Automatic Speech Recognition I

안인규 (Inkyu An)

Speech And Audio Recognition (오디오 음성인식)

https://mairlab-km.github.io/

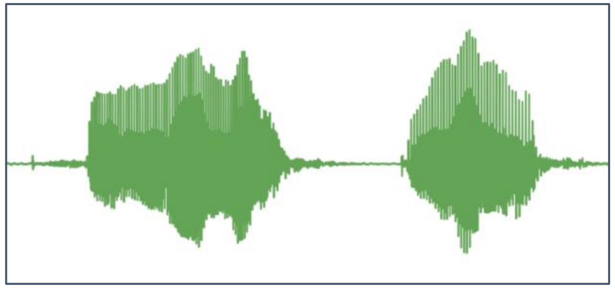
This lecture material refers to
https://github.com/yandexdataschool/speech_course?tab=readme-ov-file and
https://github.com/markovka17/dla

MAIR Lab

Automatic Speech Recognition

- Task:
  - Transform speech from audio to text

- Also known as:
  - SST (Speech to Text)



Hello world

Automatic Speech Recognition

- Metrics: WER (Word Error Rate)
  - **Target**: the quick brown fox jumps over a lazy dog
  - **Prediction**: the quick brow an fox jumps over lazy dog

Automatic Speech Recognition

- Metrics: WER (Word Error Rate)
  - **Target**: the quick brown fox jumps over a lazy dog
  - **Prediction**: the quick brow an fox jumps over lazy dog

1. S – substitutions
2. D – deletions
3. I – insertions
4. N – total words in target

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

Automatic Speech Recognition

- Metrics: WER (Word Error Rate)
  - **Target**: the quick brown fox jumps over a lazy dog
  - **Prediction**: the quick brow an fox jumps over lazy dog

1. S – substitutions
2. D – deletions
3. I – insertions
4. N – total words in target

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

CER (Character Error Rate)

- The same tings, but at the character level

Questions:

- What is more important?
- What is more difficult to minimize?
- WER > 1?
- N = 0 ?

Automatic Speech Recognition

- Metrics: WER (Word Error Rate)
  - **Target**: the quick brown fox jumps over a lazy dog
  - **Prediction**: the quick brow an fox jumps over lazy dog

1. S – substitutions
2. D – deletions
3. I – insertions
4. N – total words in target

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

CER (Character Error Rate)

- The same tings, but at the character level

Questions:

- What is more important? → D > S > I (정보 손실, 의미 왜곡, 가독성 저하 측면)
- What is more difficult to minimize? → S (음향적 유사성, 언어적 모호성)
- WER > 1? → Possible (Why?)
- N = 0 ? → Undefined or NaN

Automatic Speech Recognition

- Data: Wall Street Journal (WSJ0, WSJ1)
  - Domain: WSJ articles read aloud by journalists, high-quality audio recorded with 2 mics
  - Parts:
    - train: 78k samples, totally 73h
    - test: 8.2k samples, total 8h
  - Features:
    - read speech
    - up to several minutes of audio
    - complex newspaper language
    - commercial: no free access

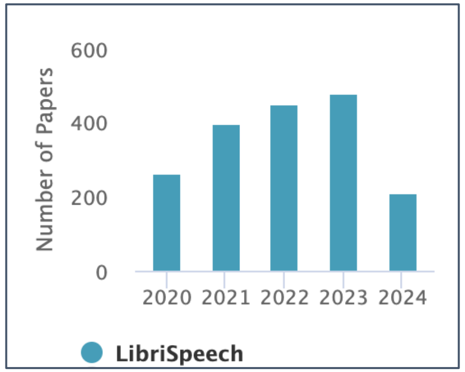
Sample:

Automatic Speech Recognition

- Data: Libri Speech (2015)
  - **Domain**: audio books from LibriVox project
  - **Parts**:
    - train: 960h / test: 11h / dev: 11h
    - clean / other (noise)
  - **Features**:
    - read speech with ASR model alignment
    - 10-20s of audio
    - literary language
    - several sentences per sample
    - balanced by speakers

subset	hours	per-spk minutes	female spkrs	male spkrs	total spkrs
dev-clean	5.4	8	20	20	40
test-clean	5.4	8	20	20	40
dev-other	5.3	10	16	17	33
test-other	5.1	10	17	16	33
train-clean-100	100.6	25	125	126	251
train-clean-360	363.6	25	439	482	921
train-other-500	496.7	30	564	602	1166

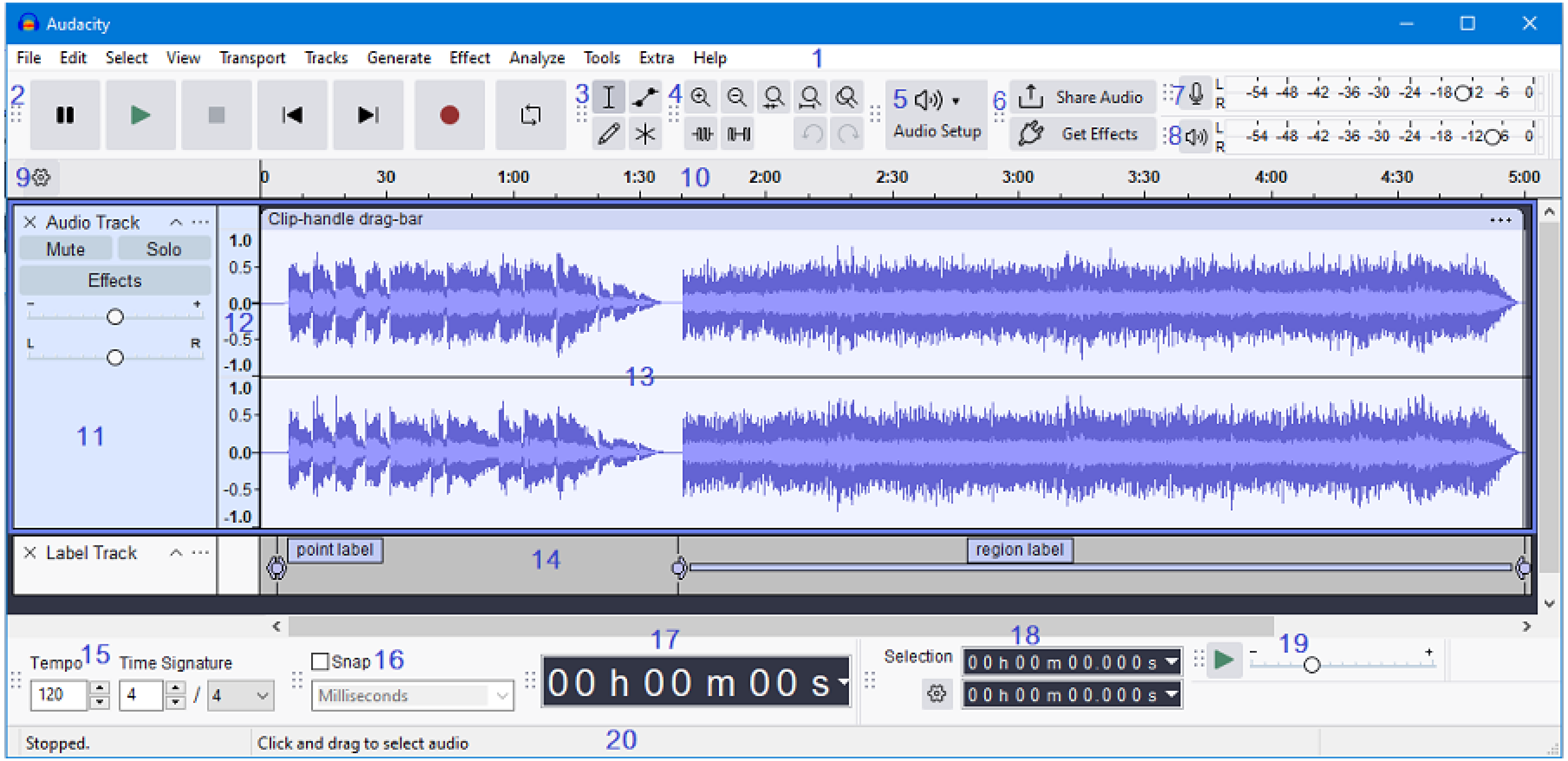
	test clean	test other
model	1.4	2.48
human	5.83	12.69



Sample (clean): Sample (other):

Automatic Speech Recognition

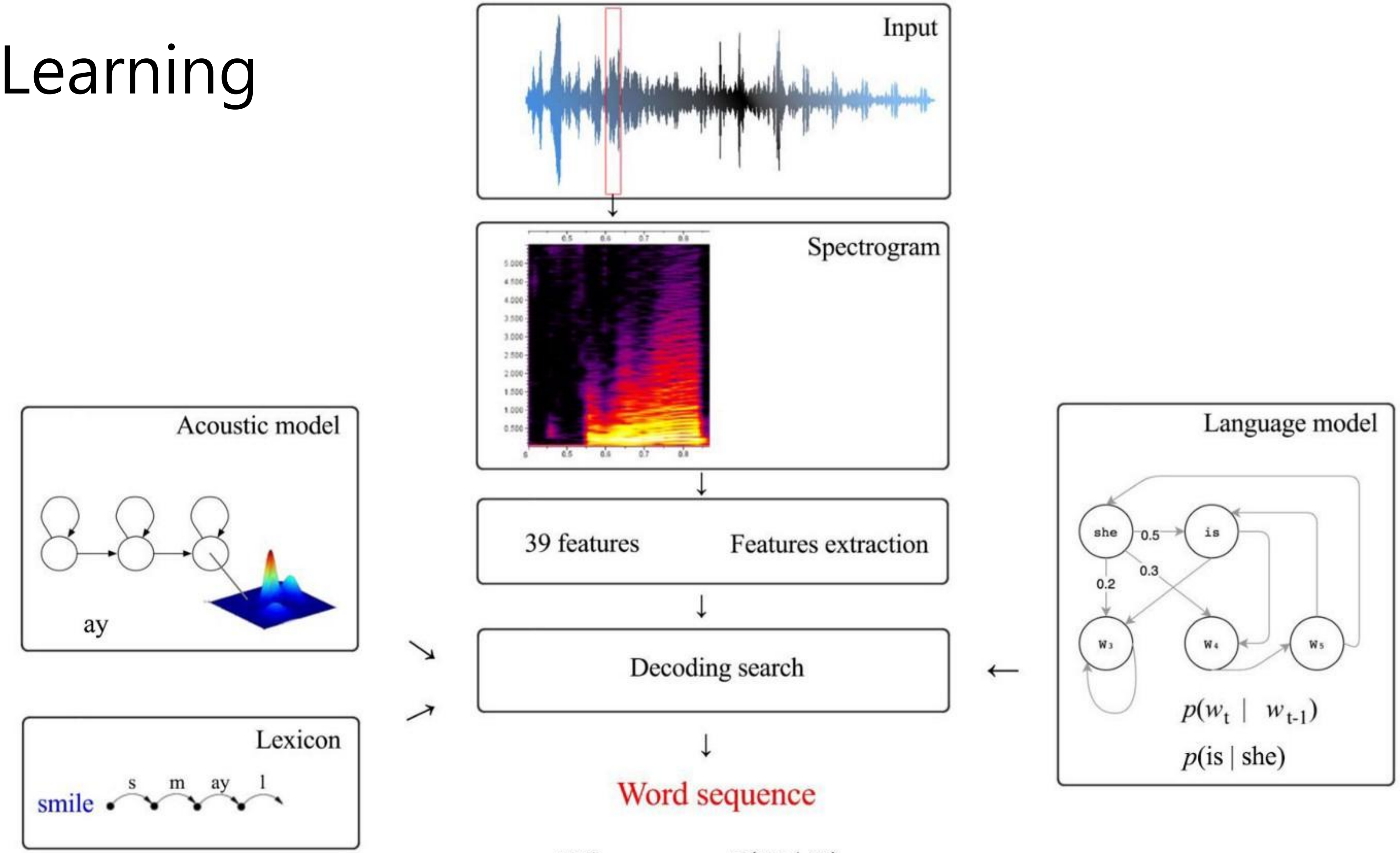
- Audio Visualization tool: Audacity (Free)



https://www.audacityteam.org/

Automatic Speech Recognition

- Before Deep Learning



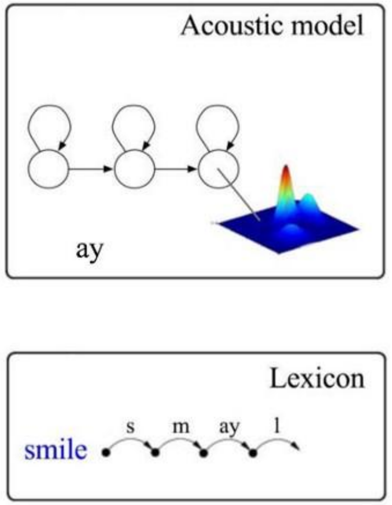
$$\mathbf{W}^* = \arg\max_{\mathbf{W}} P(\mathbf{W} \mid \mathbf{X})$$

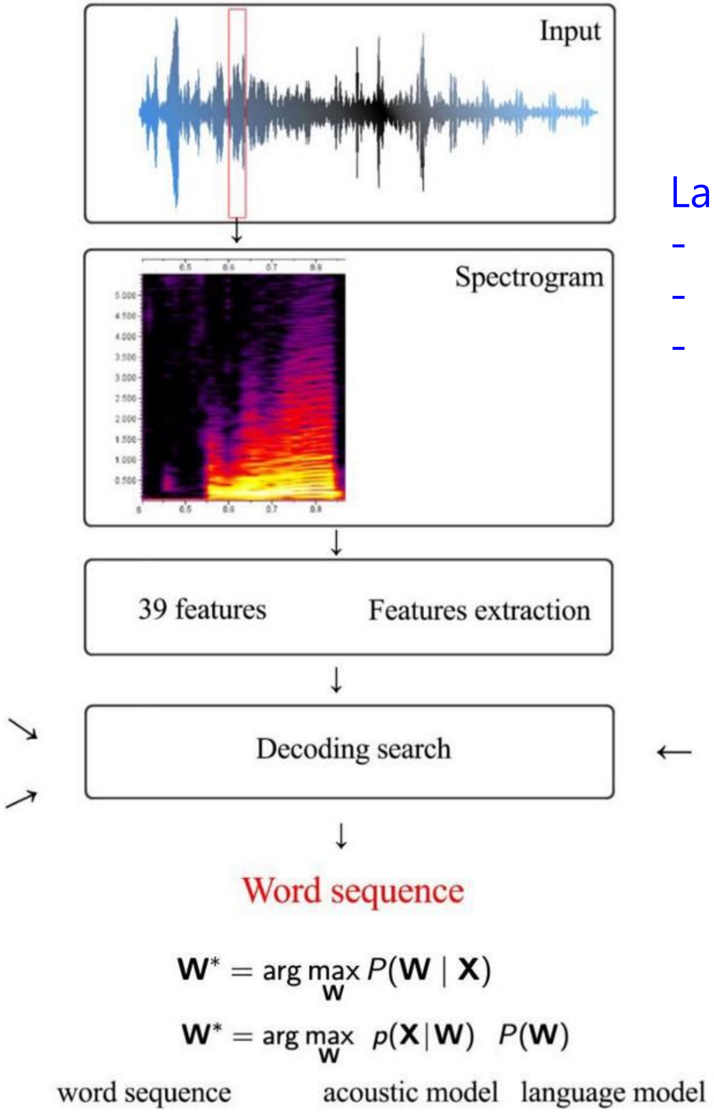
$$\underbrace{\mathbf{W}^*}_{\text{word sequence}} = \arg\max_{\mathbf{W}} \underbrace{p(\mathbf{X} \mid \mathbf{W})}_{\text{acoustic model}} \underbrace{P(\mathbf{W})}_{\text{language model}}$$

Automatic Speech Recognition

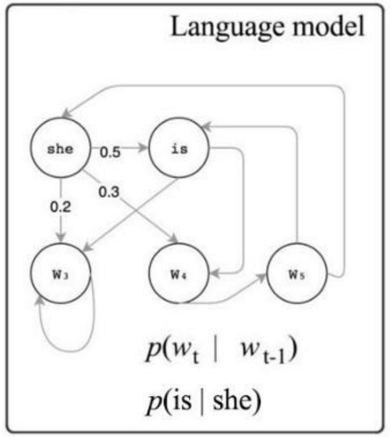
- Before Deep Learning

Acoustic model (음향 모델)
- Feature → Phoneme (음소) 확률을 mapping
- HMM (Hidden Markov Model) + GMM (Gaussian Mixture Model)





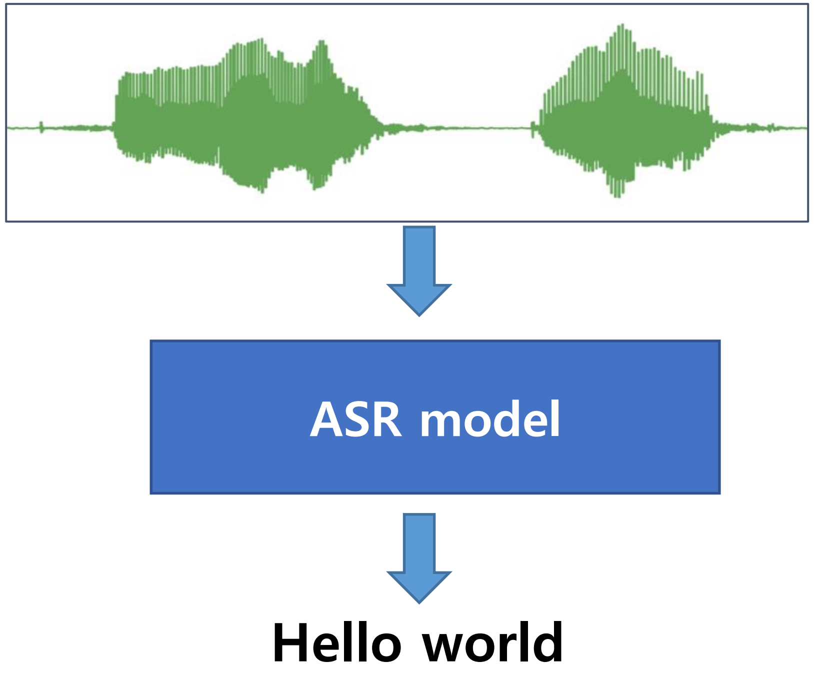
Language model (언어 모델)
- Word sequence의 확률을 계산
- n-gram
- Acoustic model이 헷갈릴 경우, 문맥 확률을 활용



Lexicon (발음 사전)
- 단어 → 음소 sequence mapping
- Acoustic model을 통해 구한 음소 확률을 단어로 연결하는 역할

DL-based ASR

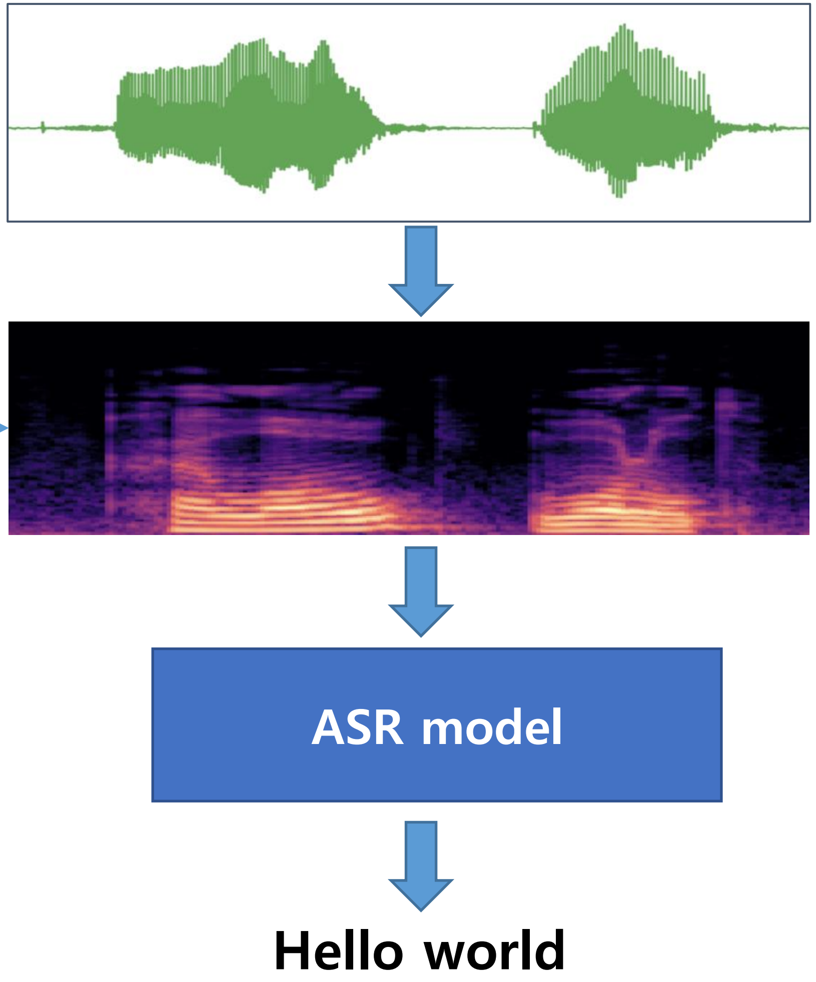
- What do we want?
  - Build an ASR model



DL-based ASR

- What do we want?
  - Build an ASR model

  - We ca extract audio features
    - Spectrogram
    - **Mel spectrogram**
    - …

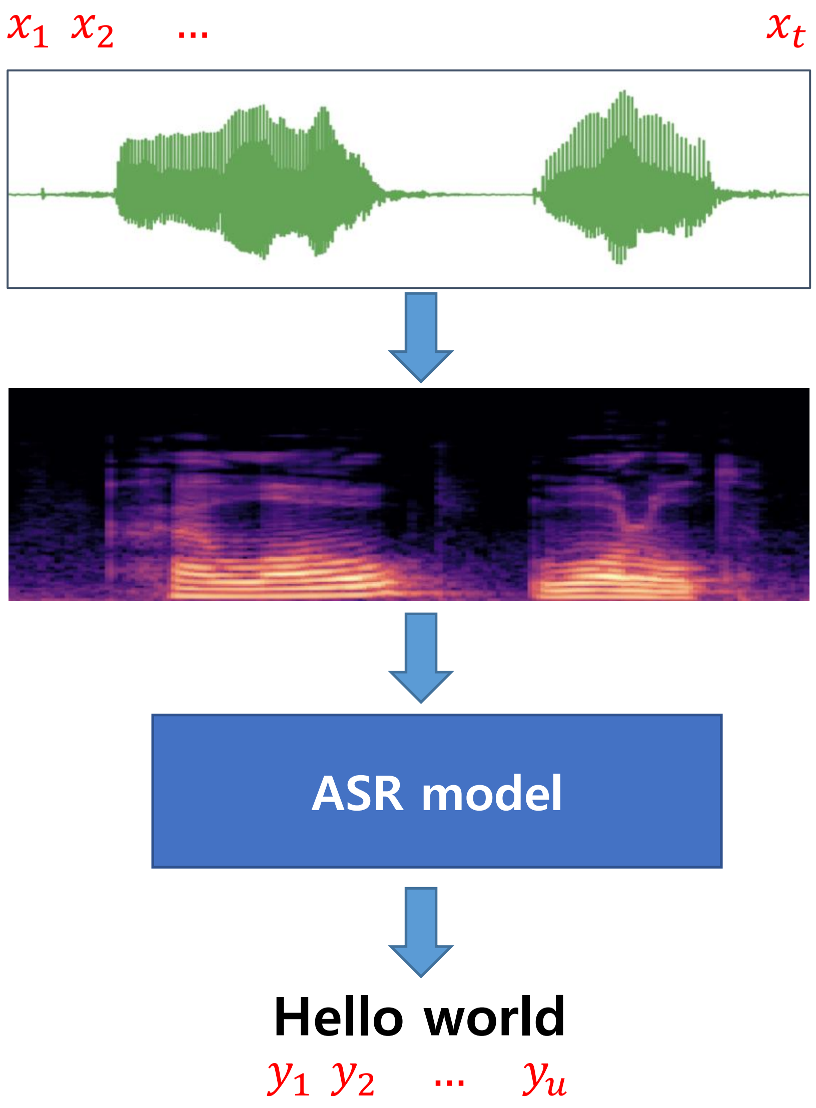


DL-based ASR

- What do we want?
  - Build an ASR model

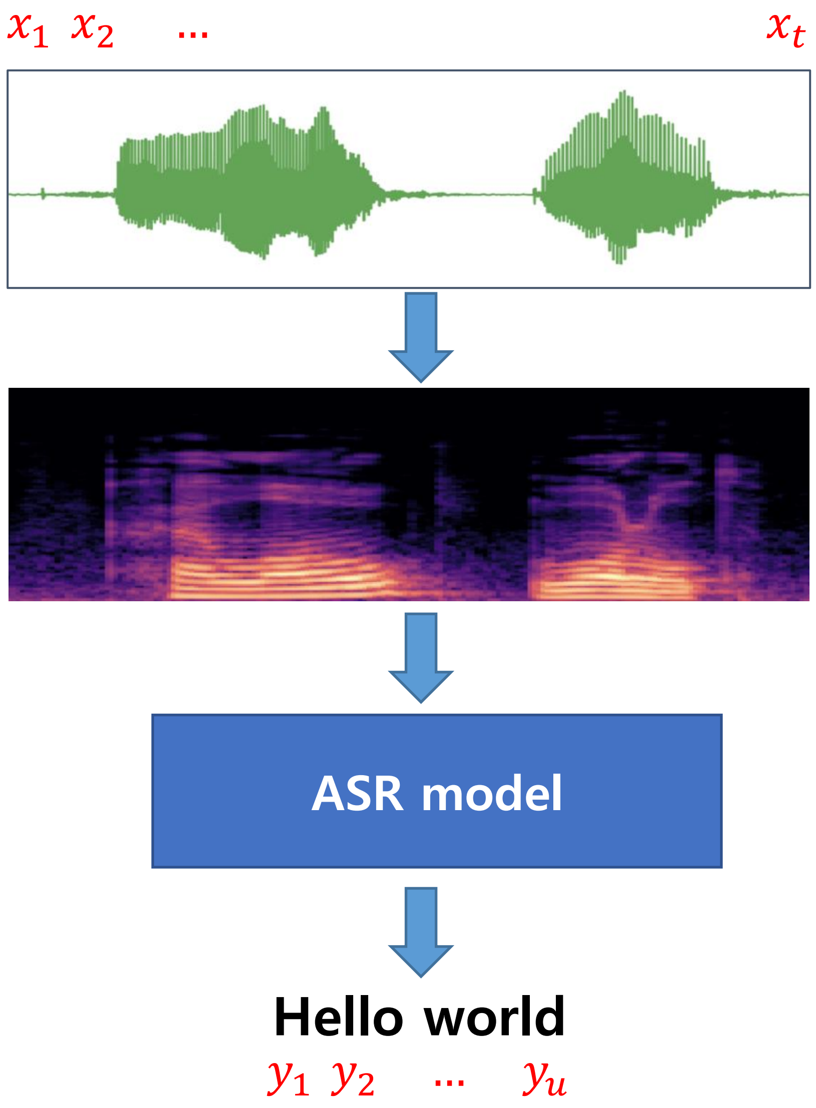
  - We ca extract audio features
    - Spectrogram
    - **Mel spectrogram**
    - …

  - We have
    - variable input lengths, $x_1, x_2, \cdots, x_t$
    - variable output length, $y_1, y_2, \cdots, y_u,\ u \leq t$



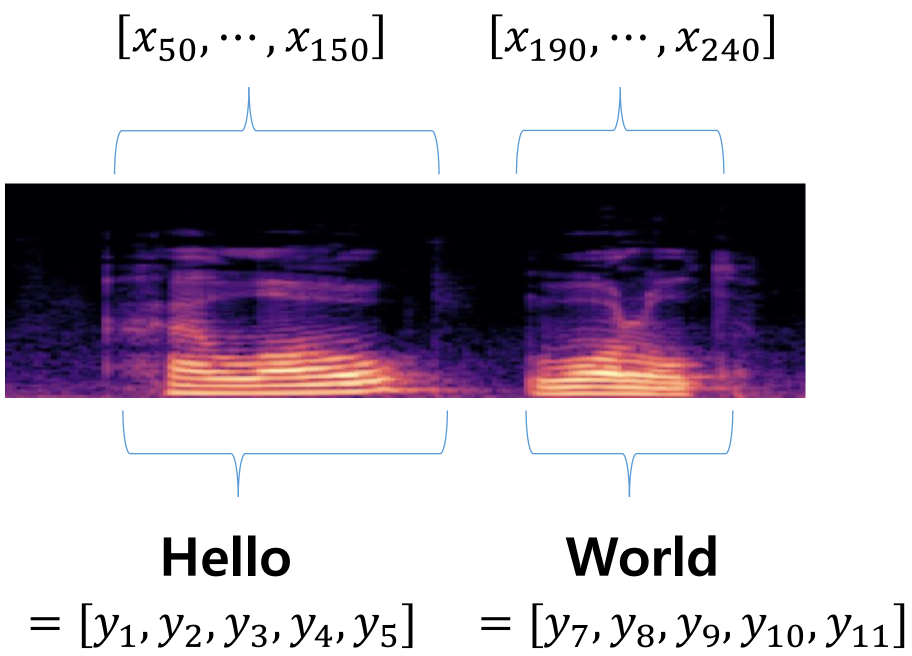
DL-based ASR

- What do we want?
  - Build an ASR model
  - We ca extract audio features
    - Spectrogram
    - **Mel spectrogram**
    - …
  - We have
    - variable input lengths, $x_1, x_2, \cdots, x_t$
    - variable output length, $y_1, y_2, \cdots, y_u,\ u \leq t$
  - **Problem**: x and y are misaligned!



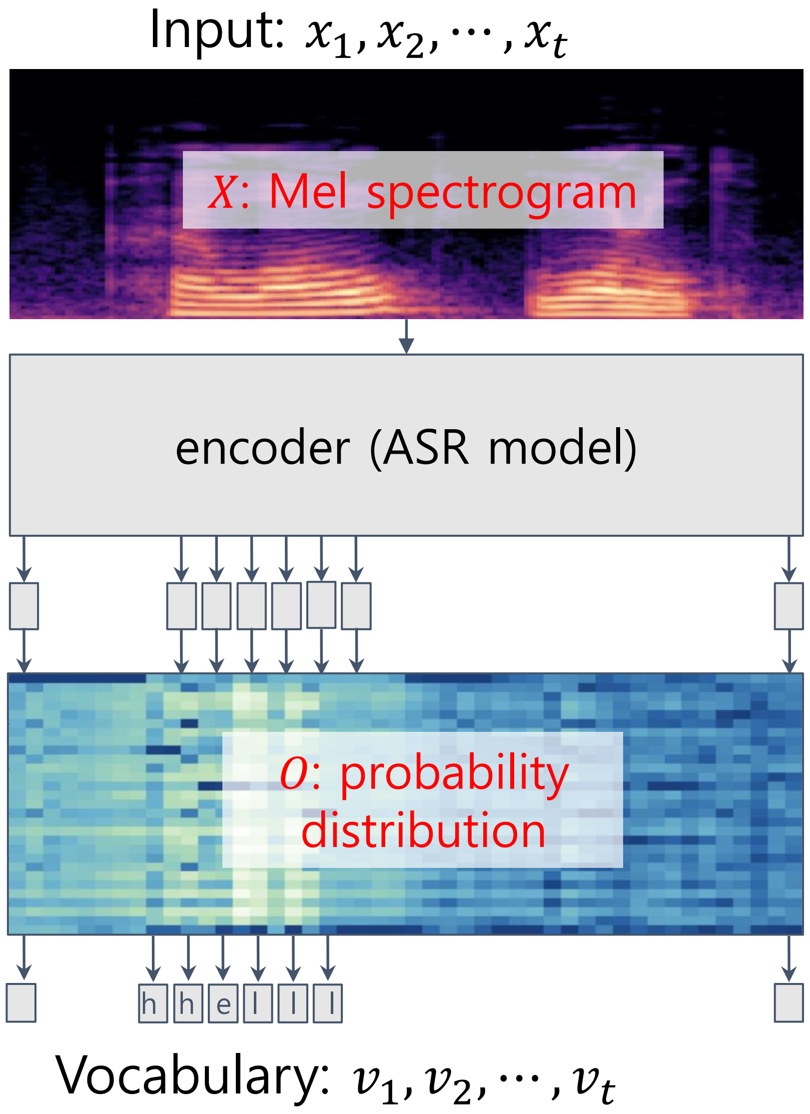
DL-based ASR

- What do we want?
  - Build an ASR model
  - We ca extract audio features
    - Spectrogram
    - **Mel spectrogram**
    - …
  - We have
    - variable input lengths, $x_1, x_2, \cdots, x_t$
    - variable output length, $y_1, y_2, \cdots, y_u,\ u \leq t$
  - **Problem**: x and y are misaligned!



DL-based ASR

- Naive Approach
  - Build a model that predicts probability distribution for each x_j over vocabulary V
  - $O = P(V \parallel X) = Model(X)$
  - $X \in \mathbb{R}^{seq_{len} \times mel_{len}}, O \in \mathbb{R}^{seq_{len} \times size_{voca}}$
  - then, we can easily predict output as argmax over the time axis
  - $Y^* = \mathrm{argmax}(O), \mathrm{Y} \in \mathbb{R}^{seq_{len}}$
  - $Y^* = \{h, h, e, l, l, l, \cdots\}$



DL-based ASR

- Naive Approach
  - Build a model that predicts probability distribution for each x_j over vocabulary V

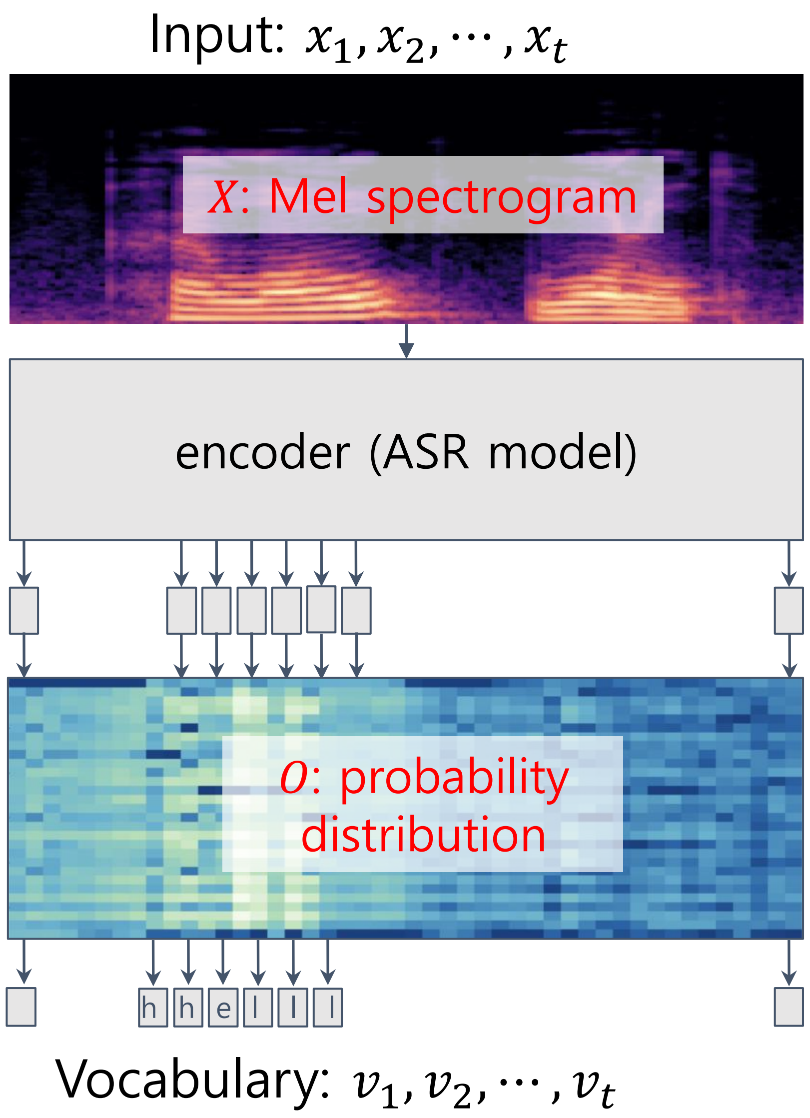
- $O = P(V \parallel X) = Model(X)$
- $X \in \mathbb{R}^{seq_{len} \times mel_{len}}, O \in \mathbb{R}^{seq_{len} \times size_{voca}}$

Encoding

  - then, we can easily predict output as argmax over the time axis

- $Y^* = \text{argmax}(O), \mathrm{Y} \in \mathbb{R}^{seq_{len}}$
- $Y^* = \{h, h, e, l, l, l, \cdots\}$

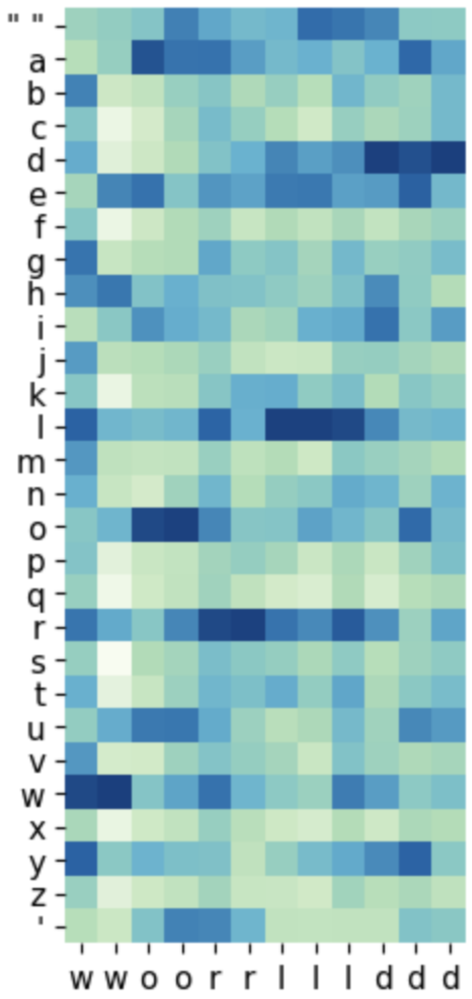
Decoding



DL-based ASR

- What do we want?
  - Thus, how can we obtain $Y^* \in \mathbb{R}^{seq_{len}}$

O: probability distribution



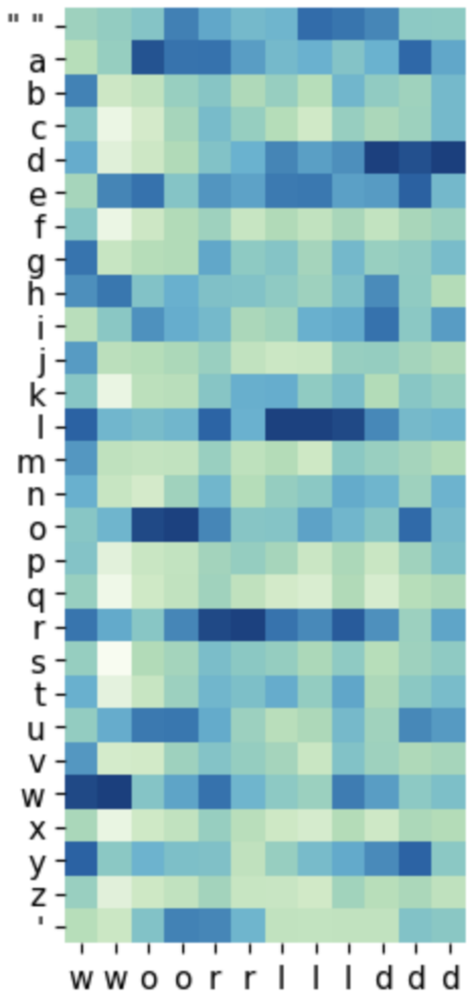
DL-based ASR

- What do we want?
  - Thus, how can we obtain $Y^* \in \mathbb{R}^{seq_{len}}$

Merge consecutive letters

- **Issues?**
  - Multiple consecutive letters in a target word (e.g., hello)
  - Silence between works and letters (e.g., breathing, lip-smacking)

O: probability distribution



DL-based ASR

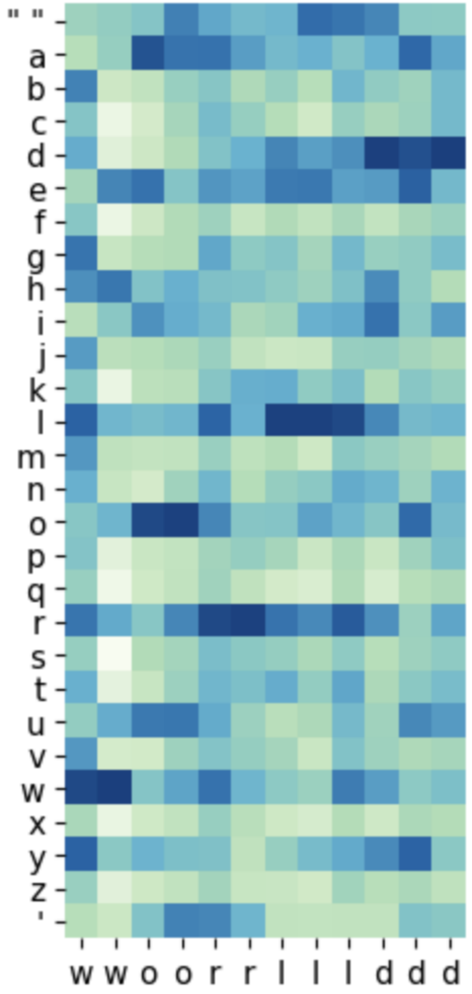
- What do we want?
  - Thus, how can we obtain $Y^* \in \mathbb{R}^{seq_{len}}$

Merge consecutive letters

- **Issues?**
  - Multiple consecutive letters in a target word (e.g., hello)
  - Silence between works and letters (e.g., breathing, lip-smacking)

Blank symbol: ε

O: probability distribution



DL-based ASR

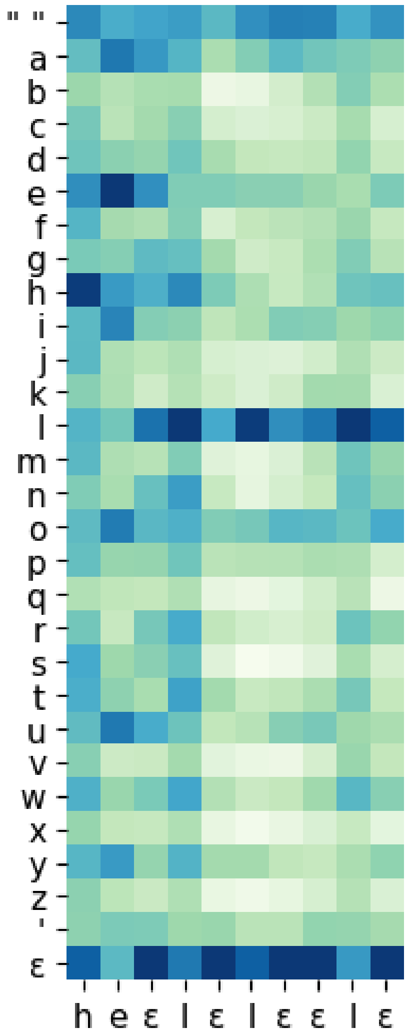
- What do we want?
  - Add special symbol to vocabulary
    - Blank symbol: ε
  - <u>Train model in such a way, to predict blank symbol in issue case</u>
  - How to deal with blank symbol while decoding
    - Merge all repeated symbols into one

      h ε e l l ε ε l ε o → h ε e l ε l ε o

    - Delete blanks

      h ε e l ε l ε o → h e l l o

O: probability distribution



DL-based ASR

- How should we define a loss?
  - We have valid paths that decode into the target

h	ε	e	l	ε	l	ε	o	o	o
h	ε	e	l	l	l	ε	l	o	o
ε	h	h	h	e	l	ε	l	o	o

Hello

  - and non-valid path

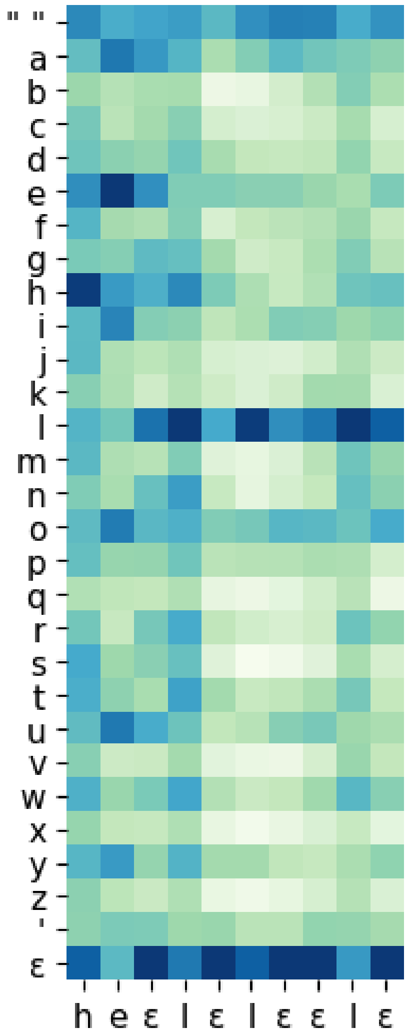
h	ε	e	**l**	**ε**	**l**	**ε**	**l**	o	o

$\rightarrow$ **Helllo**

ε	h	h	h	e	l	ε	l	**ε**	**ε**

$\rightarrow$ **?**

O: probability distribution



DL-based ASR

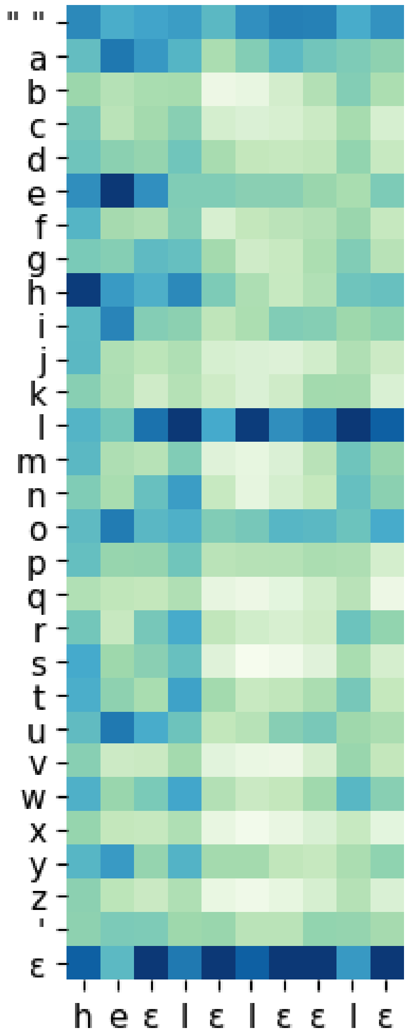
- How should we define a loss?
  - We have valid paths that decode into the target

h	ε	e	l	ε	l	ε	o	o	o
h	ε	e	l	l	l	ε	l	o	o
ε	h	h	h	e	l	ε	l	o	o

Hello

  - we don’t care what path the model selected, thus, we maximize probabilities for all of them
  - <u>for each valid path, we can compute its probability from the output matrix</u>

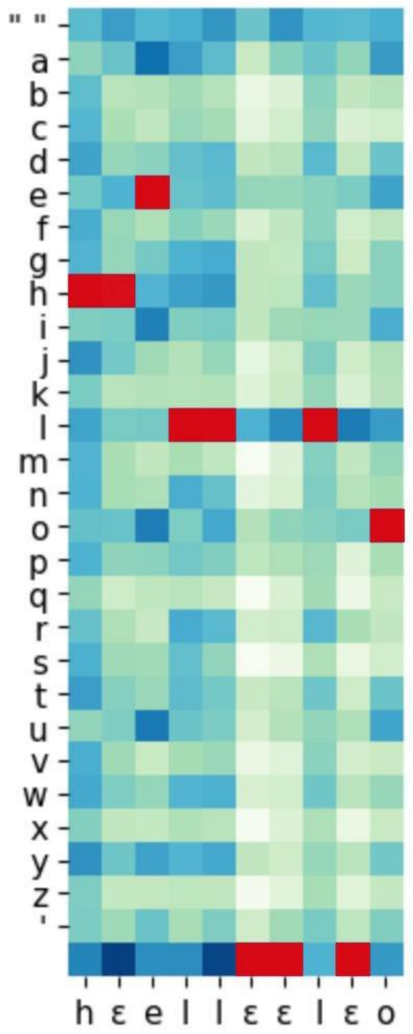
O: probability distribution



DL-based ASR

- How should we define a loss?
  - we don’t care what path the model selected, thus, we maximize probabilities for all of them
  - <u>for each valid path, we can compute its probability from the output matrix</u>
    - $P(h\ h\ e\ l\ l\ \varepsilon\ \varepsilon\ l\ \varepsilon\ o) = 0.00123$

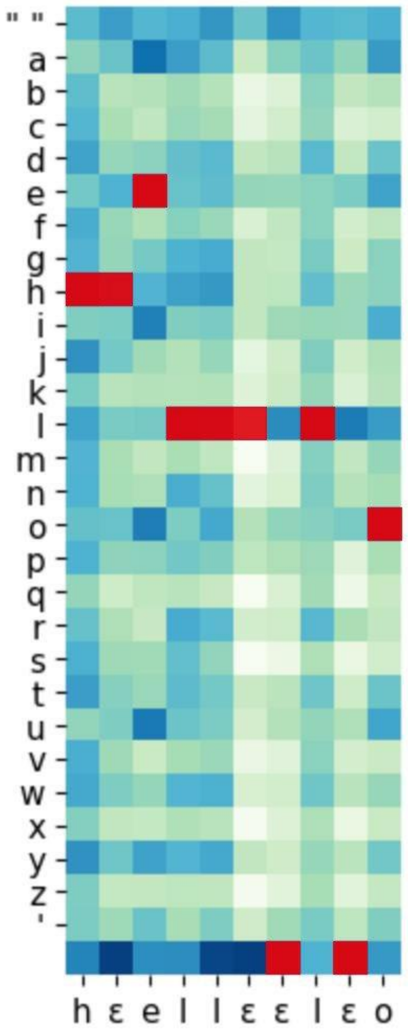
O: probability distribution



DL-based ASR

- How should we define a loss?
  - we don't care what path the model selected, thus, we maximize probabilities for all of them
  - for each valid path, we can compute its probability from the output matrix
    - $P(h\ h\ e\ l\ l\ \varepsilon\ \varepsilon\ l\ \varepsilon\ o) = 0.00123$
    - $P(h\ h\ e\ l\ l\ l\ \varepsilon\ l\ \varepsilon\ o) = 0.00112$

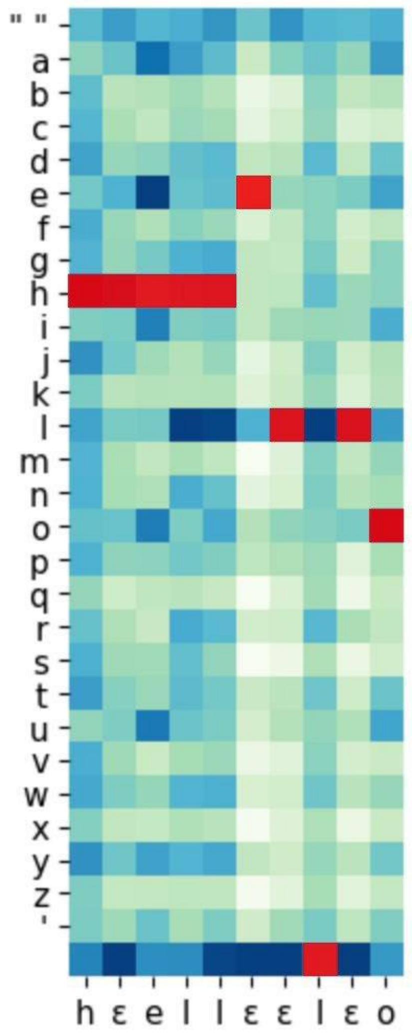
O: probability distribution



DL-based ASR

- How should we define a loss?
  - we don't care what path the model selected, thus, we maximize probabilities for all of them
  - <u>for each valid path, we can compute its probability from the output matrix</u>
    - $P(h\,h\,e\,l\,l\,\varepsilon\,\varepsilon\,l\,\varepsilon\,o) = 0.00123$
    - $P(h\,h\,e\,l\,l\,l\,\varepsilon\,l\,\varepsilon\,o) = 0.00112$
    - $P(h\,h\,h\,h\,h\,e\,l\,\varepsilon\,l\,o) = 0.00001$
  - and for many others …

O: probability distribution



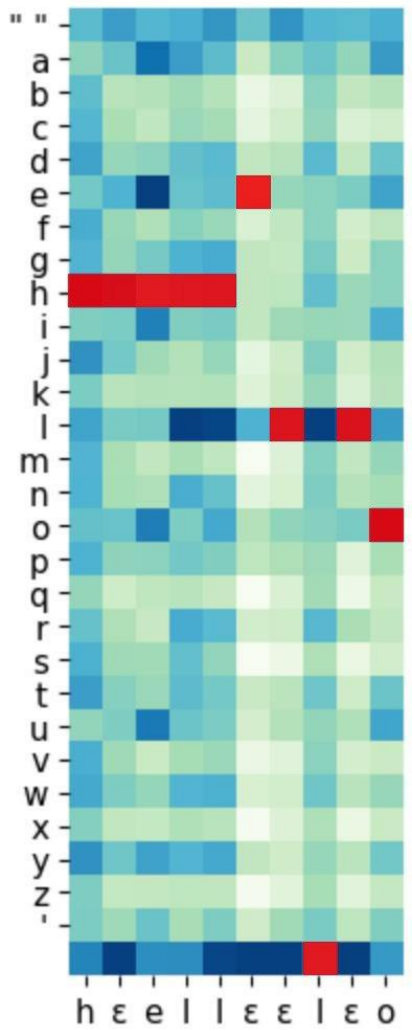
DL-based ASR

- How should we define a loss? → **CTC Loss**
  - we don't care what path the model selected, thus, we maximize probabilities for all of them
  - for each valid path, we can compute its probability from the output matrix
    - $P(h\ h\ e\ l\ l\ \varepsilon\ \varepsilon\ l\ \varepsilon\ o) = 0.00123$
    - $P(h\ h\ e\ l\ l\ l\ \varepsilon\ l\ \varepsilon\ o) = 0.00112$
    - $P(h\ h\ h\ h\ h\ e\ l\ \varepsilon\ l\ o) = 0.00001$

$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

The CTC conditional **probability** | **marginalizes** over the set of valid alignments | computing the **probability** for a single alignment step-by-step.

O: probability distribution



DL-based ASR

- Efficient CTC Loss Compute

$$\text{target} = \text{ab}$$

$$\begin{aligned} p(\text{out=ab})_T = \;& p(\text{out=ab, last\_char} = \text{b})_T \\ +\;& p(\text{out=ab, last\_char} = \varepsilon)_T \end{aligned}$$

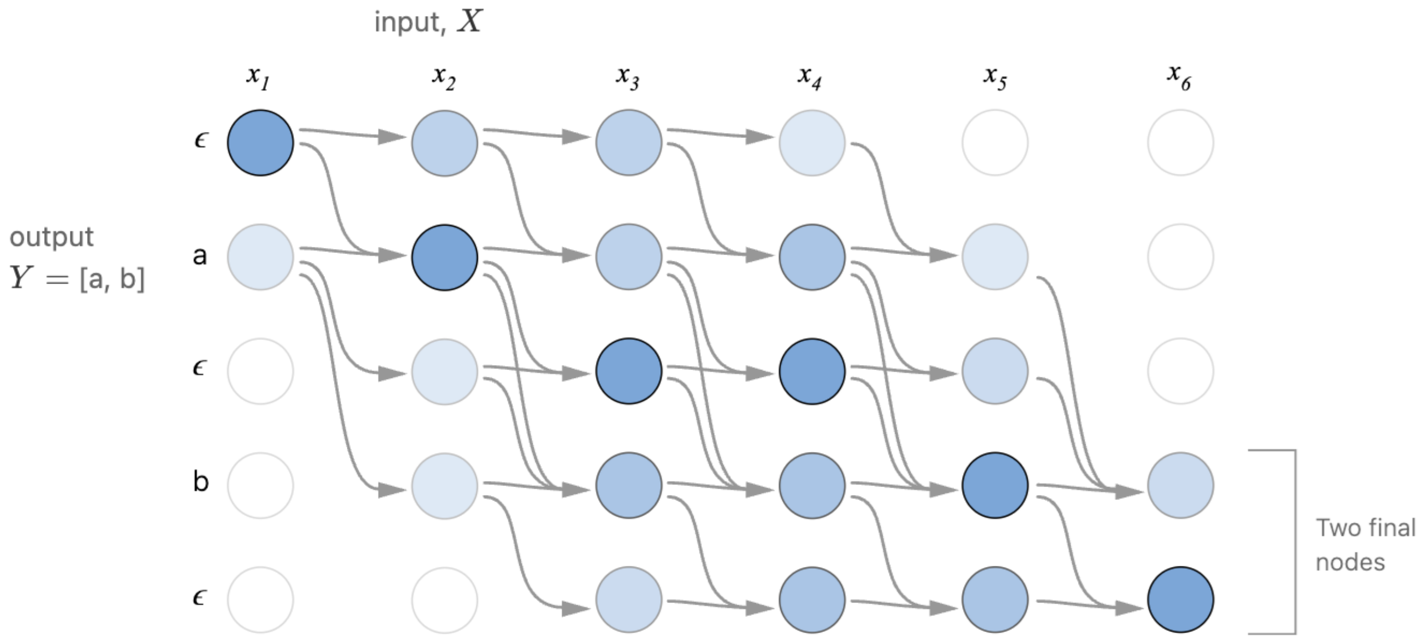
$$\begin{aligned} p\left(\text{out=ab, last\_char} = \text{b}\right)_T = \;& \\ & p(\text{prefix=a, last\_char} = \text{b})_{T-1} \cdot p(\text{char=b})_T \\ +\;& p(\text{prefix=a, last\_char} = \varepsilon)_{T-1} \cdot p(\text{char=b})_T \\ +\;& p(\text{prefix=a, last\_char} = \text{a})_{T-1} \cdot p(\text{char=b})_T \end{aligned}$$

$$\begin{aligned} p\left(\text{out=ab, last\_char} = \varepsilon\right)_T = \;& \\ & p(\text{prefix=a, last\_char} = \text{b})_{T-1} \cdot p(\text{char=}\varepsilon)_T \\ +\;& p(\text{prefix=ab, last\_char} = \varepsilon)_{T-1} \cdot p(\text{char=}\varepsilon)_T \end{aligned}$$

DL-based ASR

- Efficient CTC Loss Compute

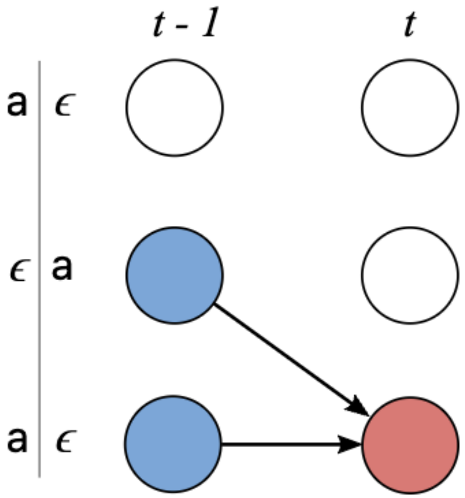
$$Z = [\epsilon,\ y_1,\ \epsilon,\ y_2,\ \ldots,\ \epsilon,\ y_U,\ \epsilon]$$



Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

DL-based ASR

- Efficient CTC Loss Compute: Case 1



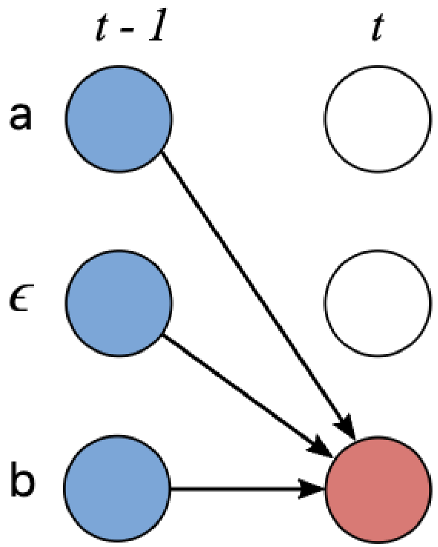
$$\alpha_{s,t} = (\alpha_{s-1,t-1} + \alpha_{s,t-1}) \cdot p_t(z_s \mid X)$$

The CTC probability of the two valid subsequences after $t-1$ input steps.

The probability of the current character at input step t.

DL-based ASR

- Efficient CTC Loss Compute: Case 2



$$\alpha_{s,t} = (\alpha_{s-2,t-1} + \alpha_{s-1,t-1} + \alpha_{s,t-1}) \cdot p_t(z_s \mid X)$$

The CTC probability of the three valid subsequences after $t-1$ input steps.

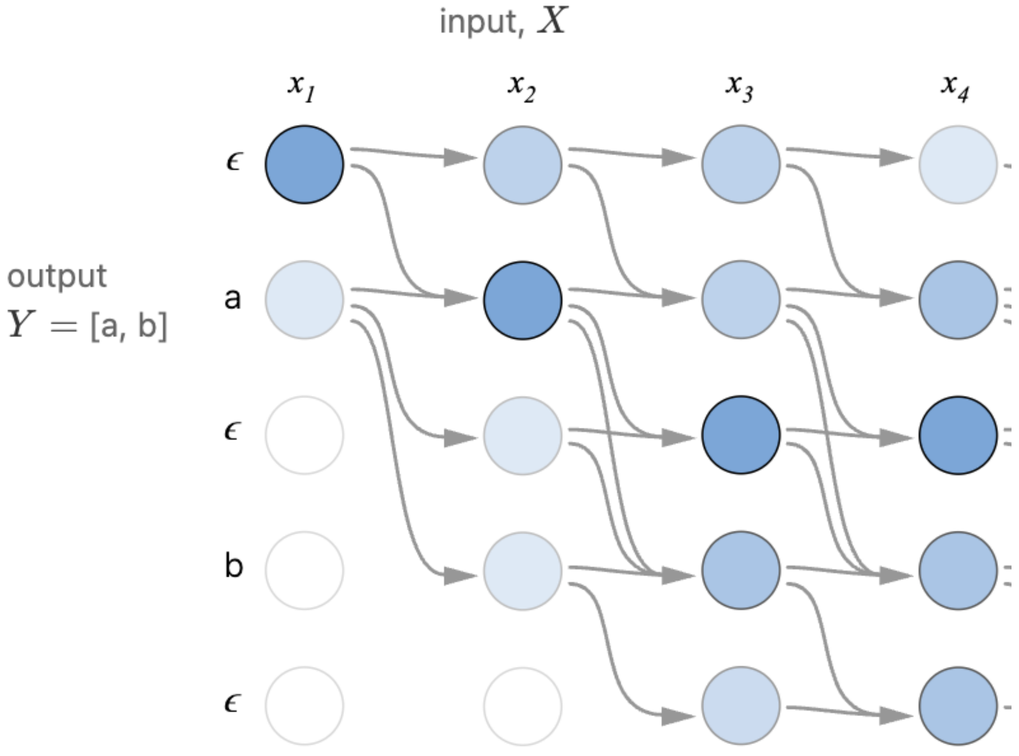
The probability of the current character at input step t.

DL-based ASR

t	t=1	t=2	t=3	t=4
$\mathcal{E}$	0.6	0.2	0.3	0.4
a	0.3	0.5	0.2	0.1
b	0.1	0.3	0.5	0.5

O: probability distribution

- Efficient CTC Loss Compute: Example
  - Output $Y = [a, b]$



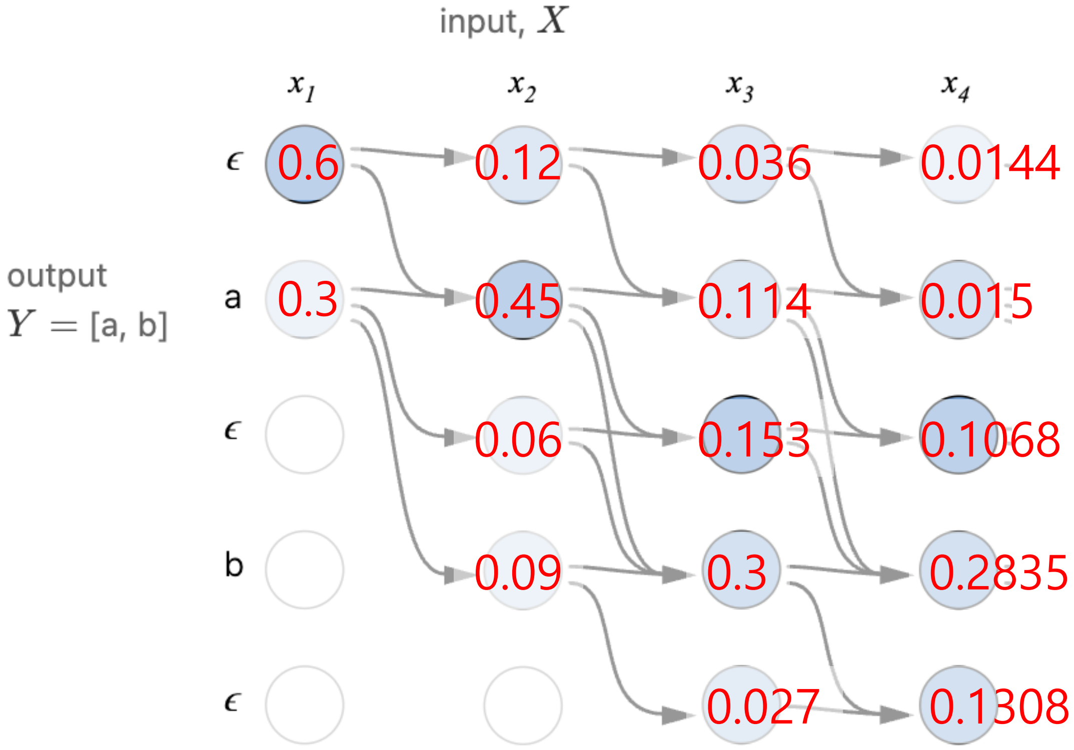
Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

DL-based ASR

t	t=1	t=2	t=3	t=4
$\mathcal{E}$	0.6	0.2	0.3	0.4
a	0.3	0.5	0.2	0.1
b	0.1	0.3	0.5	0.5

O: probability distribution

- Efficient CTC Loss Compute: Example
  - Output $Y = [a, b]$



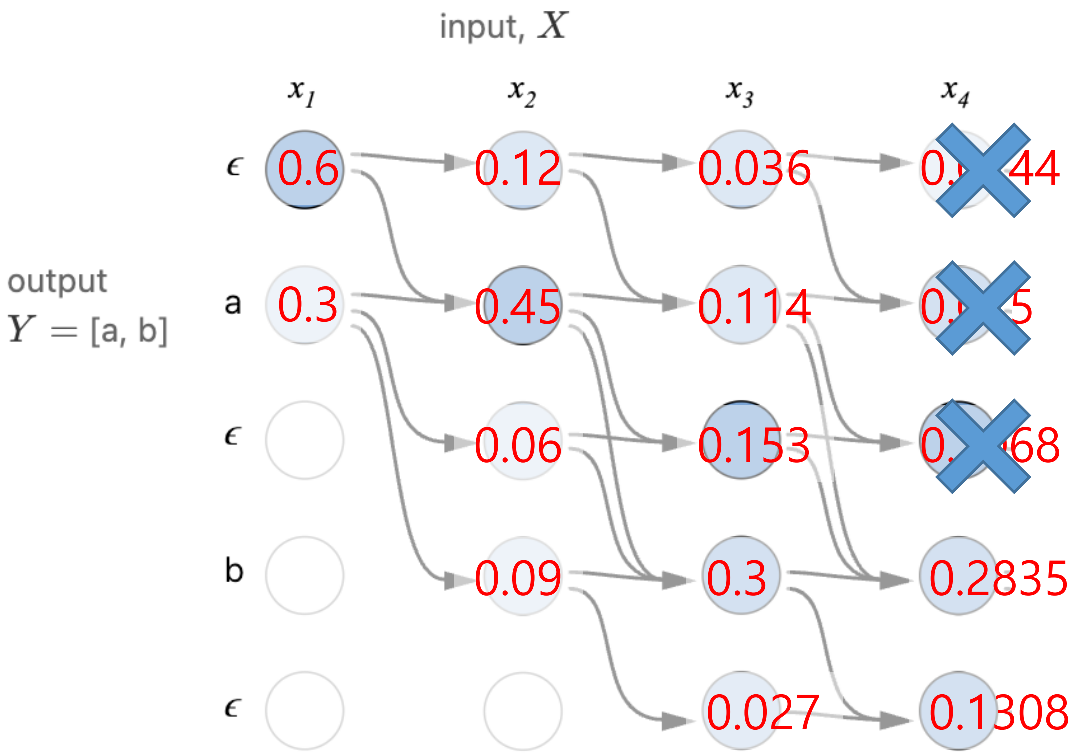
Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

DL-based ASR

t	t=1	t=2	t=3	t=4
$\mathcal{E}$	0.6	0.2	0.3	0.4
a	0.3	0.5	0.2	0.1
b	0.1	0.3	0.5	0.5

O: probability distribution

- Efficient CTC Loss Compute: Example
  - Output $Y = [a, b]$



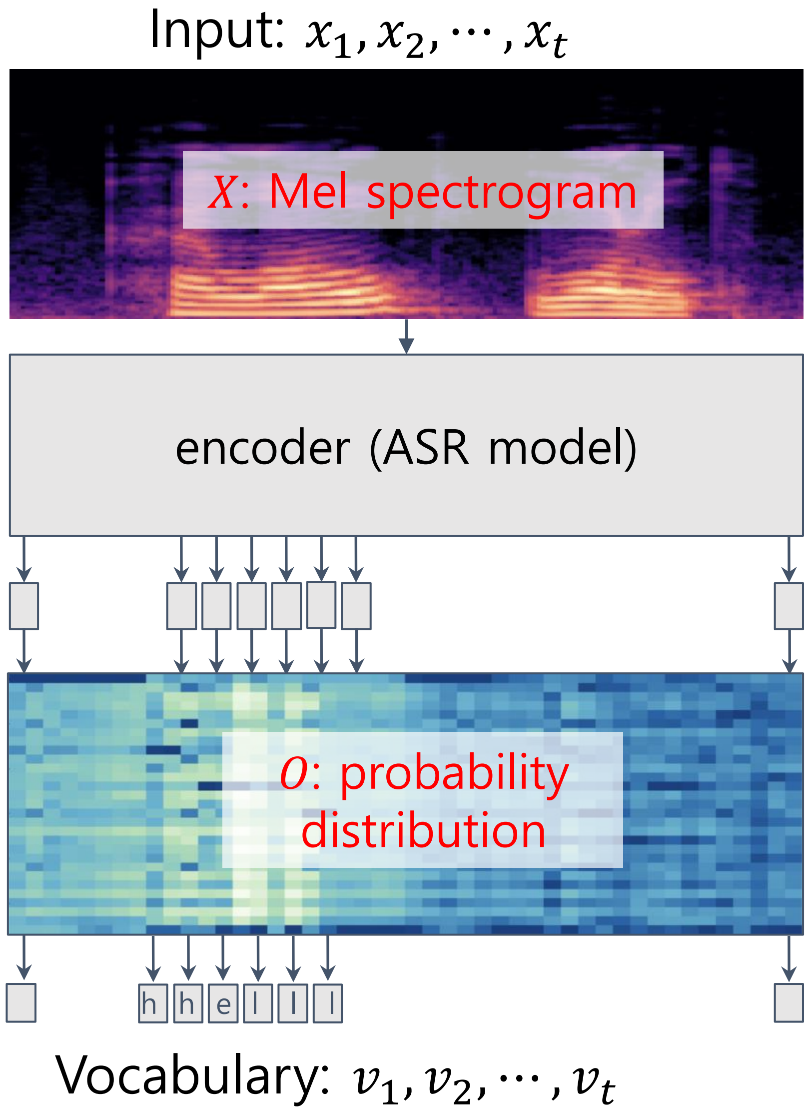
Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

0.2835+0.1308=0.4143

Forward DP
(Dynamic Programming)

DL-based ASR

- CTC properties
  - **No language context**: outputs are independent, meaning that each letter does not know about others
  - **Streamable**: we don't need full audio, to start predicting (how to inference?)
  - **Produces alignment**: between audio and target



DL-based ASR

- CTC inference
  - argmax is fast, but:
    - **Problems:**

    - $P([b, b, b]) = 0.5$ ← argmax
    - $P([a, a, \epsilon]) = 0.4$
    - $P([a, a, a]) = 0.2$
    - …

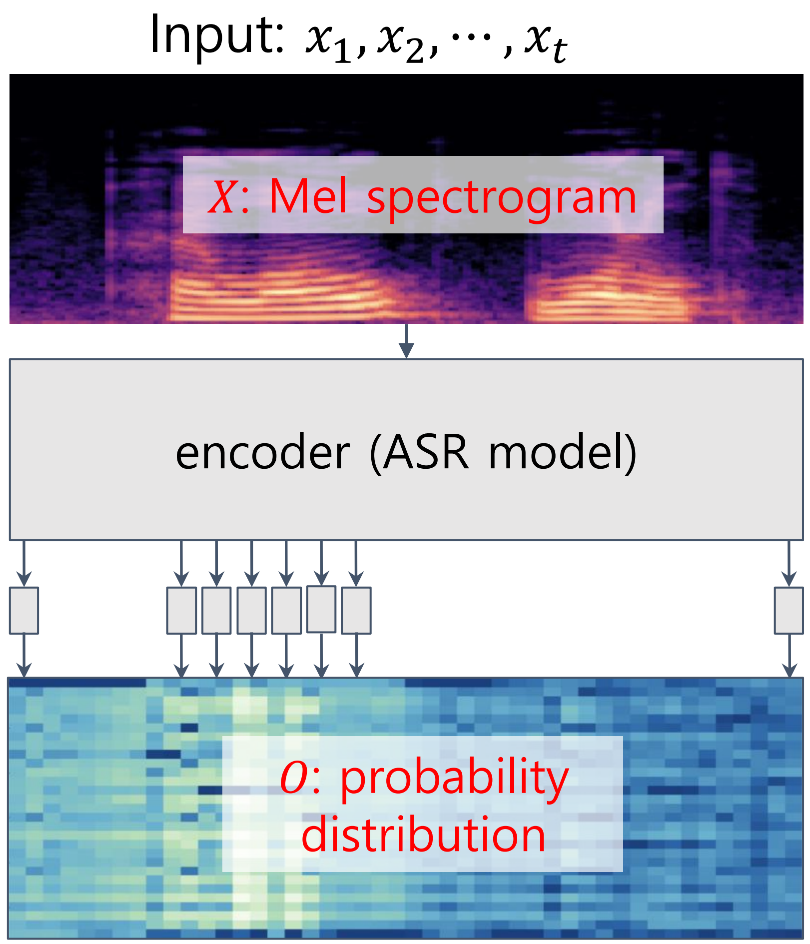
    - **but:**

    - $p("a") = 0.6$
    - $p("b") = 0.5$

We can't compute all paths

DL-based ASR

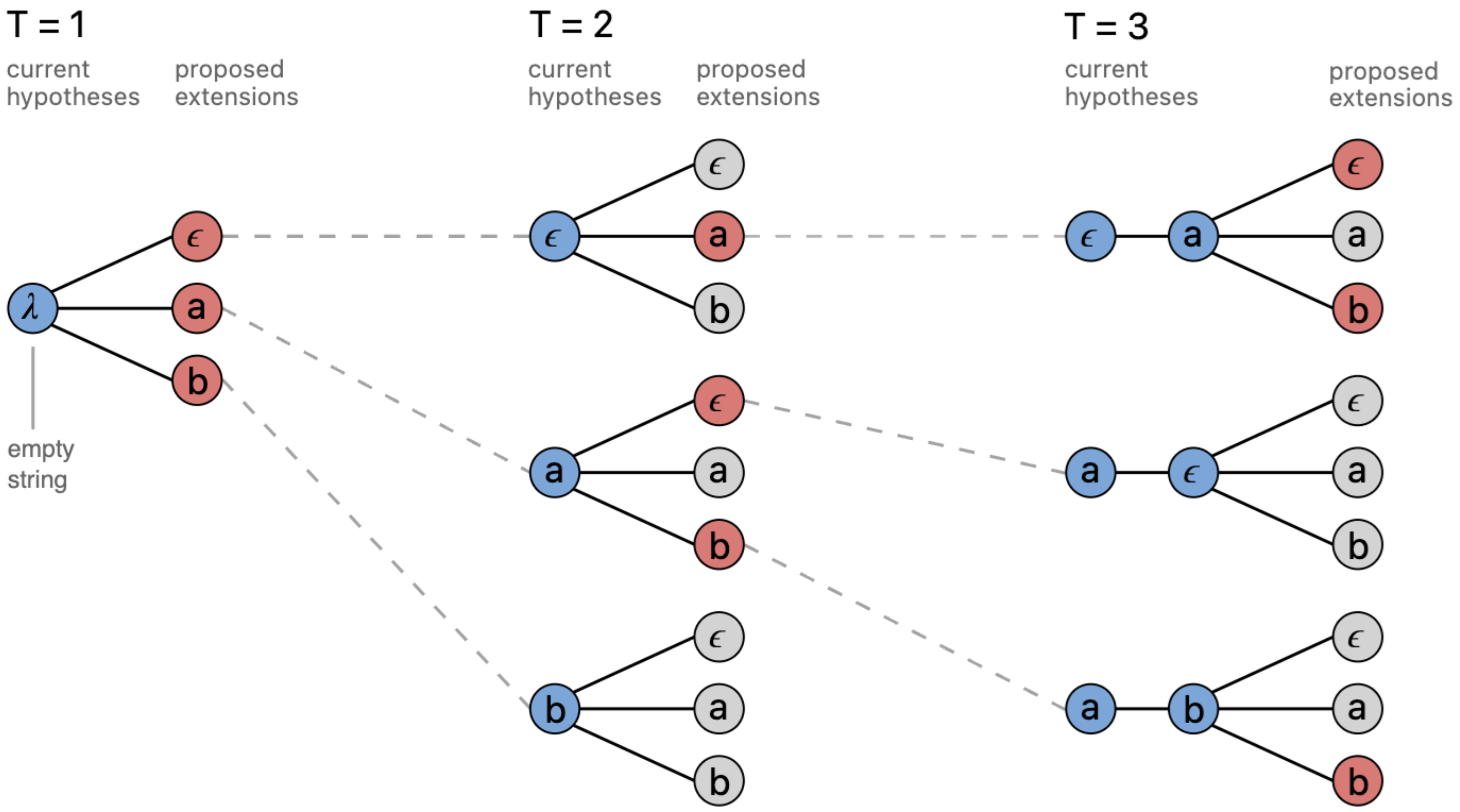
- CTC Beam Search
  - We want to decode an output matrix into a list of text predictions with probabilities
  - Has parameter: beam size >= 1
  - Tradeoff between:
    - **Taking argmax prediction**: easy and fast to compute, but not perfectly accurate
    - **Computing sum of probabilities of all possible paths**: Perfect quality, but <u>infeasible</u>



DL-based ASR

- CTC Beam Search (Beam size=3)

Beam search step:
- expand beam
- truncate beam

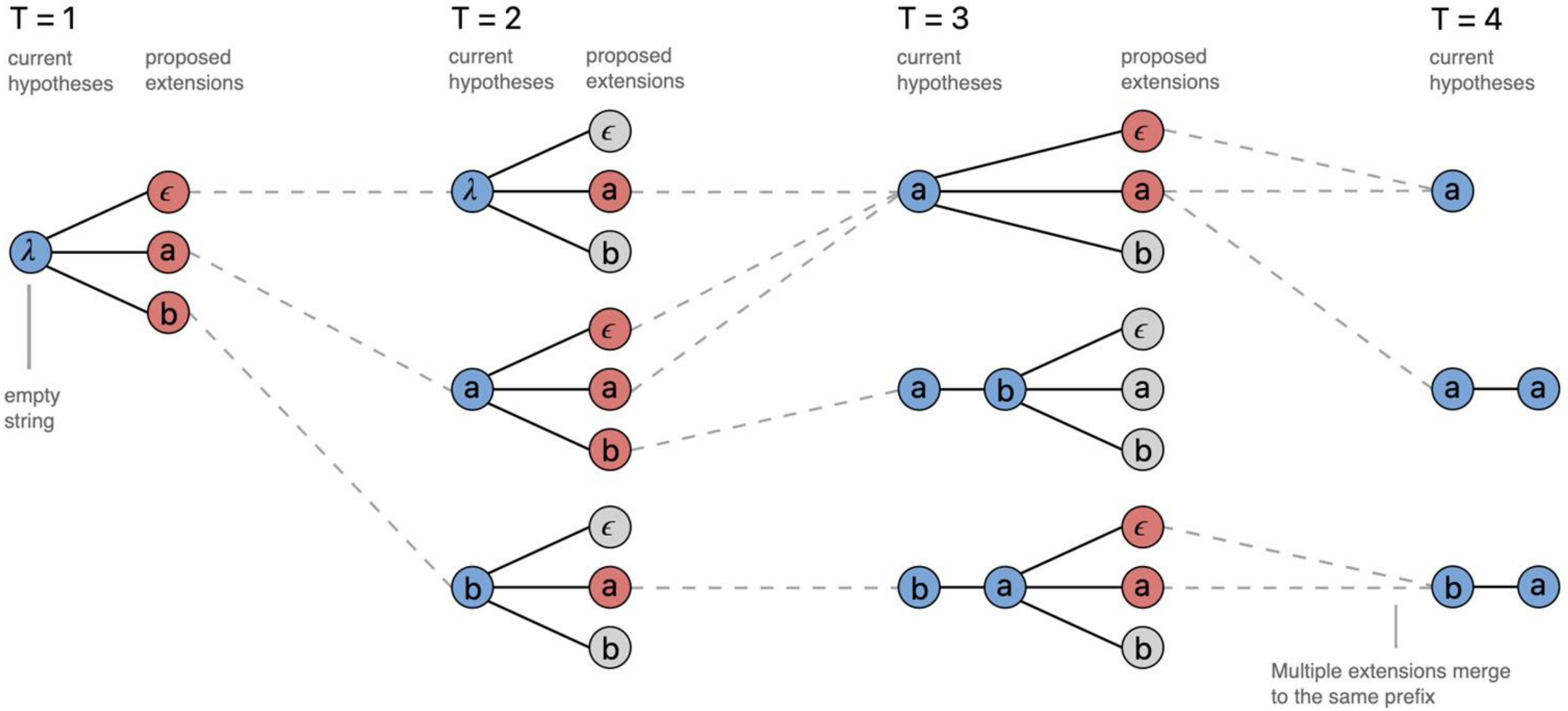


A standard beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three.

DL-based ASR

Beam search step:
- expand beam
- merge paths
- truncate beam

- CTC Beam Search (Beam size=3)



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

DL-based ASR

- CTC Summary

* Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks, A. Graves et. al, 2006

CTC Properties

- conditional independence
- streamable
- produces alignment
- fast inference

CTC Decoding

- allows to reduce fixed len predictions to variable length predictions
- provides mapping from path to string

How to train:

- compute encoder predictions - output matrix
- efficiently compute sum of probs of all paths corresponding to target
- maximize this sum

How to evaluate:

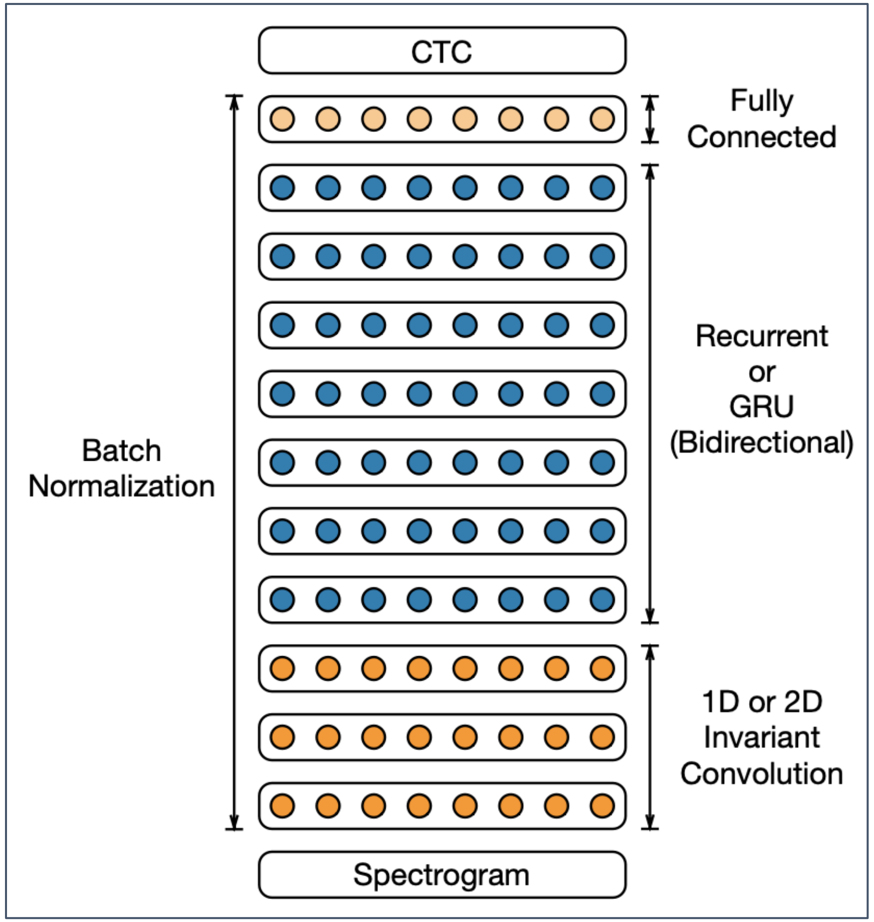
- compute encoder predictions - output matrix
- argmax / beam search
- CTC decode

Beam Search:

- expand beam
- merge paths
- truncate beam
- repeat

DL-based ASR

- Encoders: Deep Speech 2
  - **Deep Speech 2: End-to-End Speech Recognition in English and Mandarin,** Dario Amodei et al., 2015
    - 11,9k hours of training data
    - 35M parameters
    - no torch in 2015 - many challenges
    - super human quality on clean sets
    - 3-5 days on 16 GPUs

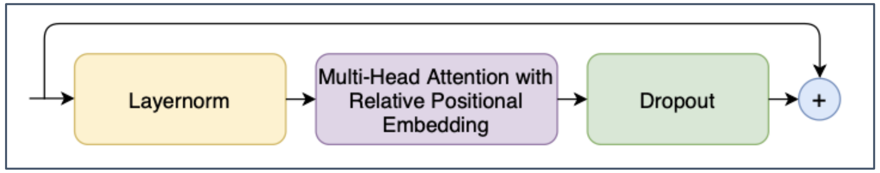


Read Speech			
Test set	DS1	DS2	Human
WSJ eval'92	4.94	3.60	5.03
WSJ eval'93	6.94	4.98	8.08
LibriSpeech test-clean	7.89	5.33	5.83
LibriSpeech test-other	21.74	13.25	12.69

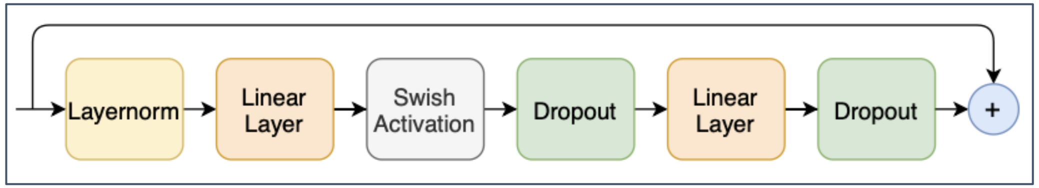
DL-based ASR

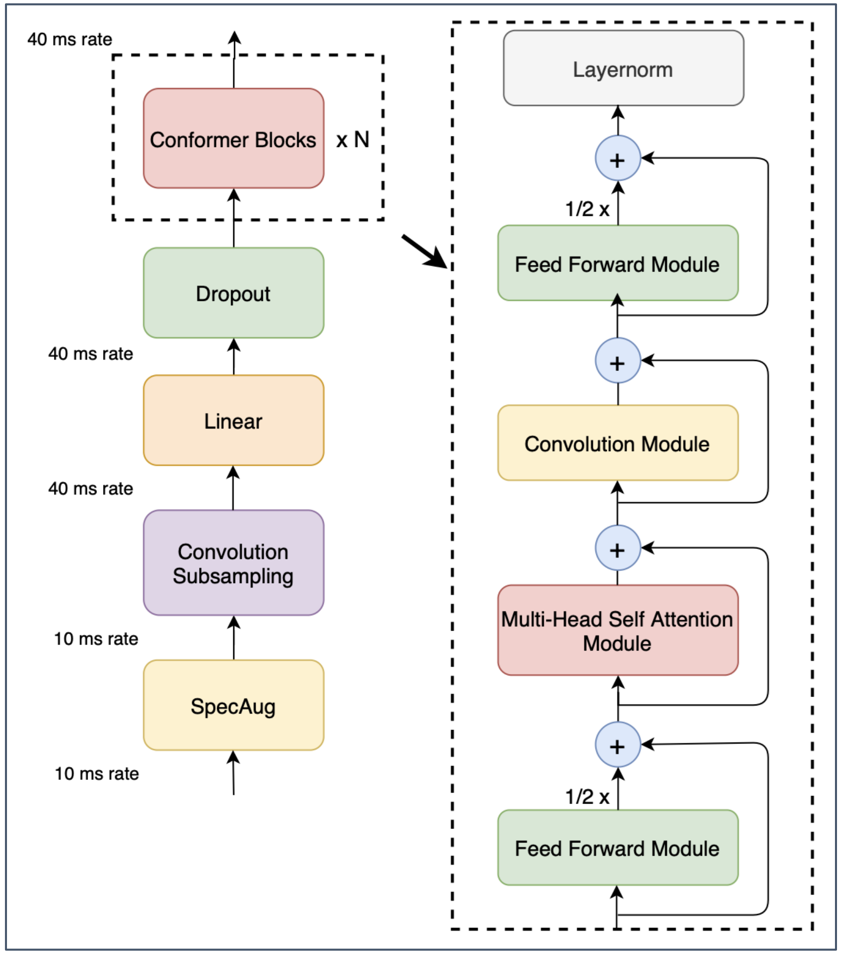
- Encoders: Conformer
  - **Conformer: Convolution-augmented Transformer for Speech Recognition, Anmol Gulati, Google, 2020**
    - Transformers
    - 1D depthwise separable convolution
    - <u>convolutions for local</u>, attention for global dependencies

Multi-head self-attention model



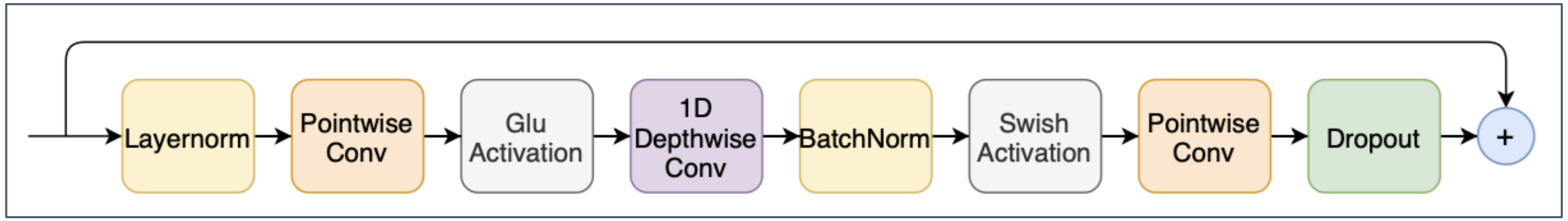
Feed forward module





DL-based ASR

- Encoders: Conformer
  - **Conformer: Convolution-augmented Transformer for Speech Recognition, Anmol Gulati, Google, 2020**

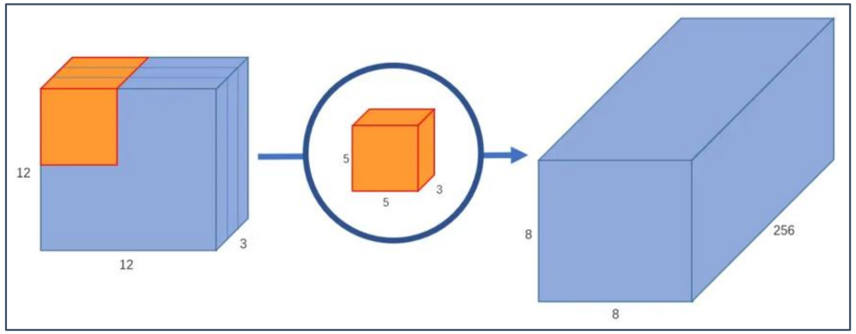


Convolution module

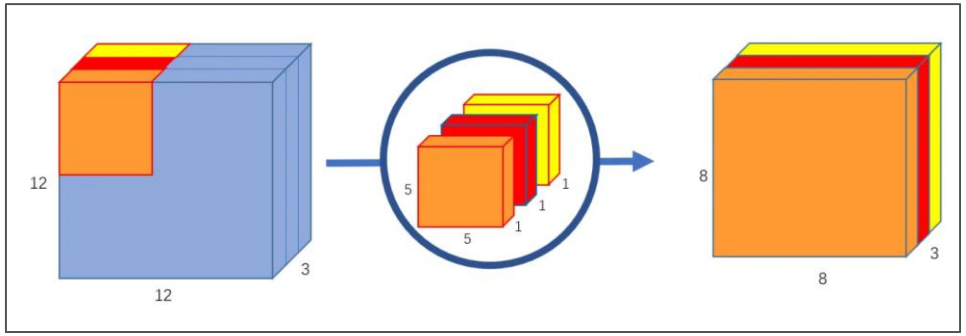
DL-based ASR

- Encoders: Conformer
  - **Conformer: Convolution-augmented Transformer for Speech Recognition, Anmol Gulati, Google, 2020**

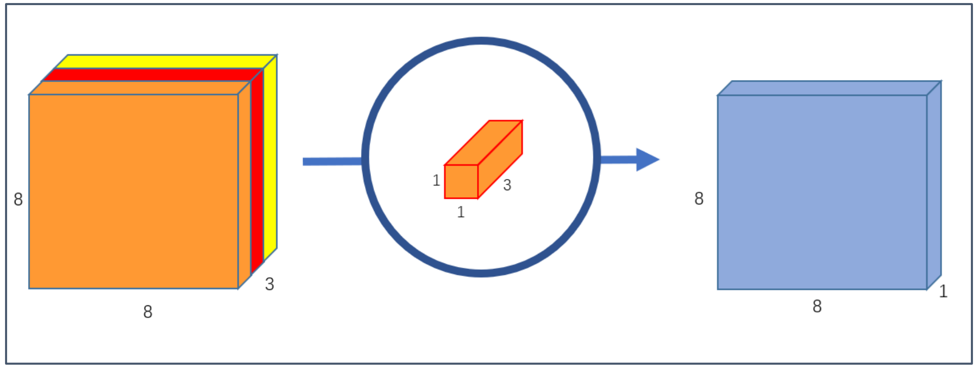
Standard Convolution



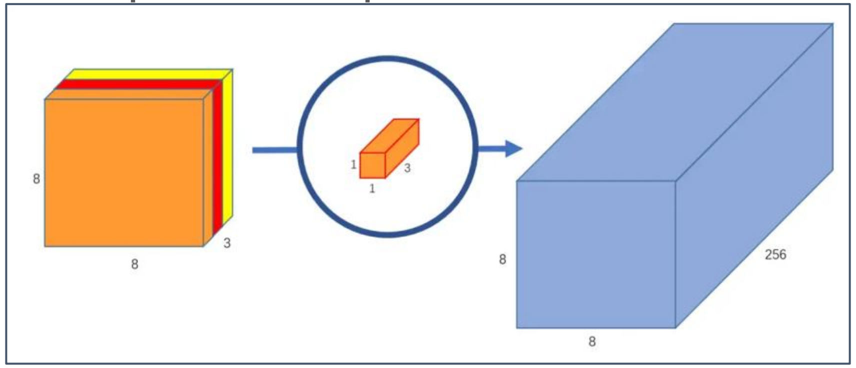
Depthwise Convolution



Pointwise Convolution



Depthwise Separable Convolution



DL-based ASR

- Subword Tokenization
  - motivation:
    - a lot of characters have different pronunciation in different contexts (e.g. cheese and character)
    - generated <u>tokens</u> are conditionally independent of each other (in case of CTC)
    - reducing the target sentence length by subword tokenization, we are trying to sidestep the CTC loss sequence length limitation
    - less compute leads to faster training and inference
    - better generalization leads to better quality
  - subword tokenization:
    - Byte Pair Encoding (BPE)
    - Word Piece
    - Unigram

DL-based ASR

- Subword Tokenization: BPE (Byte Pair Encoding)

Algorithm 1 Byte-pair encoding [30]

Input: set of strings D, target vocab size K
procedure BPE(D, K)
  $V \leftarrow$ all unique characters in D
  while $|V| < K$ **do** ▷ *Merge tokens*
    $t_L, t_r \leftarrow$ Most frequent bigram in D
    $t_{NEW} \leftarrow t_L + t_R$ ▷ *Make new token*
    $V \leftarrow V + [t_{NEW}]$
    Replace each occurrence of t_L, t_r in D with t_{NEW}
  end while
  return V
end procedure

aaabdaaabac

1. *add* **Z** <- aa
   Zabd**Z**abac

2. *add* **Y** <- ab
   ZYd**ZY**ac

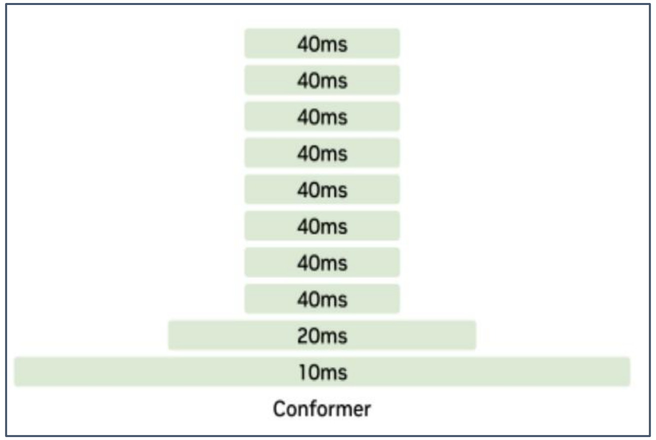
3. *add* **X** <- ZY
   Xd**X**ac

Example: low, lowest, newer → add "lo"=**A**, **A**w, **A**west, newer → ...

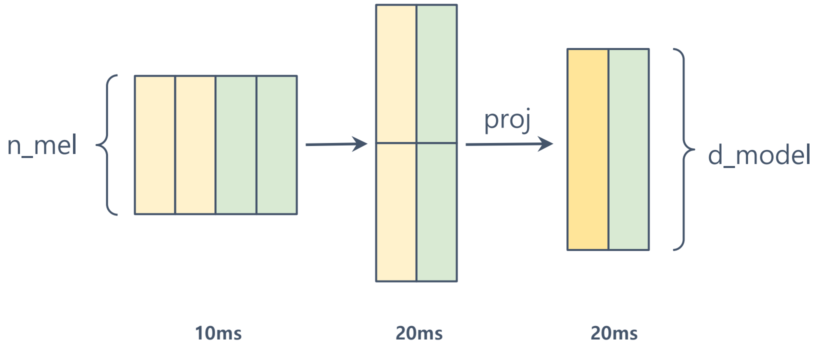
V: l, o, w, n, e, s, t, r   **V**: **A**, w, n, e, s, t, r

DL-based ASR

- Subsampling
  - motivation:
    - speed up training and inference
    - subword tokenization can work better, because we will encode more data per frame



  - options:
    - 1d conv with stride
    - 2d conv and projection
    - stacking consecutive frames and projection
    - and many others..



DL-based ASR

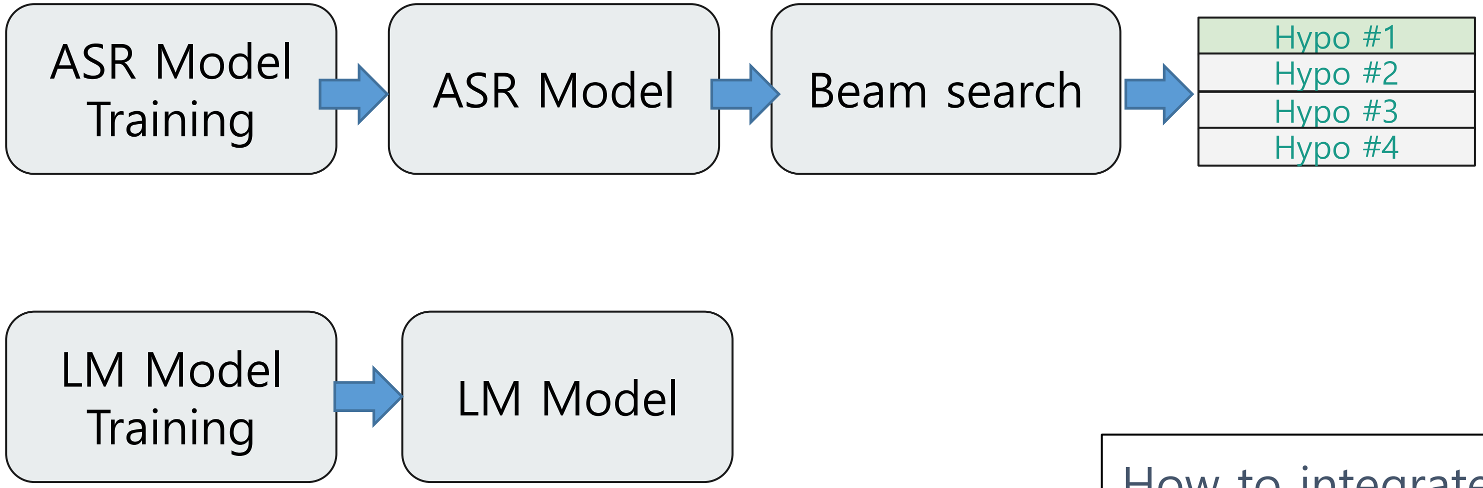
- Language models (LM)
  - motivation:
    - spelling of a word heavily depends on its context
    - LMs can add context dependency in CTC beam search
    - external language models typically saw more data
  - examples:
    - simple: n-gramms, Kneser–Ney smoothing and others
    - complex: neural networks
      e.g: Bert, GPT, LLaMA

hypo 1: let's go ***two*** a movie (am score: 0.21)
hypo 2: let's go ***to*** a movie (am score: 0.19)
hypo 3: let's go ***too*** a movie (am score: 0.13)

P(let's go ***two*** a movie) = 0.01 (lm score)
P(let's go ***to*** a movie) = 0.6 (lm score)

DL-based ASR

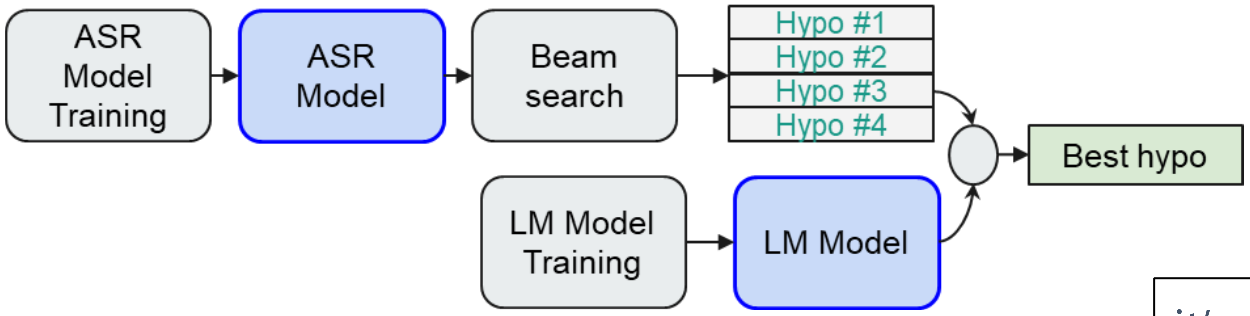
- Language models (LM)



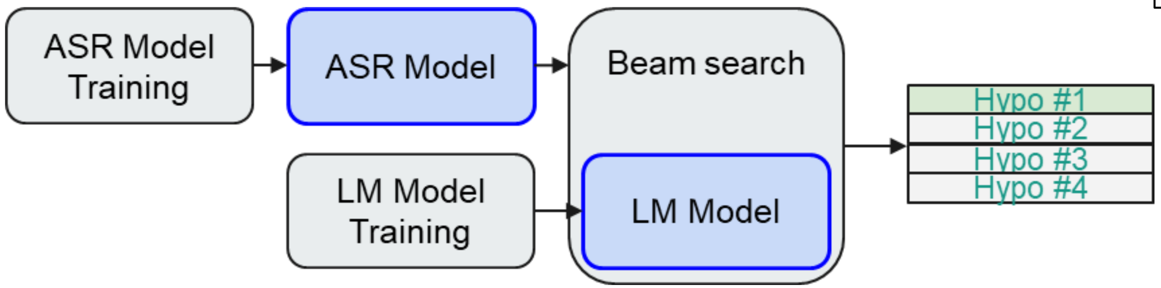
How to integrate LM?

DL-based ASR

- Language models (LM)
  - Second pass rescoring:



  - Shallow fusion:



it's better to use:

- large model (ASR) for rescoring
- light model (ASR) for shallow fusion