pi_t=l'_u} P(\pi|O)$$

- Поскольку это может быть любой из токенов l' , то по формуле полной вероятности для любого $1 \leq t \leq T$ справедлива формула

$$P(l|O) = \sum_{u=1}^{2L+1} \alpha(t, u)\beta(t, u)$$

Connectionist Temporal Classification (CTC)

Функция потерь и обучение

- **Функция потерь (CTC loss)** – суммарный логарифм вероятностей по всем обучающим парам $(O, l) \in S$, взятый с обратным знаком:

$$\mathcal{L}_{CTC} = - \sum_{(O, l) \in S} \log P(l|O) = - \sum_{(O, l) \in S} \mathcal{L}(l|O)$$

- С помощью довольно несложных выкладок можно получить формулу для производных функции потерь по выходам последнего слоя (входам софтмакса):

$$\frac{\partial \mathcal{L}(l|O)}{\partial z_i(t)} = y_i(t) - \frac{1}{P(l|O)} \sum_{u: l'_u = i} \alpha(t, u) \beta(t, u)$$

- Таким образом, для back-propagation тоже необходимо вычислить все forward и backward вероятности.

Connectionist Temporal Classification (CTC)

Распознавание (декодирование)

- По последовательности O хотелось бы найти такую последовательность l , которая наиболее вероятна:

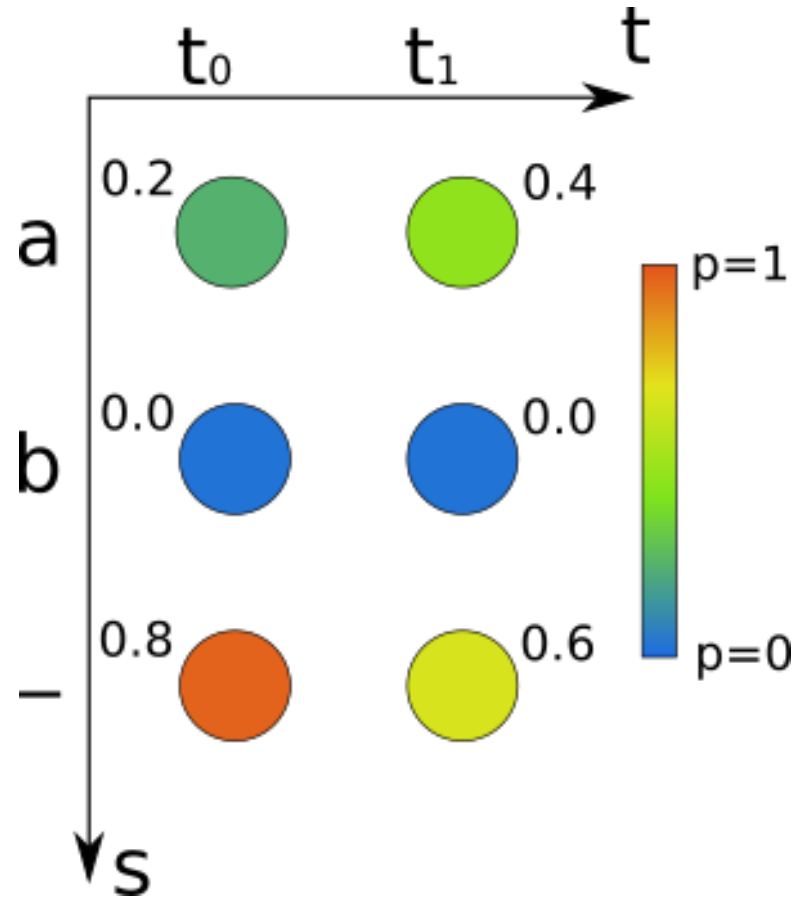
$$l^* = \arg \max_l P(l|O)$$

- Точный и эффективный алгоритм для этого пока неизвестен. Но существуют несколько альтернатив:
 - Best-path search** (жадный): найти наиболее вероятный путь π и преобразовать его в $l = F(\pi)$
 - Prefix search**: на каждом шаге расширять только тот префикс выходной строки, для которого суммарная вероятность всех расширений максимальна.
 - Prefix search находит наилучшую последовательность l , но число префиксов может расти экспоненциально.
 - Выход: отсечка (pruning) маловероятных префиксов.
 - Минус**: возможная потеря лучшей последовательности. **Плюс**: возможность использования внешней ЯМ.
 - Использование сети просто в качестве **акустической модели** и **beam-search по WFST-графу** (как в гибриде).

Connectionist Temporal Classification (CTC)

Свойства CTC:

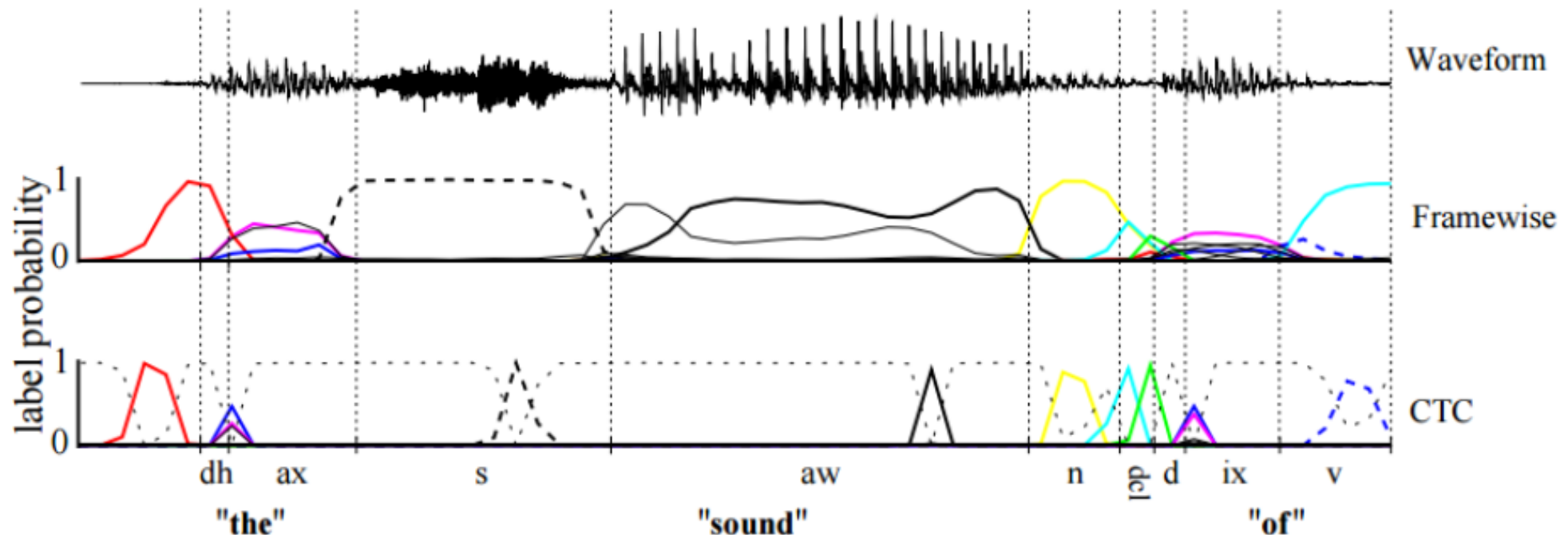
- Жадный алгоритм может найти путь π , который соответствует не лучшей строке l :



Connectionist Temporal Classification (CTC)

Свойства CTC:

- Жадный алгоритм может найти путь π , который соответствует не лучшей строке l :
- Распределения на выходе нейронной сети обычно «острые» (peaky):



Connectionist Temporal Classification (CTC)

Свойства CTC:

- Жадный алгоритм может найти путь π , который соответствует не лучшей строке l :
- Распределения на выходе нейронной сети обычно «острые» (peaky):
- Возможные элементы алфавита:
 - Фонемы: CTC-сеть лучше всего использовать просто как акустическую модель в WFST-фреймворке
 - Графемы: «буквы», пробел и знаки препинания. Возможно появление буквосочетаний, не являющихся словами
 - Целые слова: требуется очень много данных для обучения (система от Google училась на 125 000 часов)
 - Компромиссный вариант – subword units (слоги, буквосочетания): возможно хорошо обучать на умеренных объемах данных, реже порождают несуществующие слова.
 - Наиболее популярный способ выбора subword units - Byte Pair Encoding (BPE), легко регулировать количество в зависимости от объема обучающих данных
- Для лучшей сходимости рекомендуется подавать на обучение примеры, отсортированные по увеличению длительности.

План лекции

- Недостатки современных гибридных систем
- Connectionist Temporal Classification (CTC)
- **RNN-Transducer (RNN-T)**
- Encoder-Decoder-системы с механизмом внимания (AED)
- Комбинации end-to-end-подходов
- Прочее

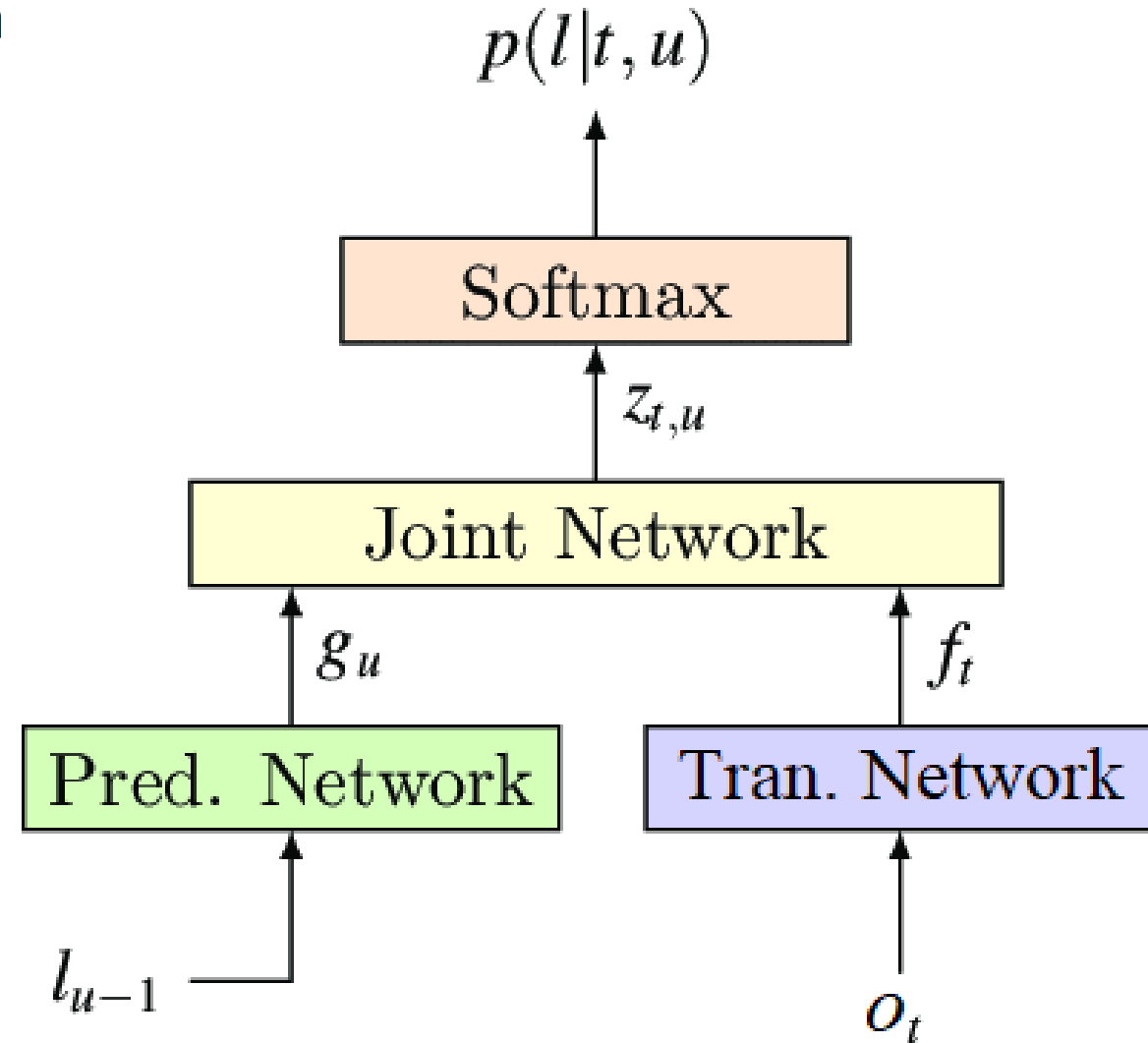
RNN-Transducer (RNN-T)

A. Graves. Sequence transduction with recurrent neural networks, 2012

- CTC работает только для последовательностей токенов, которые **короче** последовательности наблюдений
- CTC умеет только предсказывать текущий токен алфавита **по наблюдениям**
- CTC считает токены расширенного алфавита **независимыми** на разных фреймах
- Было бы здорово уметь еще предсказывать следующий токен по предыдущим – т.е. внедрить в модель языковую информацию
- Идея – сделать три различные сети:
 - **Transcription Network** (TN или Encoder) для предсказания текущего токена по акустике (признакам)
 - **Prediction Network** (PN) для предсказания текущего токена по последовательности предыдущих токенов
 - **Joint Network** (JN), которая объединяет предсказания TN и PN для финального предсказания
- Таким образом, TN – аналог АМ, PN – аналог ЯМ, а JN – аналог декодера ASR

RNN-Transducer (RNN-T)

- TN выдает последовательность «transcription vectors» f_t на каждом фрейме o_t
- PN выдает вектор g_u апостериорных вероятностей для u -го элемента l
- JN тоже выдает апостериорные вероятности $P(k|u, t)$ токенов алфавита или пустого символа \emptyset , но уже для пары индексов (t, u)
- Выдача сетью \emptyset соответствует переходу на следующий фрейм без смены индекса u
- Выдача непустого символа происходит без смены фрейма t



RNN-Transducer (RNN-T)

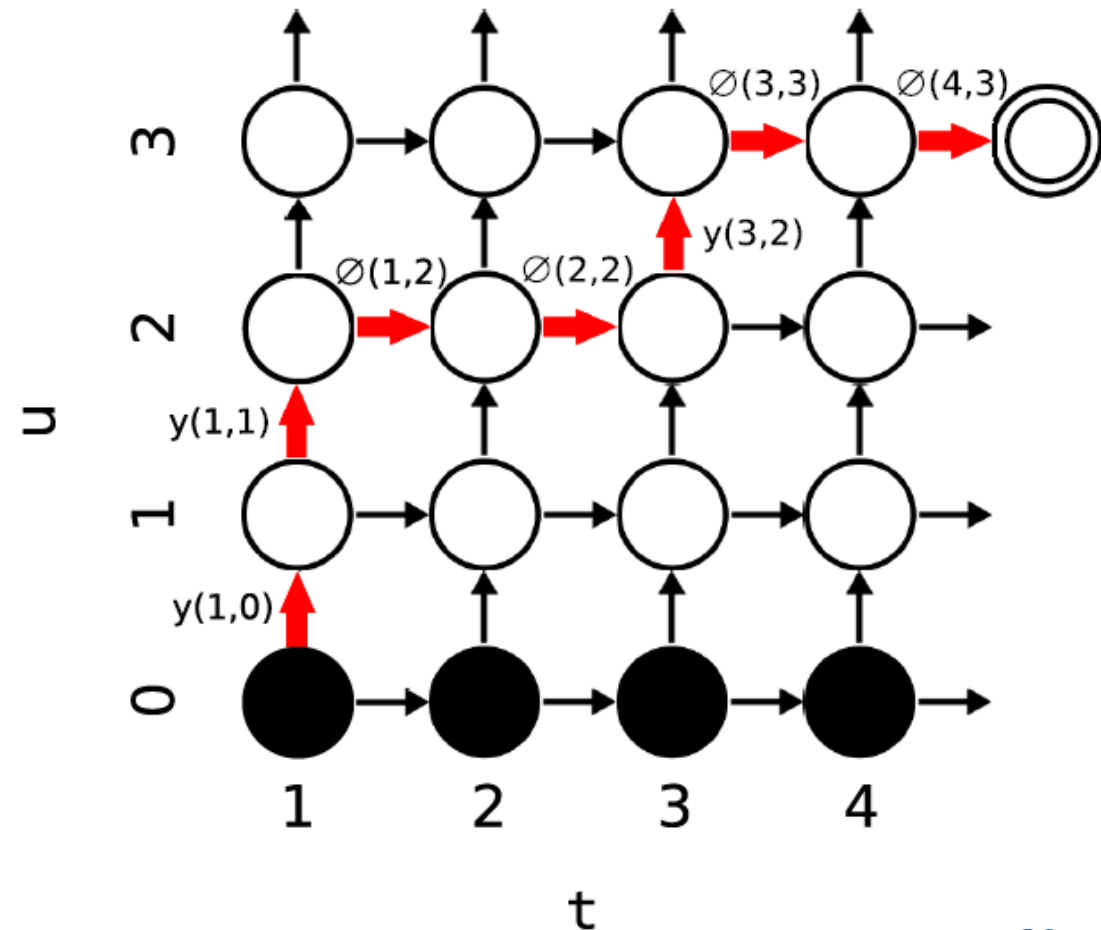
Базовый алгоритм работы RNN-T:

1. Положить $t = 1, u = 0, l_0 = \emptyset$
2. Подать o_1 на TN, вычислить f_1 , подать l_0 на PN, вычислить g_0
3. Подать f_t и g_u на JN, вычислить набор апостериорных вероятностей $P(k|u, t)$
4. Если максимальная вероятность $P(k^*|u, t)$ соответствует пустому символу \emptyset ,
 - при $t < T$ положить $t \leftarrow t + 1$, подать o_t на TN, вычислить f_t
 - в противном случае перейти к шагу 7
5. Иначе положить $u \leftarrow u + 1, l_u = A_{k^*}$, подать l_u на PN, вычислить g_u
6. Перейти к шагу 3
7. Выдать последовательность l .

RNN-Transducer (RNN-T)

Выравнивания и вычисление вероятности последовательности

- **Выравнивание** – последовательность y произвольной длины из алфавита $A \cup \{\emptyset\}$
- Каждому выравниванию соответствует последовательность из A , получаемая простым выкидыванием всех \emptyset : $l = B(y)$
- Если выходная последовательность длиной U , получена на T фреймах, то ей соответствуют выравнивания длиной $T + U$ на сети справа:
- Здесь использованы обозначения
 - $y(t, u) = P(l_{u+1}|t, u)$, $\emptyset(t, u) = P(\emptyset|t, u)$



RNN-Transducer (RNN-T)

Forward-Backward алгоритм для RNN-T:

- Определим **forward-вероятность** $\alpha(t, u)$ как вероятность выдать символы $l_{[1..u]}$, обработав transcription vectors $f_{[1..t]}$
- Легко понять, что справедлива рекуррентная формула
$$\alpha(t, u) = \alpha(t - 1, u)\phi(t - 1, u) + \alpha(t, u - 1)y(t, u - 1), \quad \alpha(1, 0) = 1.$$
- Если вычислены все $\alpha(t, u)$, то можно найти $P(I|O) = \alpha(T, U)\phi(T, U)$.
- Аналогично $\beta(t, u)$ это вероятность выдать символы $l_{[(u+1)..U]}$, обработав $f_{[t..T]}$:
$$\beta(t, u) = \beta(t + 1, u)\phi(t, u) + \beta(t, u + 1)y(t, u), \quad \beta(T, U) = \phi(T, U), \quad P(I|O) = \beta(1, 0)$$
- Можно получить еще одну формулу для вероятности полной последовательности:

$$P(I|O) = \sum_{t, u: t+u=n} \alpha(t, u)\beta(t, u), \quad \text{для любого } n = 1, \dots, T + U$$

RNN-Transducer (RNN-T)

Функция потерь и обучение

- Формально **функция потерь** совпадает с CTC loss:

$$\mathcal{L}_{RNNT} = - \sum_{(O,I) \in \mathcal{S}} \log P(I|O) = - \sum_{(O,I) \in \mathcal{S}} \mathcal{L}(I|O)$$

- Для производных функции потерь по выходам **Joint Network** можно получить следующую формулу:

$$\frac{\partial \mathcal{L}(I|O)}{\partial P(k|t,u)} = \frac{\alpha(t,u)}{P(I|O)} \begin{cases} \beta(t+1,u), & \text{если } k = \emptyset \\ \beta(t,u+1), & \text{если } k = l_{u+1} \\ 0, & \text{иначе} \end{cases}$$

- Таким образом, вычислив все $\alpha(t,u)$ и $\beta(t,u)$, можно выполнять **back-propagation** через **Joint Network** и далее через **Prediction Network** и **Transcription Network**

RNN-Transducer (RNN-T)

Распознавание с RNN-T:

- Как и в CTC, задача распознавания – найти наиболее вероятную последовательность:

$$l^* = \arg \max_l P(l|O)$$

- Базовый (жадный) алгоритм – обычно работает намного лучше, чем для CTC
- Beam search организовать несколько сложнее, т.к. для каждой гипотезы необходимо хранить внутреннее состояние Prediction Network
- Использование внешней языковой модели:
 - Shallow fusion: $l^* = \arg \max_l (\log P(l|O) + \lambda \log P_{LM}(l))$
 - Cold fusion, deep fusion – внешняя ЯМ (обычно нейронная) встраивается в модель, скрытое состояние ЯМ объединяется со скрытым состоянием RNN-T внутри JN и такая модель доучивается уже с информацией от внешней ЯМ

RNN-Transducer (RNN-T)

Детали обучения:

- RNN-T может выдавать по несколько токенов за фрейм. Обычно это количество принудительно ограничивают (как правило, порог 3 работает хорошо)
- Если данных для обучения немного, а емкость PN и JN достаточно велика, они могут просто «выучить» все обучающие последовательности токенов – **переобучение**
- При обучении на вход PN поступают истинные токены обучающей последовательности, а при распознавании – распознанные, т.е. присутствует **mismatch**
- Чтобы этого избежать, при обучении с вероятностью ϵ подают токен, сгенерированный на предыдущем шаге, а с вероятностью $(1 - \epsilon)$ – истинный, причем ϵ постепенно возрастает в ходе обучения. Этот подход называется **scheduled sampling**

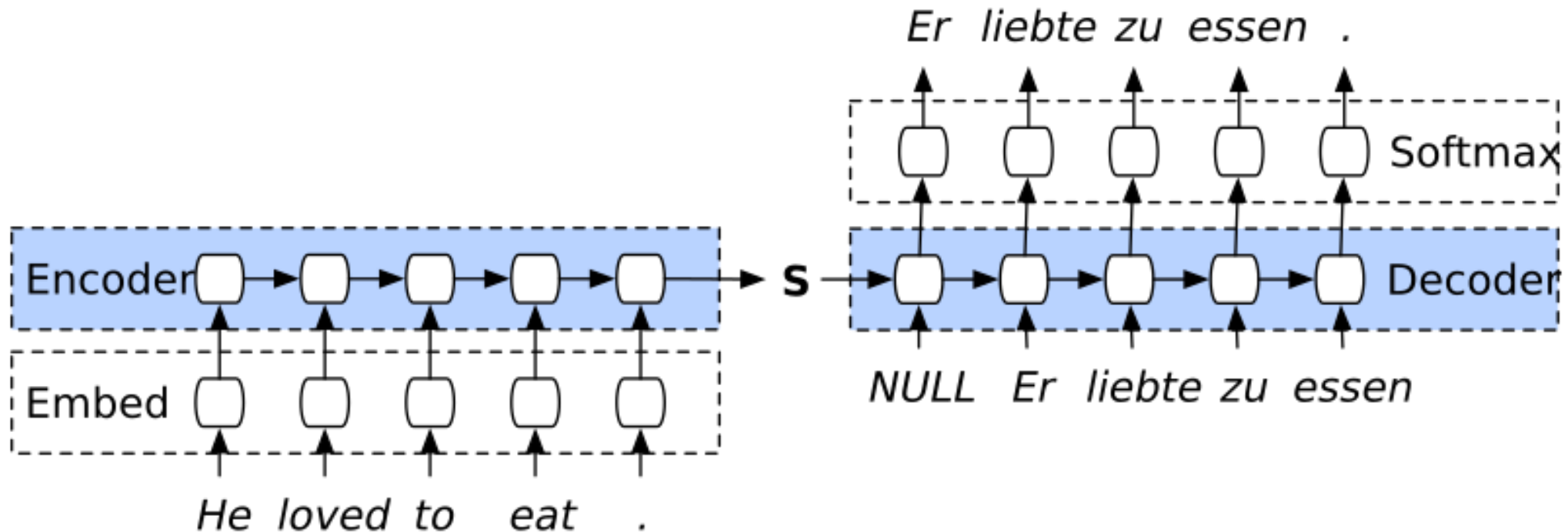
План лекции

- Недостатки современных гибридных систем
- Connectionist Temporal Classification (CTC)
- RNN-Transducer (RNN-T)
- **Encoder-Decoder-системы с механизмом внимания (AED)**
- Комбинации end-to-end-подходов
- Прочее

Attention-based Encoder-Decoder (AED) системы

Классическая Encoder-Decoder-архитектура

- Впервые предложена для задач [sequence-to-sequence](#) (seq2seq), в частности, для машинного перевода:

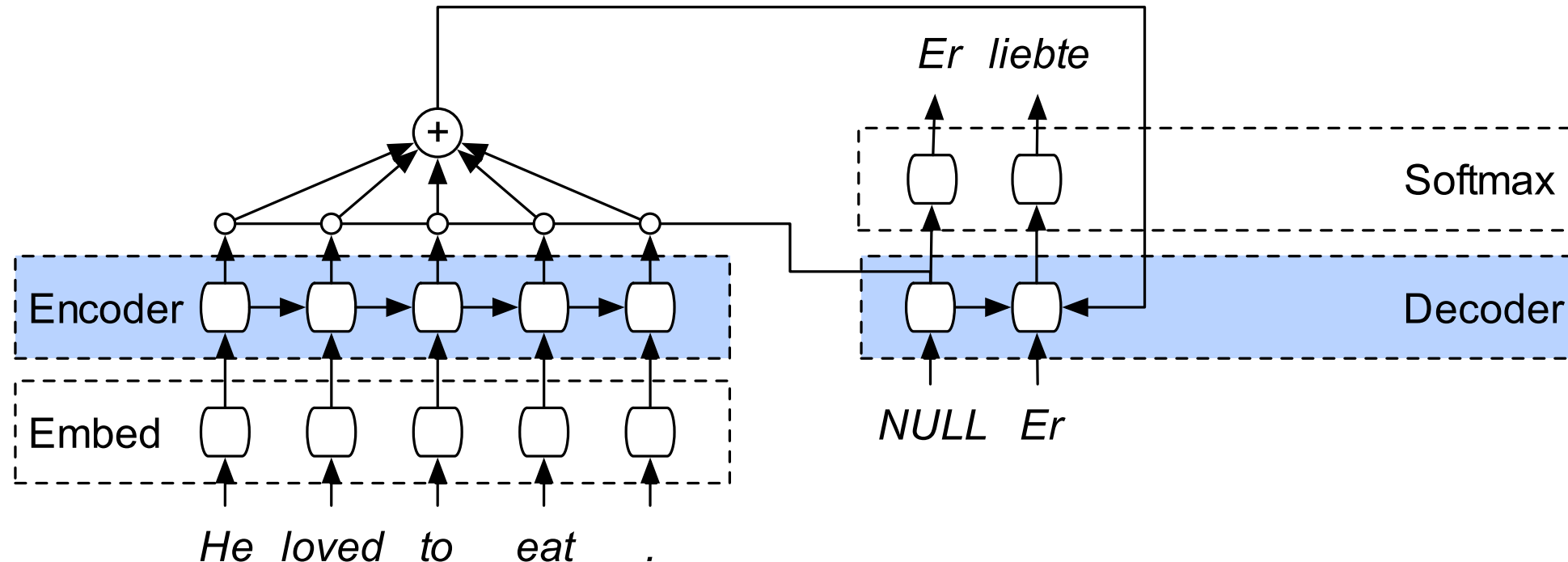


- Недостаток: плохо справляется с длинными последовательностями

Attention-based Encoder-Decoder (AED) системы

Добавление модуля внимания (attention):

- На каждом шаге декодера смотрим на **всю** входную последовательность и собираем информацию от векторов скрытых состояний **энкодера**



- Механизм внимания строит вероятностное «**выравнивание**» между входом и выходом

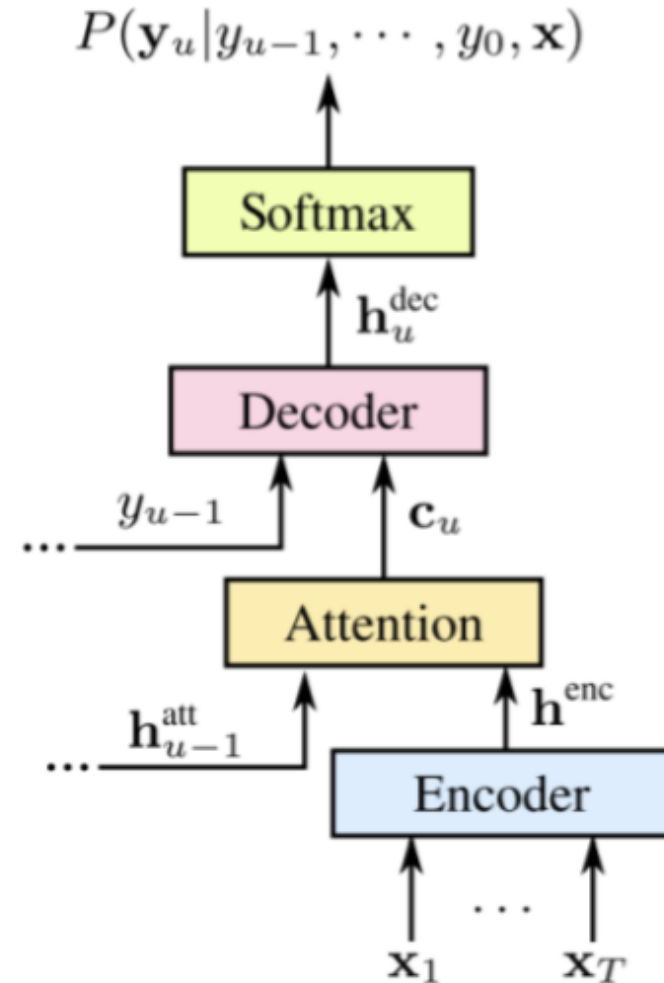
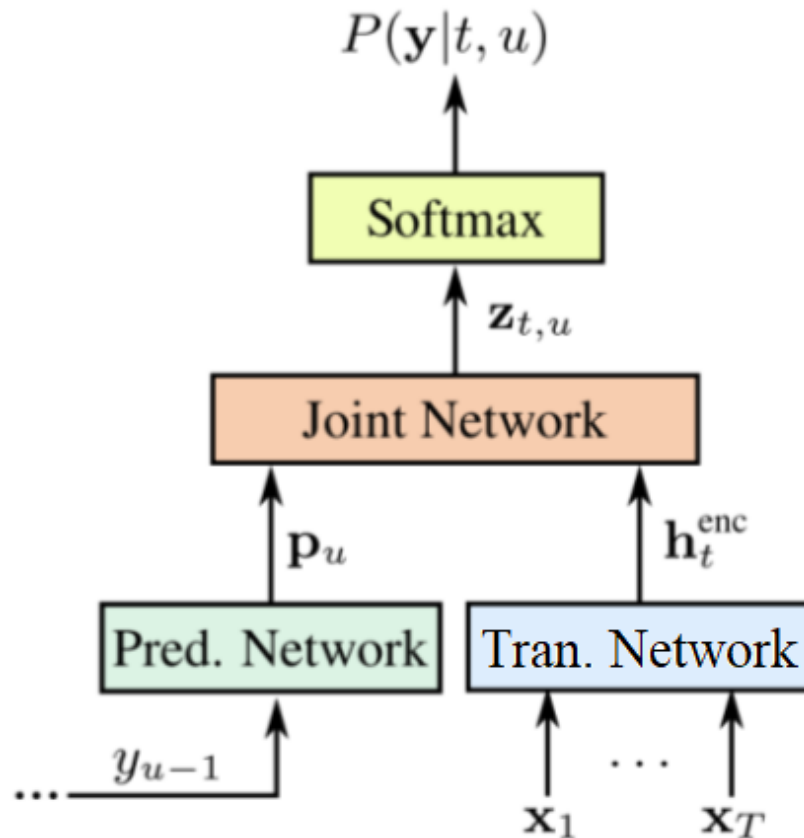
Attention-based Encoder-Decoder (AED) системы

Особенности механизма внимания для речи

- Длины последовательностей могут сильно различаться. Если учиться только на коротких, потом длинные плохо распознаются и наоборот. Чтобы устранить эту зависимость, используют **позиционную информацию** в attention
- То, куда было обращено внимание на предыдущем шаге, важно для нового шага. Поэтому прошлый вектор скрытого состояния attention часто используют при вычислении нового attention-вектора
- В задачах машинного перевода выравнивание не обязано быть «**МОНОТОННЫМ**». В речи – обязано, но базовый attention этого не гарантирует. Поэтому часто используется специальные варианты (например, **Monotonic Chunkwise Attention**)
- Для ускорения часто диапазон вычисления attention ограничивают окном вокруг фрейма, на котором предыдущий attention был максимален.

Attention-based Encoder-Decoder (AED) системы

Сравнение RNN-T и AED:



Attention-based Encoder-Decoder (AED) системы

Функция потерь и обучение

- Декодер выдает непосредственно вероятности очередного символа при заданной истории и наблюдениях $P(l_u | l_{u-1}, \dots, l_0, O)$
- Поэтому легко найти апостериорную вероятность $P(\mathbf{l} | O)$, которую максимизирует ASR:

$$P(\mathbf{l} | O) = \prod_{u=1}^U P(l_u | l_{u-1}, \dots, l_0, O)$$

- Следовательно **функция потерь**

$$\mathcal{L} = - \sum_{(O, \mathbf{l}) \in \mathcal{S}} \log P(\mathbf{l} | O) = - \sum_{(O, \mathbf{l}) \in \mathcal{S}} \sum_{u=1}^{U_{\mathbf{l}}} \log P(l_u | l_{u-1}, \dots, l_0, O)$$

непосредственно зависит от выходов декодера. Поэтому производные вычисляются элементарно

Attention-based Encoder-Decoder (AED) системы

Распознавание

- Жадный алгоритм – обычно работает достаточно хорошо
- Beam search – с теми же оговорками, что для RNN-T
- Использование внешней ЯМ:
 - Shallow fusion, cold fusion, deep fusion
 - В стандартной целевой функции $\log P(\mathbf{l}|O) + \lambda \log P_{LM}(\mathbf{l})$ первое слагаемое зависит от длины последовательности \mathbf{l} . Поэтому используются альтернативы:

$$\frac{\log P(\mathbf{l}|O)}{|\mathbf{l}|} + \lambda \log P_{LM}(\mathbf{l}) \quad \text{или} \quad \log P(\mathbf{l}|O) + \lambda \log P_{LM}(\mathbf{l}) - \gamma |\mathbf{l}|$$

Attention-based Encoder-Decoder (AED) системы

Listen, Attend and Spell (LAS). Google, 2015

- Архитектура, которая лежит в основе многих современных AED-систем
- Энкодер (listener) – «пирамидальная» BLSTM, для уменьшения длины входной последовательности
- Описывается следующими соотношениями:

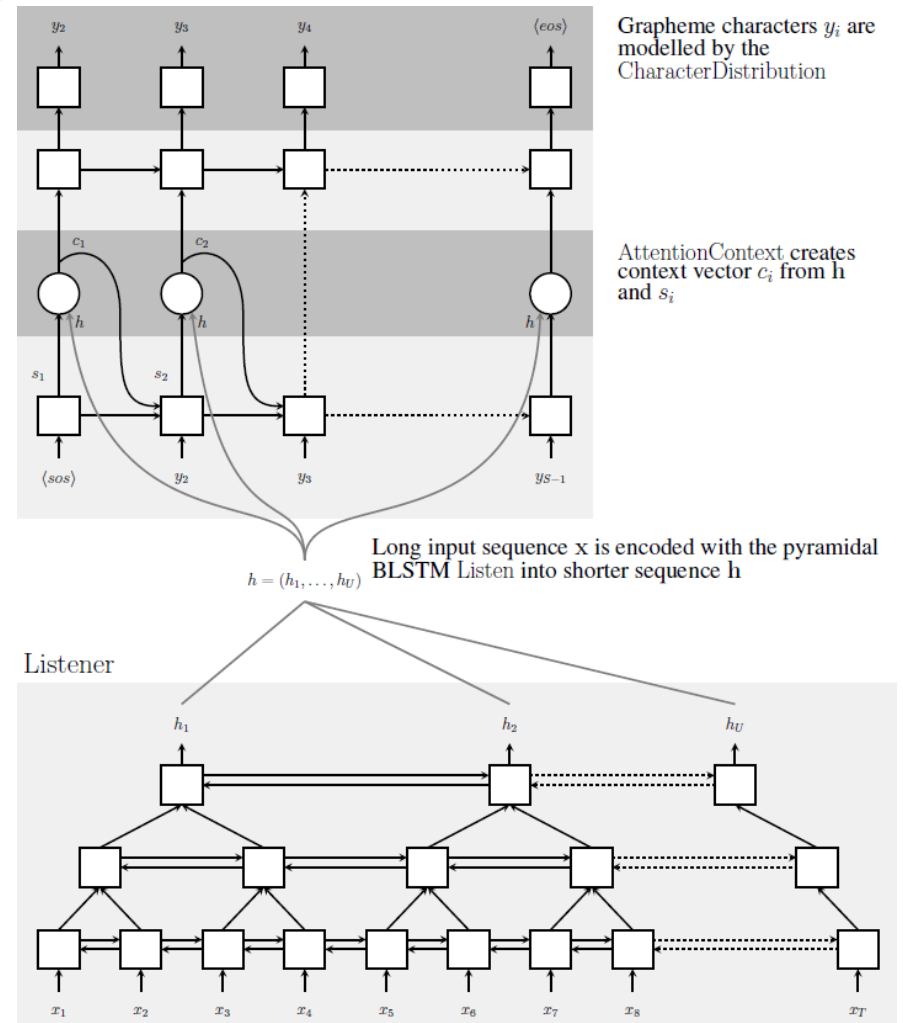
$$\mathbf{h} = \text{Listen}(\mathbf{x}), \quad P(\mathbf{y}|\mathbf{x}) = \text{AttendAndSpell}(\mathbf{h}, \mathbf{y})$$

где AttendAndSpell – декодер с attention вида

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

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

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



План лекции

- Недостатки современных гибридных систем
- Connectionist Temporal Classification (CTC)
- RNN-Transducer (RNN-T)
- Encoder-Decoder-системы с механизмом внимания (AED)
- Комбинации end-to-end-подходов
- Прочее

Комбинация различных end-to-end подходов

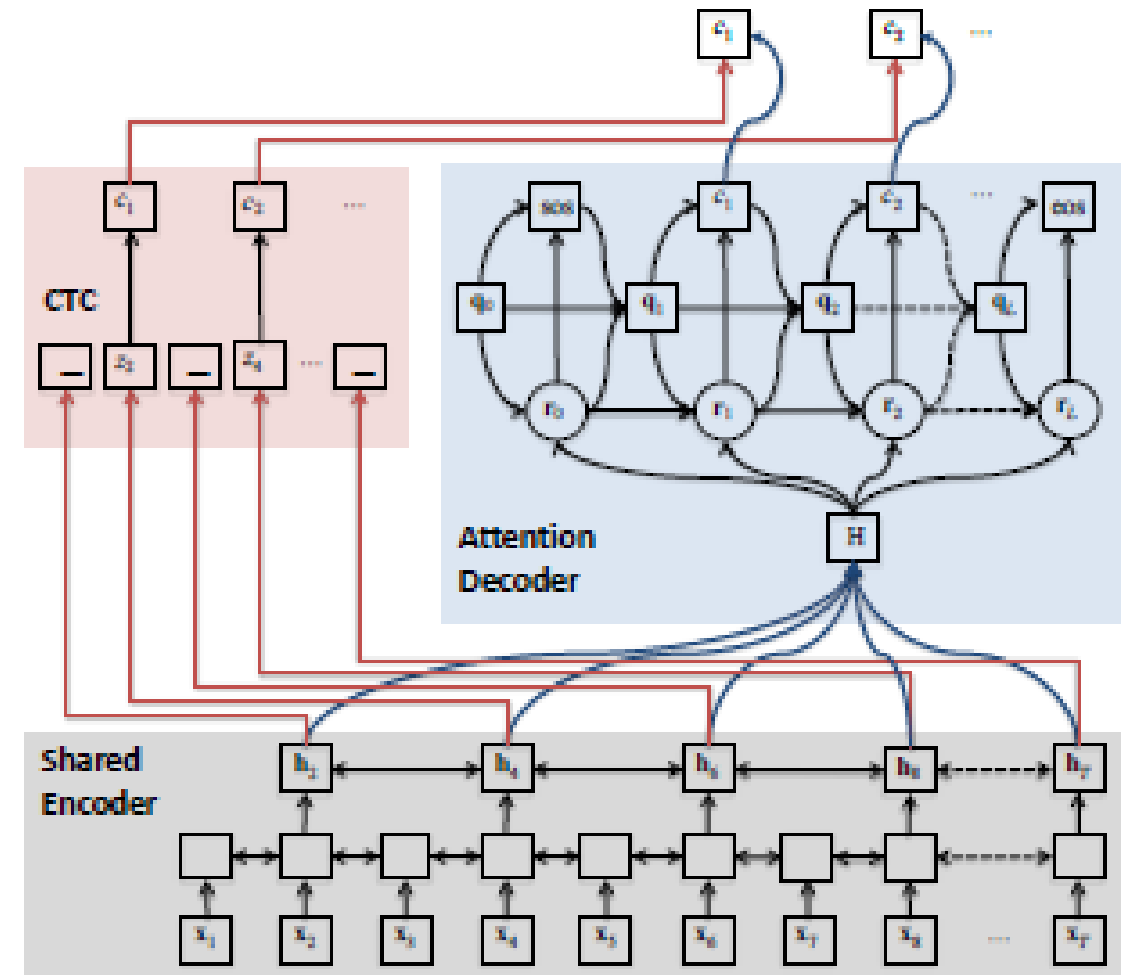
Мотивация

- Различные E2E-системы обладают своими достоинствами и недостатками
- Идея – как-то скомбинировать различные подходы, чтобы нивелировать недостатки и подчеркнуть достоинства
- Способы комбинирования:
 - На уровне объединения loss-ов
 - На уровне архитектуры
- Наиболее распространенный вариант: CTC+attention

Комбинация различных end-to-end подходов

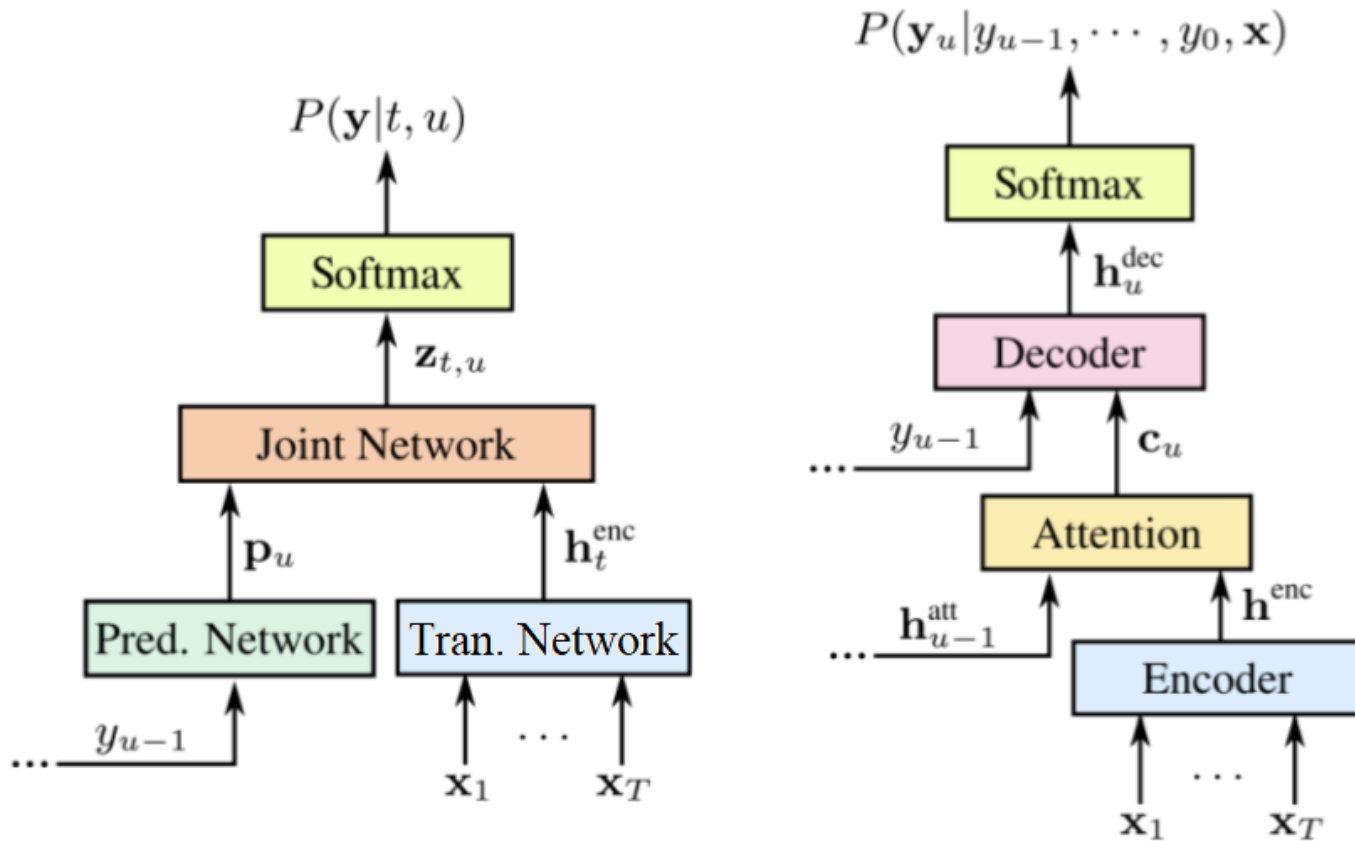
CTC+attention

- У сети есть общее «туловище» (энкодер) и 2 «головы» - в одной вычисляется CTC-loss, другая – декодер с attention
- Loss'ы объединяются с весом:
$$\mathcal{L}(\mathbf{l}, \mathbf{O}) = \lambda \log P_{CTC}(\mathbf{l}|\mathbf{O}) + (1 - \lambda) \log P_{att}(\mathbf{l}|\mathbf{O})$$
- При распознавании используется та же целевая функция: $\mathbf{l}^* = \arg \max_{\mathbf{l}} \mathcal{L}(\mathbf{l}, \mathbf{O})$
- Двухпроходный алгоритм: сначала только attention decoder, потом оставшиеся гипотезы рескорятся полным loss-ом.



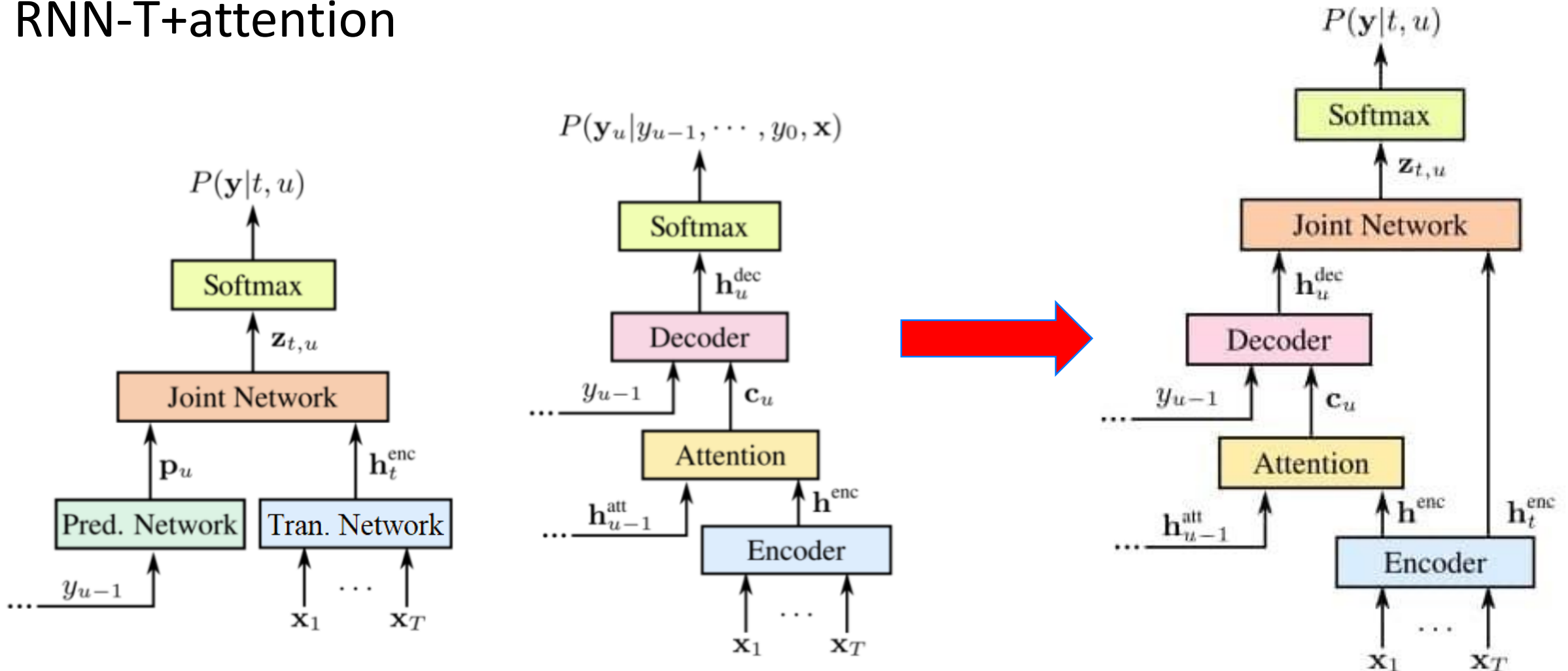
Комбинация различных end-to-end подходов

RNN-T+attention



Комбинация различных end-to-end подходов

RNN-T+attention



План лекции

- Недостатки современных гибридных систем
- Connectionist Temporal Classification (CTC)
- RNN-Transducer (RNN-T)
- Encoder-Decoder-системы с механизмом внимания (AED)
- Комбинации end-to-end-подходов
- Прочее

Byte-Pair Encoding (BPE)

BPE – способ построения алфавита из subword units (swu)

Алгоритм: вход – набор текстов, размер алфавита N , выход – алфавит и лексикон

1. Разбить все слова на графемы и приписать токен $</w>$ в конце каждого слова
2. Добавить все графемы и токен $</w>$ в алфавит
3. Перебрать все пары соседних токенов в словах и найти наиболее частотную
4. Склеить элементы этой пары в новый токен и добавить его в алфавит
5. Заменить все вхождения этой пары в обучающие данные на новый токен
6. Если текущий размер алфавита меньше N , перейти к шагу 3
7. Если какие-то токены больше не встречаются в словах – выкинуть из алфавита
8. Выдать алфавит и представление слов токенами

Byte-Pair Encoding (BPE)

Построение транскрипций для новых слов

Есть список слов, и хочется их представить с помощью имеющегося BPE-алфавита

1. Разбить слова на графемы и добавить к каждому слову токен `</w>`
2. Для всех BPE-токенов в порядке убывания длины:
 - Найти последовательности графем в словах, соответствующие данному токену и заменить их на токен
3. Если в словах остались графемы, не входящие в BPE-алфавит, заменить их на специальный unknown-токен `<unk>`
4. Полученный лексикон можно использовать при WFST-декодировании

WFST-декодирование для end-to-end

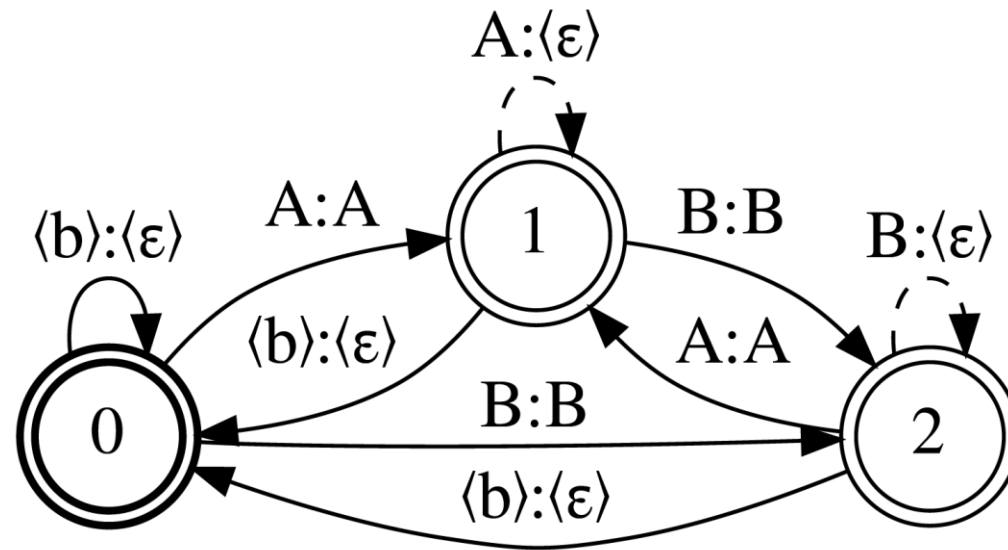
Отличия от гибридных моделей:

- В end-to-end системах оптимизируется вероятность распознавания целой фразы, поэтому значимость учета контекста (фонемного, графемного) сильно ниже
- Для гибрида требовалось построить WFST-граф $H \circ C \circ L \circ G$, где H отвечает за топологию НММ, а C — за контекстную зависимость
- Здесь нет НММ, а есть просто токены (графемы, BPE, слоги и т.п.) без контекста
- Строится простейший WFST-граф T , описывающий все возможные пути по токенам на последовательности фреймов
- Тогда WFST-граф распознавания описывается композицией $T \circ L \circ G$
- Обычно TLG-граф намного меньше аналогичного HCLG-графа и декодирование на нем значительно быстрее!

WFST-декодинг для end-to-end

Пример Т-графа для CTC:

- Для простоты будем считать, что у нас только 3 токена: A, B и b (blank)



- Фактически этот WFST осуществляет преобразование из «пути» в выходную последовательность токенов

Архитектуры, применяемые в end-to-end системах

- Первоначальные end-to-end системы строились на базе RNN, в частности, LSTM
 - Alex Graves and Navdeep Jaitly. **Towards end-to-end speech recognition with recurrent neural networks**, 2014
 - Hagen Soltau, Hank Liao, and Hasim Sak. **Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition**, 2016

Архитектуры, применяемые в end-to-end системах

- Первоначальные end-to-end системы строились на базе RNN, в частности, LSTM
 - Alex Graves and Navdeep Jaitly. **Towards end-to-end speech recognition with recurrent neural networks**, 2014
 - Hagen Soltau, Hank Liao, and Hasim Sak. **Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition**, 2016
- Потом выяснилось, что длинный контекст можно неплохо моделировать и сверточными сетями
 - R. Collobert, C. Puhersch, and G. Synnaeve, **Wav2letter: an end-to-end convnet-based speech recognition system**, 2016.
 - V. Liptchinsky, G. Synnaeve, and R. Collobert, **Letter-based speech recognition with gated convnets**, 2017.
 - Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M. Cohen, Huyen Nguyen, Ravi Teja Gadde, **Jasper: An End-to-End Convolutional Neural Acoustic Model**, 2019

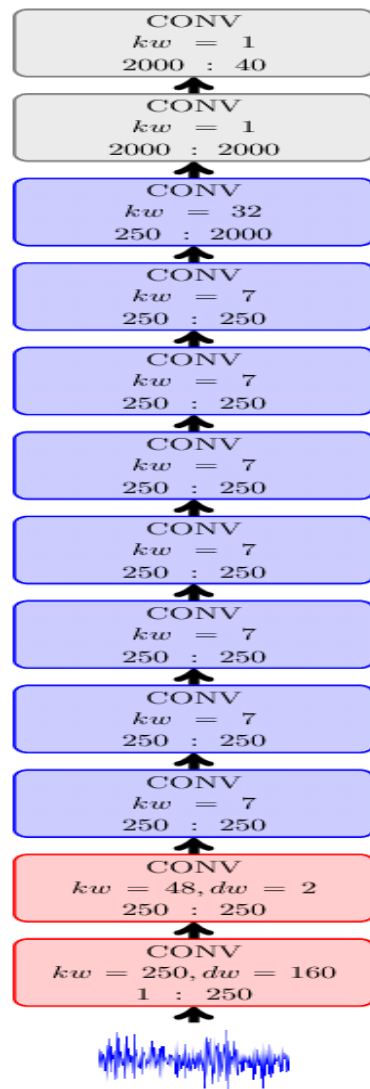
Архитектуры, применяемые в end-to-end системах

- Первоначальные end-to-end системы строились на базе RNN, в частности, LSTM
 - Alex Graves and Navdeep Jaitly. **Towards end-to-end speech recognition with recurrent neural networks**, 2014
 - Hagen Soltau, Hank Liao, and Hasim Sak. **Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition**, 2016
- Потом выяснилось, что длинный контекст можно неплохо моделировать и сверточными сетями
 - R. Collobert, C. Puhersch, and G. Synnaeve, **Wav2letter: an end-to-end convnet-based speech recognition system**, 2016.
 - V. Liptchinsky, G. Synnaeve, and R. Collobert, **Letter-based speech recognition with gated convnets**, 2017.
 - Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M. Cohen, Huyen Nguyen, Ravi Teja Gadde, **Jasper: An End-to-End Convolutional Neural Acoustic Model**, 2019
- В 2018 наступила «эра» трансформеров
 - Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, **Attention Is All You Need**, 2017
 - Linhao Dong, Shuang Xu, Bo Xu, **Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition**, 2018

Архитектуры, применяемые в end-to-end системах

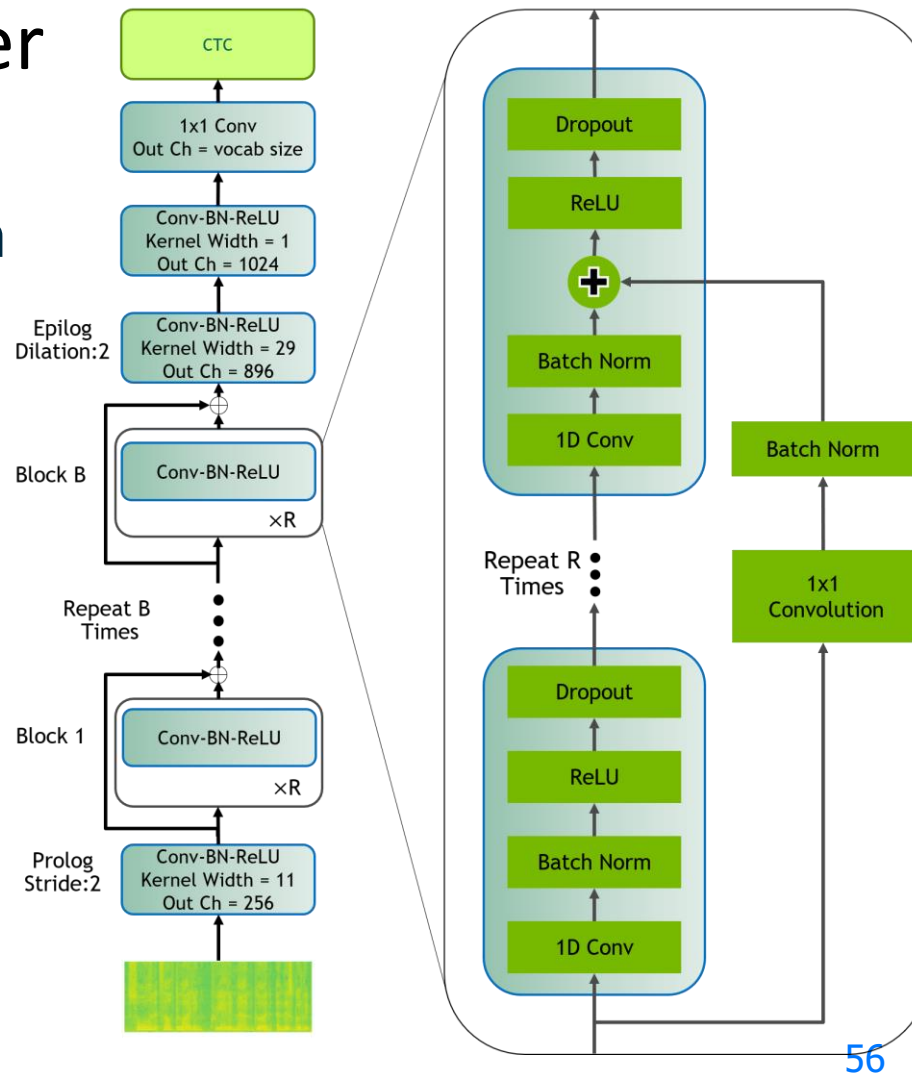
Wav2letter Convnet

- 1D-свертки
- kw – kernel width
- dw – stride
- Если на входе спектр или MFCC, то первый слой не нужен



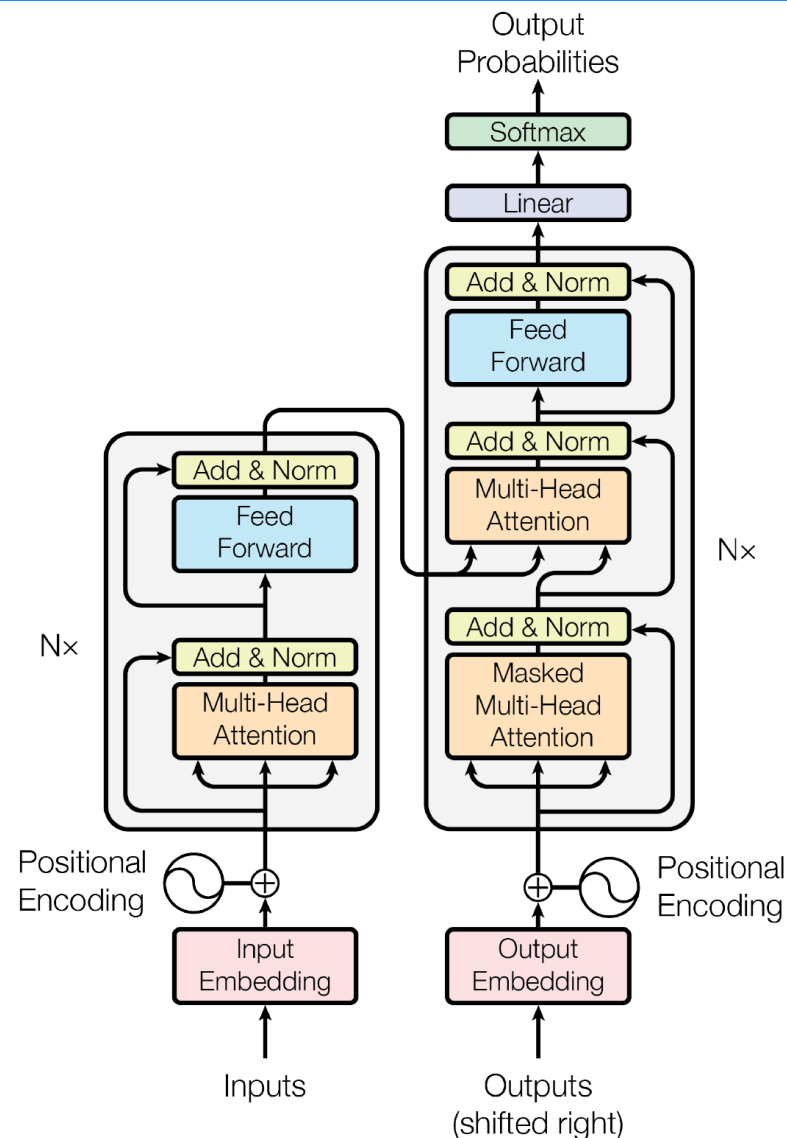
NVIDIA Jasper

- Сверточные блоки с batch normalization и residual connections



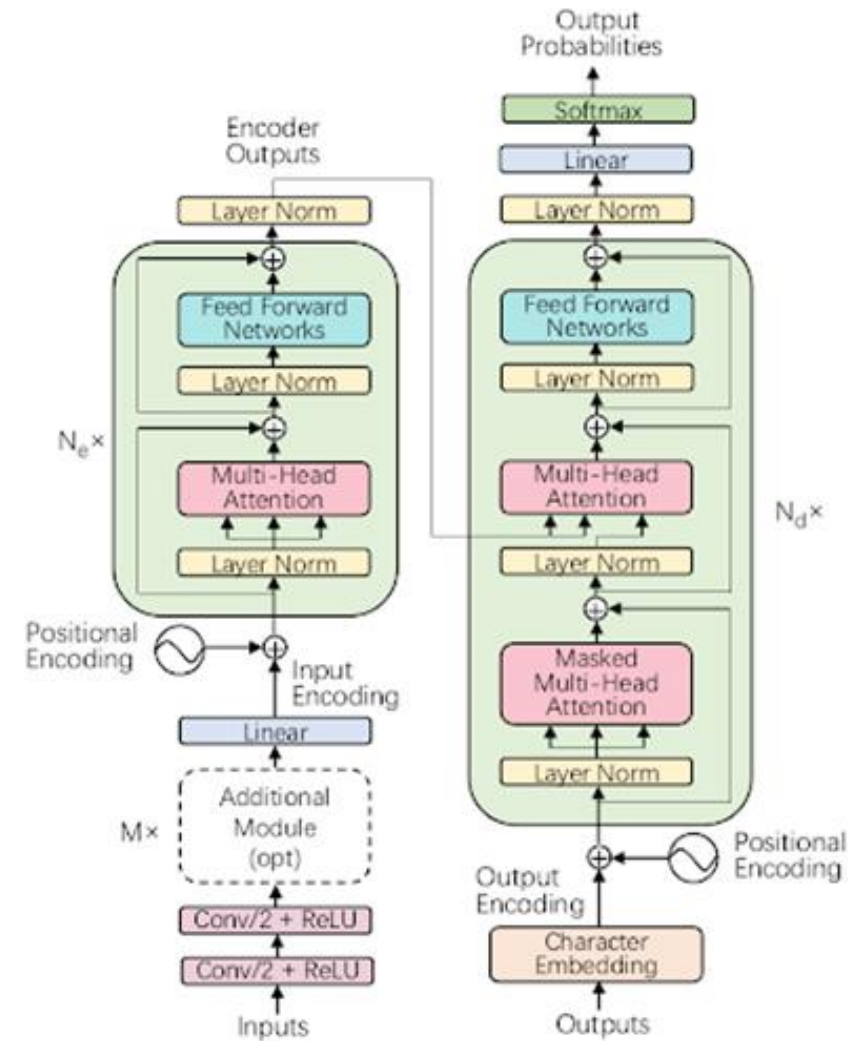
Архитектуры, применяемые в end-to-end системах

Оригинальный трансформер из работы “Attention is all you need”:



Архитектуры, применяемые в end-to-end системах

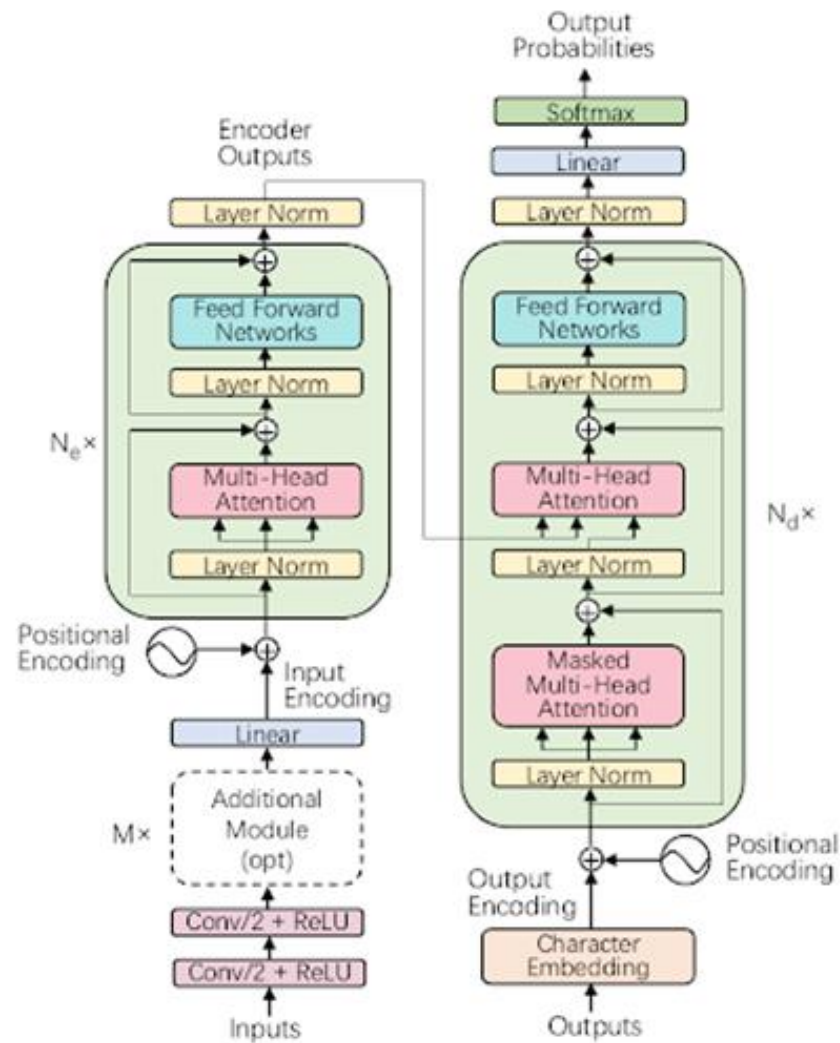
Speech Transformer, 2018:



Архитектуры, применяемые в end-to-end системах

Speech Transformer, 2018:

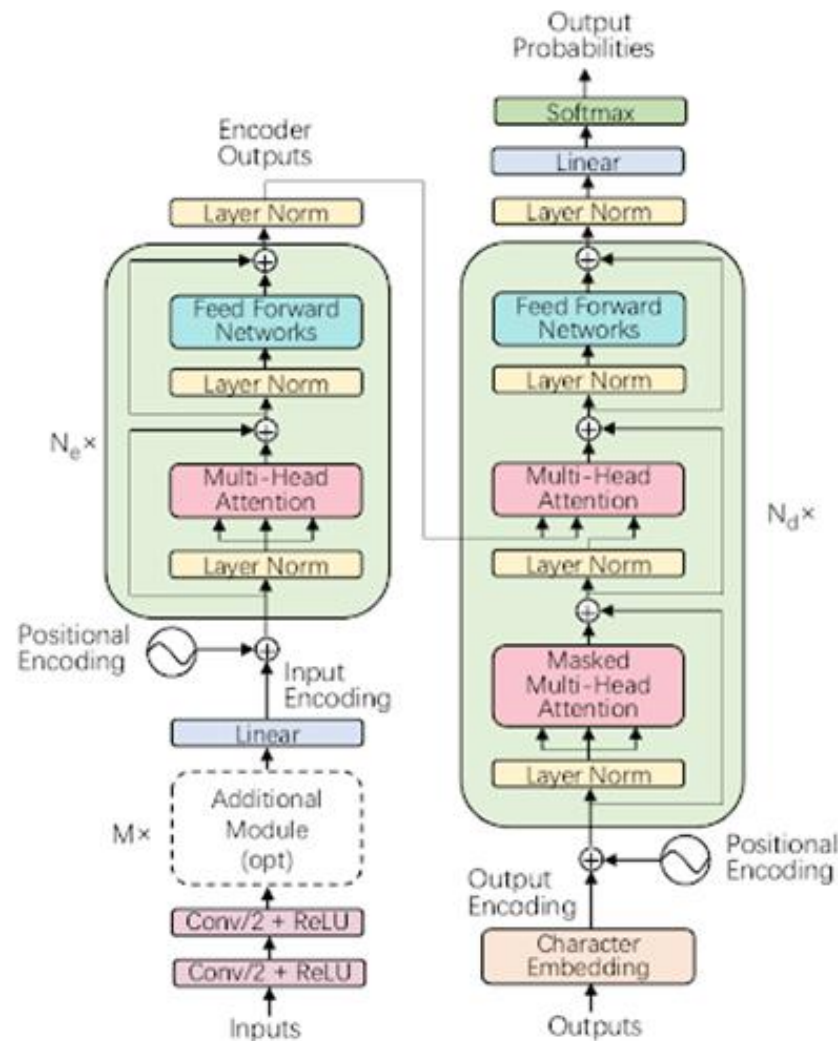
- Интеграция Speech Transformer с CTC loss (2019)



Архитектуры, применяемые в end-to-end системах

Speech Transformer, 2018:

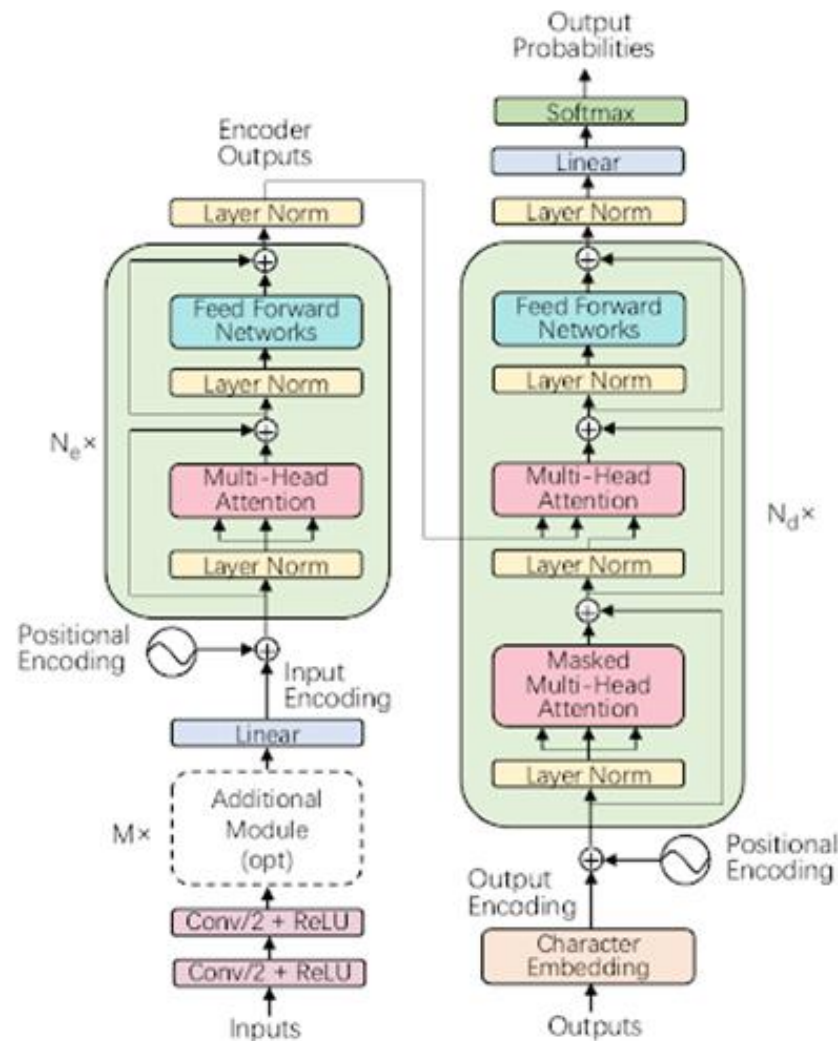
- Интеграция Speech Transformer с CTC loss (2019)
- Отказ от синусоидального absolute Positional Encoding, замена его на relative PE или на conv/pooling-слои (2019)



Архитектуры, применяемые в end-to-end системах

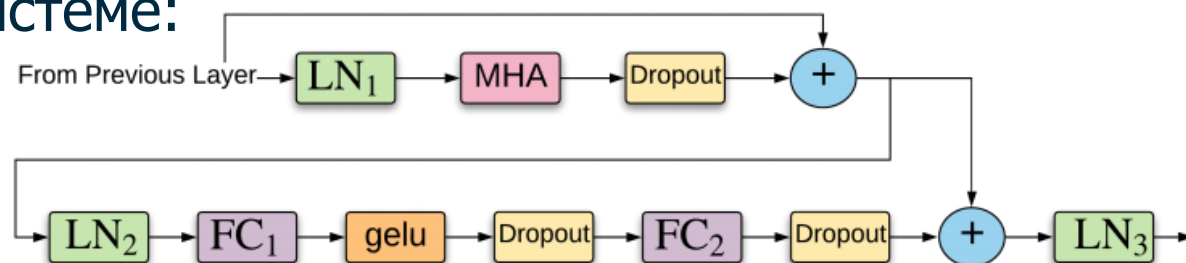
Speech Transformer, 2018:

- Интеграция Speech Transformer с CTC loss (2019)
- Отказ от синусоидального absolute Positional Encoding, замена его на relative PE или на conv/pooling-слои (2019)
- Использование только encoder-блоков трансформера в CTC и RNN-T (Transformer-Transducer) системах (2019-2020)

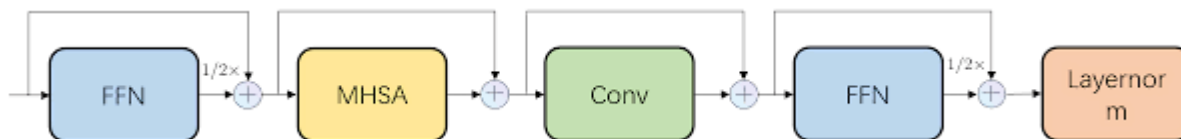


Архитектуры, применяемые в end-to-end системах

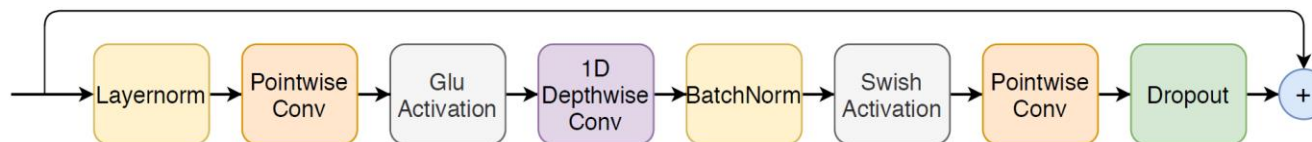
- Типичный transformer-encoder блок в ASR-системе:



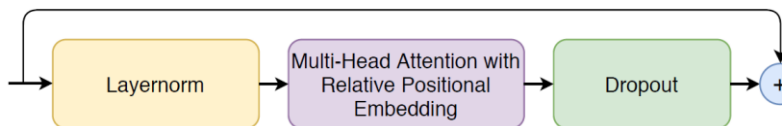
- Блок Convolutional Transformer (Conformer), 2020:



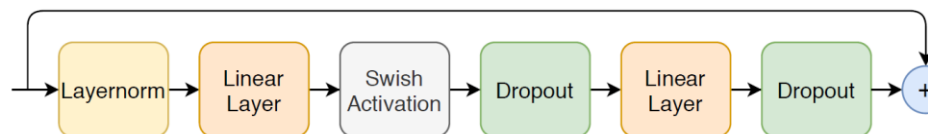
- Convolution module:



- Multi-head self-attention module:



- Feedforward module:





Спасибо
за внимание!

Вопросы?