Automatic Speech Recognition

Pavel Bogomolov, developer at Yandex

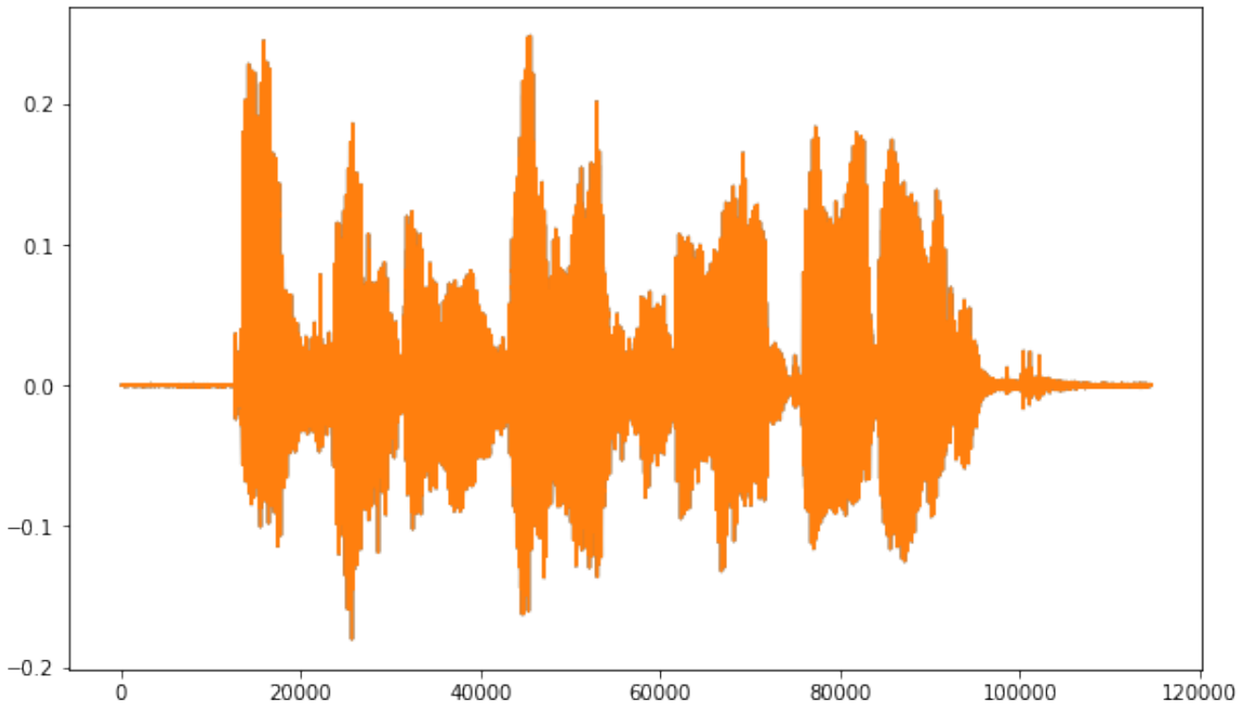
Outline

- Problem definition
- Quality assessment
- Datasets
- End-to-end ASR
- Listen, Attend and Spell
- CTC loss
- Language models

Problem definition

- Input: sound
- Output: the text spoken on the record
- Vocabulary: all words of the language (unlike spotter)

Problem definition



"потише сделай чего орешь то"

Quality assessment

- The performance is easier to measure than in machine translation, since there is only one true output

- If we write the words of the hypothesis under the words of reference, then accuracy = #correct / len(reference)

- The length of the output can differ from the reference, therefore we still need a more robust metric

Example

- Reference text: алиса включи пожалуйста детские сказки
- Predicted text: алиса включи детские сказки
- Accuracy = 2/5 = 0.4 (error: 0.6)
- How can we fix that?

Word Error Rate (WER)

- Align the two sentences minimizing word-wise Levenshtein distance
- Three types of errors: substitutions (S), insertions (I), deletions (D)
- WER = (S + I + D) / N, where N is the total number of words in the reference text
- Can it be higher than 100%?
- There are other, less commonly used metrics (CER, SER)

Example

- Reference text: алиса включи пожалуйста детские сказки
- Predicted text: алиса включи детские сказки
- Accuracy = 2/5 = 0.4 (error: 0.6)
- WER = (0 + 0 + 1) / 5 = 0.2

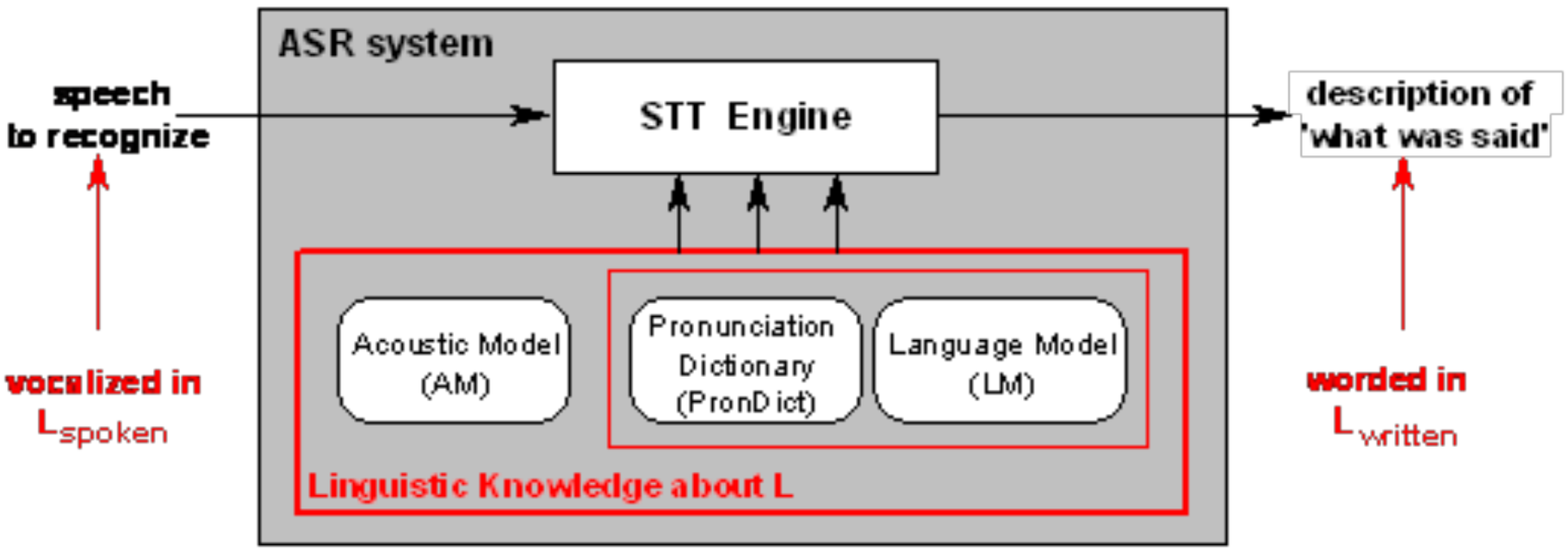
Open datasets

- Librispeech: large-scale (1000 hours) corpus of read English speech
- WSJ: 73 hours of read sentences from Wall Street Journal
- Switchboard: 260 hours, telephone conversations
- TIMIT: a total of 6300 sentences (5.4 hours), consisting of 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States.

"Closed" datasets

- Companies have much more data
- One of the largest datasets, Librispeech (100 GB), contains only 1000 hours of speech
- Meanwhile, Google has datasets of 12500 and even 125000 hours
- The more data and resources, the higher the final quality

Conventional ASR



Conventional ASR

- Acoustic model (AM) provides phonetic representation of speech
- Pronunciation dictionary contains phonetic transcriptions of words
- Language model (LM) computes the probability of a given sequence of words
- All these components are trained separately and require expert knowledge and careful tuning, but the final quality can surpass end-to-end systems to this day (e.g. Librispeech SOTA)
- The pipeline is similar to statistical machine translation, with its translation and language models

End-to-End ASR

- The task more formally:
- The input is a sequence of frames (x_1, ... x_N)
- We need to produce a sequence of characters (or, more generally, any tokens: words, BPE, etc.) (h_1, ..., h_L), minimizing WER on the test set

End-to-End ASR

- The task more formally:

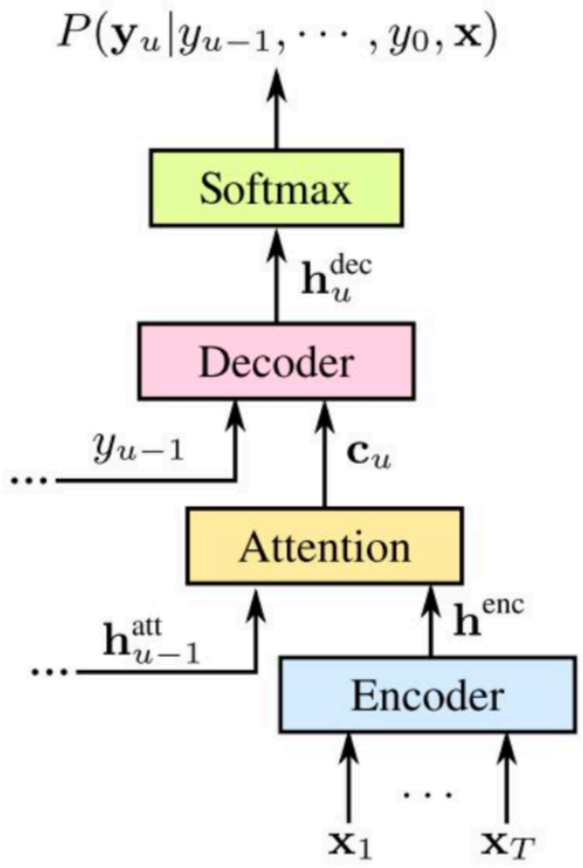
- The input is a **sequence** of frames (x_1, ... x_N)

- We need to produce a **sequence** of characters (or, more generally, any tokens: words, BPE, etc.) (h_1, ..., h_L), minimizing WER on the test set

Does this seem familiar?

Listen, Attend and Spell

Attention-based Encoder-Decoder Models

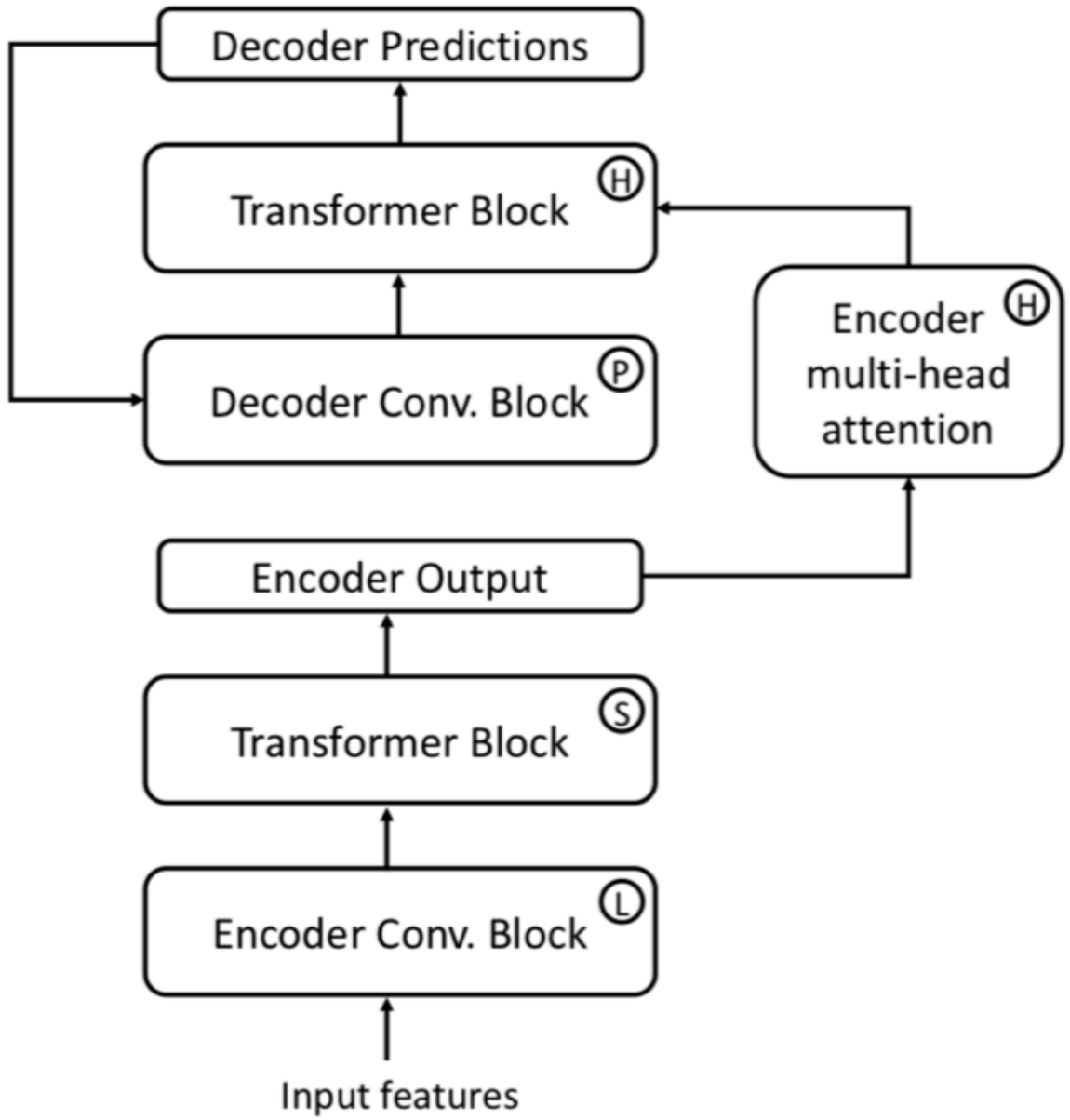


- Encoder (analogous to AM):
  - Transforms input speech into higher-level representation
- Attention (alignment model):
  - Identifies encoded frames that are relevant to producing current output
- Decoder (analogous to PM, LM):
  - Operates autoregressively by predicting each output token as a function of the previous predictions

Listen, Attend and Spell

- A sequence-to-sequence model with attention
- The original paper by Google proposed using LSTMs for encoding and decoding, but any architectures will work
- Even transformers!

Transformers with convolutional context for ASR



Transformers with convolutional context for ASR

- Code: https://github.com/pytorch/fairseq/tree/master/examples/speech_recognition

Advantages of LAS

- The model takes into account the previously produced characters at each step, leading to more coherent results

- Attention lets the decoder obtain more information about the whole record

Offline vs online ASR

- In dialogue systems, low latency is crucial
- Systems which need the whole record before they can start decoding are called "offline"
- "Online" systems can produce outputs incrementally, leading to much lower latency

Shortcomings of LAS

- This model is offline-only, since we need the features of the whole record for decoding

- Training requires tricks like teacher forcing and scheduled sampling

- The most important one is the autoregressive nature of the decoder, which means that the inference time scales with the length of the output

Frame-level predictions

- Idea: make predictions for each frame and then reconstruct the answer from them, alleviating the need for an autoregressive decoder

- We can use character-level alignments and train models using frame-wise cross entropy

- But alignments are too difficult to obtain

Connectionist Temporal Classification

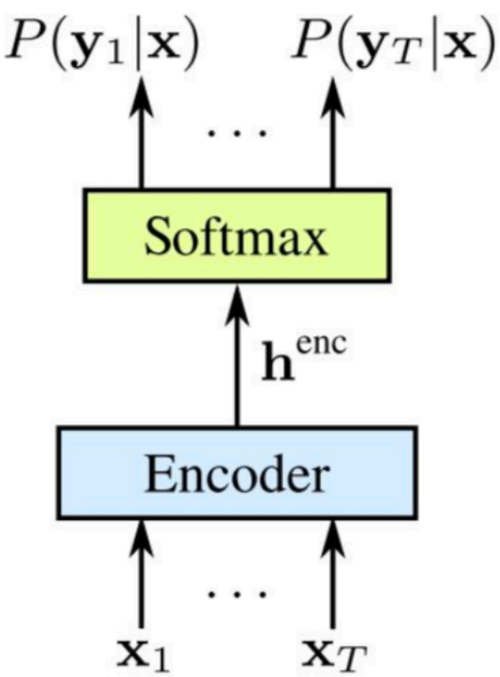
- We predict a character for each frame, augmenting the vocabulary with a blank symbol (-)

- Remove duplicate consecutive characters, then remove blanks to obtain the final prediction

- Example: lllllllllooooooo---sssss-ss -> loss

- The network predicts probability distributions for every frame

Connectionist Temporal Classification



B B **c** B B **a** **a** B B **t**
B **c** **c** B **a** B B B B **t**
...
B **c** B B **a** B B **t** **t** B

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\hat{\mathbf{y}} \in \mathcal{B}(\mathbf{y},\mathbf{x})} \prod_{t=1}^{T} P(\hat{y}_t|\mathbf{x})$$

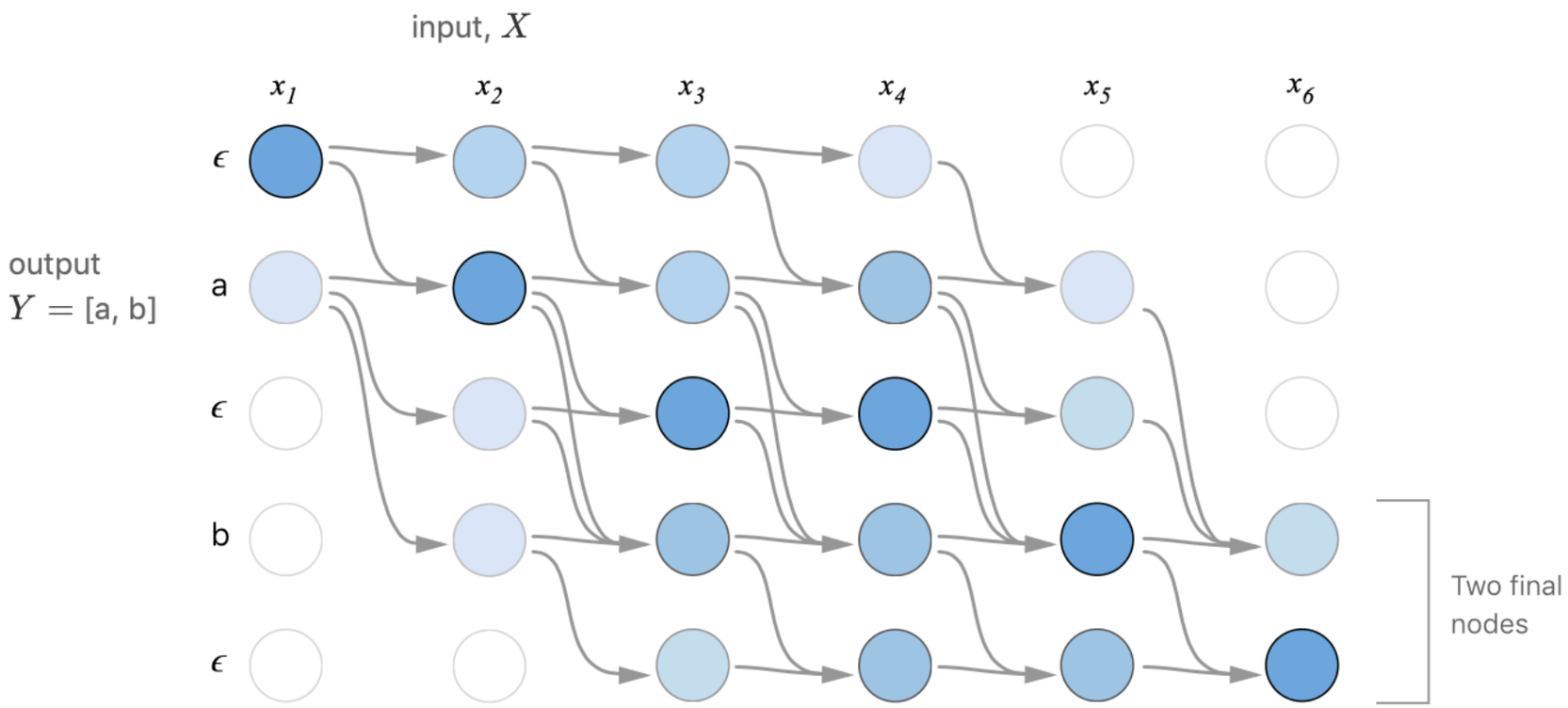
Connectionist Temporal Classification

- Loss: -log P(all paths which decode as target)

- Turns out, this loss is differentiable!

- We can use dynamic programming to calculate the loss and the gradients

Connectionist Temporal Classification

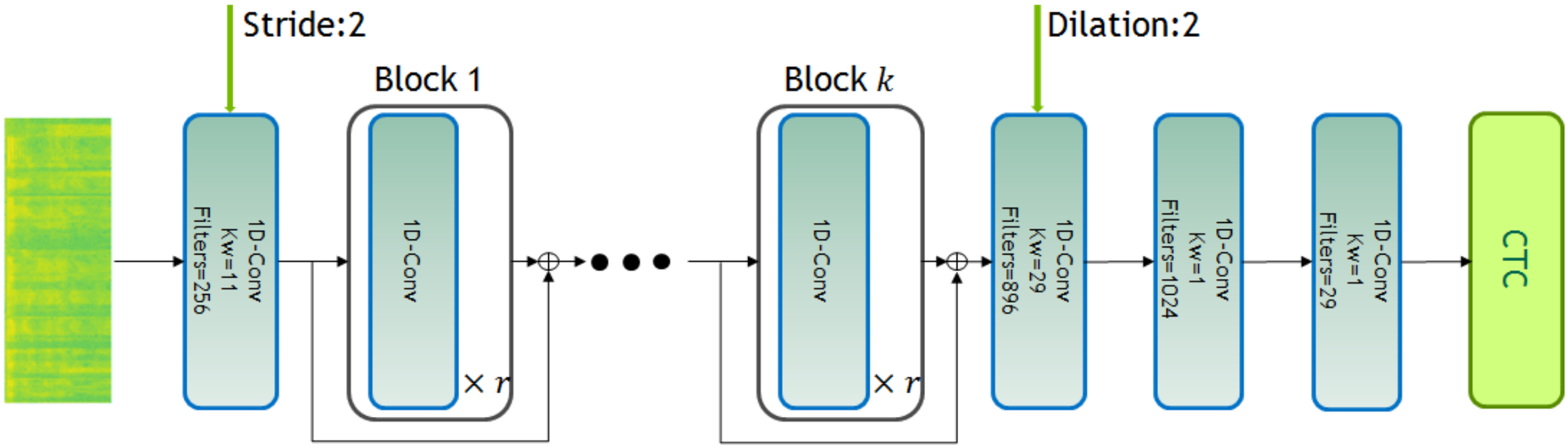


Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

Connectionist Temporal Classification

- A very thorough explanation of how exactly CTC works: https://distill.pub/2017/ctc/

Jasper


Stride:2
Block 1
Block k
Dilation:2
1D-Conv
Kw=11
Filters=256
1D-Conv
× r
1D-Conv
Kw=29
Filters=896
1D-Conv
Kw=1
Filters=1024
1D-Conv
Kw=1
Filters=29
CTC

Jasper

- PyTorch: https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/SpeechRecognition/Jasper

- TensorFlow: https://github.com/NVIDIA/OpenSeq2Seq

Advantages of CTC

- Simplicity: the training process is pretty straightforward, does not need teacher forcing

- Speed: non-autoregressive model => latency depends only on input length

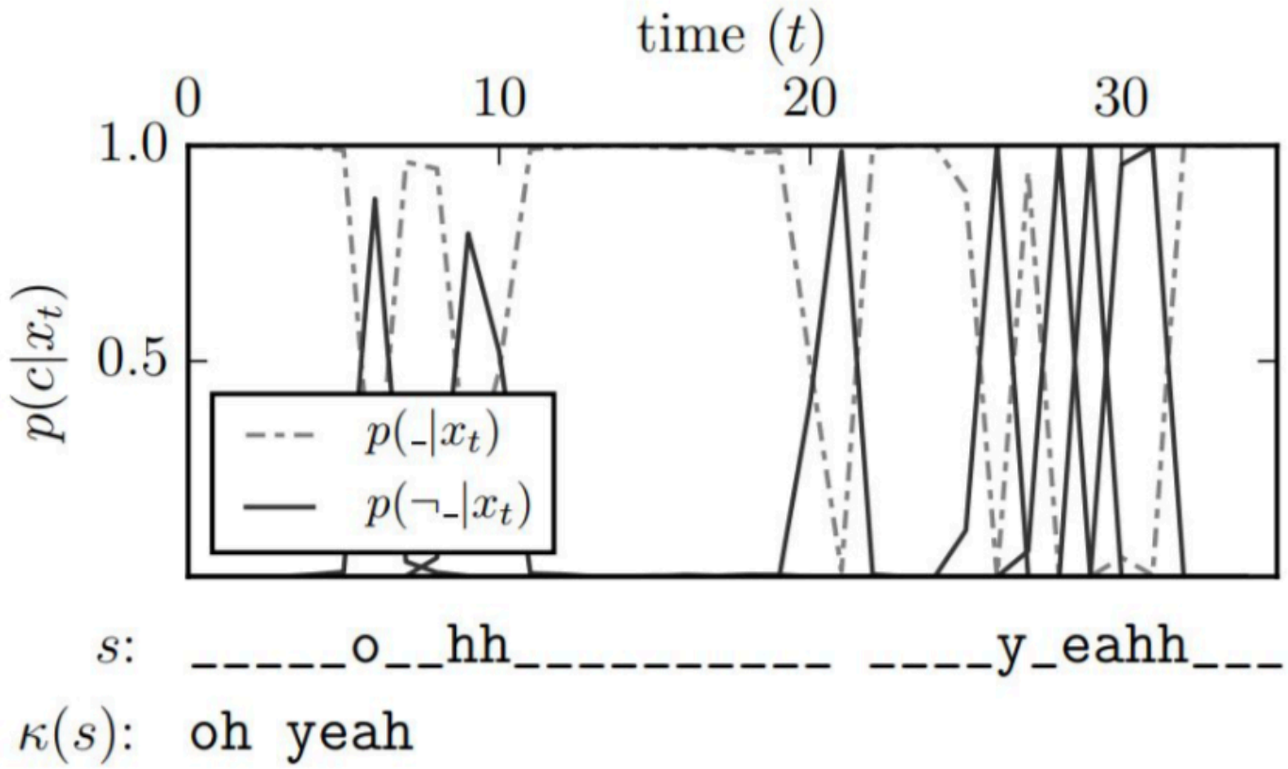
- Streaming: depending on the encoder architecture, may be applied online, producing predictions incrementally

- Adaptiveness: can be more robust to domain shift (provided the acoustics are the same)

Shortcomings of CTC

- The network outputs at different frames are conditionally independent, can produce undesirable results (especially in cases without normalization, e.g. "AAA" vs "triple A")
- To remedy this, CTC-trained models are usually decoded using beam search with an external language model
- Overconfidence: activations are "spiky"

Shortcomings of CTC



Language models

- While there exist large text corpuses, end-to-end ASR training utilizes relatively small amounts of text from audio transcriptions

- LM-rescoring significantly improves the quality of attention-based models (https://arxiv.org/abs/1910.14659v1)

- CTC models need LM-decoding because of the conditional independence of frame predictions

Using LMs for ASR

- We can use an LM to rescore the top-n hypotheses in the last state of beam search (rescoring)

- We can also use an LM during the decoding, either to modify the scores in the beam, or to feed more information to the decoder directly (fusion)

LM-rescoring

Model	dev		test	
	clean	other	clean	other
baseline (100-best)	7.17	19.79	7.26	20.37
GPT-2 (117M, cased)	5.39	16.81	5.64	17.60
BERT (base, cased)	5.17	16.44	5.41	17.41
RoBERTa (base, cased)	**5.03**	**16.16**	**5.25**	**17.18**
GPT-2 (345M, cased)	5.15	16.48	5.30	17.26
BERT (large, cased)	4.96	16.26	5.25	16.97
RoBERTa (large, cased)	**4.75**	**15.81**	**5.05**	**16.79**
oracle (100-best)	2.85	12.21	2.81	12.85

Table 1: WERs on LibriSpeech after reranking. Baseline lists and oracle numbers are from Shin et al. (2019).

LM-rescoring

Model	dev		test	
	clean	other	clean	other
baseline (100-best)	7.17	19.79	7.26	20.37
uni-SANLM	6.08	17.32	6.11	18.13
bi-SANLM	5.52	16.61	5.65	17.44
BERT (base, Libri. only)	**4.63**	**15.56**	**4.79**	**16.50**
BERT (base, cased)	5.17	16.44	5.41	17.41
BERT (base, uncased)	5.02	16.07	5.14	16.97
+ adaptation, 380k steps	**4.37**	**15.17**	**4.58**	**15.96**
oracle (100-best)	2.85	12.21	2.81	12.85

Table 4: WERs on LibriSpeech after reranking. Baseline, SANLM, and oracle numbers are from Shin et al. (2019).

LM-fusion

- Shallow fusion - add up log-probabilities of acoustic and language models in inference to select top-hypotheses

- Deep fusion - train LM and AM separately, then add a new layer combining their hidden states to produce the output distribution

- Cold fusion - feed LM-logits to the decoder in a special gated way; allows changing LMs without retraining the acoustic model

Thank you!

Useful links

- https://www.cs.toronto.edu/~graves/icml_2006.pdf - CTC
- https://arxiv.org/abs/1508.01211 - Listen, Attend and Spell
- https://arxiv.org/abs/1904.03288 - Jasper
- https://arxiv.org/abs/1904.11660 - Transformers for ASR
- https://arxiv.org/abs/1910.14659v1 - LM-rescoring
- https://arxiv.org/abs/1807.10857 - LM-fusion

Contact info

- E-mail: pibogomolov@yandex.ru

- Telegram: @bobrosoft98