Yandex

SCHOOL OF DATA ANALYSIS

ASR II: Discriminative State-Space Models I

Andrey Malinin

14th March 2022

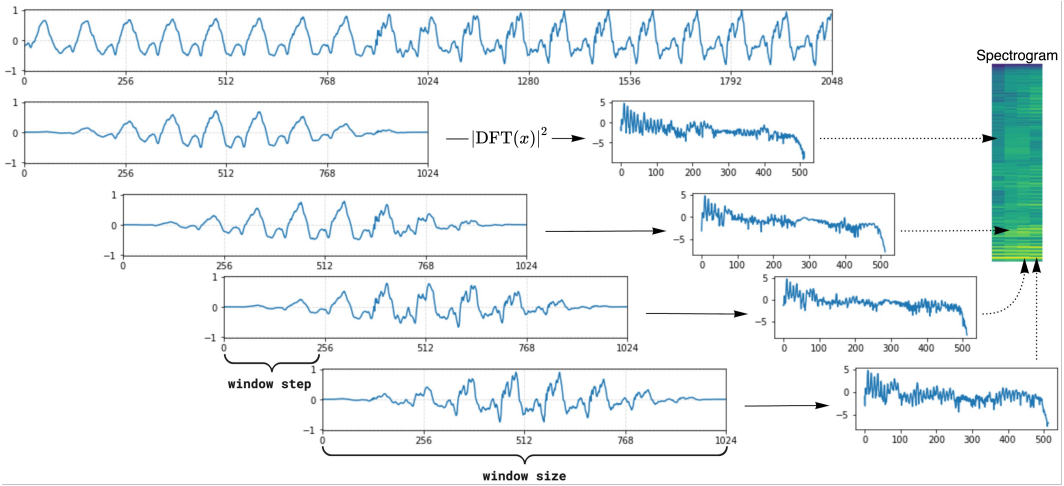
Story so far

In this previous episode...

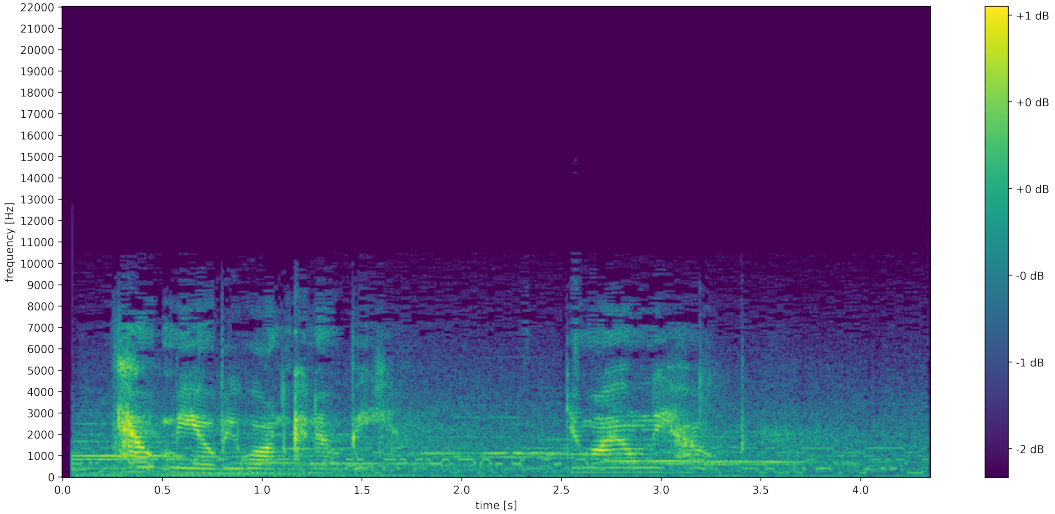
Recap - Data Processing

- Transform audio into Mel-Spectrogram
  - Yields sequence of acoustic features or frames $\boldsymbol{X}_{1:T}$
  - This decreases the time-resolution (downsamples) of the audio data
  - Length of each frame depends on audio sample-rate and hop-length
- Transform text into a sequence of speech units $\omega_{1:L}$
  - Speech units can be phonemes, gramphes, syllables, etc..
  - Speech units must be compact and easy to back to / from words

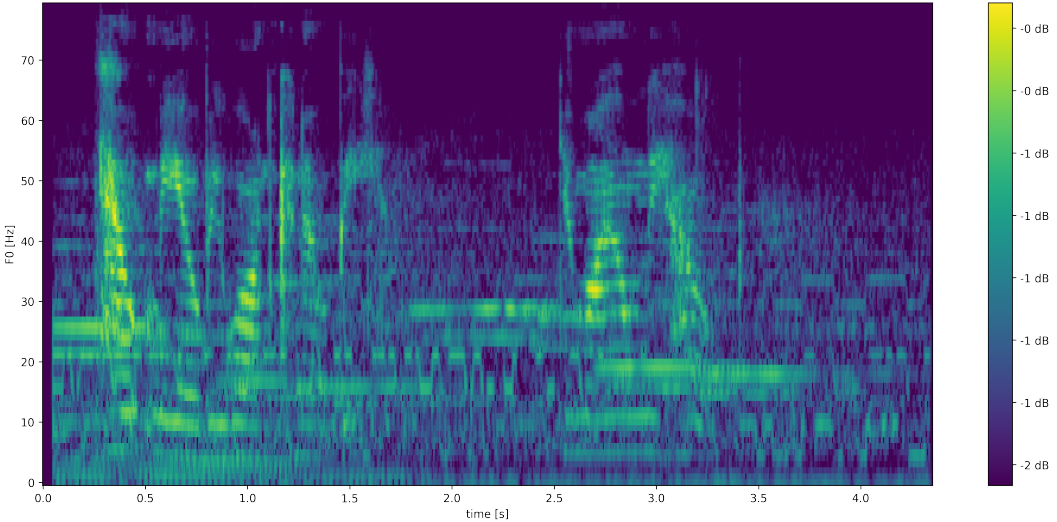
DSP Reminder - Audio to Spectrogram via STFT



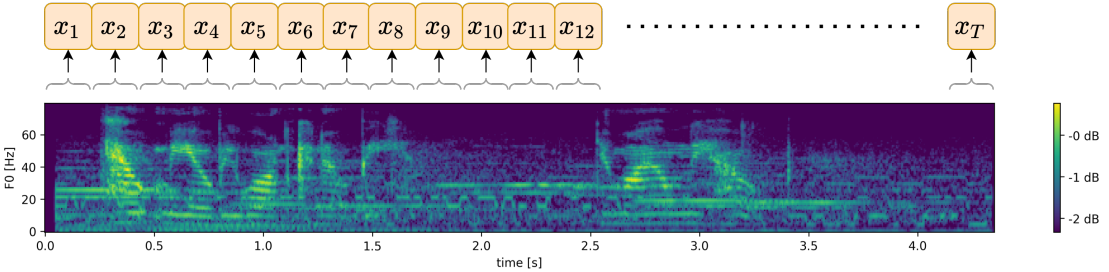
DSP Reminder - Spectrogram



DSP Reminder - Mel-Spectrogram



DSP Reminder - Acoustic Features



Extract **acoustic features**, **observations** or **frames** $\boldsymbol{X}_{1:T} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\}$

- At a 22.05KHz sample-rate each second is 22050 samples long!
- DFT down-samples $\rightarrow$ At 512 sample window-step each $\boldsymbol{x}_t$ is $\sim$25ms of audio.
- Acoustic sequence $\boldsymbol{X}_{1:T}$ is long - 43 frames / second @22KHz & 512 frame overlap

Can use either each time-step or further down-sample via convolutions, for example.

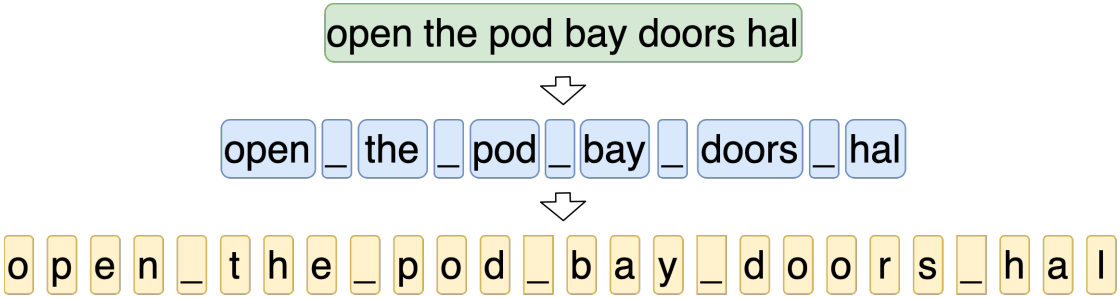
Text Processing - sub-word speech units

- Define a mapping $\mathcal{M}$ between words $\boldsymbol{w}$ and speech units $\boldsymbol{\omega}_{1:L}$.

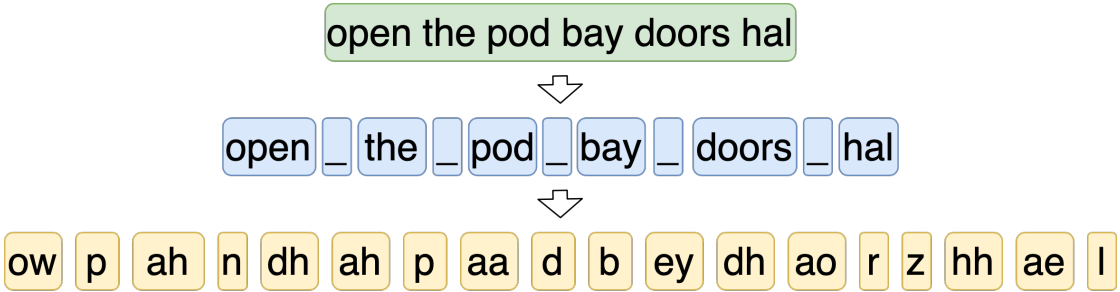
$$\{\boldsymbol{\omega}_{1:L_q}^{(q)}\}_{q=1}^{Q} = \mathcal{M}(\boldsymbol{w}), \quad \{\boldsymbol{w}^{(p)}\}_{p=1}^{P} = \mathcal{M}^{-1}(\boldsymbol{\omega}_{1:L})$$

- For some choices of speech units this mapping is not 1-to-1 (Q>1, P>1)
  - Different word pronunciations → different phoneme sequences
  - Different graphemic spellings (centre vs center)
- Desirable properties of speech units →
  - Well defined, simple mapping between words and speech units
  - Compact, even for large vocabulary, and extensible to OOV words.
  - Account for phonetic variability
- Choice depends on language properties, size of training data, type of ASR system

Mapping between words and speech units



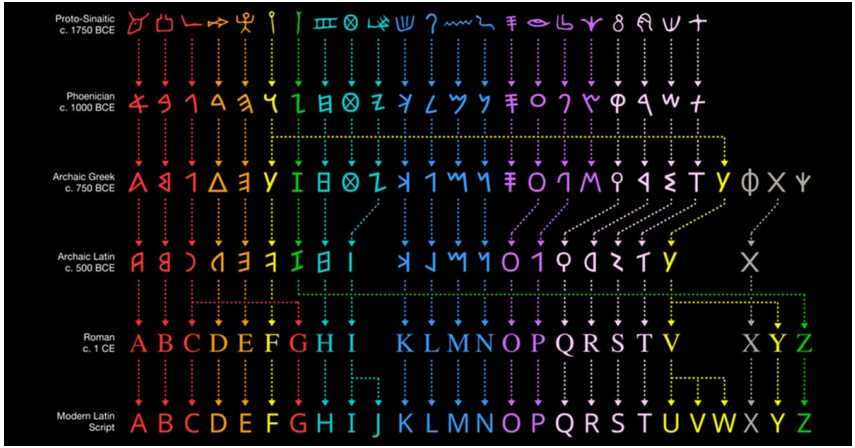
Mapping between words and speech units



Syllabic (basic Katakana)

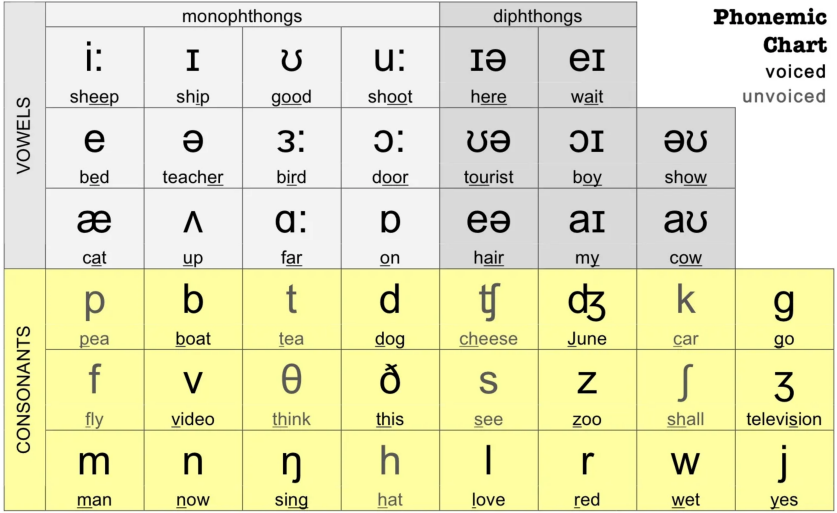
	a	i	u	e	o
Vowels	ア *a*	イ *i*	ウ *u*	エ *e*	オ *o*
k	カ *ka*	キ *ki*	ク *ku*	ケ *ke*	コ *ko*
s	サ *sa*	シ *shi*	ス *su*	セ *se*	ソ *so*
t	タ *ta*	チ *chi*	ツ *tsu*	テ *te*	ト *to*
n	ナ *na*	ニ *ni*	ヌ *nu*	ネ *ne*	ノ *no*
h	ハ *ha*	ヒ *hi*	フ *fu*	ヘ *he*	ホ *ho*
m	マ *ma*	ミ *mi*	ム *mu*	メ *me*	モ *mo*
y	ヤ *ya*		ユ *yu*		ヨ *yo*
r	ラ *ra*	リ *ri*	ル *ru*	レ *re*	ロ *ro*
w	ワ *wa*				ヲ *wo*
ン *n*					

Graphemic (Alphabet)



Matt Baker. Usefulcharts.com

Phonetic (IPA)

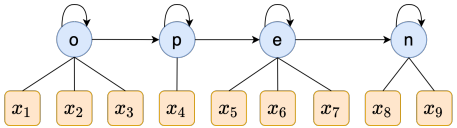


Recap - Data Processing

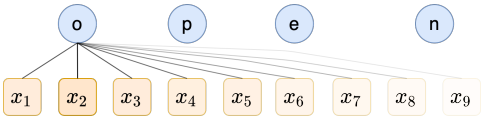
- Transform audio into Mel-Spectrogram
  - Yields sequence of acoustic features or frames $\boldsymbol{X}_{1:T}$
  - This decreases the time-resolution (downsamples) of the audio data
  - Length of each frame depends on audio sample-rate and hop-length
- Transform text into a sequence of speech units $\omega_{1:L}$
  - Speech units can be phonemes, gramphes, syllables, etc..
  - Speech units must be compact and easy to back to / from words

Problem of Alignment

- Which feature vectors $\boldsymbol{X}_{1:T}$ and speech units $\boldsymbol{\omega}_{1:L}$ relate or align to each other?
  - Models need to be able to dynamically align features vectors to words.
- Two common approaches to constructing models which can align
  - State-Space models
  - **Neural Attention Mechanisms**

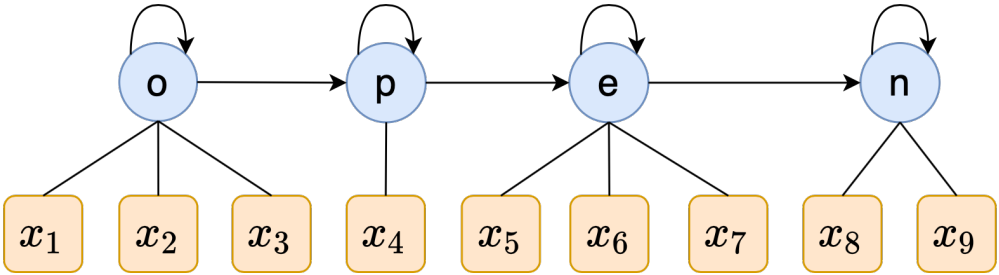


(a) State-Space

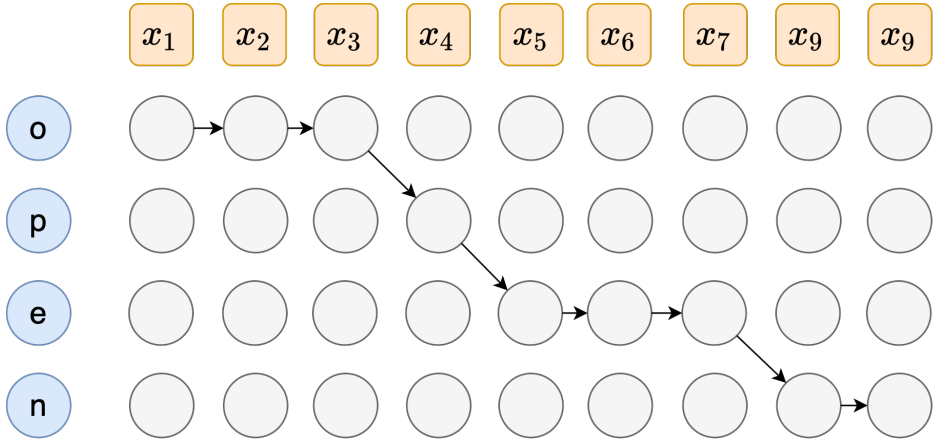


(b) Attention Mechanism

State-Space Alignment Models



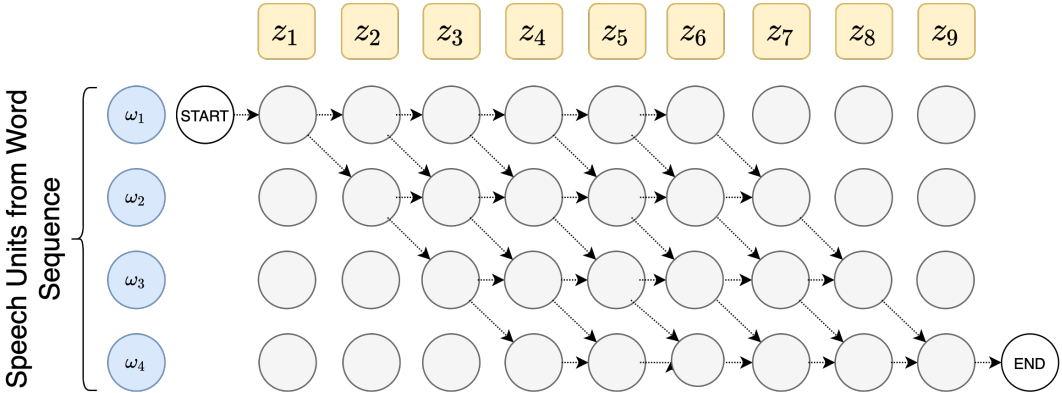
State-Space Trellis



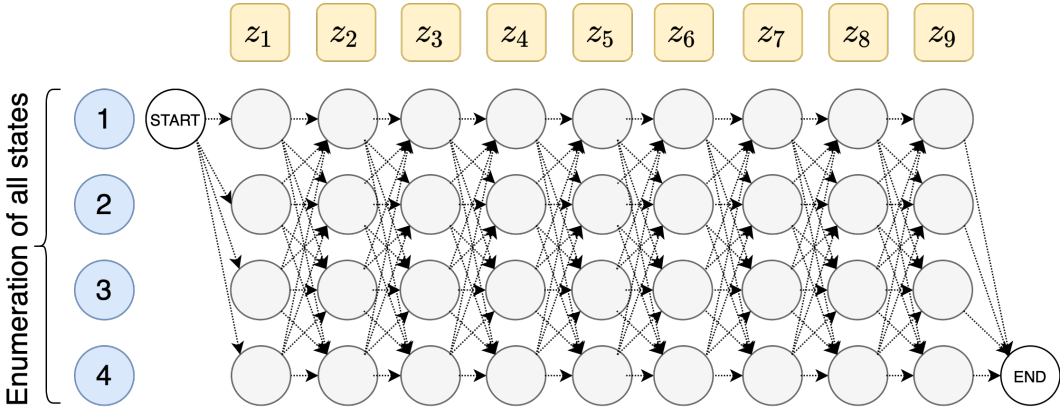
Hidden Markov Models - Trellis Structure

- Inference and Training / Alignment Trellises have difference structure!
  - Inference Trellis - Any path is valid
  - Training / Alignment Trellis - speech unit sequence defines set of valid paths

HMM Trellis - Training / Alignment Trellis



HMM Trellis - Inference Trellis



Discriminative State-Space ASR Models

- **Discriminative** ASR $\rightarrow$ "Which sequence $\hat{\boldsymbol{w}}$ is likely given the audio?"

$$\hat{\boldsymbol{w}} = \mathcal{M}^{-1}(\hat{\omega}_{1:L}), \quad \hat{\omega}_{1:L} = \arg\max_{\hat{\omega}_{1:L}} \mathrm{P}(\hat{\omega}_{1:L} | \boldsymbol{X}_{1:T}; \boldsymbol{\theta})$$

- **Generative** ASR $\rightarrow$ "Given which sequence $\hat{\boldsymbol{w}}$ is the audio mostly likely?"

$$\hat{\boldsymbol{w}} = \mathcal{M}^{-1}(\hat{\omega}_{1:L}), \quad \hat{\omega}_{1:L} = \arg\max_{\hat{\omega}_{1:L}} \mathrm{P}(\boldsymbol{X}_{\boldsymbol{1:T}} | \hat{\omega}_{1:L}; \boldsymbol{\theta}) \mathrm{P}(\hat{\omega}_{1:L}; \boldsymbol{\theta})$$

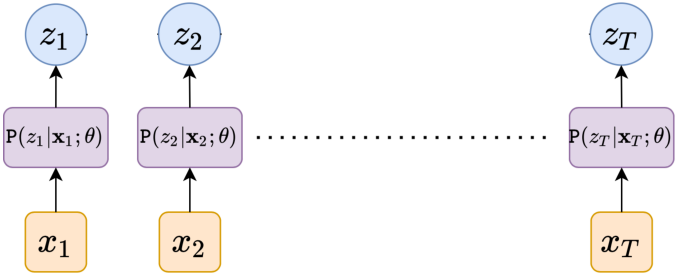
- ASR is a **discriminative** task $\rightarrow$ "Which sequence $\hat{\boldsymbol{w}}$ is likely given the audio?"

$$\hat{\boldsymbol{w}} = \mathcal{M}^{-1}(\hat{\boldsymbol{\omega}}_{1:L}), \quad \hat{\boldsymbol{\omega}}_{1:L} = \arg\max_{\hat{\boldsymbol{\omega}}_{1:L}} \mathrm{P}(\hat{\boldsymbol{\omega}}_{1:L} | \boldsymbol{X}_{1:T}; \boldsymbol{\theta})$$

- Discriminative State-Space ASR systems:

$$\mathrm{P}(\boldsymbol{\omega}_{1:L} | \boldsymbol{X}_{1:T}; \boldsymbol{\theta}) = \sum_{\boldsymbol{\pi}_{1:T} \in \mathcal{A}(\boldsymbol{\omega}_{1:L}, T)} \mathrm{P}(\boldsymbol{\pi}_{1:T} | \boldsymbol{X}_{1:T}; \boldsymbol{\theta})$$

Connectionist Temporal Classification (CTC)



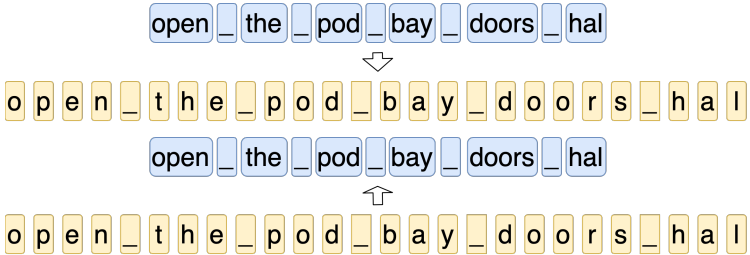
- CTC is a discriminative state-space ASR model defined as:

$$\mathrm{p}(\boldsymbol{w}|\boldsymbol{X}_{1:T};\boldsymbol{\theta}) = \sum_{\boldsymbol{\pi}_{1:T}\in\mathcal{A}(\boldsymbol{\omega}_{1:L},T)} \prod_{t=1}^{T} \mathrm{p}(z_t|\boldsymbol{x}_t;\boldsymbol{\theta}),\ \boldsymbol{\omega}_{1:L} = \mathcal{M}(\boldsymbol{w}),\ \boldsymbol{\pi}_{1:T} = \{z_t\}_{t=1}^{T}$$

Connectionist Temporal Classification (CTC) - Properties

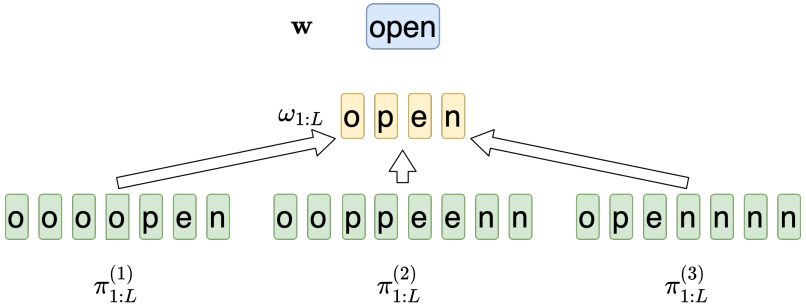
- CTC assumes all states are conditionally independent
  - Generally poor assumption, but context modelled implicitly by Neural Network.
  - State duration can also be poorly modelled
  - However, allows model to be more easily adapted by using external LM
- Alignment free - Do not need prior alignment for training
  - Train by maximizing probability of all valid alignments
- CTC models are typically graphemic with an extra black symbol ϵ.
  - Allows simple mapping between words and speech units
- CTC, like HMM allows only monotonic alignment
  - Good assumption for ASR
  - Cannot ignore any frames

Connectionist Temporal Classification (CTC)



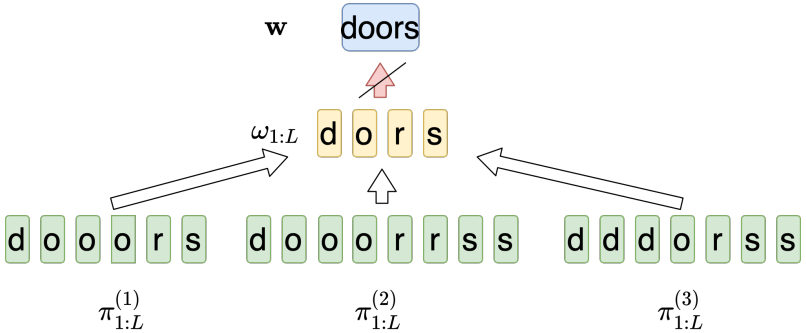
- Mapping word sequence to grapheme sequences is simple
  - We also need a procedure to map back from graphemes to words!
- A grapheme could be repeated for many states. Ex: oooopen $\rightarrow$ open
  - Could simply merge repeated graphemes!

Connectionist Temporal Classification (CTC)



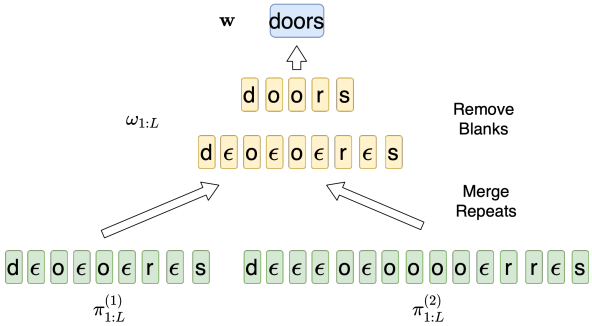
- Merging repeats is an easy operation
  - Can map many state sequences back to original grapheme sequence
  - But does this always work?

Connectionist Temporal Classification (CTC)



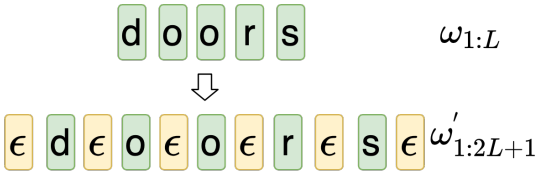
- Cannot properly restore words with repeated letters!

Connectionist Temporal Classification (CTC)



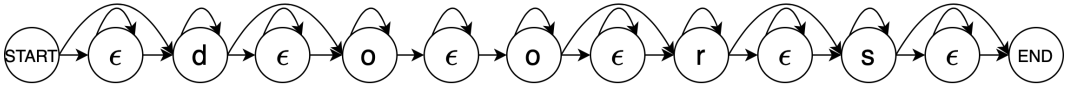
- Solve by introducing a **blank** symbol ϵ!
  - Restore works by first merging repeats, then removing blanks.
- Blanks follow a few simple rules
  - States sequence can start and end with blank
  - Blanks can come after any character, but can also be omitted - ex: doϵors

Connectionist Temporal Classification (CTC)



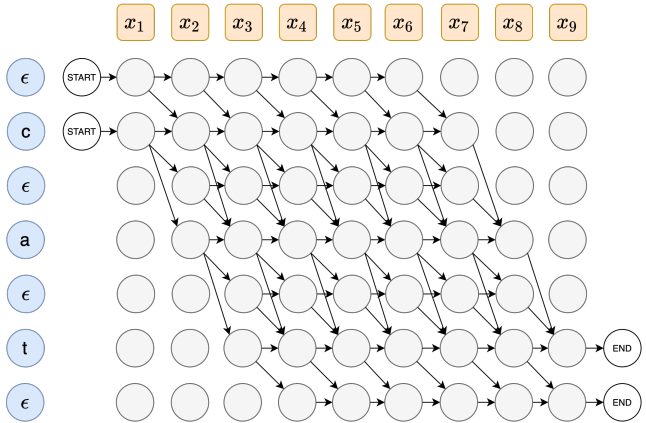
- For training use a sequence $\omega'_{1:2L+1}$ with a blank before and after each character

Connectionist Temporal Classification (CTC)



- Following state transitions in are $\omega'_{1:2L+1}$ valid:
  - Self-transitions
  - Transitions between blank and non-blank
  - Transitions between distinct non-blank labels
- Allows sequence $\omega'_{1:2L+1}$ contains all valid alignments.

Connectionist Temporal Classification (CTC) - Alignment and Training Trellis



- CTC trellis has 2 state and end states.

Ok, so how can we predict?

Greedy Decoding

- Predicting with CTC is simple as it does not model inter-state dependencies:

$$\mathrm{P}(\boldsymbol{z}_{1:T}|\boldsymbol{X}_{1:T}) = \prod_{t=1}^{T} \mathrm{P}(z_t|\boldsymbol{X}_{1:T})$$

$$\boldsymbol{\pi}^*_{1:T} = \arg\max_{\boldsymbol{\pi}_{1:T}} \prod_{t=1}^{T} \mathrm{P}(z_t = \pi_t|\boldsymbol{X}_{1:T})$$

- This yields most probable state sequence - then merge repeats and remove blanks.
- Greedy Decoding works well as CTC yield sharp distributions
- GD can still fail to find the most likely $\boldsymbol{\omega}^*_{1:L}$
  - Most likely path $\boldsymbol{\omega}^*_{1:L}$ may be composed of multiple, less likely state sequences.
- Can do better with Prefix-Decoding, but need LM for big gains.

Ok, so how can we train CTC models?

Training - CTC-Loss Function

- CTC loss is alignment free $\rightarrow$ do not need to know a-prior alignment
  - Maximize probability of all valid alignments $\boldsymbol{\pi}_{1:T} \in \mathcal{A}(\boldsymbol{\omega}_{1:L})$

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{n=1}^{N} -\ln \mathrm{P}(\boldsymbol{\omega}_{1:L_n}^{(n)} | \boldsymbol{X}_{1:T_n}^{(n)}, \boldsymbol{\theta}) \\
&= \frac{1}{N}\sum_{n=1}^{N} -\ln \sum_{\boldsymbol{\pi}_{1:T_n} \in \mathcal{A}(\boldsymbol{\omega}_{1:L_n}^{(n)})} \mathrm{P}(\boldsymbol{\pi}_{1:T_n} | \boldsymbol{X}_{1:T_n}^{(n)}, \boldsymbol{\theta}) \\
&= \frac{1}{N}\sum_{n=1}^{N} -\ln \sum_{\boldsymbol{\pi}_{1:T_n} \in \mathcal{A}(\boldsymbol{\omega}_{1:L_n}^{(n)})} \prod_{t=1}^{T_n} \mathrm{P}(\pi_t | \boldsymbol{X}_{1:T_n}^{(n)}, \boldsymbol{\theta})
\end{aligned}$$

- Number of possible paths is $O(K^{2L+1})$ $\rightarrow$ use Dynamic Programming!

Training - CTC Forward Algorithm

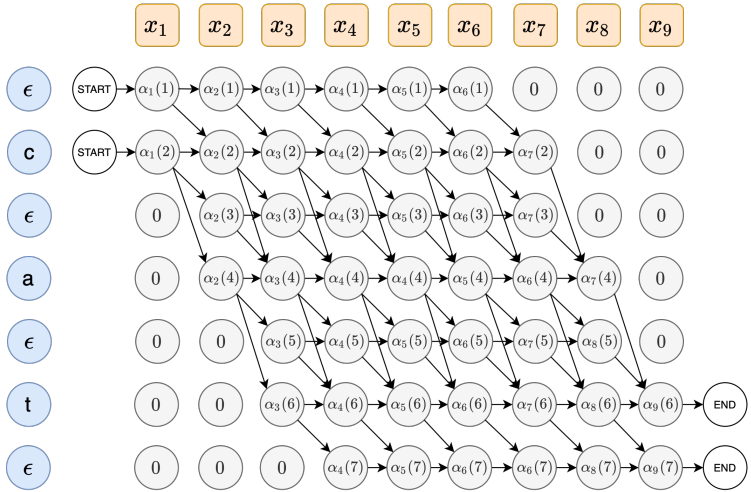
- The CTC forward algorithm recursively computes the forward variable $\alpha_t(s)$

$$\begin{aligned}\alpha_t(s) &= \mathrm{P}(\boldsymbol{\omega}_{1:s/2}, \pi_t = \omega'_s | \boldsymbol{X}_{1:T}, \boldsymbol{\theta}) \\ &= \sum_{\boldsymbol{\pi}_{1:t-1} \in \mathcal{A}(\boldsymbol{\omega}_{1:s/2}),\ \pi_t = \omega'_s} \mathrm{P}(\boldsymbol{\pi}_{1:t} | \boldsymbol{X}_{1:T}, \boldsymbol{\theta})\end{aligned}$$

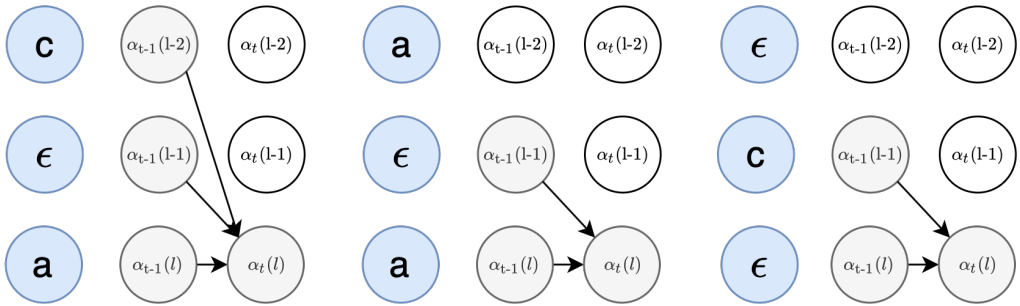
- $\alpha_t(s)$ is the probability of all paths of length t which go though state ω'_s
  - Note we consider alignments of subsequence $\boldsymbol{\omega}_{1:s/2}$!
- Final loss is given by the sum of alphas of two end states:

$$\mathrm{P}(\boldsymbol{\omega}_{1:L} | \boldsymbol{X}_{1:T}, \boldsymbol{\theta}) = \alpha_T(2L) + \alpha_T(2L+1)$$

Training - CTC Forward Algorithm



Training - CTC Forward Algorithm - Edge Cases



(a) Can skip blank

(b) Can't skip blank

(c) Go to blank

Training - CTC Forward Algorithm

$$\alpha_t(0) = 0, \forall t$$
$$\alpha_1(1) = \mathrm{P}(z_1 = \epsilon | \boldsymbol{X}_{1:T}),$$
$$\alpha_1(2) = \mathrm{P}(z_1 = \omega_2^{'} | \boldsymbol{X}_{1:T}),$$
$$\alpha_1(s) = 0, \ \forall s > 2$$
$$\alpha_t(s) = 0, \ \forall s < (2L+1) - 2(T-t) - 1$$
$$\alpha_t(s) = \begin{cases} (\alpha_{t-1}(s) + \alpha_{t-1}(s-1))\mathrm{P}(z_t = \omega_s^{'} | \boldsymbol{X}_{1:T}) & \text{if } \omega_s^{'} = \epsilon \text{ or } \omega_s^{'} = \omega_{s-2}^{'} \\ (\alpha_{t-1}(s) + \alpha_{t-1}(s-1) + \alpha_{t-1}(s-2))\mathrm{P}(z_t = \omega_s^{'} | \boldsymbol{X}_{1:T}) & \text{otherwise} \end{cases}$$

Training - CTC Backward Algorithm

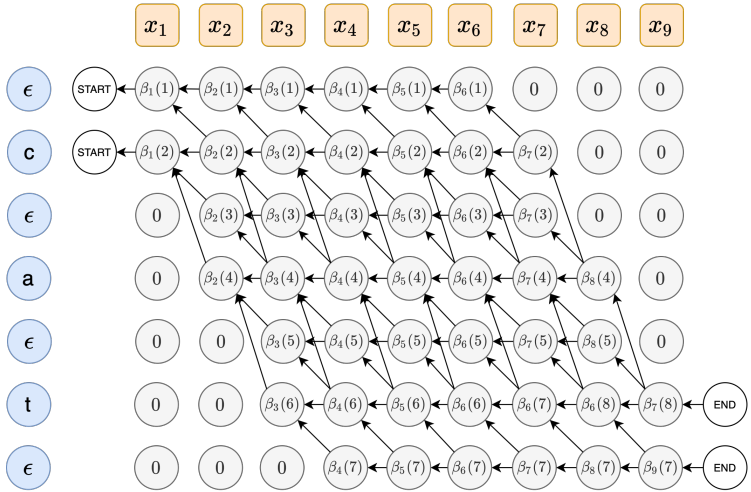
- The CTC backward algorithm recursively computes the backward variable $\beta_t(s)$

$$\begin{aligned}\beta_t(s) &= \mathrm{P}(\boldsymbol{\omega}_{s/2:L}, \pi_t = \omega_s' | \boldsymbol{X}_{1:T}, \boldsymbol{\theta}) \\ &= \sum_{\boldsymbol{\pi}_{t+1:T} \in \mathcal{A}(\boldsymbol{\omega}_{s/2:L}),\ \pi_t = \omega_s'} \mathrm{P}(\boldsymbol{\pi}_{t+1:T} | \boldsymbol{X}_{1:T}, \boldsymbol{\theta})\end{aligned}$$

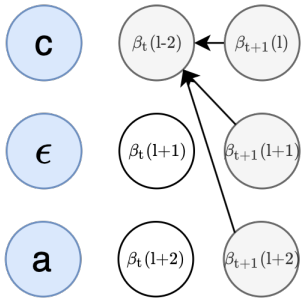
- $\beta_t(s)$ is the probability of all valid alignments $\boldsymbol{\omega}_{s/2:L}$ starting in state ω_s'
- Final loss also given by the sum of betas of two start states:

$$\mathrm{P}(\boldsymbol{\omega}_{1:L} | \boldsymbol{X}_{1:T}, \boldsymbol{\theta}) = \beta_1(1) + \beta_1(2)$$

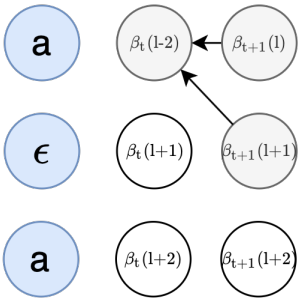
Training - CTC Backward Algorithm



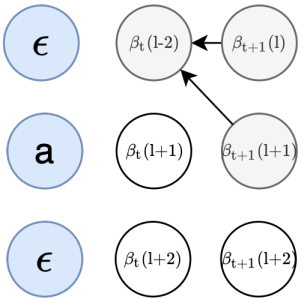
Training - CTC Backward Algorithm - Edge Cases



(a) Can skip blank



(b) Can't skip blank



(c) Go to blank

Training - CTC Backward Algorithm

$$\beta_T(2L+1) = 1$$
$$\beta_T(2L) = 1$$
$$\beta_T(s) = 0, \forall s < 2L$$
$$\beta_t(s) = 0, \ \forall s > 2t$$
$$\beta_t(2L+2) = 0, \ \forall t$$
$$\beta_t(s) = \begin{cases} (\beta_{t+1}(s) + \beta_{t+1}(s+1))\mathrm{P}(z_t = \omega_s^{'}|\boldsymbol{X}_{1:T}) & \text{if } \omega_s^{'} = \epsilon \text{ or } \omega_s^{'} = \omega_{s+2}^{'} \\ (\beta_{t+1}(s) + \beta_{t+1}(s+1) + \beta_{t+1}(s+2))\mathrm{P}(z_t = \omega_s^{'}|\boldsymbol{X}_{1:T}) & \text{otherwise} \end{cases}$$

CTC Forward-Backward Algorithm, Loss Computation and Soft-Alignment

- The probability of all paths passing through a state $\pi_t = \omega'_s$ is the product of forward and backward variables
  - This is important for the calculation of the gradient (not shown)

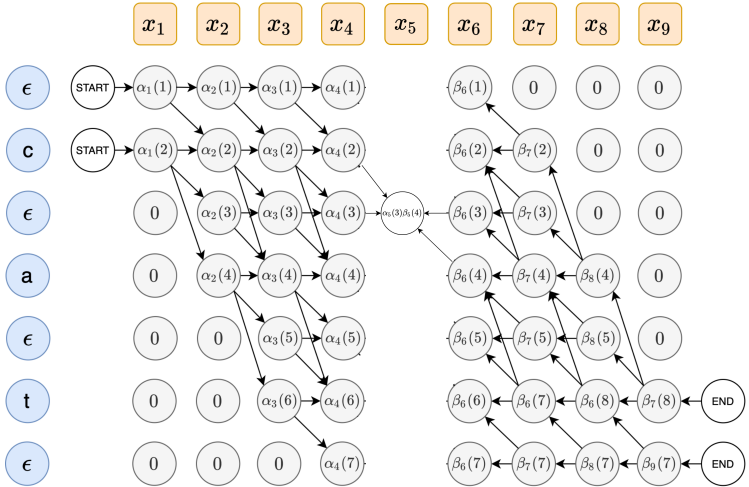
$$\alpha_t(s)\beta_t(s) = \sum_{\boldsymbol{\pi}_{1:T} \in \mathcal{A}(\boldsymbol{\omega}_{1:L}),\ \pi_t = \omega'_s} \mathrm{P}(\boldsymbol{\pi}_{1:T} | \boldsymbol{X}_{1:T}, \boldsymbol{\theta})$$

- Sum of all such products yields total probability:

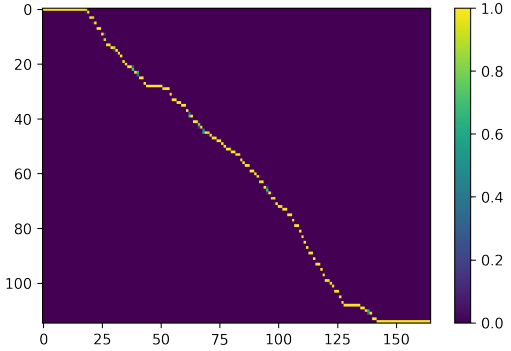
$$\sum_{s=1}^{2L+1} \alpha_t(s)\beta_t(s) = \mathrm{P}(\boldsymbol{\omega}_{1:L} | \boldsymbol{X}_{1:T}, \boldsymbol{\theta})$$

- We can also use $\alpha_t(s)\beta_t(s)$ as a measure of soft-alignment
  - Often normalize to align $= \frac{\alpha_t(s)\beta_t(s)}{\sum_{s=1}^{2L+1} \alpha_t(s)\beta_t(s)}$

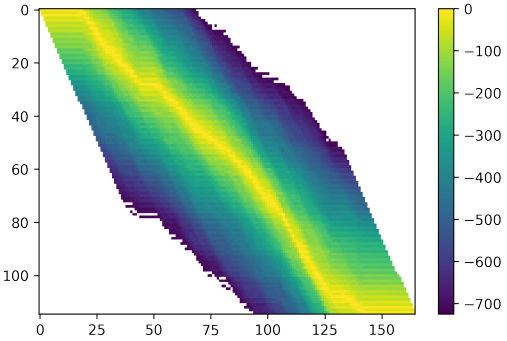
Training - CTC Forward-Backward Algorithm



CTC Soft-Alignment



(a) Soft-Alignment



(b) Log Soft-Alignment

CTC Forward-Backward Algorithm and Loss Computation

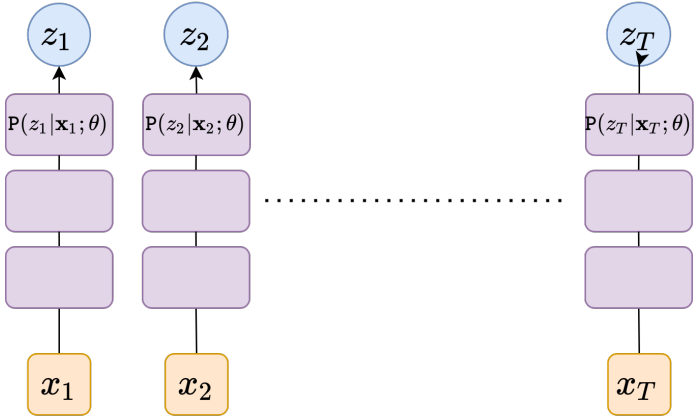
- Why use the Fw-Bw algorithm instead of just the Fw or Bw algorithms??

$$\mathrm{P}(\boldsymbol{\omega}_{1:L}|\boldsymbol{X}_{1:T},\boldsymbol{\theta}) = \underbrace{\sum_{s=1}^{2L+1}\alpha_t(s)\beta_t(s)}_{\text{Forward-Backward}} = \underbrace{\alpha_T(2L)+\alpha_T(2L+1)}_{\text{Forward}} = \underbrace{\beta_1(1)+\beta_1(2)}_{\text{Backward}}$$

- For gradient computation during training and alignment.

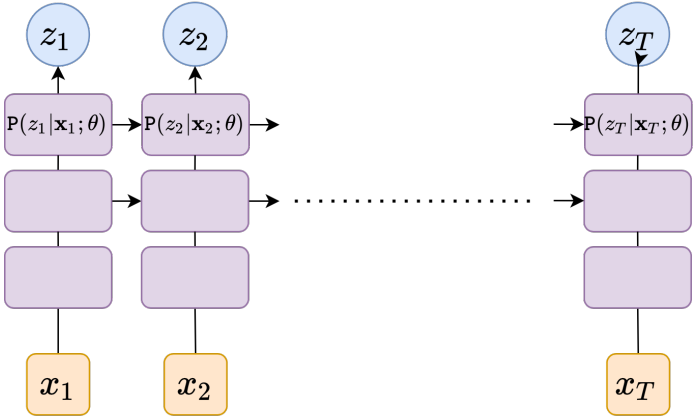
Math is cool (maybe?), but how do I build this?

Neural Architecture for CTC ASR



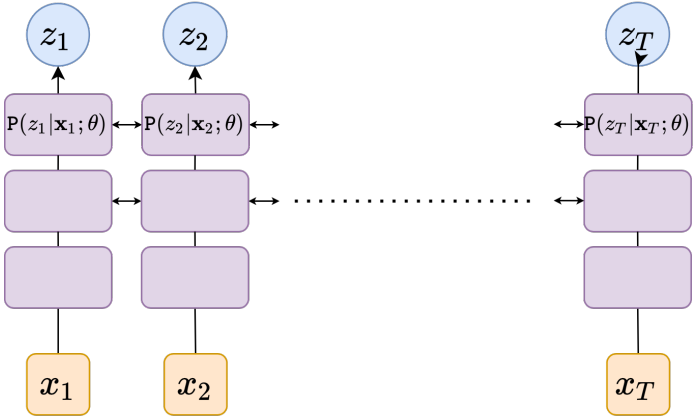
- We can use a Feed-Forward DNN, but not context modelling

Neural Architecture for CTC ASR



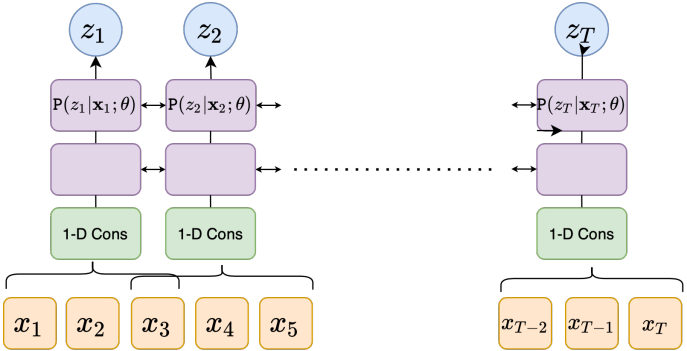
- Using Recurrent Units allows us to capture causal relations
  - Can use different units - RNN, GRU, LSTM, etc...

Neural Architecture for CTC ASR



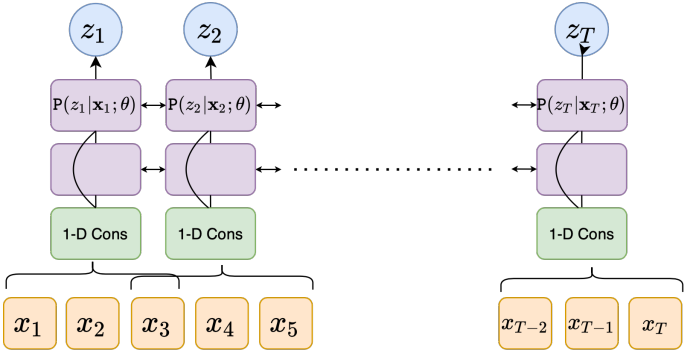
- Bidirectional Recurrent Units allows us to capture future context better

Neural Architecture for CTC ASR



- Using convolution filters low-lever representation learning
- Strided convolution allows down-sampling!

Neural Architecture for CTC ASR



- Adding residual layers improves convergence, adds regularization
  - Used in many large-scale architectures!

Neural Architecture for CTC ASR - Jasper (2019 SOTA)

- Deep Learning is like lego - can add many architectural elements
  - BatchNorm / LayerNorm
  - Skip-connection
  - Self-Attention Layers
  - Strided convolutions for pooling
  - Dropout and other stochastic layers
- Make sure more not too large for about of training data!

Neural Architecture for CTC ASR - Jasper (2019 SOTA)

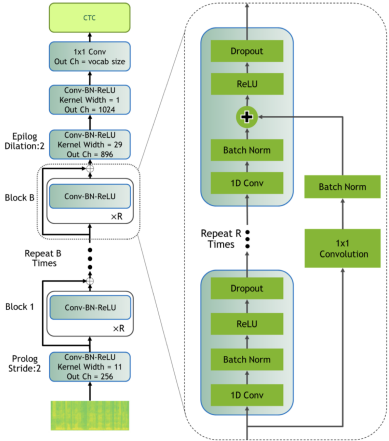


Figure 1: *Jasper BxR model: B - number of blocks, R - number of sub-blocks.*

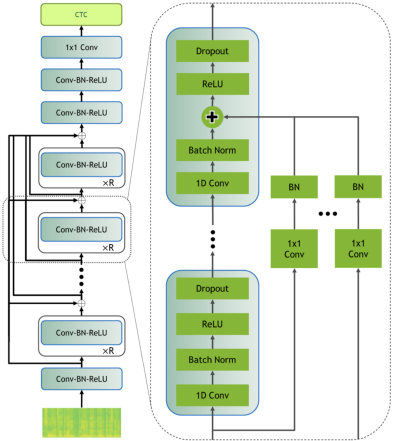


Figure 2: *Jasper Dense Residual*

So what have we learned?

Conclusion

- CTC is a discriminative state-space ASR system
  - Uses graphemic states with special blank symbol
  - 'Alignment-free' training
  - Fully factorizes - doesn't explicitly model context
- CTC Trellis has different structure from HMM trellis
  - Transfer between blanks and distinct non-blanks
- Can compute CTC loss using Forward-Backward Algorithm
- CTC and HMMs are closely related
  - CTC model is HMM with uniform prior and transition probs.
  - CTC training and inference criteria are matched
  - State duration and context modelling are poorer
  - CTC needs much more data
- Combining convolution, bidirectional recurrent and residual layers works well