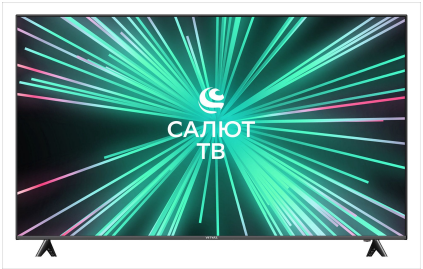
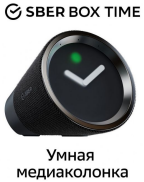
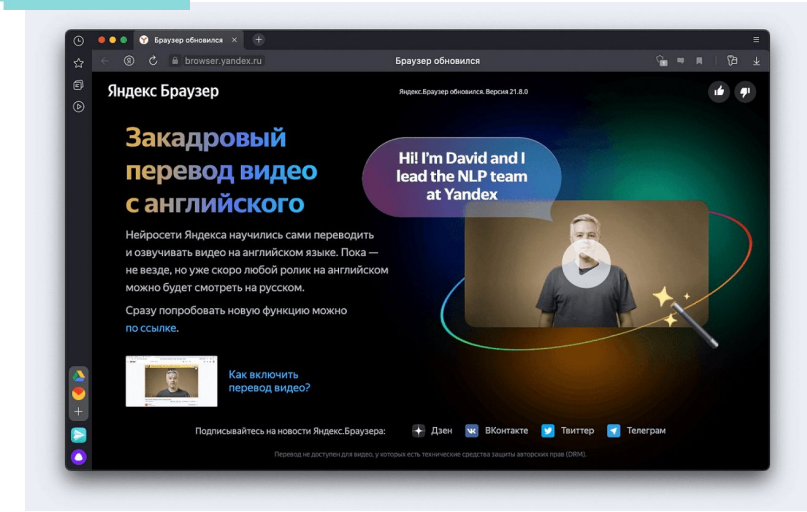
ASR

Automatic Speech Recognition Part 1

Convert speech to text







Speech recognition

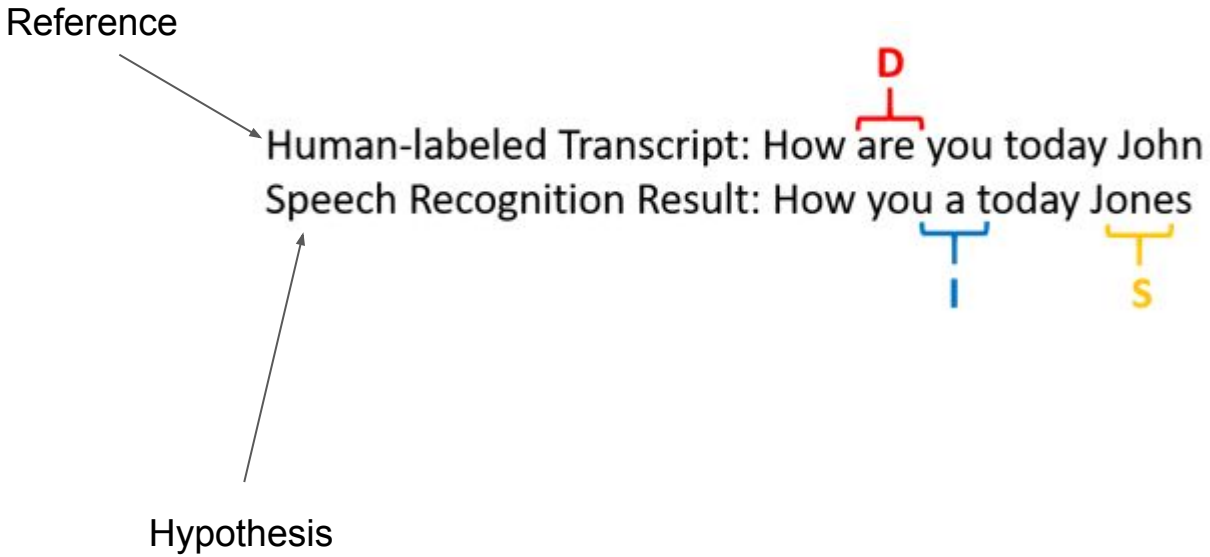
- CTC Loss
- E2E architectures
- Error Correction
- Rescoring

Metrics

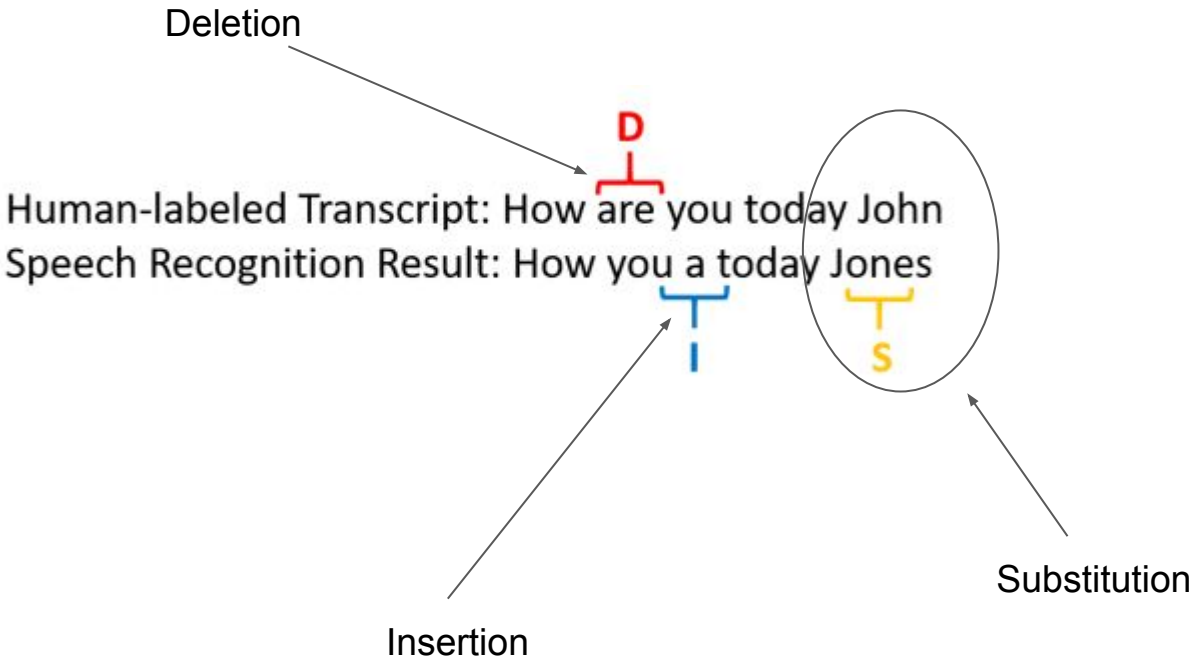
Most common ASR performance metric is Word Error Rate

$$\text{Word Error Rate} = \frac{\text{Insertions} + \text{Deletions} + \text{Substitutions}}{\text{Number of Words in Reference Transcript}}$$

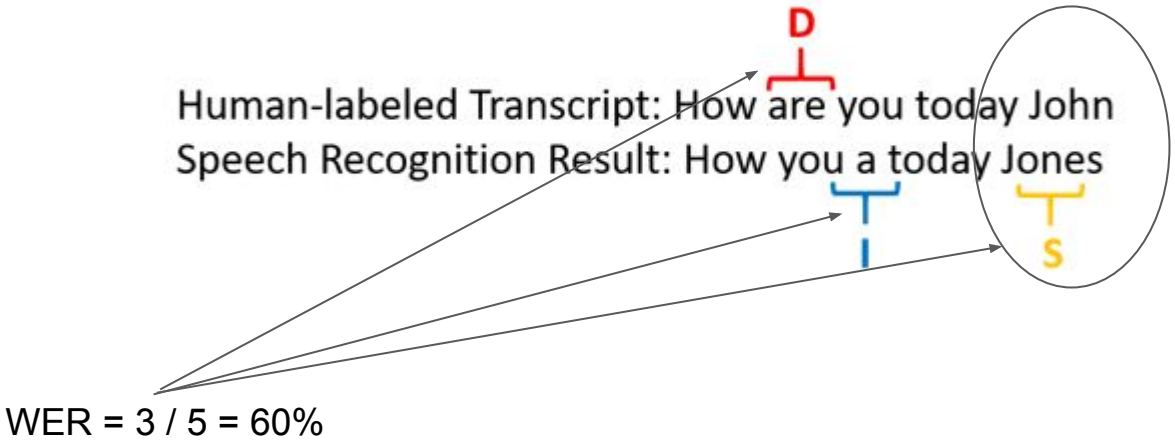
Word Error Rate



Word Error Rate



Word Error Rate



WER = 3 / 5 = 60%

Word Error Rate

Alignment

- Use Levenshtein distance to calculate difference between hypothesis and reference
- Save all transitions in Levenshtein distance algorithm to calculate number of substitutions, deletions, insertions

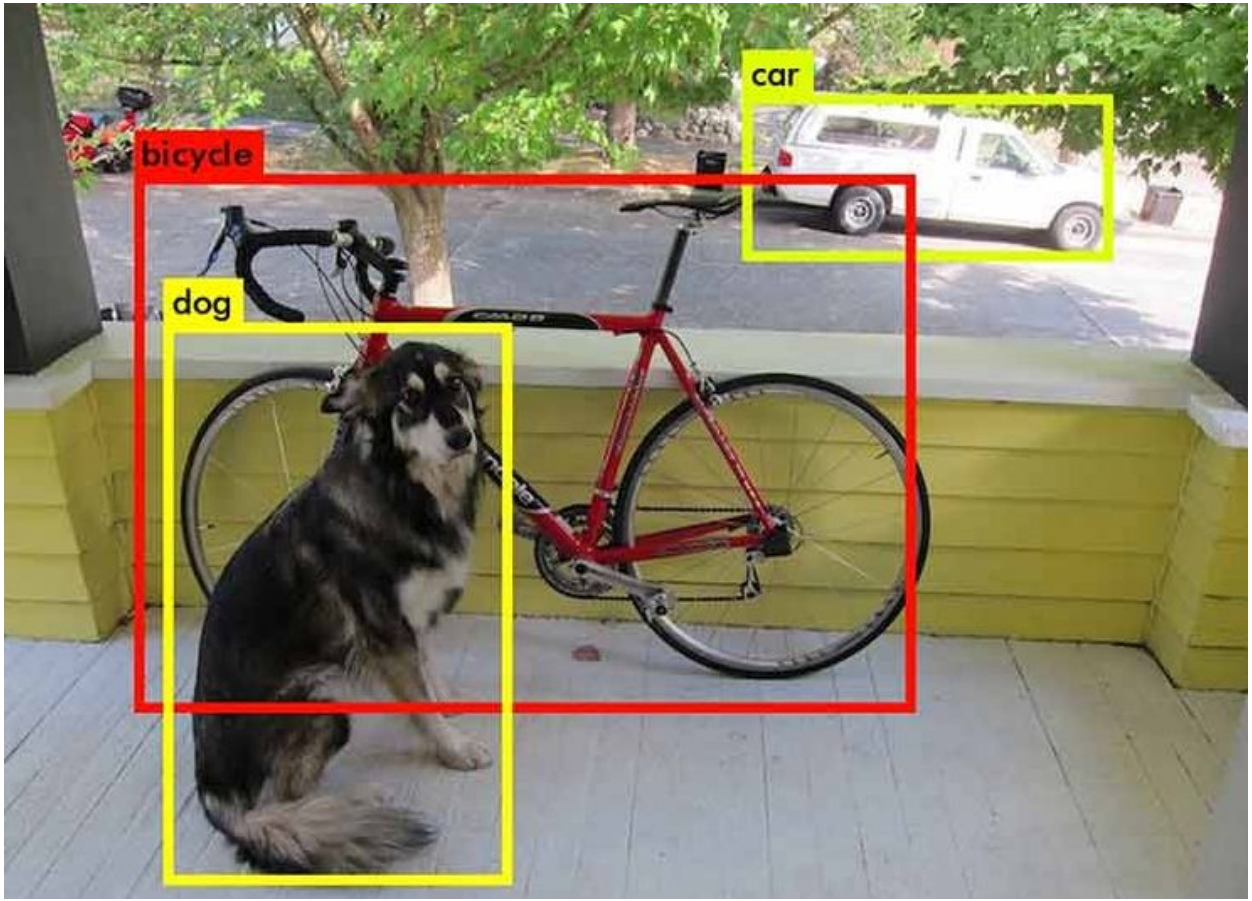
$$\mathrm{WER} = \frac{\mathrm{LevenshteinDistance(ref,\ hyp)}}{\mathrm{NumWords(ref)}}$$

- Dataset WER is sum of all mistakes divided by sum of all reference word counts
- Or sometimes it's just mean Word Error Rate of all files

$$\mathrm{WER}_{\mathrm{dataset}} = \frac{\sum_i^n (S_i + D_i + I_i)}{\sum_i^n \mathrm{NumWords(ref)}}$$

$$\mathrm{WER}_{\mathrm{dataset}} = \frac{1}{n} \sum_i^n WER_i$$

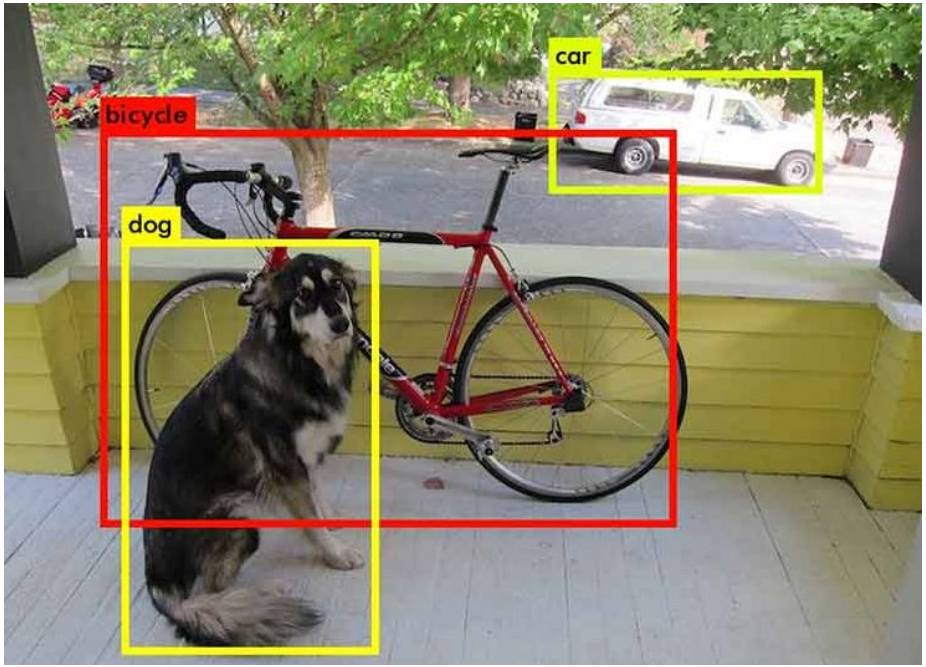
Solving ASR



Solving ASR

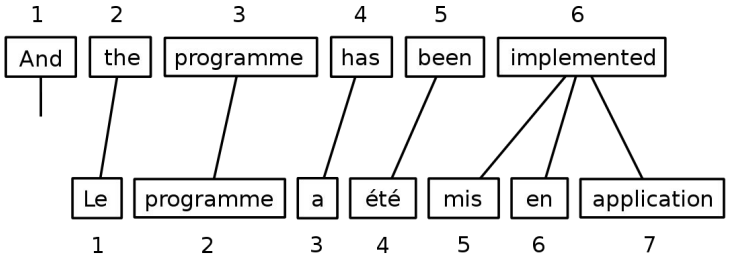
- Object detection
- Neural Network takes image outputs bounding boxes **independently**

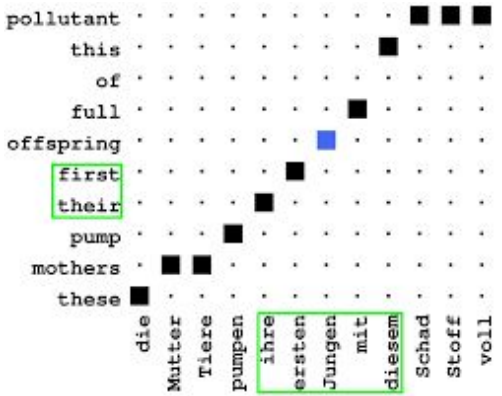
- Let’s take it to ASR task



Alignment

- Nearly all sequence-to-sequence (Seq2Seq) tasks have alignment problem
- For machine translation the objective of the word alignment task is to discover the word-to-word correspondences in a sentence pair

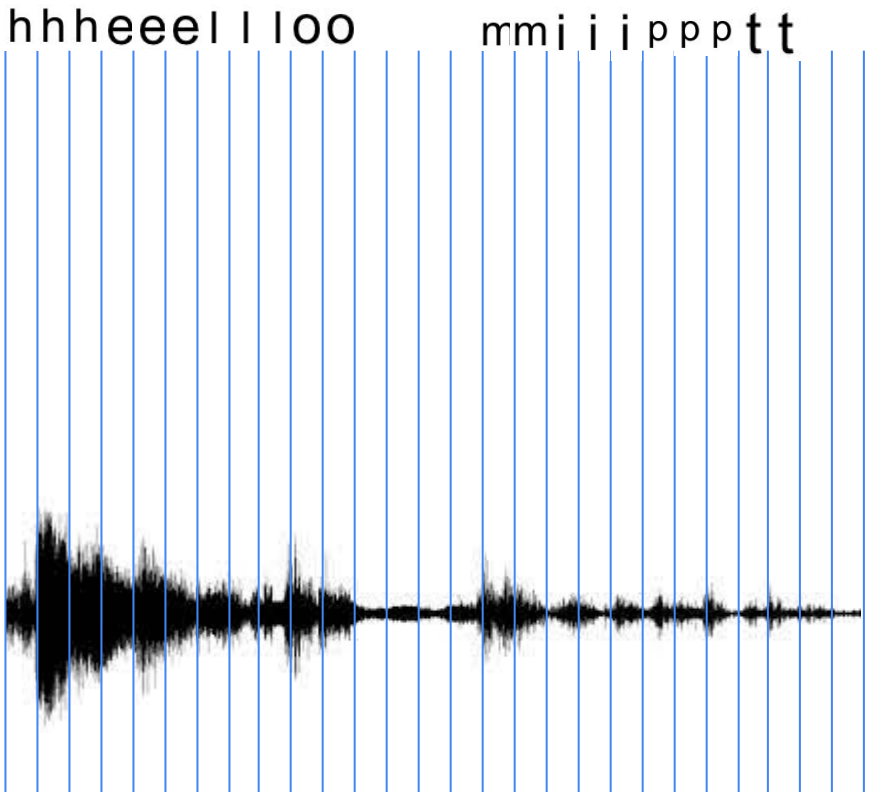




Alignment

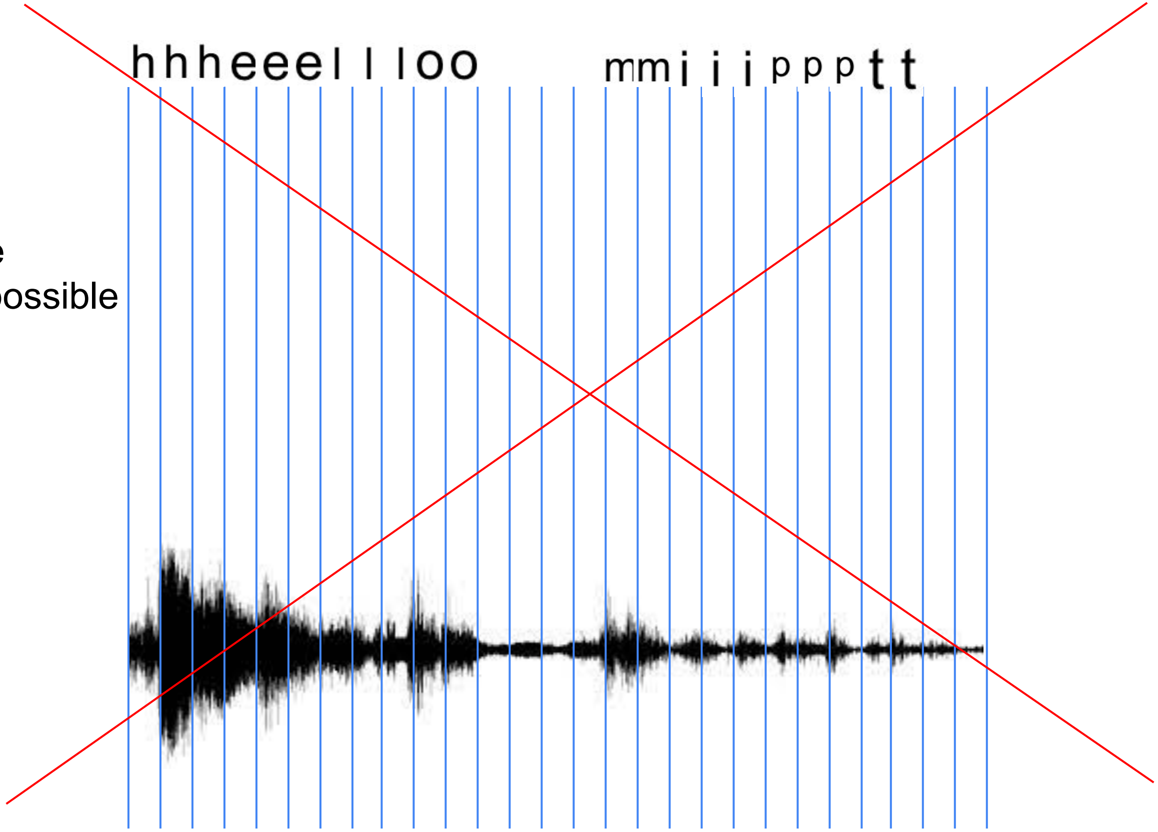
- Speech recognition alignment is much harder problem due to continuous input
- Translation problem is word-to-word task. Both source (SRC) and destination (DST) are discrete input values
- Speech recognition input is continuous

Segmentation

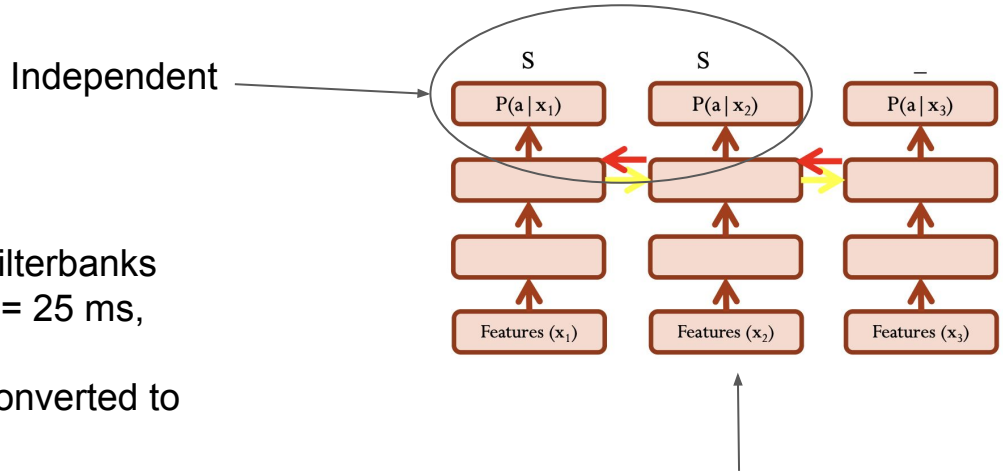


Segmentation

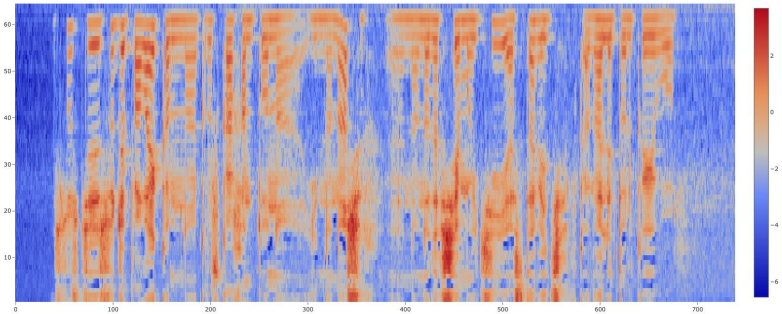
- Very expensive
- sometimes impossible



ASR

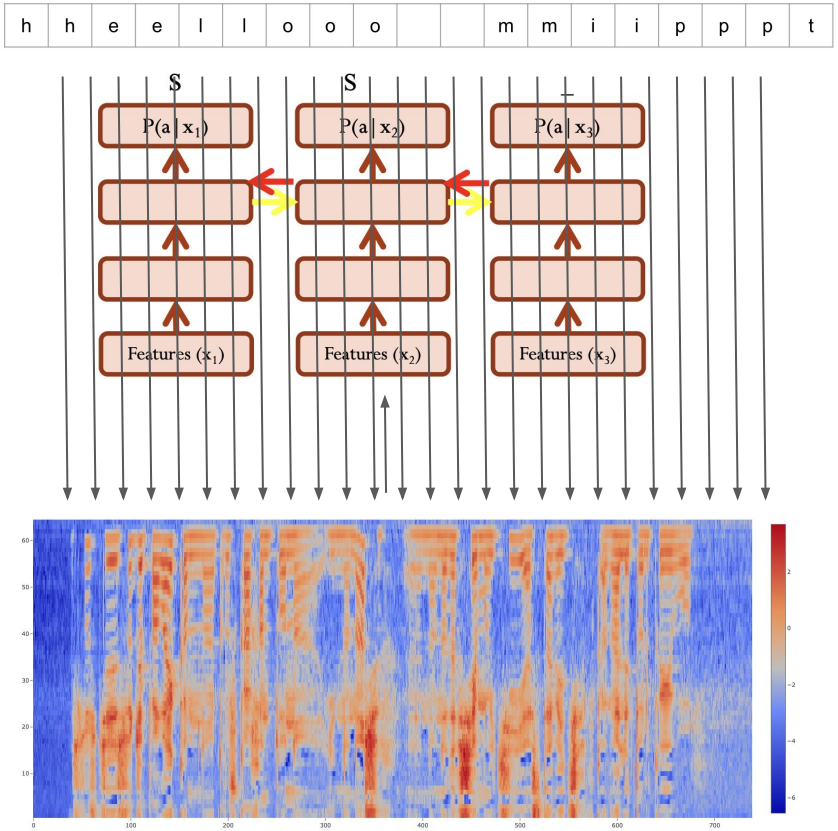


- Convert waveform to LogMel Filterbanks
- Lets use hop = 10 ms, window = 25 ms, n_mels = 64
- 10 seconds waveform will be converted to matrix [1000, 64]
- Construct neural network that somehow predicts P(A | X)
- P(a_{i} | x) does not depend on P(a_{j} | X) – **outputs are independent**



ASR

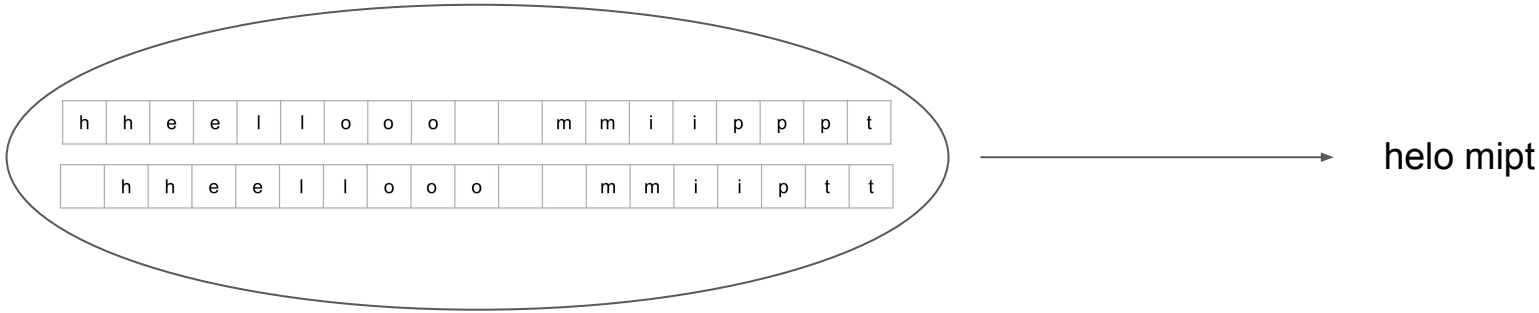
- P(a_{i} | x) does not depend on P(a_{j} | X)
  – **outputs are independent**
- Let’s allow repetition of characters



Connectionist Temporal Classification

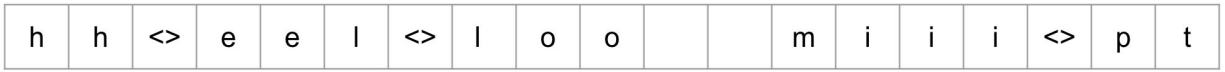
Apply rule:

- reduce multiple characters



Connectionist Temporal Classification

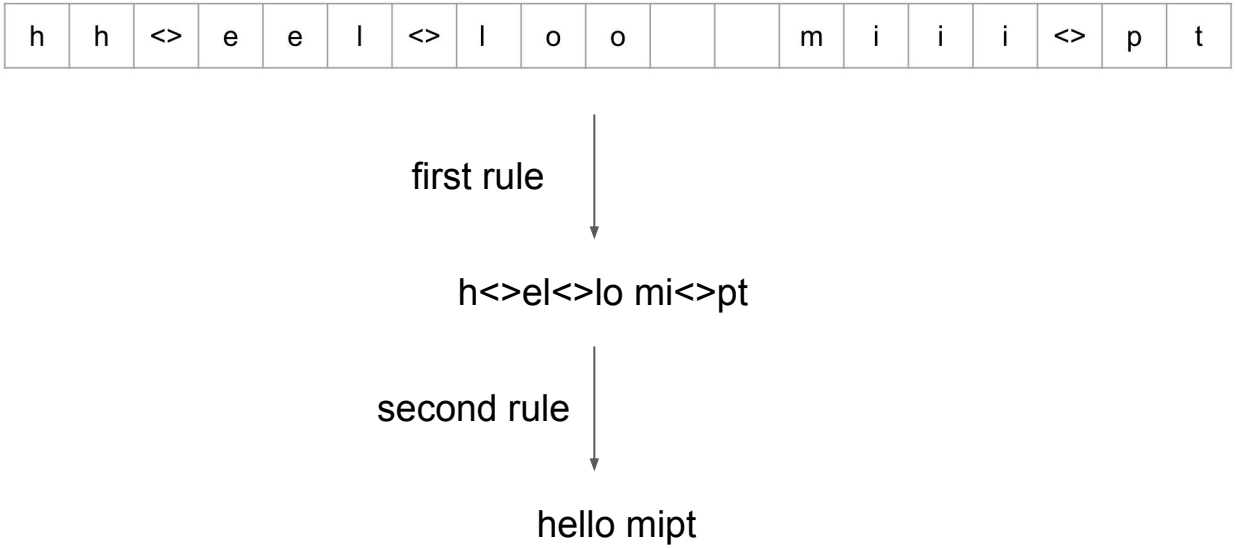
- Introduce blank character
- Means nothing
- Helps with repetition
- Add second rule: delete blanks



Connectionist Temporal Classification

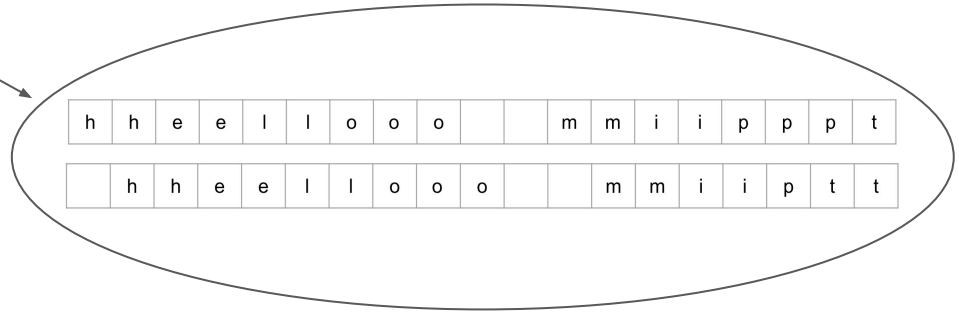
Apply 2 rules:

- reduce multiple characters
- delete blanks

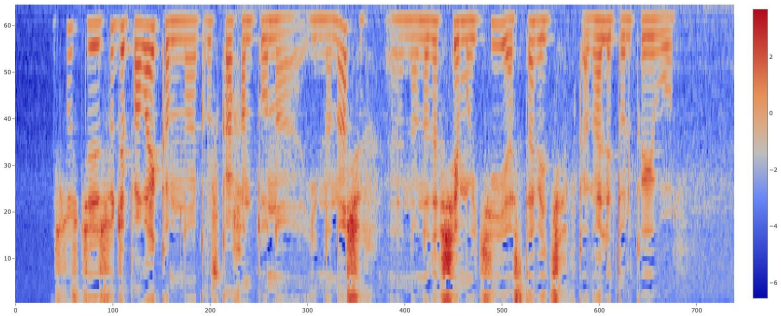


Connectionist Temporal Classification

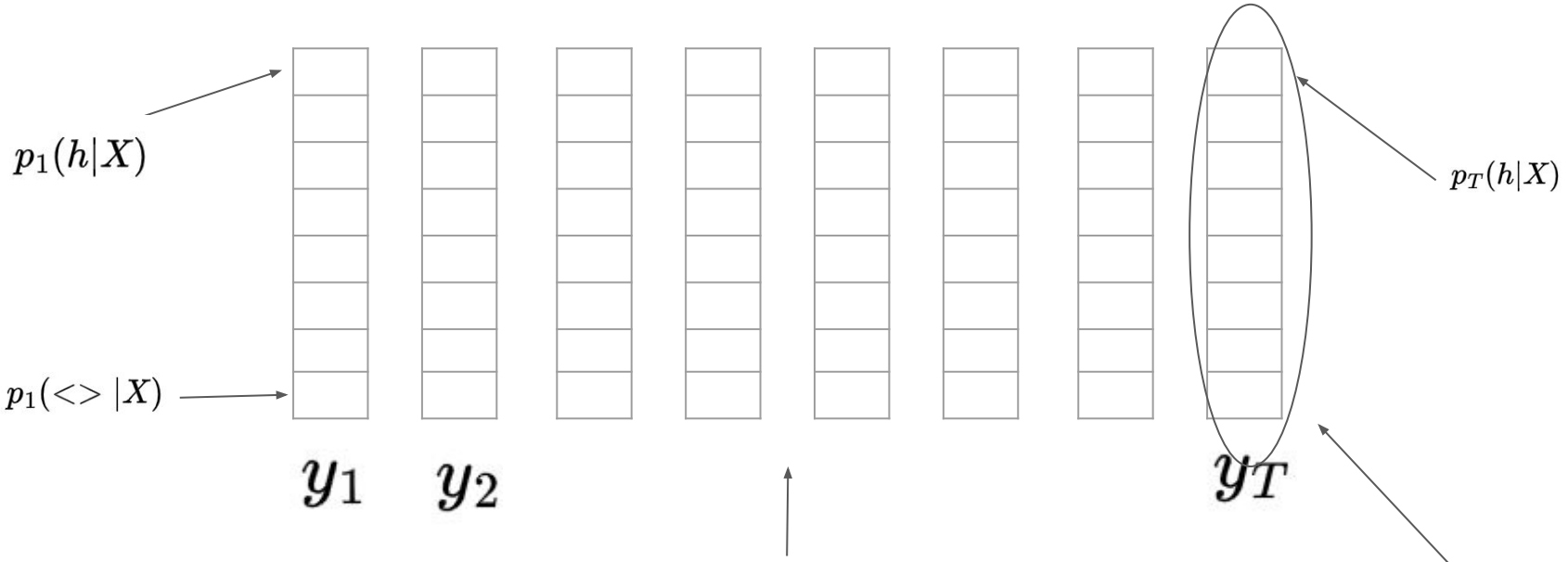
identical

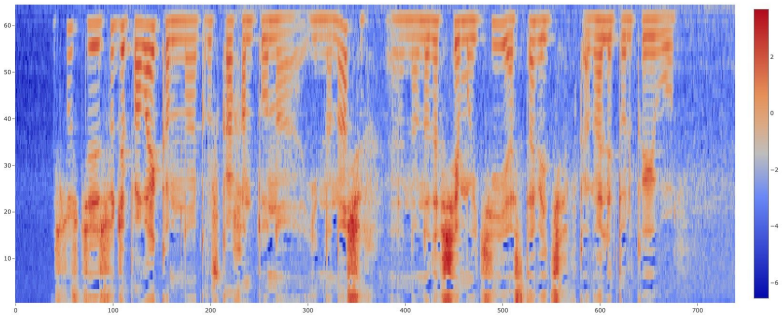


- Have to solve **alignment problem**
- Some alignments are identical



Connectionist Temporal Classification





Logits - outputs after log softmax

Connectionist Temporal Classification

Path1: “hhhel<>lo miiipt” → hello mipt

$$P(\text{”hhhel<>lo miiipt”}) = y_h^1 y_h^2 y_h^3 y_e^4 y_{<>}^5 y_l^6 y_o^7 y^8 y_m^9 y_i^{10} y_i^{11} y_i^{12} y_p^{13} y_t^{14}$$

Path2: “hhelll<>lo mipt” → hello mipt

$$P(\text{”hhelll<>lo mipt”}) = y_h^1 y_h^2 y_e^3 y_l^4 y_l^5 y_l^6 y_{<>}^7 y_l^8 y_o^9 y^{10} y_m^{11} y_i^{12} y_p^{13} y_t^{14}$$

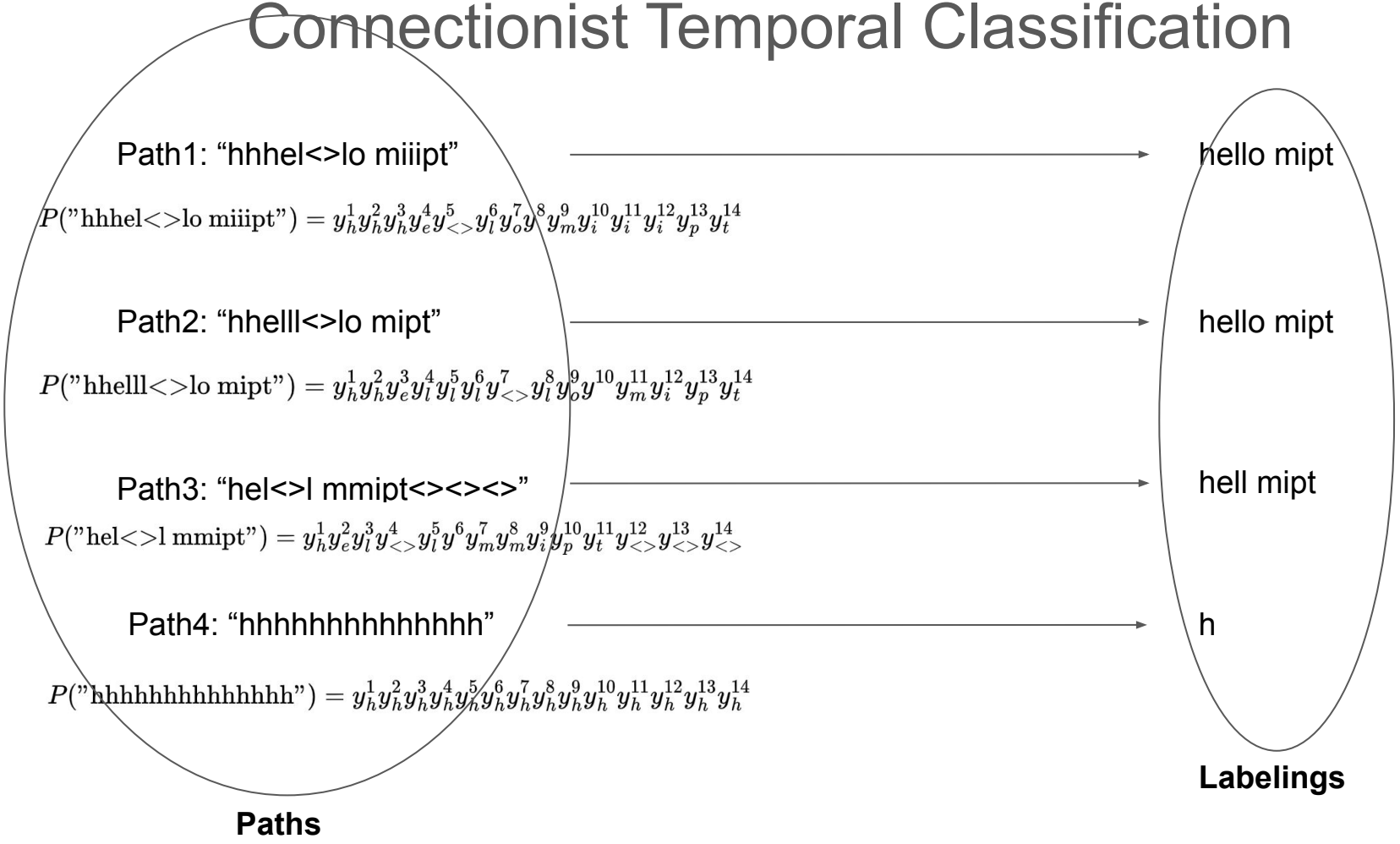
Path3: “hel<>l mmipt<><><>” → hell mipt

$$P(\text{”hel<>l mmipt”}) = y_h^1 y_e^2 y_l^3 y_{<>}^4 y_l^5 y^6 y_m^7 y_m^8 y_i^9 y_p^{10} y_t^{11} y_{<>}^{12} y_{<>}^{13} y_{<>}^{14}$$

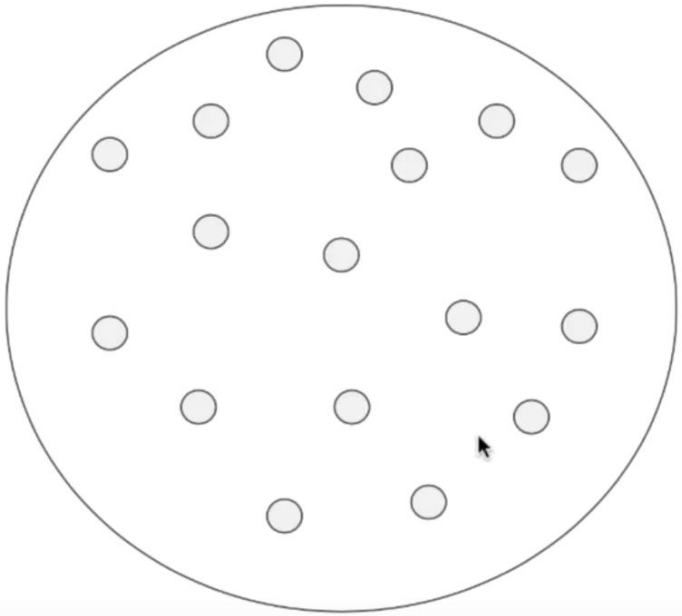
Path4: “hhhhhhhhhhhhhh” → h

$$P(\text{”hhhhhhhhhhhhhh”}) = y_h^1 y_h^2 y_h^3 y_h^4 y_h^5 y_h^6 y_h^7 y_h^8 y_h^9 y_h^{10} y_h^{11} y_h^{12} y_h^{13} y_h^{14}$$

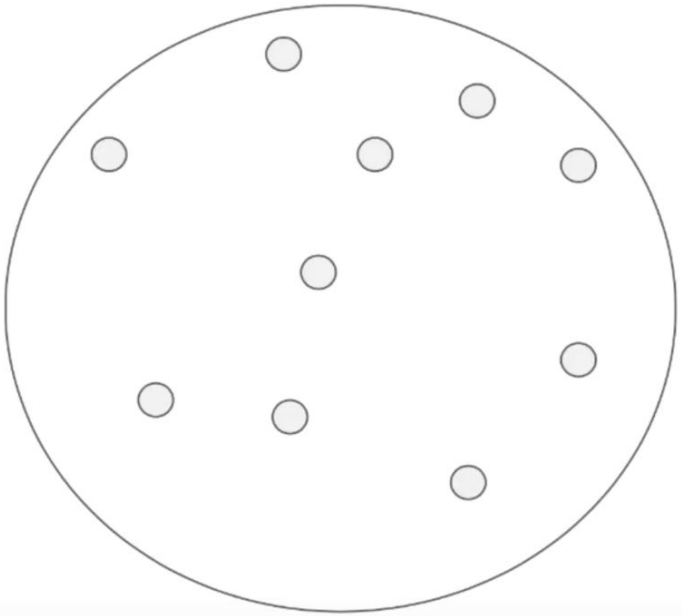
Connectionist Temporal Classification



Connectionist Temporal Classification

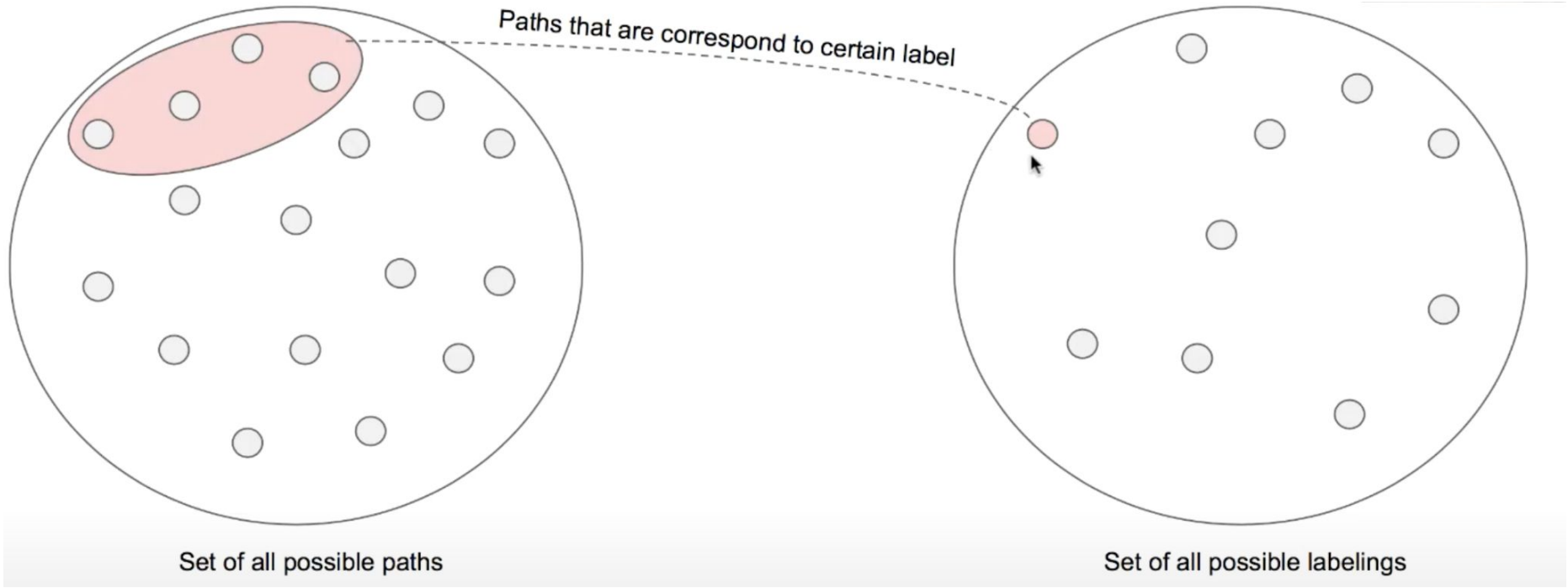


Set of all possible paths

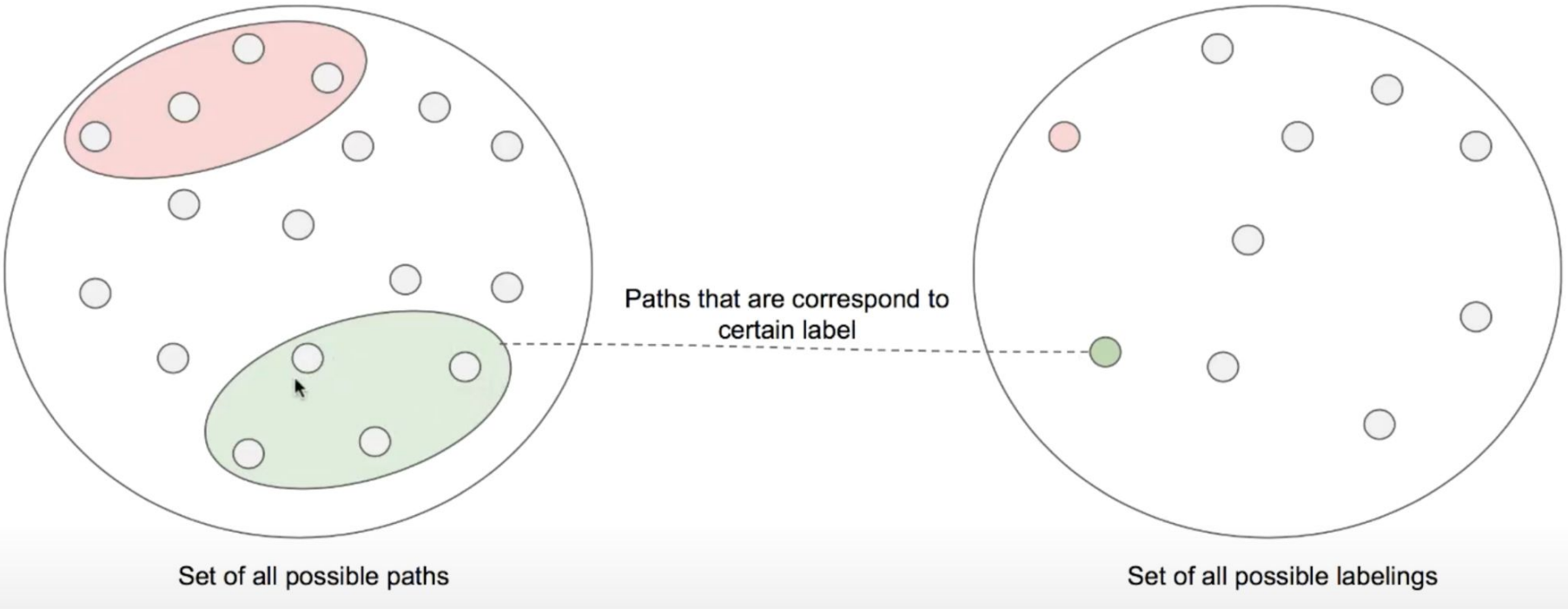


Set of all possible labelings

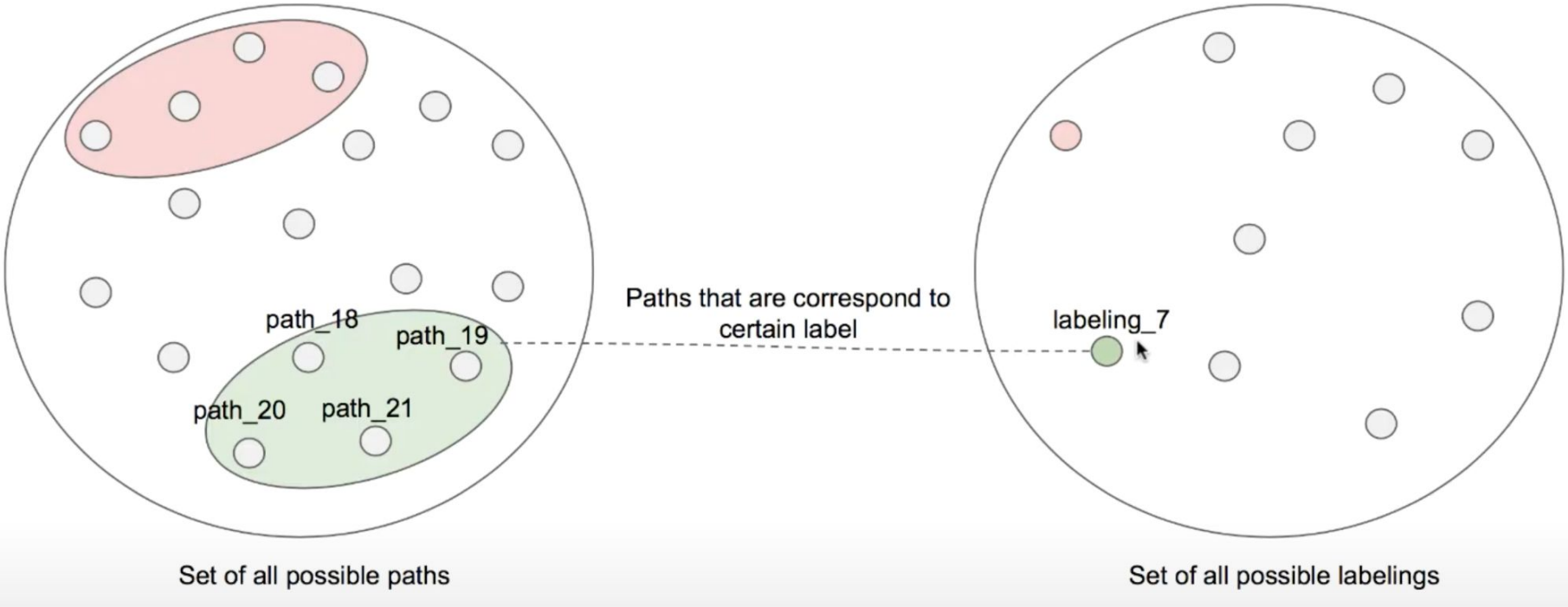
Connectionist Temporal Classification



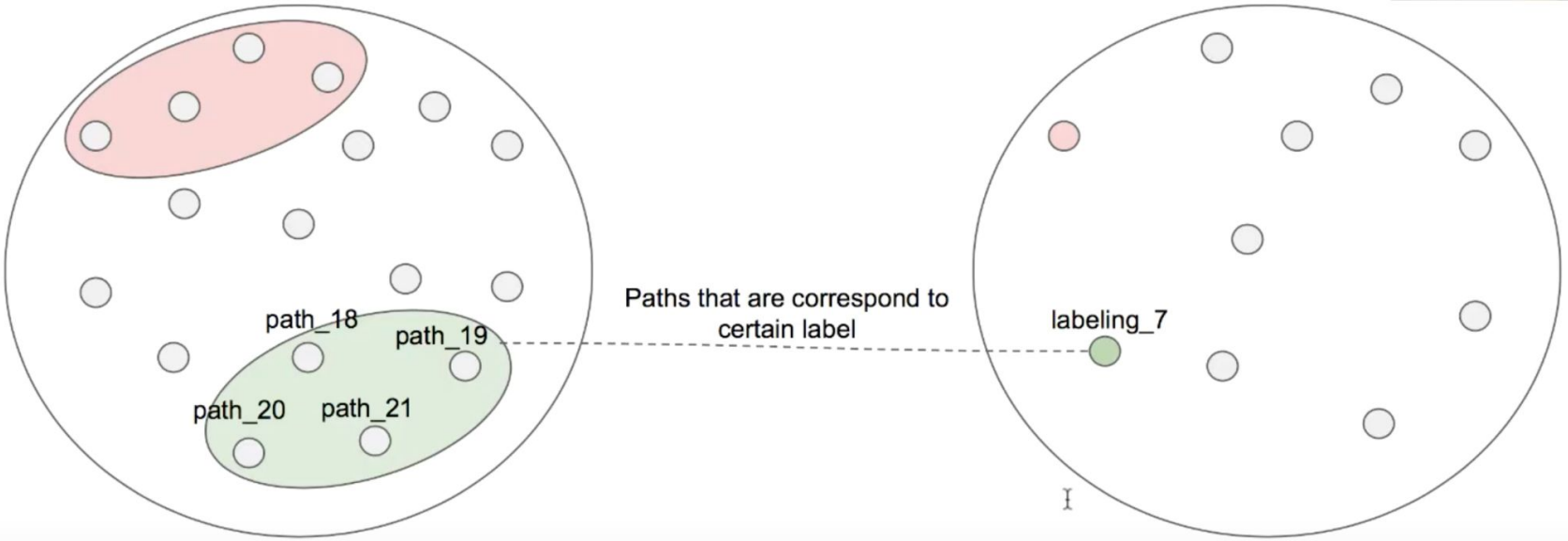
Connectionist Temporal Classification



Connectionist Temporal Classification



Connectionist Temporal Classification

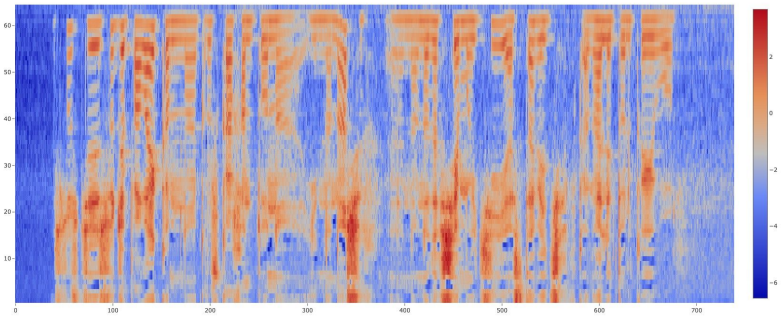


p(labeling_7) = sum of probabilities of all corresponding paths =
= p(path_18) + p(path_19) + p(path_20) + p(path_21)

Connectionist Temporal Classification

- Maximize probability just like in cross entropy

$$\mathrm{L}_{\mathrm{CTC}} = -\log P(\text{"hello mipt"})$$



Connectionist Temporal Classification

$$\mathrm{L}_{\mathrm{CTC}}(Y, R) = \sum_{C:k(C)=R} P(C|Y) = \sum_{C:k(C)=R} \prod_{t=1}^{T} p(c_t|Y)$$

Logits

Reference labeling

paths that lead to R

use conditional independence

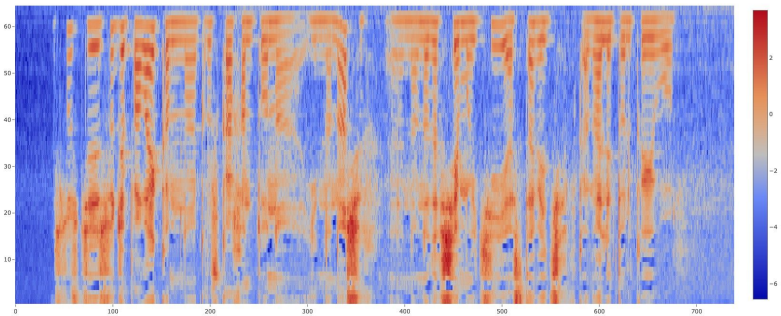
In practice we use log

$$\mathrm{L}_{\mathrm{CTC}}(Y, R) = -\ln P(\text{Paths in Y that lead to R})$$

Connectionist Temporal Classification

- Maximize probability just like in cross entropy
- Suppose we have 1000 frames and 28 letters (english alphabet & space between words & blank)
- Number of possible alignments?

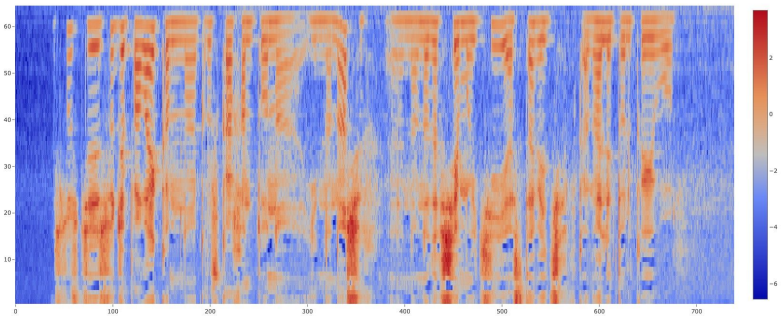
$$\mathrm{CTC}_{loss} = -\log P(\text{"hello mipt"})$$



Connectionist Temporal Classification

- Maximize probability just like in cross entropy
- Suppose we have 1000 frames and 28 letters (english alphabet & space between words & blank)
- Number of possible alignments is 28^1000
- We cannot search them all
- Fortunately, there is an efficient way to calculate probability and **gradients**
  How to calculate CTC loss efficiently.

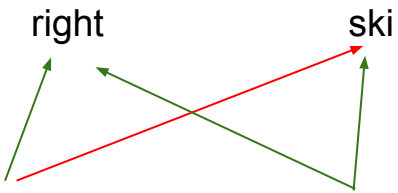
$$\mathrm{CTC}_{loss} = -\log P(\text{"hello mipt"})$$



Connectionist Temporal Classification

- CTC Loss is a sequence loss
- CTC is **alignment free**
- Outputs are **conditionally independent**
- Highly parallel compute

Units

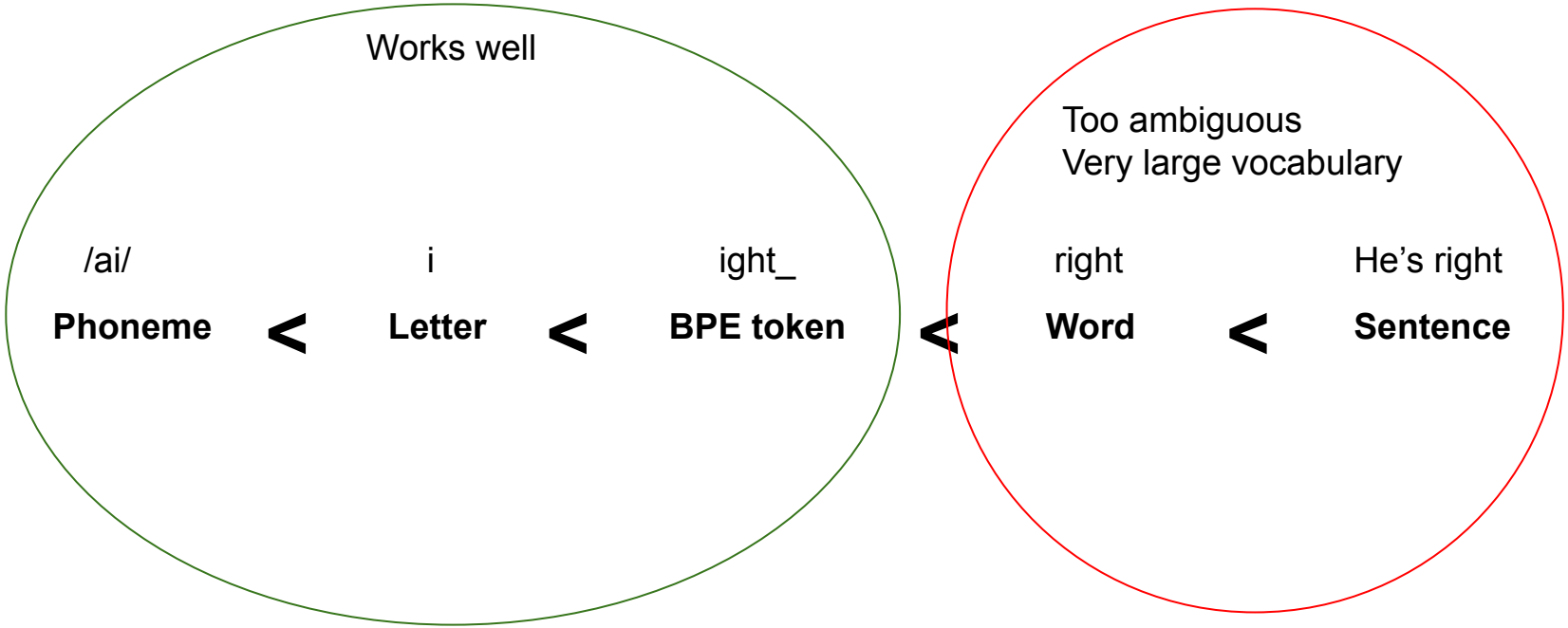


/ai/ i ight_ right He’s right

Phoneme < **Letter** < **BPE token** < **Word** < **Sentence**

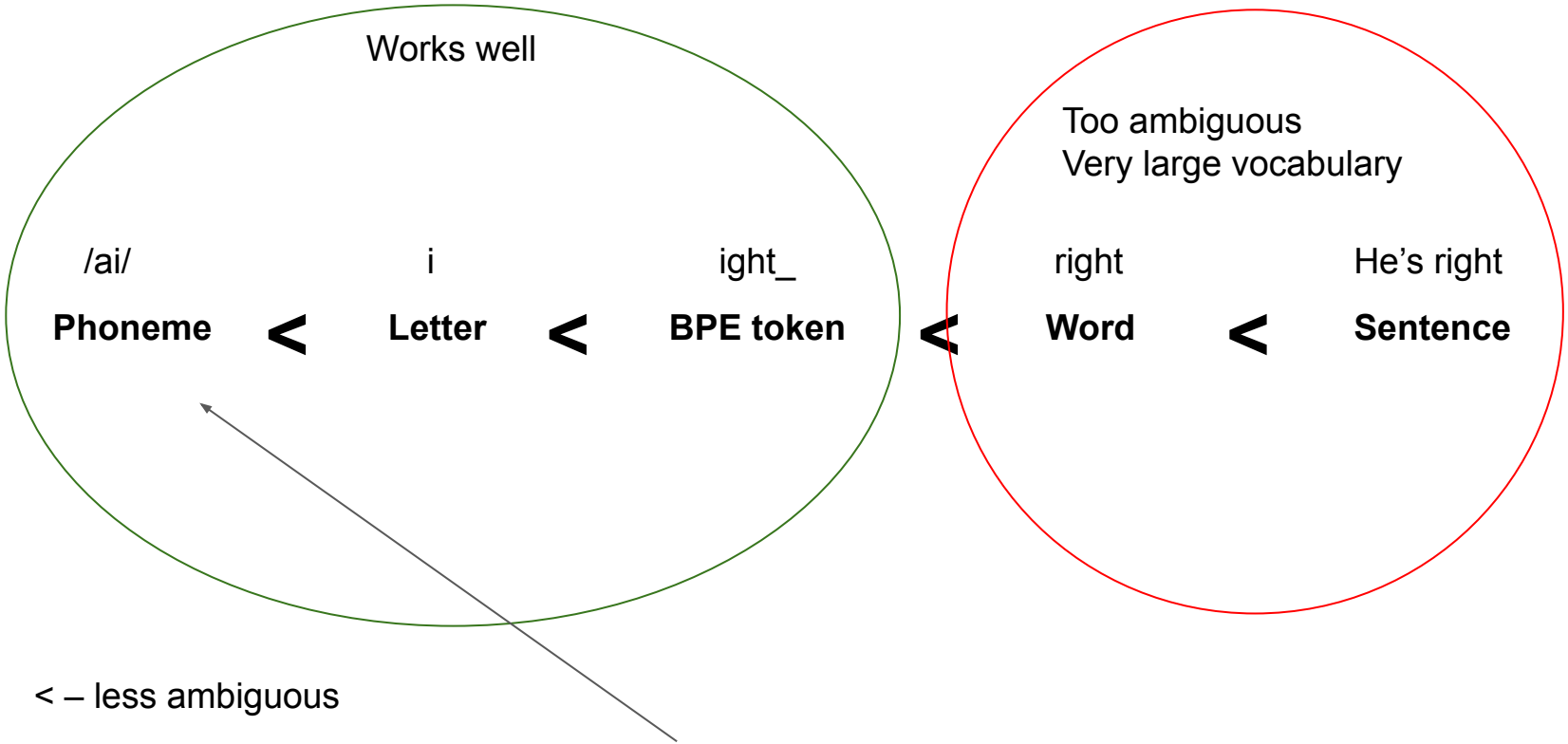
< – less ambiguous

Units



< – less ambiguous

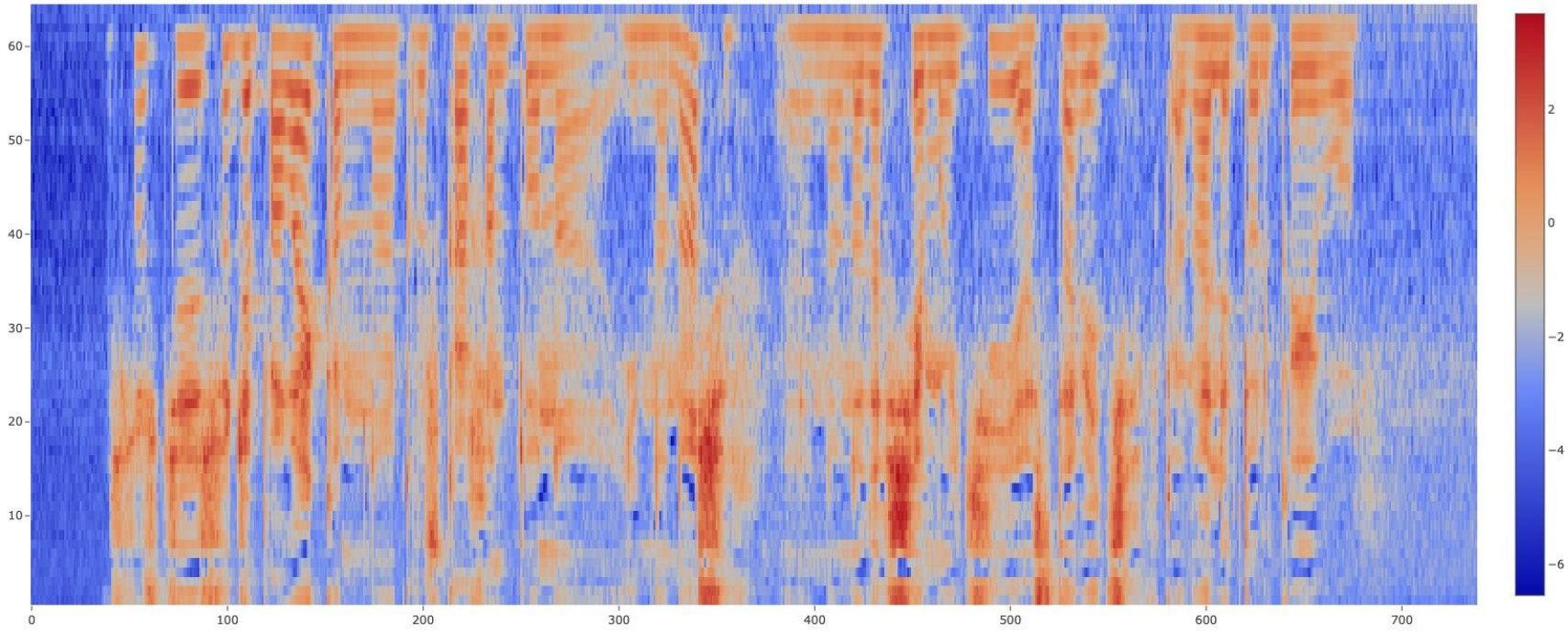
Units



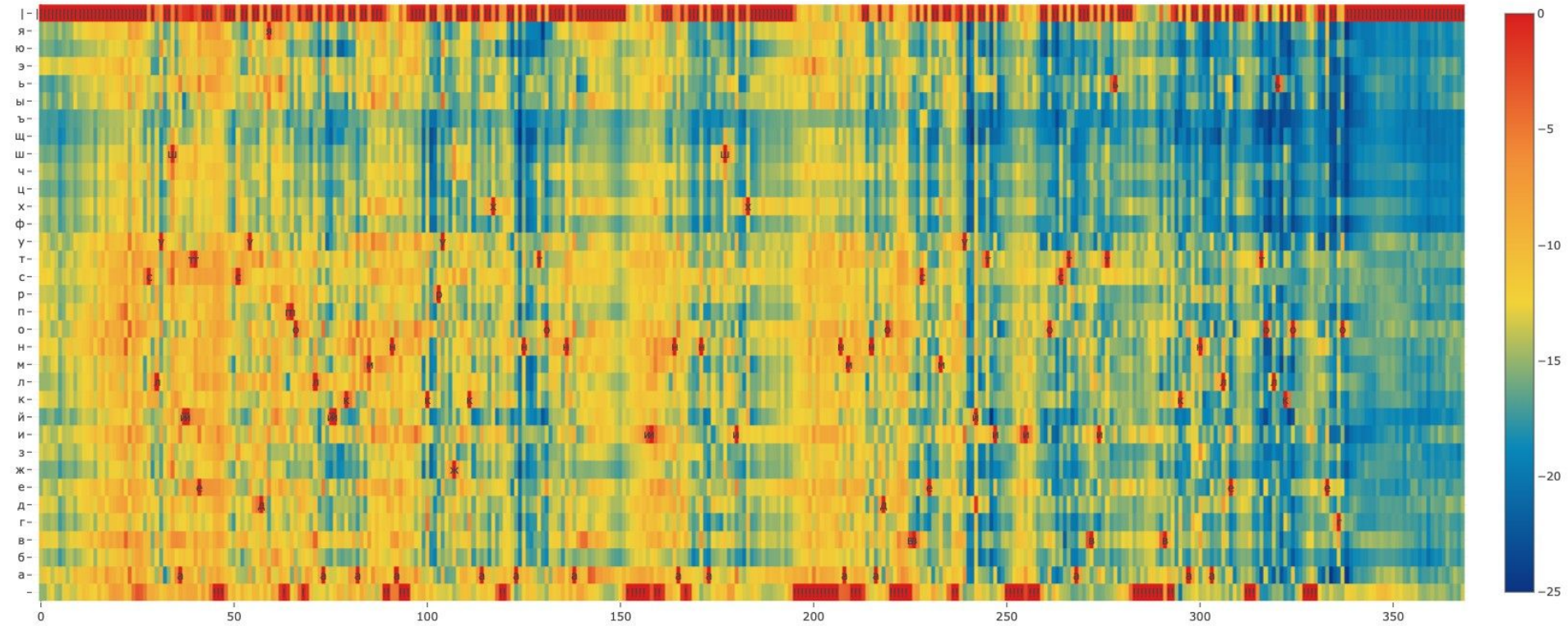
< – less ambiguous

Need an additional word to phoneme model

Connectionist Temporal Classification



letters:слушайте судя по лайкам на кружках антона и на наших нам надо всем уйти и оставить в канале только его

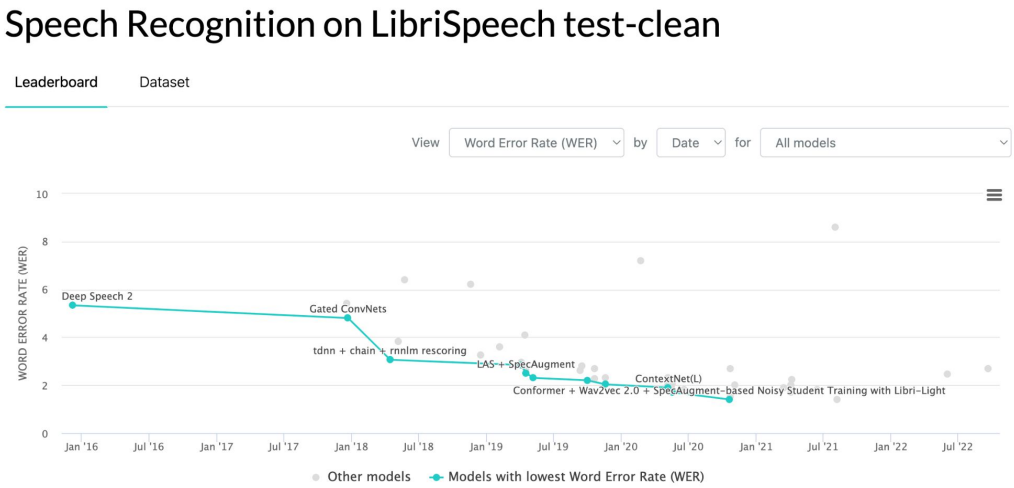


Architectures

- We now have sequence loss that is easy to compute and calculate gradients of
- Let’s use it to solve ASR

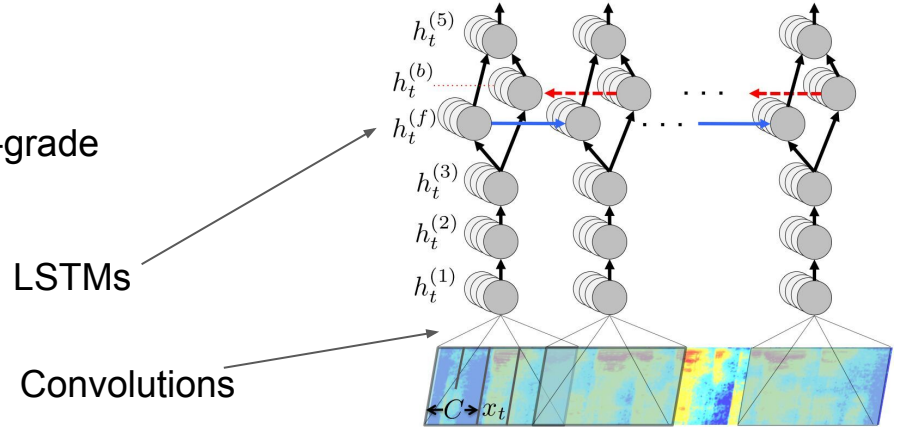
Architectures

- Librispeech is opensource dataset with audio books
- Paperswithcode is nice leaderboard for ASR models on Librispeech dataset



DeepSpeech 2

- Baidu introduced DeepSpeech 2 at 2015
- DeepSpeech 2 was one of the first production-grade CTC-ASR systems
- [DeepSpeech 2 Paper]
- Paper has a lot of cool technical (hence a bit outdated) features
- 44 place on Librispeech-test

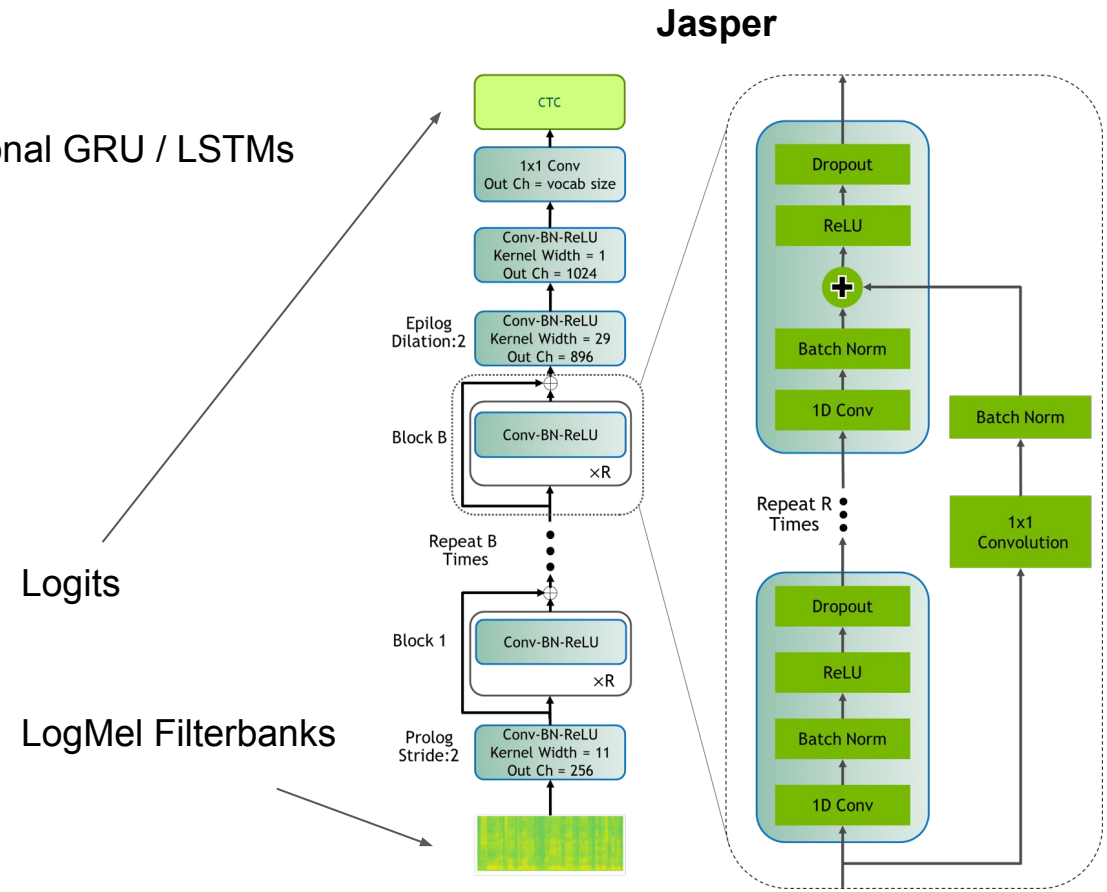


43	**Gated ConvNets**	4.8	×	Letter-Based Speech Recognition with Gated ConvNets		2017
44	**Deep Speech 2**	5.33	✓	Deep Speech 2: End-to-End Speech Recognition in English and Mandarin		2015
45	**CTC + policy learning**	5.42	×	Improving End-to-End Speech Recognition with Policy Learning		2017

Jasper

- Deepspeech 2 has a lot of bidirectional GRU / LSTMs
- Harder to train than convolutions

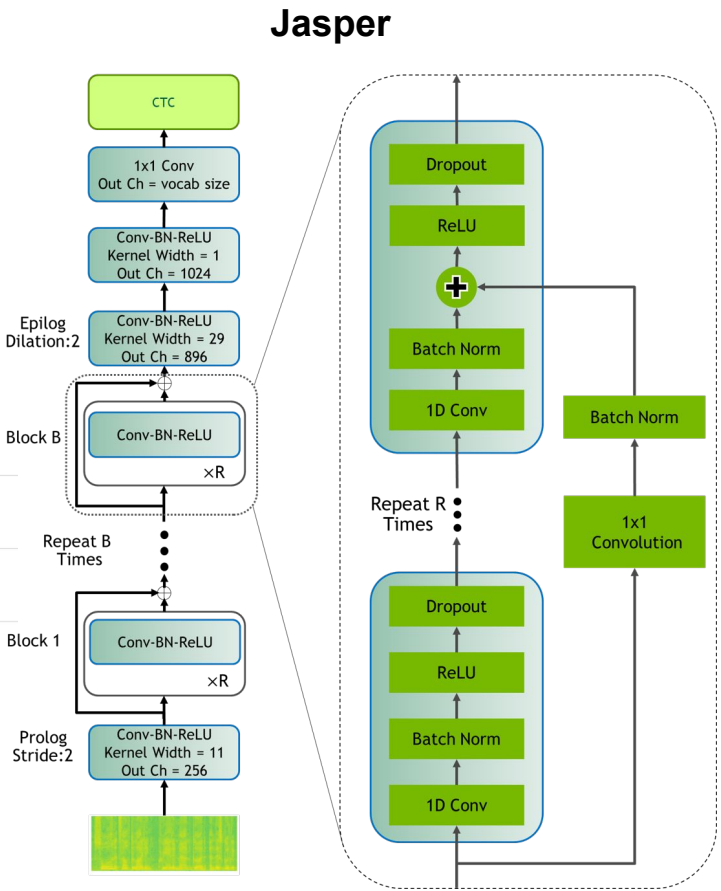
- Use 1D Convs with Batchnorm
- [Link to Jasper Paper]



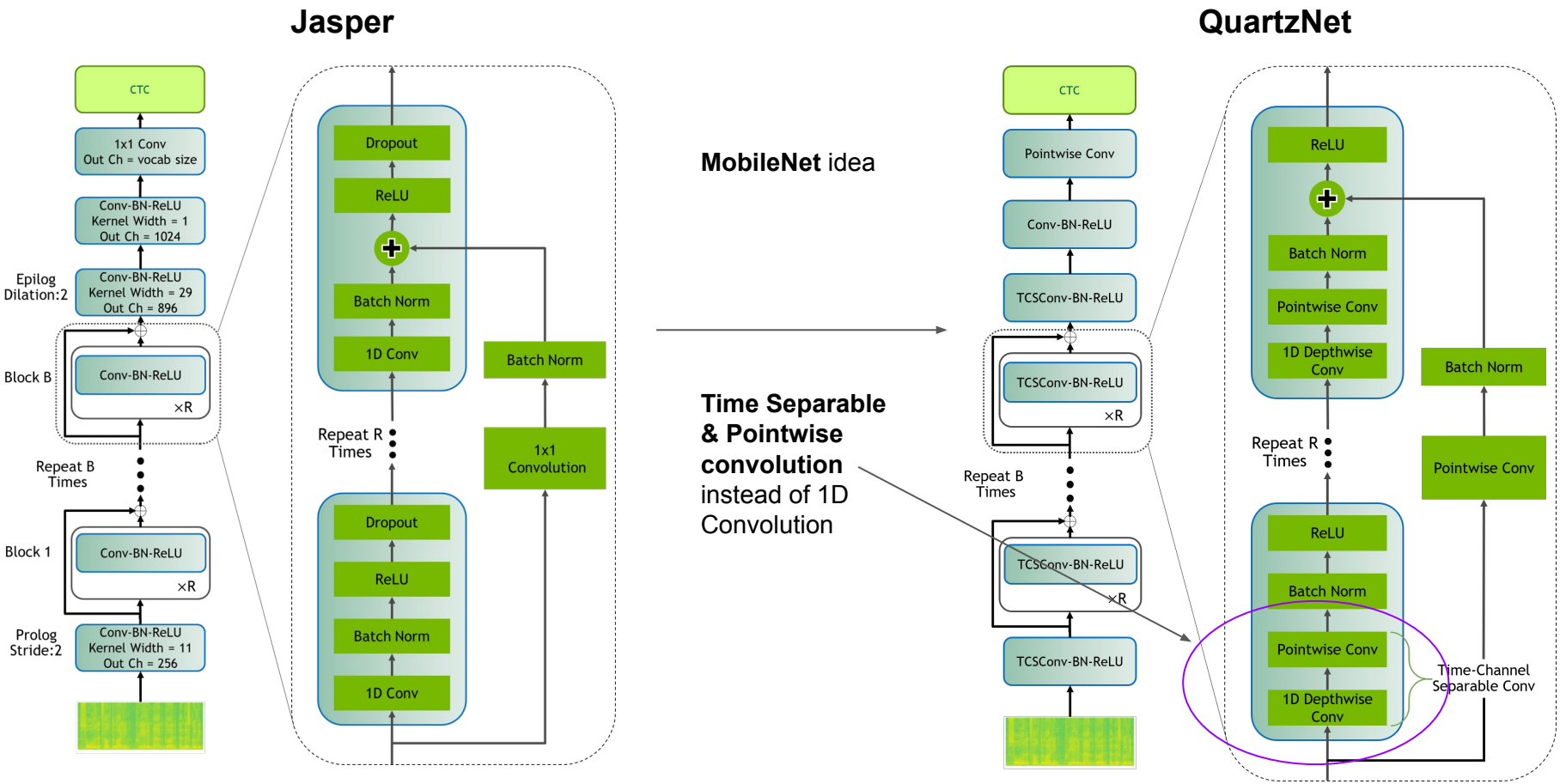
Jasper

- Link to Jasper Paper
- 10 positions higher than DeepSpeech 2
- Much faster
- Convolutional networks are truly parallel
- Due to conditional independence
  we can calculate whole output in one network call

34	**Jasper DR 10x5** (+ Time/Freq Masks)	2.84	×	Jasper: An End-to-End Convolutional Neural Acoustic Model		2019
35	**Jasper DR 10x5**	2.95	×	Jasper: An End-to-End Convolutional Neural Acoustic Model		2019



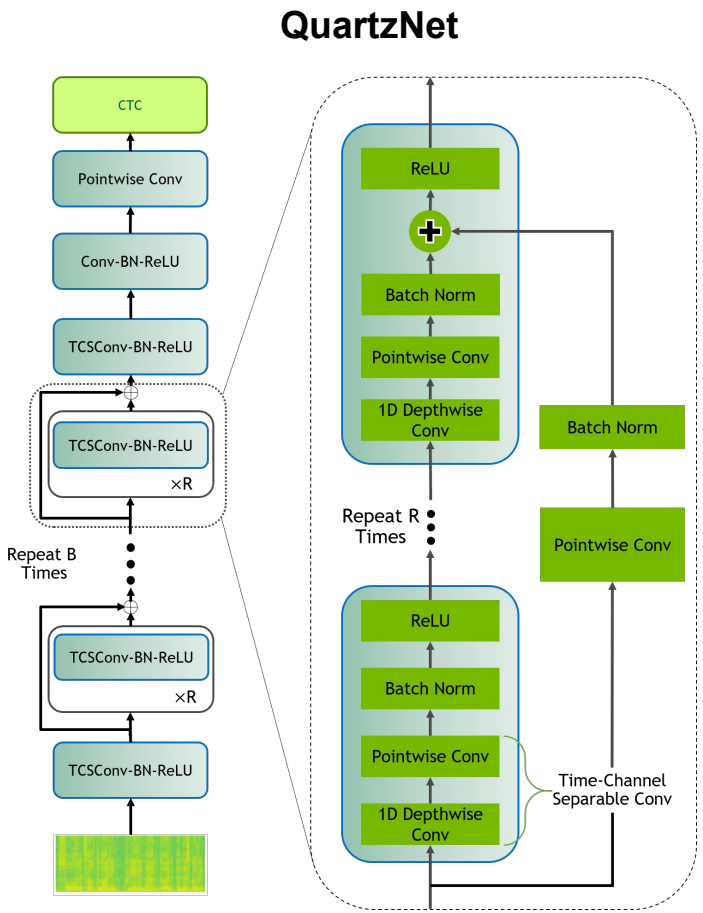
QuartzNet


Jasper
CTC
1x1 Conv
Out Ch = vocab size
Conv-BN-ReLU
Kernel Width = 1
Out Ch = 1024
Epilog
Dilation:2
Conv-BN-ReLU
Kernel Width = 29
Out Ch = 896
Block B
Conv-BN-ReLU
×R
Repeat B
Times
Block 1
Conv-BN-ReLU
×R
Prolog
Stride:2
Conv-BN-ReLU
Kernel Width = 11
Out Ch = 256
Dropout
ReLU
Batch Norm
1D Conv
Repeat R
Times
Dropout
ReLU
Batch Norm
1D Conv
Batch Norm
1x1
Convolution
MobileNet idea
Time Separable
& Pointwise
convolution
instead of 1D
Convolution
QuartzNet
CTC
Pointwise Conv
Conv-BN-ReLU
TCSConv-BN-ReLU
TCSConv-BN-ReLU
×R
Repeat B
Times
TCSConv-BN-ReLU
×R
TCSConv-BN-ReLU
ReLU
Batch Norm
Pointwise Conv
1D Depthwise
Conv
Repeat R
Times
ReLU
Batch Norm
Pointwise Conv
1D Depthwise
Conv
Batch Norm
Pointwise Conv
Time-Channel
Separable Conv

QuartzNet

- [Link to QuartzNet Paper]
- 5 positions higher than Jasper
- Convolutional networks are truly parallel
- Faster than Jasper

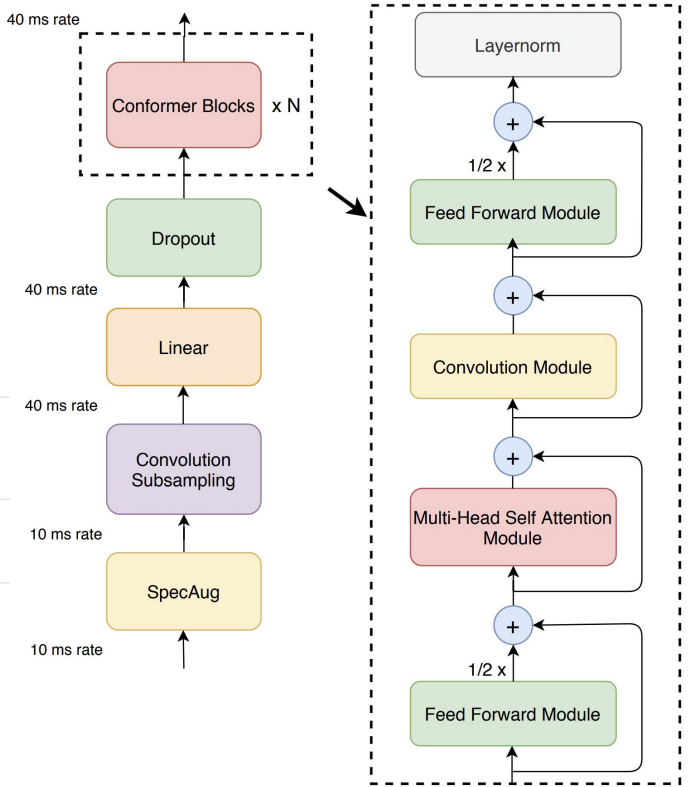
29	**QuartzNet15x5**	2.69	×	QuartzNet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions			2019

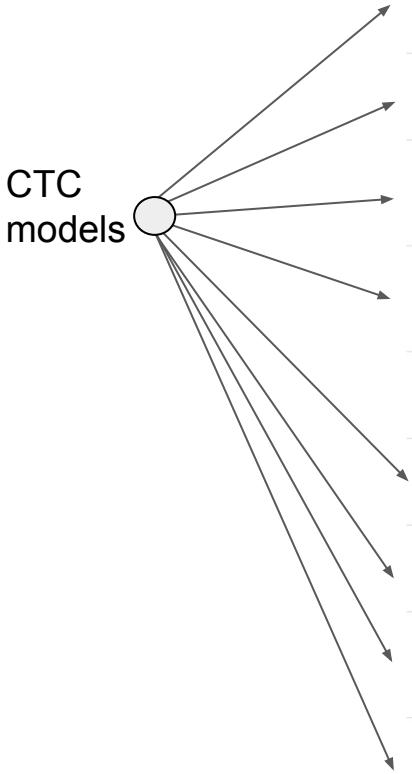


Conformer

- Link to Conformer Paper
- Use Transformer blocks instead of simple convolutions
- Transformer blocks don’t work without convolutions
- Add simple convolutions
- 20 positions higher than QuartzNet!
- Quite fast

9	**ContextNet** (L)	1.9	×	ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context			2020
10	**Conformer** (L)	1.9	×	Conformer: Convolution-augmented Transformer for Speech Recognition			2020



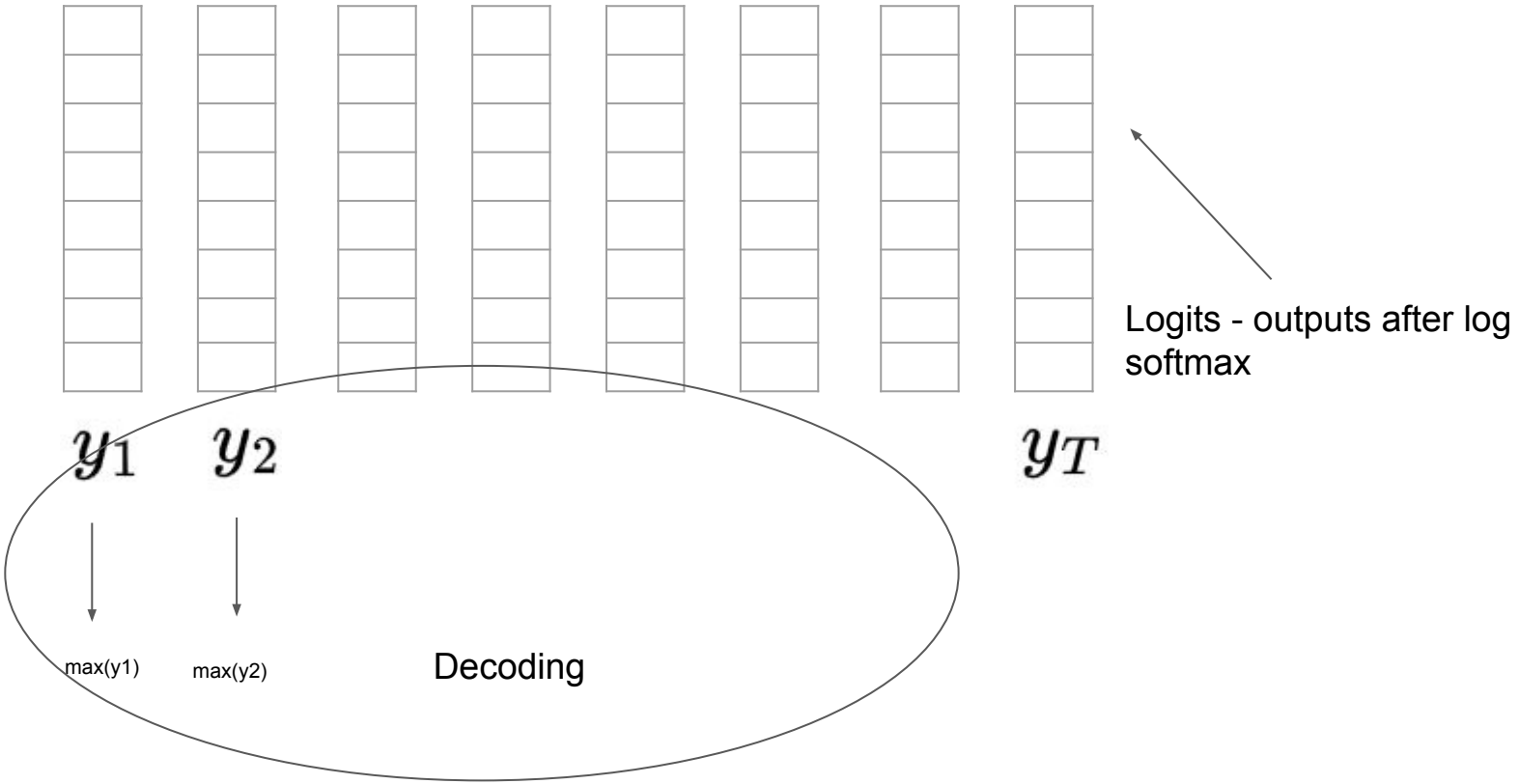


9	**ContextNet** (L)	1.9	×	ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context			2020	
10	**Conformer** (L)	1.9	×	Conformer: Convolution-augmented Transformer for Speech Recognition			2020	Conformer
11	**Transformer+Time reduction+Self Knowledge distillation**	1.9	×	Transformer-based ASR Incorporating Time-reduction Layer and Fine-tuning with Self-Knowledge Distillation			2021	Transformer
12	**ContextNet** (M)	2	×	ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context			2020	
13	**Transformer Transducer**	2.0	×	Improving RNN Transducer Based ASR with Auxiliary Tasks			2020	Transformer
14	**Conformer** (M)	2	×	Conformer: Convolution-augmented Transformer for Speech Recognition			2020	Conformer
15	**SpeechStew** (100M)	2.0	×	SpeechStew: Simply Mix All Available Speech Recognition Data to Train One Large Neural Network			2021	Conformer
16	**Conv + Transformer AM + Pseudo-Labeling** (ConvLM with Transformer Rescoring)	2.03	✓	End-to-end ASR: from Supervised to Semi-Supervised Learning with Modern Architectures			2019	Transformer
17	**Conv + Transformer AM + Iterative Pseudo-Labeling** (n-gram LM + Transformer Rescoring)	2.10	✓	Iterative Pseudo-Labeling for Speech Recognition			2020	Transformer

Recap

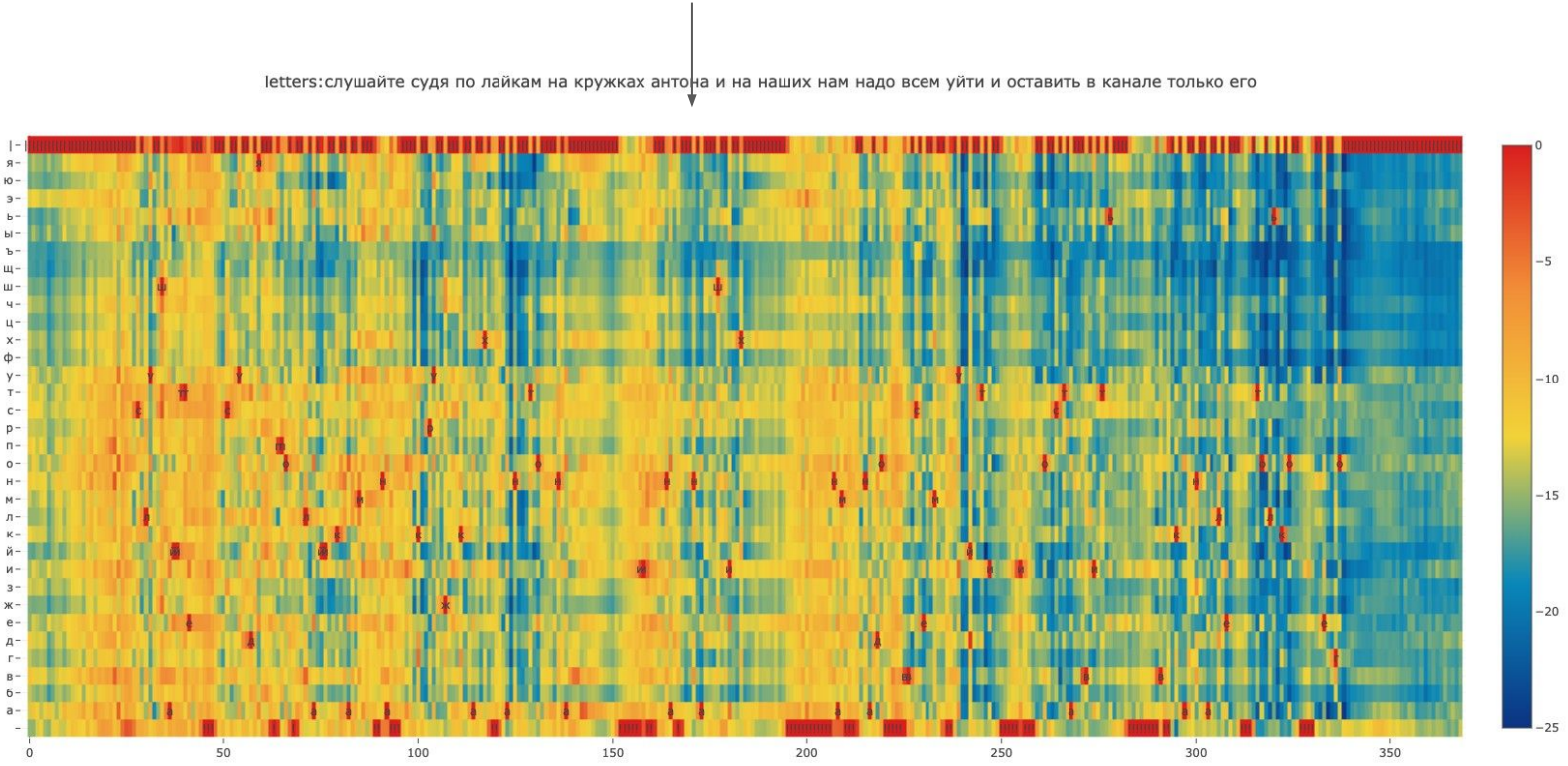
- ASR metric - Word Error Rate
- New sequence loss - CTC loss
- Produce conditionally independent outputs
- Decode CTC using 2 simple rules: reduce multiple letters, remove blank
- Use Phonemes, Letters, BPE
- Fastest approach to ASR: parallel forward pass
- LSTM models: DeepSpeech 2
- Convolutional models: Jasper, QuartzNet
- Transformer models: Conformer

Error Correction



Error Correction

All outputs are conditionally independent



Error Correction

Example Results (WSJ)

YET A REHBILITATION CRU IS ONHAND IN THE BUILDING LOOGGING BRICKS PLASTER AND BLUEPRINS FOUR FORTY TWO NEW BETIN EPARTMENTS

YET A REHABILITATION CREW IS ON HAND IN THE BUILDING LUGGING BRICKS PLASTER AND BLUEPRINTS FOR FORTY TWO NEW BEDROOM APARTMENTS

THIS PARCLE GUNA COME BACK ON THIS ILAND SOM DAY SOO

THE SPARKLE GONNA COME BACK ON THIS ISLAND SOMEDAY SOON

TRADE REPRESENTIGD JUIDER WARANTS THAT THE U S WONT BACKCOFF ITS PUSH FOR TRADE BARIOR REDUCTIONS

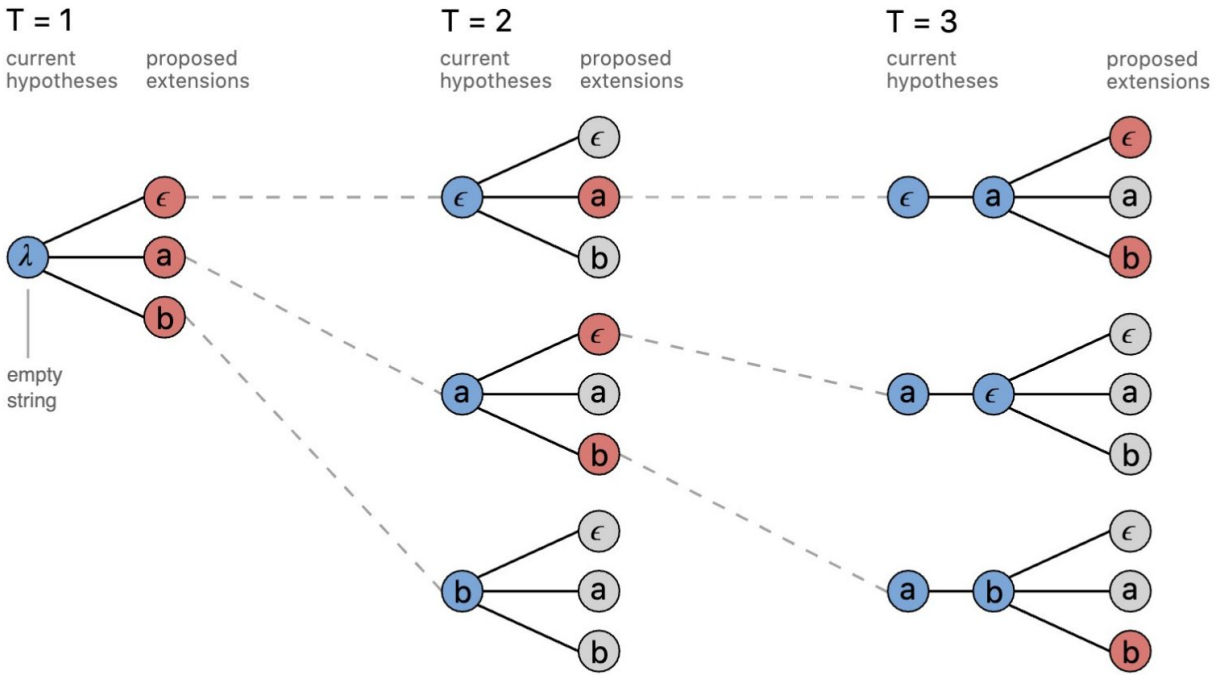
TRADE REPRESENTATIVE YEUTTER WARNS THAT THE U S WONT BACK OFF ITS PUSH FOR TRADE BARRIER REDUCTIONS

TREASURY SECRETARY BAGER AT ROHIE WOS IN AUGGRAL PRESSED FOUR ARISE IN THE VALUE OF KOREAS CURRENCY

TREASURY SECRETARY BAKER AT ROH TAE WOOS INAUGURAL PRESSED FOR A RISE IN THE VALUE OF KOREAS CURRENCY

Beam Search

- Greedy decoding is not the best
- Create search graph
- Try to extract new information from it

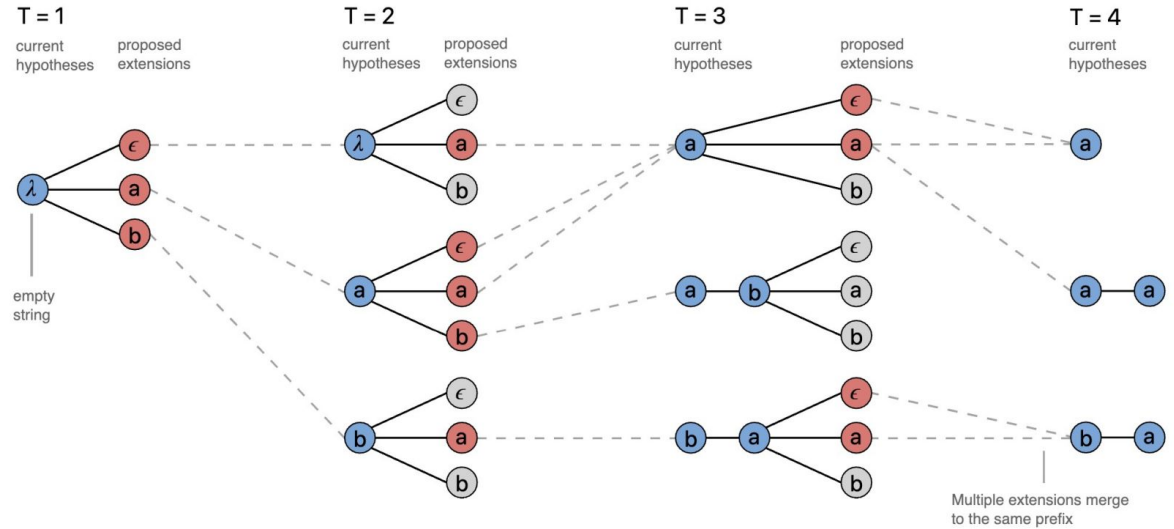


A standard beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three.

http://web.stanford.edu/class/cs224s/lectures/224s.22.lec10.pdf

Beam Search

- Create search graph
- Some paths will collapse due to 2 CTC rules
- Use LogSumExp to find probability



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

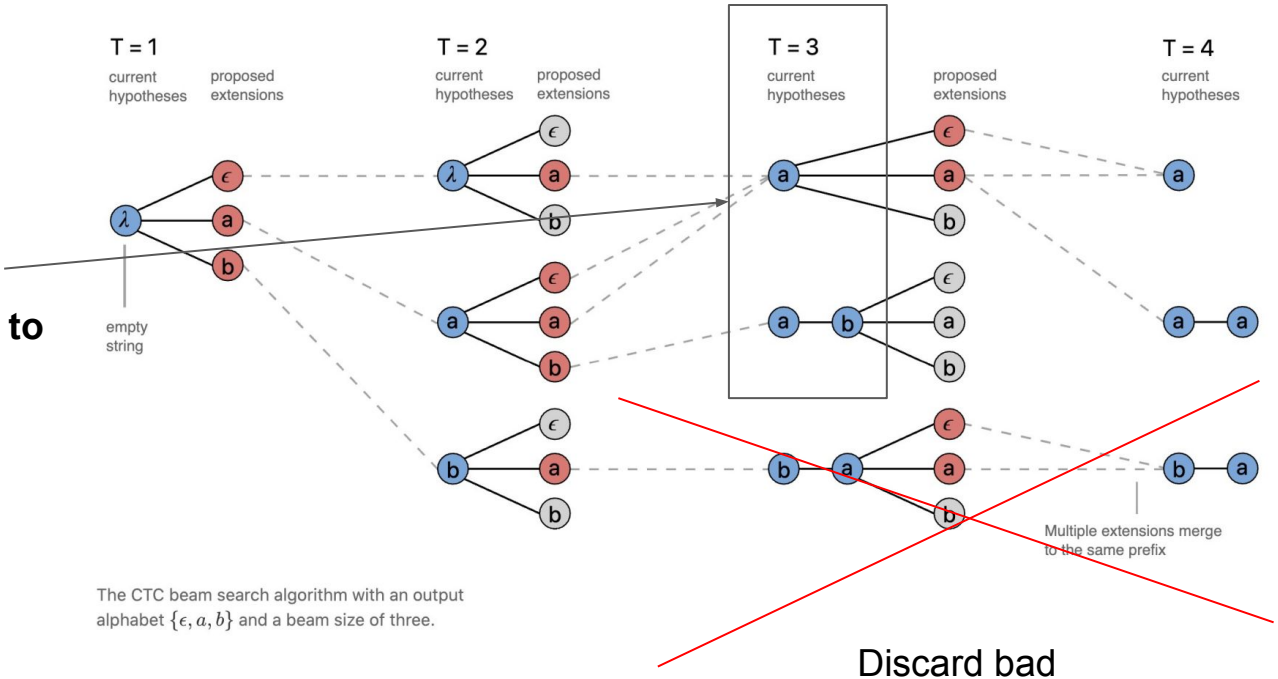
http://web.stanford.edu/class/cs224s/lectures/224s.22.lec10.pdf

Beam Search

- Use LogSumExp to find probability
- Cannot search whole graph
- Limit search with beam

- **Sort each stack according to probability**

- BeamSize = 2 means keep 2 hypotheses on each step



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

http://web.stanford.edu/class/cs224s/lectures/224s.22.lec10.pdf

Language models

$$P(w_1, w_2, w_3, \dots w_n) = P(w_1|w_2, \dots w_n) \cdot P(w_2|w_1, w_3, \dots w_n) \dots P(w_n|w_1, w_2, \dots, w_{n-1})$$

- Given a sentence S: sequence of words w1, w2…wn
- Need to calculate P(S)
- Probability distribution over sentences

Use Markov assumption

$$P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2, w_1) \dots P(w_n|w_{n-1}, w_{n-2} \dots)$$

Language models

- Factorize probability of sentence using Markov Assumption
- Sequence of words size of n is called **n-gram**
- Use n-gram counts to estimate probability

$$P(w_1, w_2, w_3, \dots w_n) = P(w_1|w_2, \dots w_n) \cdot P(w_2|w_1, w_3, \dots w_n) \dots P(w_n|w_1, w_2, \dots, w_{n-1})$$

Use Markov assumption

$$P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2, w_1) \dots P(w_n|w_{n-1}, w_{n-2} \dots)$$

$$P(w_3|w_1, w_2) = \frac{Count(w_1, w_2, w_3)}{Count(*, *, *)}$$

Language models

$$P(w_3|w_1, w_2) = \frac{Count(w_1, w_2, w_3)}{Count(*, *, *)}$$

- Count n-grams over large corpora of text
- We can now estimate probability of sentence

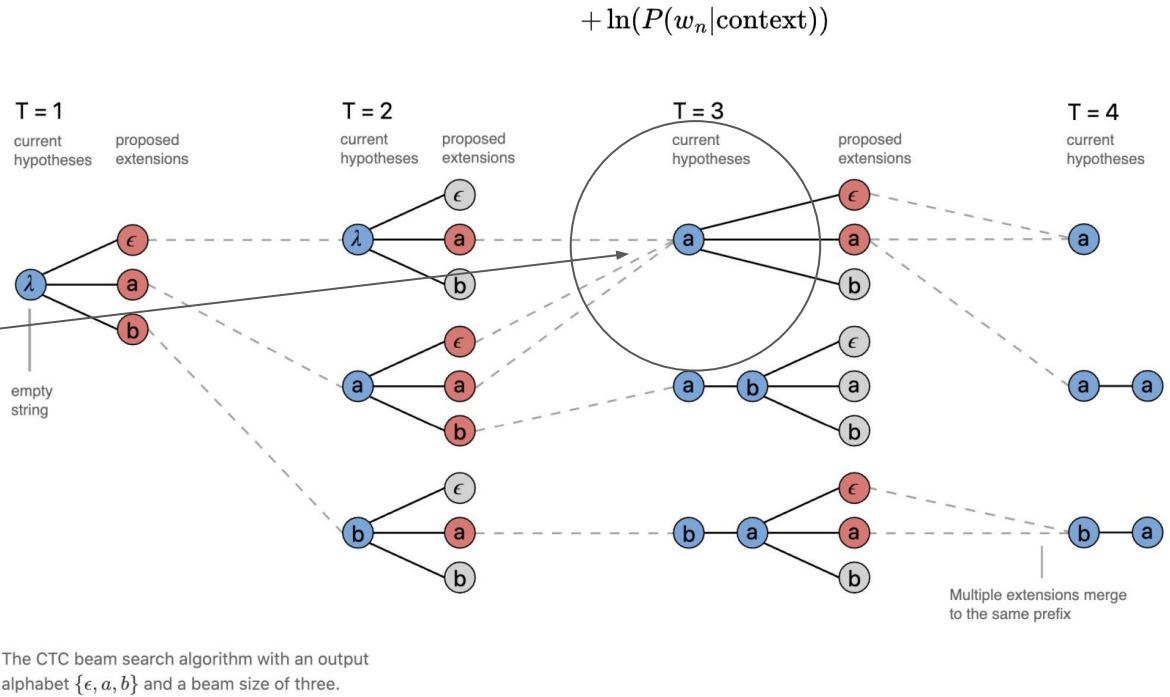
$$\text{Unigram LM} : p(w_1^N) = \prod_{n=1}^{N} p(w_n)$$

$$\text{Bigram LM} : p(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-1})$$

$$\text{Trigram LM} : p(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-2}, w_{n-1})$$

Beam Search

- How can we apply it to ASR?
- Score each prefix with language model
- If it word based, score when word is emitted
- Sort stack using **CTC probability** and **LM probability**



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

Beam Search

Before LM

- use LogSumExp, collapse CTC paths, extract best hypotheses
- Discard bad hypotheses

After LM

- Add language model score to introduce **new information** to Beam Search
- Sort according to weighted sum
- Discard bad hypotheses
- Need to estimate probability thousands of times

Slower

$$\text{score}(y_1, \dots y_n) = \ln \text{CTCScore}(y_{\leq t})$$

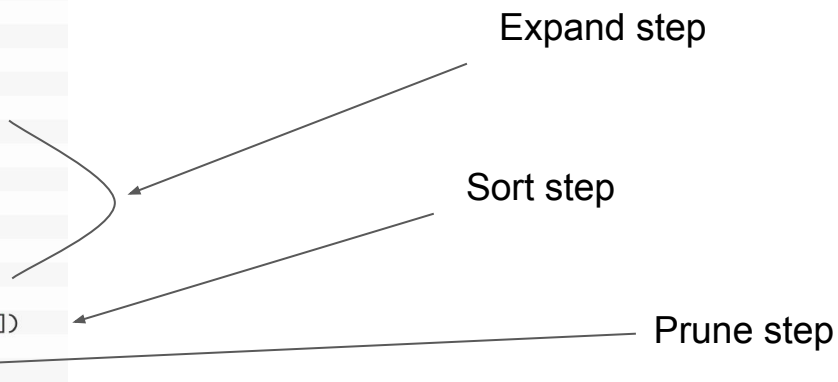
$$\text{score}(y_1, \dots y_n) = \ln \text{CTCScore}(y_{\leq t}) + \alpha \ln P(y_t | y_{t-1} \dots)_{LM}$$

Beam Search

```python
from math import log
from numpy import array
from numpy import argmax

# beam search
def beam_search_decoder(data, k):
    sequences = [[list(), 0.0]]
    # walk over each step in sequence
    for row in data:
        all_candidates = list()
        # expand each current candidate
        for i in range(len(sequences)):
            seq, score = sequences[i]
            for j in range(len(row)):
                candidate = [seq + [j], score - log(row[j])]
                all_candidates.append(candidate)
        # order all candidates by score
        ordered = sorted(all_candidates, key=lambda tup:tup[1])
        # select k best
        sequences = ordered[:k]
    return sequences

# define a sequence of 10 words over a vocab of 5 words
data = [[0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1],
        [0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1],
        [0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1],
        [0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1],
        [0.1, 0.2, 0.3, 0.4, 0.5],
        [0.5, 0.4, 0.3, 0.2, 0.1]]
data = array(data)
# decode sequence
result = beam_search_decoder(data, 3)
# print result
for seq in result:
    print(seq)
```



https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/

Beam Search

Approach	WER
Deep Speech 2 (no decoding)	22.83
Deep Speech 2 (4-gram LM, beam size of 512)	5.59
ESPnet (no decoding)	12.34
ESPnet (no LM, beam size of 20)	11.56
Kaldi TDNN (Chap. 8)	4.44

LM	LibriSpeech		WSJ	
	test-clean	test-other	93-test	92-eval
-	4.19	10.98	8.97	6.37
4-gram	3.21	8.04	5.57	3.51
T-XL	2.96	7.53	4.82	2.99

Oracle Metrics

Final hypotheses after beam search

- hypo1, score = -10
- hypo2, score = -12
- hypo3, score = -13
- hypo4, score = -100

Let's compute **Oracle Word Error Rate**

$$\text{OracleWER}(\text{Hypotheses}, \text{Ref}) = \min_{\text{hyp} \in \text{Hypotheses}} \text{WER}(\text{hyp}, \text{Ref})$$

Oracle WER is always less or equal to WER

How to improve beam search?

Model Error

WER = 10%
OWER = 1%

Correct answer appears in N-best.
Model does not discriminate good enough

Additional information (better LM, better formula) helps

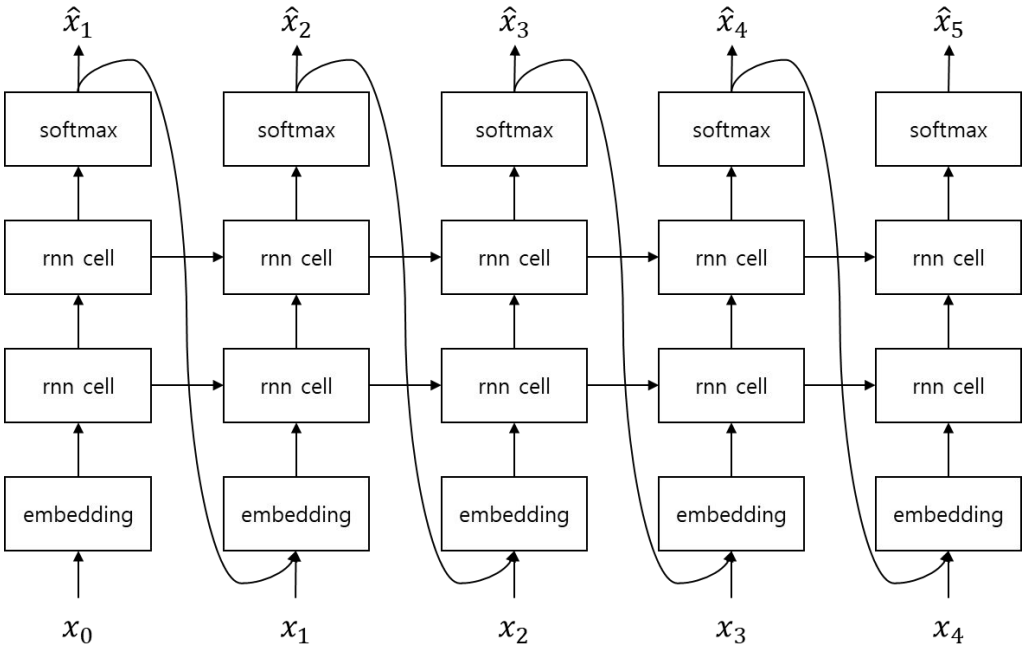
Search Error

WER = 10%
OWER = 9.5%

Correct answer does not appear in N-best.
New information would not help

use blank weight, length penalty

Neural Language Models



- Train neural network to predict next word
- Use LSTM/Transformer
- No expensive data needed

LM	LibriSpeech		WSJ	
	test-clean	test-other	93-test	92-eval
-	4.19	10.98	8.97	6.37
4-gram	3.21	8.04	5.57	3.51
T-XL	2.96	7.53	4.82	2.99

Models

NGram Language Model

- Statistical language model
- Estimates probability with N-grams counts
- Very fast
- Large RAM usage

Neural Language Model

- Neural Language Models (BERT, GPT-3)
- Estimates probability with neural network
- Uses GPU, quite slow
- Large RAM usage
- Superior quality

Models

NGram Language Model

- Statistical language model
- Estimates probability with N-grams counts
- Very fast
- Large RAM usage

Use during Beam Search

Neural Language Model

- Neural Language Models (BERT, GPT-3)
- Estimates probability with neural network
- Uses GPU, quite slow
- Large RAM usage
- Superior quality

Use after Beam Search

Rescoring

Final hypotheses after beam search

- hypo1, score = -10
- hypo2, score = -12
- hypo3, score = -13
- hypo4, score = -100

$$\text{score}(y_1, \dots y_n) = \ln \text{CTCScore}(y_{\leq t}) + \alpha \ln P(y_t | y_{t-1} \dots)_{LM}$$

Rescoring

Final hypotheses after beam search

- hypo1, score = -10
- hypo2, score = -12
- hypo3, score = -13
- hypo4, score = -100

$$\text{score}(y_1, \dots y_n) = \ln \text{CTCScore}(y_{\leq t}) + \alpha \ln P(y_t | y_{t-1} \dots)_{LM}$$

$$+ \ln P(y | y_{t-1} \dots)_{\text{neural LM}}$$

Use neural language model to **rescore** final NBest

Recap

- Use greedy decoding to get fastest results
- Beam search yields better quality
- Keep track of BeamSize hypotheses at a time
- Some speed degradation is expected
- Use NGram language model to introduce new information
- Use smaller models in beam search to get N-best
- Rescore N-best using big models

Next time

- Encoder-Decoder
- RNN-T
- Unsupervised Pretraining (Wav2Vec2.0)