Non-Autoregressive Transformers in Automatic Speech Recognition

George Farah

george.farah@rwth-aachen.de

Selected Topics in Human Language Technology and Pattern Recognition

February 22, 2022

Human Language Technology and Pattern Recognition
Computer Science Department, RWTH Aachen University

Literature overview

- **Attention is All you Need (2017) by Vaswani et al.**
  The first paper to present the concepts of transformer and emphasized the importance of self-attention mechanism.
- **Listen and Fill in the Missing Letters: Non-Autoregressive Transformer for Speech Recognition (2020) by Chen et al.**
  Presented the concept of Non-Autoregressive transformer in the context of ASR using conditional masked language model.
- **Mask CTC: Non-Autoregressive End-to-End ASR with CTC and Mask Predict (2020) by Higuchi et al.**
  Extended the concept of NAT ASR by using CTC loss.
- **Spike-Triggered Non-Autoregressive Transformer for End-to-End Speech Recognition (2020) by Tian el al.**
  Introduces a Spike-Triggered CTC approach to ASR.

Automatic Speech Recognition

- **End to End Automatic Speech Recognition (ASR) turn the audio input speech into the corresponding text sequence.**
- **Popular end to end sequence modeling are LSTMs or RNNs.**

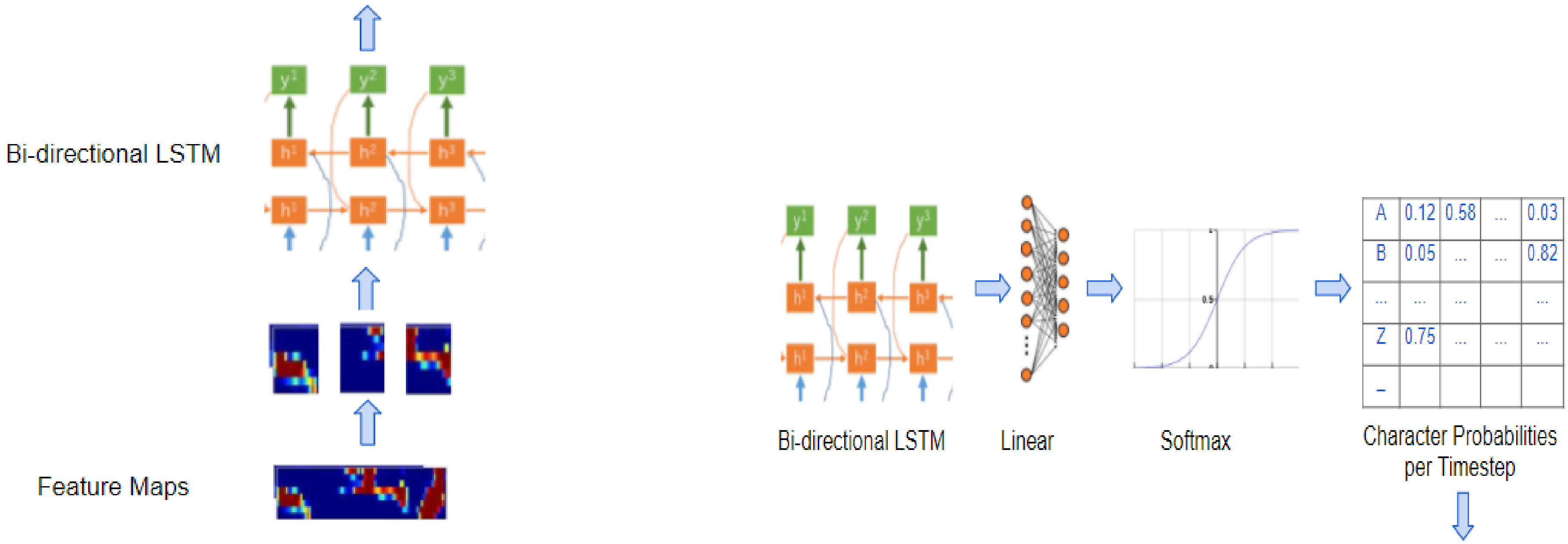

Figure: Example of LSTM based ASR [1]

Self-Attention

- Transformer concept presented in 2017 in the Paper "Attention is All you Need". Transformers is a model architecture relying entirely on self attention mechanism [2].
- Main advantages include enhancement in learning long range dependencies in the sequence, speed up in the training phase.

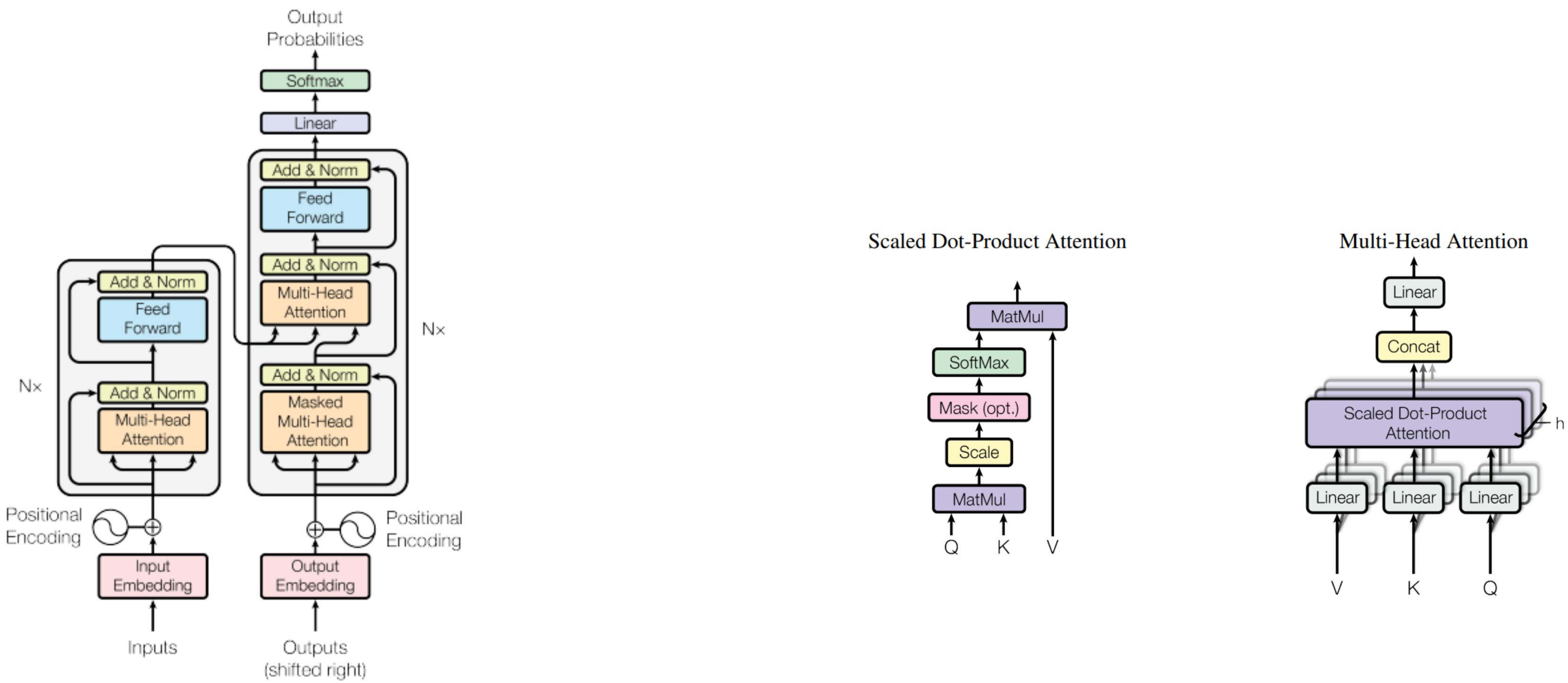

Figure: Transformer model architecture and attention mechanism [2]

- **In the training phase, transformer perform training in parallel because of the self attention matrix nature, and the ground history tokens being fed into the decoder.**
- **During Inference the decoding is done in Autoregressive manner, the target sequence is generated character by character from left to right.**
- **The generated sequence at point y_t depends on the predictions from the previous sequence y_{t-1}. So no parallelisation in possible.**

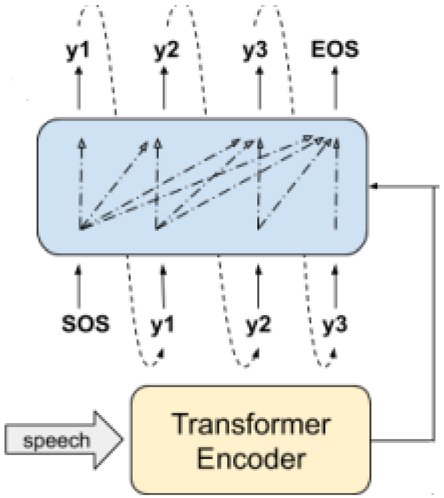

Figure: Decoding Phase in Transformers [3]

Non-Autoregressive Models

- **Non-Autoregressive Models (NAT) were presented to introduce parallel computing into the inference phase.**
- **One of the most common method is Audio Conditional Masked Language Model (A-CMLM) with mask predict which was presented in the context of machine translation and extended to ASR in the paper: Listen and Fill in the Missing Letters.**
- **This new method requires adjustments to the decoder in the training phase.**

Training method: Audio-Conditional Masked Language Model (A-CMLM)

- **This new method requires adjustments to the decoder in the training phase.**
- **Replace the ground truth tokens with some other information like partial decoding results. [4]**
- **Output Probabilities in the Autoregressive (AR) framework is presented as the following:**

$$P(y_t | y_{<t}, f_t(h))$$

- y_t **is the next token in the sequence,** $y_{<t}$ **are previous history tokens and** $f_t(h)$ **is the attention mechanism with hidden representation** h**.**

CMLM

- Random tokens in the decoder input $y_{<t}$ are replaced with special masked tokens $<$ MASK $>$
- The network now has to predict the original unmasked tokens.
- This can be presented mathematically as the following:

$$P(y_{T_M}|y_{T_U}, x) = \prod_{t \in T_M} P(y_t|y_{T_U}, f_t(h))$$

- T_M, T_U are sets of masked and unmasked tokens. Important assumption in this expression is conditionally independent so it could be computed in parallel.

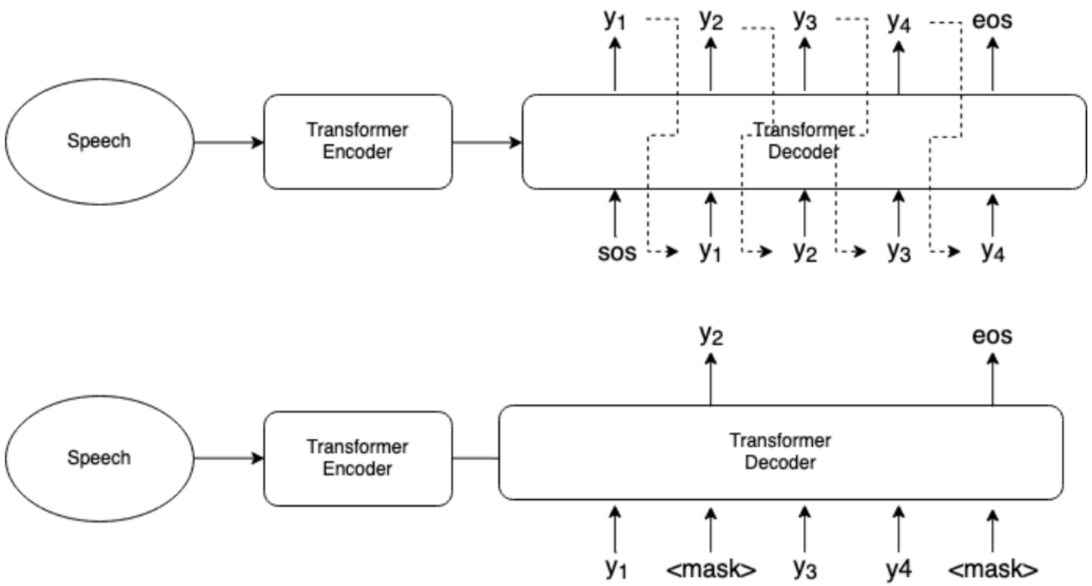

Figure: Comparing training and decoding (dashed lines) between NAT and AT [4]

Decoding Method

- **Main idea is to predict the whole sequence within a constant number of iterations independent on the output sequence length.**
- **The decoder is fed with Mask tokens for all time steps and then make predictions on them at each iteration**
- **Easy first: make predictions at each iteration and keep the most confident ones without replacing them.**
- **Mask-predict: make predictions at each iteration and keep the most probable ones, replace previous predictions if they were more less confident.**

Easy First vs Mask Predict

- **Easy first: In first iteration $\hat{y}_0$ decoder is fed with mask tokens, afterwards we keep most confident ones and update them in y_1:**

$$\hat{\mathrm{y}}_t^1 = \begin{cases} \textbf{arg max}_V P(\mathrm{y}_t^1|.) & t \in \textbf{largest}_C(max_V P(\mathrm{y}^1|.)) \\ \hat{\mathrm{y}}_t^0 & \textbf{otherwise} \end{cases}$$

- $C = L/K$. L **sequence length,** K **number of iterations**
- **Mask Predict: Also start with masked tokens, see the most probable token for each output, use it as a confidence score and replace least confident ones by mask tokens.**

$$\hat{\mathrm{y}}_t^k = \begin{cases} < \textbf{MASK} > & t \in \textbf{smallest}_C(\textbf{max}_V P(\mathrm{y}_t^k|.)) \\ \textbf{arg max}_V P(\mathrm{y}_t^k|.) & \textbf{otherwise} \end{cases}$$

- $C = L * (1 - \frac{k}{K})$
- **After getting prediction we update the probabilities of all previously masked tokens.**

Decoding Example

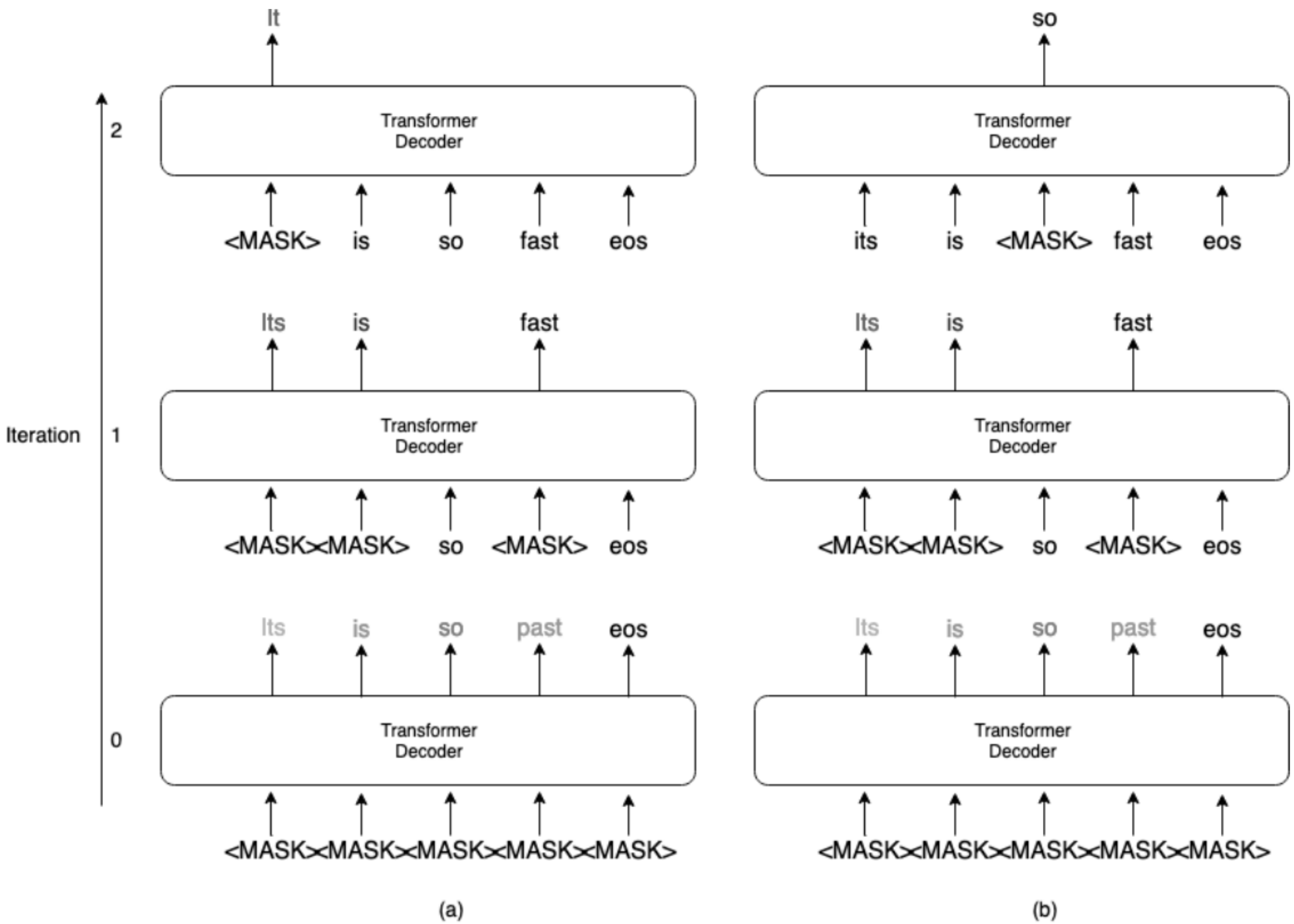

Figure: left: Easy Predict, right: Mask Predict. Notice the difference in predicting the token "so". [4]

Sequence Length Problem

- NAT needs to resolve the issue of the predicted sequence length.
- EOS token needs to be predicted at the end of the speech. However a fixed sequence length must be predefined.
- This becomes an issue if the predefined length is shorter than the actual sequence length which might lead to deletion.

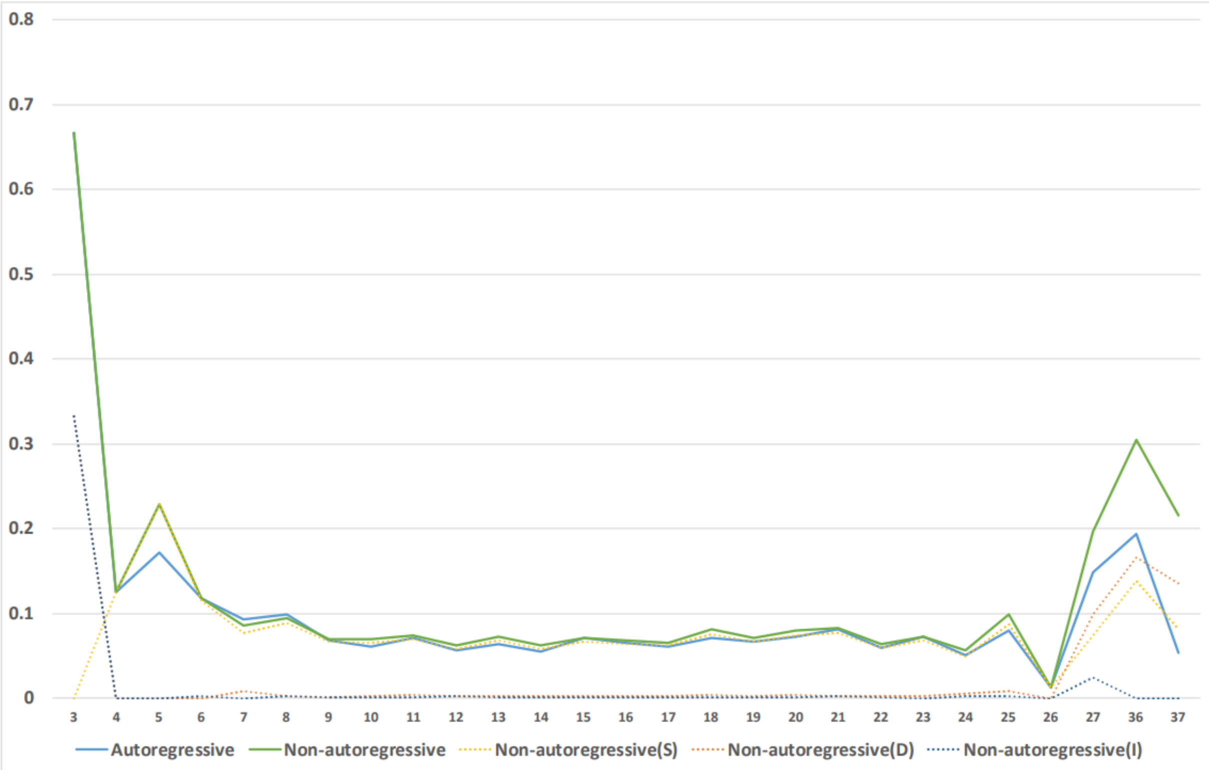

Figure: Comparing different output sequence lengths error rates and the cause of the error. [4]

CTC

- Connectionist temporal classification (CTC) encodes sequence with $<$ BLANK $>$ tokens that represent a blank character.
- At each time step t of the input, an possible output sequence alignment is created
- Each alignment is filled with a distribution of blank or character tokens and the total output is collapsed.
- The output is found by marginalising over all possible alignments to find the most probable.

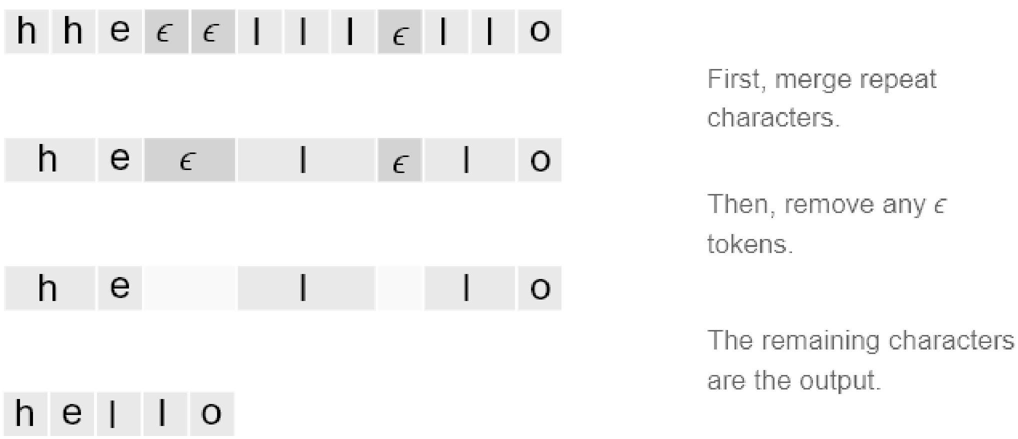

Figure: Process of filling and finding the CTC output of the word "Hello", ϵ represent blank tokens. [5]

CTC-Mask Approach

- **CTC predicts frame level alignment between input and output sequence.**
- **In this approach in ASR, CTC loss function is combined with the CMLM loss to train the network.**

$$P_{CTC}(Y|X) = \sum_{A \in \beta^{-1}(Y)} P(A|X)$$

$$\mathcal{L} = \gamma log P_{CTC}(Y|X) + (1-\gamma) log P_{cmlm}(Y_{T_M}|Y_{T_U}, X)$$

Where A is predicted alignment with blank tokens, γ is a tunable parameter and $\beta^{-1}(Y)$ is the set of all possible allignemtns.

Decoding

- **The CTC outputs are considered as the initial sequence for decoding.**
- **CTC outputs are obtained through a greedy search algorithm so the inference is Non-Autoregressive .**
- **A certain probability threshold is set to check whether a prediction is confident enough and need to be masked or not.**
- **The tokens that were masked get predicted by conditioning on the more confident ones, using easy first.**

Decoding Example

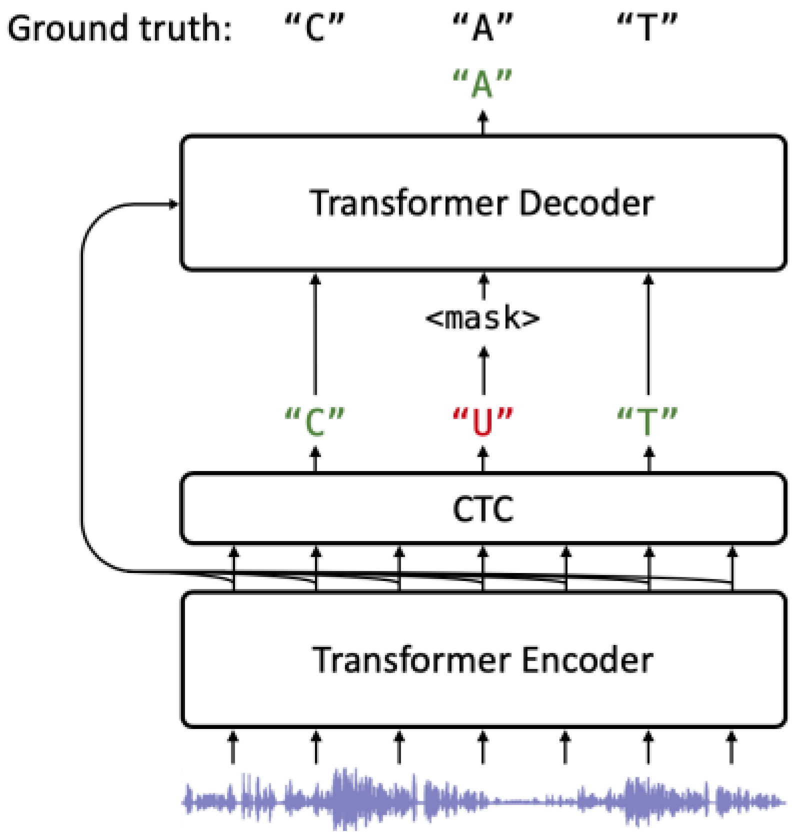

Figure: Decoding Procedure of Mask-CTC [6]

Decoding Example

Ground truth

instead they favor unannounced checks by roving rather than in house inspectors focusing on critical control points in seafood processing

Greedy CTC inference

instead they favor un anounced checks by roving rather than in house anspectors focusing on critical control points in sefood processing

Proposed CTC masking & Iterative decoding ($P_{\text{thres}} = 0.999, K = 3$)

instead they favor un__noun__d ch__ks by roving rather than _n_house _nspectors focusing on crit_cal control points in __food processing
instead they favor unannoun__d ch__ks by roving rather than _n_house inspectors focusing on critical control points in __food processing
instead they favor unannounc_d ch_cks by roving rather than in house inspectors focusing on critical control points in __food processing
instead they favor unannounced checks by roving rather than in house inspectors focusing on critical control points in sifood processing

Figure: Decoding Example on the WSJ Corpus. Red characters after CTC inference are low confidence tokens. [6]

Spike Triggered NAT

- **CTC approach might generate duplicated tokens and a large number of blanks during inference which slows down the computation**
- **Instead, add a spike trigger into the CTC: If a certain threshold is passed that a token is not blank, the trigger is recorded.**
- **Instead of using the mask approach, the triggered spikes are used as input, which encode more information than empty masked tokens.**
- **No need for CMLM approach rather with CTC and cross entropy joint loss.**

Spike NAT: Model Architecture

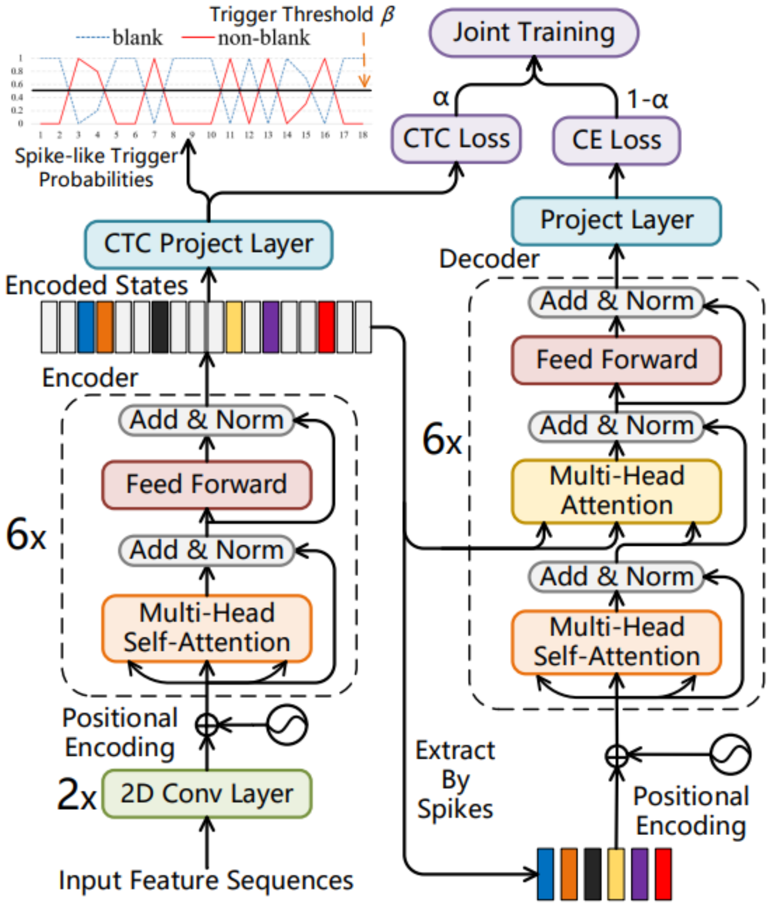

Figure: Spike triggered model architecture. Notice spike triggered encoding fed into the decoder. [7]

Spike Triggered NAT

- **The model is trained jointly:**

$$\mathcal{L} = \begin{cases} \alpha L_{CTC} + (1-\alpha) L_{CE} & T' \geq T \\ L_{CTC} & T' < T \end{cases}$$

- **T' and T are the predicted and target sequence length, α is the weight of joint , L_{CE} is cross entropy loss.**
- **This method of training guarantee to generate the correct sequence since we take into account the case where the predicted length is larger or smaller than target length.**
- **During inference the model select the token with the highest probability at each position.**
- **The authors also include a transformer based language model into the decoding to enhance the results.**

Results and Experiments

- **The authors of Listen and Fill in Missing Letter use the Open Source Mandarin speech corpus (AISHELL) and Corpus of Spontaneous Japanese (CSJ) in their experiments.**
- **They avoided using Latin language corpus because of the problem of sequence length.**

System	Dev CER	Test CER	Real Time Factor
Baseline (Transformer)	6.0	6.7	1.44
Easy first(K=1)	6.8	7.6	0.22
Easy first(K=3)	6.4	7.1	0.22
Mask-predict(K=1)	6.8	7.6	0.22
Mask-predict(K=3)	6.4	7.2	0.24

Table: Comparison between baseline Transformer, CMLM with Easy First and Mask Predict on AISHELL Corpus. CER is the Character Error Rate, K is the number of itnerations. [4]

- **The AR transformer delivers better performance but the NAT deliver up to 7 times speed up. For CSJ Corpus they attain almost identical performance difference.**

Mask CTC experiments

- Due to the introduced CTC module, the experiments deliver good performance on Latin language corpus. [6]
- The authors do their experiment on the Wall Street Journal Corpus (WSJ)

System	Iterations	dev93 WER	eval92 WER	RTF
AR CTC	L	14.4	11.3	0.97
MASK CTC ($P_{thres} = 0.0$)	1	16.3	12.9	0.03
Mask CTC	5	15.5	12.2	0.05
Mask CTC	10	15.5	12.1	0.07
Mask CTC	#mask	15.4	12.1	0.13

Table: Word Error Rate (WER) and RTF comparision between different CTC modules on WSJ Corpus. The threshold used is 0.999, and γ 0.3 in the loss function. [6]

- for $P_{thres} = 0$ the output of the greedy CTC are used as final predictions (no masking). #mask mean the at each iteration one masked token is predicted.
- The (AR) CTC Transformer still delivers the best performance, but NAT also deliver reasonable performance with much speed up up to 20-30 times.

Notes on Experiments

- **In the CTC paper experiments on the CSJ corpus are conducted, they deliver good results. However, there was no difference between Mask-CTC and only CTC NAT modules due to the nature of the Japanese characters.**
- **In the Spike-Triggered paper, experiments are conducted on the AISHELL Corpus. They deliver similar performance to the CMLM model, but with up to 40 times speed up.**

Conclusion

- **NAT is a method introduced to speed up the inference process.**
- **An conditional masked language model is introduced to avoid dependency on previous tokens in prediction.**
- **One main issue is identifying sequence length.**
- **Methods such as using CTC loss and spike triggered CTC are introduced to deal with this issue.**
- **The models deliver a noticeable speed up in decoding but reduction in performance.**

Thank you for your attention

George Farah

`kitza@i6.informatik.rwth-aachen.de`

Reference

- "Audio deep learning made simple: Automatic speech recognition (asr), how it works." `https://towardsdatascience.com/audio-deep-learning-made-simple-automatic-speech-recognition-`
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.
- X. Song, Z. Wu, Y. Huang, C. Weng, D. Su, and H. Meng, "Non-autoregressive transformer asr with ctc-enhanced decoder input," 2021.
- N. Chen, S. Watanabe, J. Villalba, P. Zelasko, and N. Dehak, "Listen and fill in the missing letters: Non-autoregressive transformer for speech recognition," *IEEE Signal Processing Letters*, vol. 28, p. 121–125, 2020.
- A. Hannun, "Sequence modeling with ctc." `https://distill.pub/2017/ctc/`, 2017.
- Y. Higuchi, S. Watanabe, N. Chen, T. Ogawa, and T. Kobayashi, "Mask ctc: Non-autoregressive end-to-end asr with ctc and mask predict," 2020.

Z. Tian, J. Yi, J. Tao, Y. Bai, S. Zhang, and Z. Wen, “Spike-triggered non-autoregressive transformer for end-to-end speech recognition,” 2020.