INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Lattice generation in attention-based speech recognition models

Michał Zapotoczny, Piotr Pietrzak, Adrian Łańcucki, Jan Chorowski

University of Wroclaw, Poland
jch@cs.uni.wroc.pl

Abstract

Attention-based neural speech recognition models are frequently decoded with beam search, which produces a tree of hypotheses. In many cases, such as when using external language models, numerous decoding hypotheses need to be considered, requiring large beam sizes during decoding. We demonstrate that it is possible to merge certain nodes in a tree of hypotheses, in order to obtain a decoding lattice, which increases the number of decoding hypotheses without increasing the number of candidates that are scored by the neural network. We propose a convolutional architecture, which facilitates comparing states of the model at different pi The experiments are carried on the Wall Street Journal dataset, where the lattice decoder obtains lower word error rates with smaller beam sizes, than an otherwise similar architecture with regular beam search.

Index Terms: speech recognition, beam search, artificial neural networks, attention-based models, lattice generation, decoding

1. Introduction

Beam search is the dominant decoding algorithm for attention-based speech recognition models. Its computational cost $\mathcal{O}(N)$ grows with the number N of hypotheses considered in parallel. In many cases small beams are sufficient, and increasing the beam size does not lead to further improvements. However, there are certain scenarios in which large beams are desired in order to increase the ambiguity of hypotheses:

- using an external language model requires beams of a few hundred [1],
- storing decodings for use in later stages, e.g. in two-pass decoding when the hypotheses are re-scored with a larger language model, or
- feeding the hypotheses to another system, e.g. a keyword spotter [2] or a translator [3], to name a few.

Beam search generates a tree of decodings, in which the starting symbol is in the root and the rightmost symbols of all the hypotheses are in the leaves. In contrast, classical ASR systems return lattices [4] – directed acylcic graphs (DAGs) in which every path is a valid decoding. A lattice with N edges may contain combinatorially many hypotheses.

In this contribution, we demonstrate how to generate lattices with a sequence-to-sequence (seq2seq) system [5], like the one shown in Figure 1. We show that lattices may yield more accurate decodings with fewer beams, which can be observed in a decreased word error rate (WER). We analyze the effective beam size of our method, which we define as the average number of valid paths through the lattice. In addition, our lattices manifest a lower oracle WER, which may be useful in certain applications such as indexing [2].

2. Background

2.1. Seq2seq ASR Models

Sequence-to-sequence models transduce an input sequence $X = (x_1, \ldots, x_n)$ to an output sequence $Y = (y_1, \ldots, y_m)$, typically with the use of recurrent neural networks (RNNs). An RNN encoder is able to summarize the input sequence into a fixed-size hidden representation, while an RNN decoder generates the output sequence in an autoregressive fashion. The attention mechanism [6, 7] aligns the source sequence to the target one by allowing the decoder to selectively attend to all outputs of the encoder.

Seq2seq speech recognition models are trained on utterance-transcript pairs with the goal of maximizing the probability of observing the correct transcript. The number of input speech frames is much larger than the output symbols (e.g. letters, phonemes). For a new utterance, a set of potential transcripts is scored by the model, and the most probable one is typically chosen as the system's response. Typically, scoring all potential transcripts is unfeasible, and heuristic procedures that consider only a small number of hypotheses, such as beam search, have to be used.

Traditional speech recognition systems employ the token passing algorithm for efficient decoding [4]. During operation a discrete hidden state which captures the dependency on past outputs is maintained. Whenever two decoding hypotheses are represented with the same state, they can be merged and treated as one for subsequent operations. This allows building lattices, which compactly encode combinatorialy many interpretations of an input. Seq2seq systems use recurrent decoders, which compress the history of past outputs into continuous hidden states. Different states of an RNN can never be exactly equal, prohibiting merging of hypotheses. In practice, the token passing algorithm reduces to beam search, producing only a tree of hypotheses. Our model recovers the lattice generation capability by replacing the RNN with a Temporal Convolutional Network, described next, that has a finite and adjustable receptive field length. It makes it possible to exactly compare and merge two hypotheses which differ on sufficiently distant locations.

2.2. Temporal Convolutional Networks

Apart from recurrent models, the class of Convolutional autoregressive models apply 1-D convolutions in the temporal domain [8, 9]. Temporal Convolutional Network (TCN) [10] is an end-to-end neural language model, made of 1-D causal convolutions, which attend only to the past elements of a sequence. It is parametrized with a kernel $F = (f_1, \ldots, f_k)$, where $f_i \in \mathbb{R}^d$, and k is the number of previous elements considered used to make a new prediction. A single application of a kernel to sequence X of d-dimensional vectors is [10]

$$F(s) = (X *_h F)(s) = \sum_{i=0}^{k-1} f_i \circ x_{s-h \cdot i},$$

http://dx.doi.org/10.21437/Interspeech.2019-2667

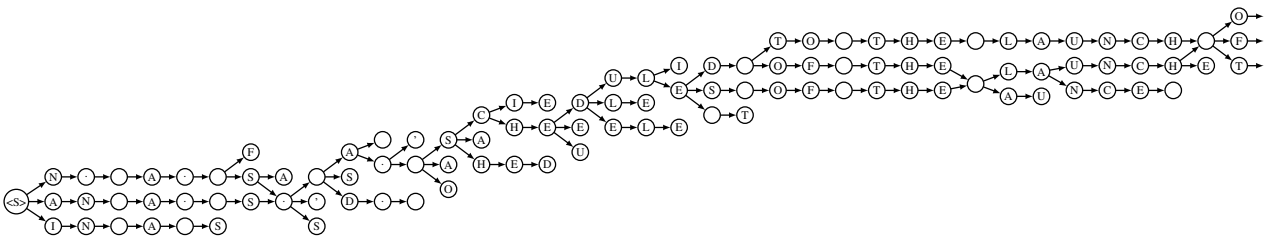

Figure 1: *A sample decoding lattice generated with the proposed algorithm. During beam search, nodes in the hypotheses tree can be heuristically merged, based on the state of the model. The lattice is shown truncated for clarity.*

where $\circ$ denotes element-wise product, and h is the dilation of the convolution.

TCN stacks causal convolutions with increasing dilation and residual connections, padding the input sequence when necessary. Thanks to the dilations, the receptive field of a TCN can grow exponentially with the depth of the convolutional stack. By controlling the depth and the dilations of the network one can precisely set the receptive field of the network, ensuring that sufficiently distant inputs do not influence the output of the model.

3. Using TCNs in Seq2seq Models

We now describe how to employ the TCN decoder in an attention-based seq2seq system. The main changes concern the addition of a location specific attention mechanism. By design, the TCN decoder has a finite receptive field length. This implies that its behavior is the same for two sequences with same suffixes. To disambiguate between two repetitions of the same passage, the attention mechanism has to track its position in the source sequence.

The details of our seq2seq model are as follows. It combines a standard encoder with a TCN decoder endowed with an attention mechanism. Let $H = (h_1, \dots h_N)$ be the output of the encoder. The initial step of every decoding iteration is to calculate the new model state c_t

$$c_t = TCN(emb(\hat{Y}_{t-h}), \dots, emb(\hat{Y}_{t-1})), \tag{1}$$

where emb is the embedding of characters, and $\hat{Y}_t$ is the network's output at time t. Then, attention mechanism is computed using encoded frames h_i, decoder state c_t, and previous attention history $(a_1, \dots, a_{t-1})$ as

$$e_{t,i} = v^T tanh(Uh_i + Vc_t + w_t^i),$$
$$a_t = SoftMax(e_t),$$

for $i = (1 \dots N)$, where $U \in \mathbb{R}^{d_{att} \times d_{enc}}$, and $V \in \mathbb{R}^{d_{att} \times d_{hid}}$.

Every w_t is computed by applying a 1-D convolution with kernel $K \in \mathbb{R}^{d_{att} \times k}$ over the attention history $(a_1, \dots, a_{t-1})$. The kernel is calculated with a linear projection $K = \bar{K} c_t$. Padding with zeros and truncating is applied where necessary. The attention mechanism linearly projects each input frame h_i using a matrix U, then compares it with a linear transformation the last hidden state of the decoder, Vc_t. This allow the model to select all frames containing information relevant at the moment according to TCN.

Since TCN has a limited memory, it cannot discern between repetitions of the same sounds. Location-specific attention, expressed with w_t^i, captures the idea that the attention head should move forward, based on the state of the language model. This motivates applying a convolutional operation to the sequence a_{t-1}, where the kernel of the operation depends on c_t. Due to the monotonicity of the speech/transcript alignment, we assume that the weights can only be moved forward, and by no more than a constant distance k. Lastly, local attention keeps the model from looping on repeated segments (see Section 4.3 for details).

Finally, the attention scores are used to compute glimpses $g_t = \sum_{j=1}^{N} a_{t,j} h_j$, which are then fed to a fully connected layer together with the model state c_t

$$p_{acoustic} = SoftMax(FC(g_t, c_t)), \tag{2}$$

where $p_{acoustic}$ represents a probability distribution over output symbols.

3.1. Beam search decoding

A beam search decoder with beam size b maintains a tree of b hypotheses of length $t-1$ at any given time t. It extends each of them with every possible symbol from the alphabet Σ, generating $b \times |\Sigma|$ new hypotheses of length t. The most probable b hypotheses are retained, and the rest is discarded. Upon encountering the end-of-sequence token, a hypothesis is moved to the final set of hypotheses.

Transcripts are scored by multiplying probabilities of their symbols, and decoders tend to favor shorter paths as a side effect. In models equipped with the attention mechanism, longer transcripts can be promoted with an additional coverage score [1]. Let $A \in \mathbb{R}^{m \times n}$ be the matrix of attention scores, with its element $a_{t,i}$ denoting attention score for the ith encoder output at time t. The coverage counts the number of frames which received a cumulative attention higher than τ, and promotes transcripts that explain many frames of the input

$$cov(A) = |\{a_{t,i} \,:\, a_{t,i} > \tau\}|.$$

Moreover, we employ an external language model, yielding the following decoding score of an candidate output y:

$$s(y) = -\log p_{\text{acoustic}}(y|x) - w_{\text{l}} \log p_{\text{l}}(y) + w_{\text{cov}}\, cov(A),$$

where $w_{lm}, w_{cov} \in \mathbb{R}$ are the weights of the external language model and attention coverage, and p_{lm} is the probability assigned by the language model.

3.2. Generating Lattices by Merging Hypotheses

Algorithm 1 illustrates our lattice construction routine. It is similar to beam search with the main difference that, for every processed node, it checks whether it can be contracted with any of the existing nodes in the lattice. Through these contractions, the lattice structure emerges.

Algorithm 1 Lattice generation

for $t \in 1, \dots N-1$ **do**
  hypotheses$[t] \leftarrow$ NEXT_LETTER(hypotheses$[t-1]$)
  hypotheses$[t] \leftarrow$ TOP_K(hypotheses[t])
  for $b \in$ hypotheses$[t]$, $l \in$ lattice **do**
    if CONTRACTIBLE(b, l) **then**
      if SCORE$(b) <$ SCORE(l) **then**
        hypotheses $\leftarrow$ hypotheses $- b$
      else
        lattice $\leftarrow$ lattice $+ b - l -$ DESCENDANTS(l)
    else
      lattice $\leftarrow$ lattice $+ b$

The state of our model during decoding is a tuple (c_t, a_t, l_t) of states of the TCN, the attention mechanism (i.e., the frames selected during the previous step), and the external language model, respectively. The function CONTRACTIBLE in Algorithm 1 implements the conditions for contracting two nodes. Simply put, it requires the model states in both nodes to be equivalent, which means that the states of TCN, attention and the language model all have to be equivalent.

The states of the language model and the TCN are discrete. They depend on the recent emission history and can be compared precisely. Should we use an RNN in place of the TCN, the hidden states would have to be matched approximately with a similarity function.

The state of the attention mechanism consists of the weights assigned by the attention mechanism to the frames of the utterance at the previous timestep. We define the similarity between two sets of attention weights a and a' as

$$sim(a, a') = \sum_{i=0}^{N-1} min(a_i, a'_i).$$

Because $\sum a = 1$, it follows that $sim(a, a') \in [0, 1]$. Two sets of attention weights are equivalent if $sim(a, a') > 0.8$.

4. Experiments

We carry out our experiments on the Wall Street Journal (WSJ) dataset, with a typical split into *Si284*, *Dev93* and *Eval92* train, dev and test sets. During decoding, we use the extended WSJ trigram language model [11]. We train our models using Adam optimizer [12] with learning rate 0.0004. To combat overfitting, we apply regularization by injecting Gaussian weight noise with $\sigma = 0.2$ for the encoder and $\sigma = 0.02$ for the decoder.

The encoder is composed of a stack of 2-D convolutional layers with kernel size 7x7, 32 features, hard *tanh* activation over the range $[0, 20]$ and Batch Normalization [13]. The strides are 1×2 and 3×1, respectively. Convolutional layers are followed by 4 BiLSTM layers with 320 units in every direction, outputs of which are added.

The decoder uses TCN with hidden state of size 384, kernel size $k = 3$, and 2 levels of dilation with $h = (1, 2)$ which results in a receptive field length of 7 characters. The attention has hidden size 64 and uses kernel of size 32. Through preliminary experiments, we have set the weights of the coverage and language model to $w_{\text{cov}} = 0.8$, $\tau = 0.25$, and $w_{\text{l}} = 0.75$. Lastly, local attention masking was set to $[-10, 50]$.

We provide the source code of our implementation at `https://github.com/chorowski-lab/pytorch-asr`.

Table 1: *Decoding WER (%) on the WSJ dataset with the standard beam search (Beam), and the proposed algorithm (Lattice). Both models use the extended trigram language model [11].*

	Eval92		*Dev93*	
Beam size	*Beam*	*Lattice*	*Beam*	*Lattice*
1	14.20	14.20	17.87	17.87
2	10.42	10.12	13.71	13.44
5	8.29	7.83	11.83	11.67
10	7.55	7.44	11.23	11.02
20	7.39	7.16	10.64	10.51
35	7.20	7.05	10.44	10.19
50	7.14	7.00	10.34	10.13
70	7.16	7.12	10.34	10.12
100	7.16	7.14	10.21	10.09
150	7.16	6.98	10.09	10.12

4.1. Decoding with Generated Lattices

We first compare out lattice decoder with the standard beam search decoder, using the same underlying model trained on the WSJ corpus. The results on the *Dev93* set are shown in Table 1, and as WER curves in Figure 2. Our lattice decoder shows similar performance to the beam search decoder with a $3\times$ smaller beam (50 vs 150). In the tested setting, it yields lower error rates for beam sizes ranging from 20 up to 150. Both methods are equivalent for beam size 1, hence no noticeable gains are visible for small beam sizes. Conversely, wide beams do not bring any improvements, as they lead to collecting less favorable hypotheses.

4.2. Decoding with an Oracle

In order to estimate how thorough a given set of hypotheses is with respect to the ground truth transcript, we search for the closest hypothesis to the ground truth in a given decoding lattice or a tree. For ordinary and lattice decoders, we score similarity of their hypotheses to the ground truth with WER, and report the average number of hypotheses generated for a given beam

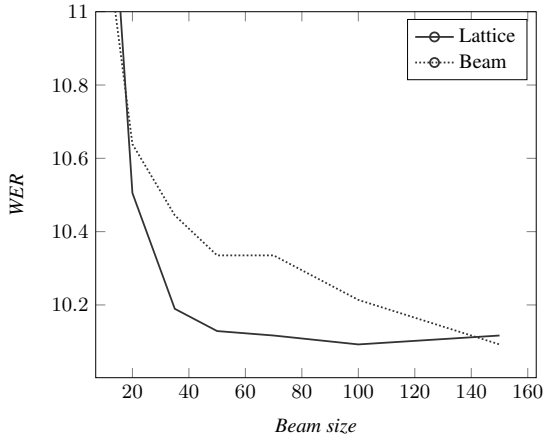

Figure 2: *WER curve (evaluated on Dev93 WSJ subset) when decoding with different beam sizes*

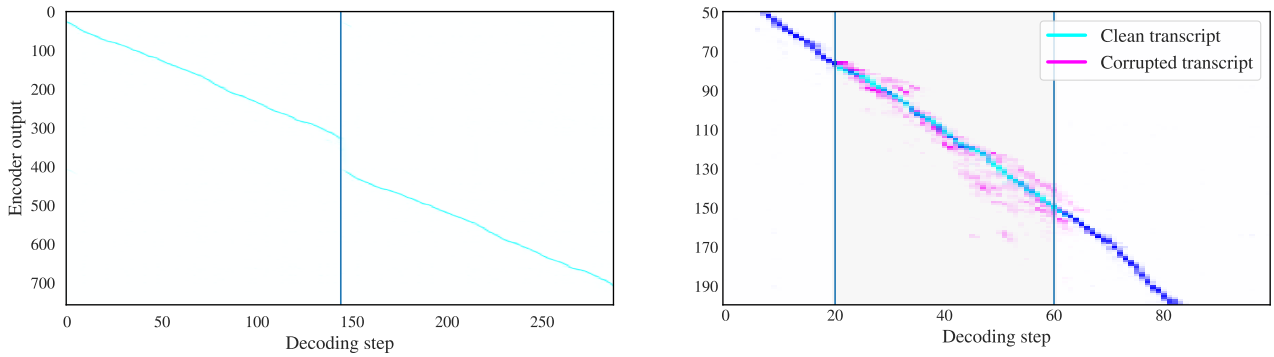

Figure 3: *Attention weights during decoding.* ***Left:*** *A transcript concatenated with itself. Thanks to the local attention component, the model is able to differentiate between identical segments located at different points in time.* ***Right:*** *Attention weights for two transcripts, one of which has a randomly perturbed segment (in gray). Weights align perfectly, creating an opportunity for contraction of hypotheses.*

size (Table 2).

Table 2: *Oracle WER computed over hypotheses found with beam search and lattice search. Lattices encode combinatorially many hypotheses and often contain a hypothesis which is closer to ground truth than the one found using beam search. The results are averaged over WSJ Eval92 set utterances.*

Beam size	*Beam*	*Lattice*	*# Hyps in lattice*
2	10.8	9.6	5.98
5	7.7	5.8	231.53
10	6.1	4.67	12707.75

4.3. Node Contractions

The TCN-based decoder has a fixed-size receptive field, which limits the number of past characters it can attend to. We found that too large receptive fields inhibit contractions of nodes in the hypotheses tree, while too narrow fields interfere with the language model. However, if two identical strings of characters in the transcript are distant, the model could fail to discern between them and merge their corresponding nodes in the lattice, resulting either in dropping the text between the repetition or in a cycle. We demonstrate that this is prevented by the local attention mechanism.

For a given test utterance-transcript pair (x, y), we have concatenated it with itself creating (xx, yy). Figure 3 (left) shows decoding of this utterance. While decoding different copies, the attention mechanism focuses on the corresponding copy of the input, resulting in low sim score.

To investigate the function which recognizes possible contractions in the lattice, we devise the following experiment. For a given transcript Y from the test set, we randomly perturb its segment $Y_{t_1:t_2}$ which is wider than the receptive field of the TCN. We replace characters in the segment at random to create a corrupted $\tilde{Y}$. We then decode the model using both transcripts, and record its attention weights (Figure 3 right). The attention weights over the perturbed segment are visibly dispersed. The segment is longer than the 7 character long receptive field of the model. However, it is able to recover completely, yielding the exact same attention weights as in the original transcript in a few steps after t_2. Thus, if those were two diverged transcript within a one lattice, they would be contracted.

5. Related Work

A number of deep neural models successfully replace recurrent neural networks with convolutional ones. For instance, [8, 14] construct strong language models. In [9], a full seq2seq model is realized through convolutions. However, it does not include the attention mechanism.

Related to our work is the model described in [15]. It uses a single encoder and two decoders: one trained with Connectionist Temporal Classification [16], and one using attention. The model has the capability of generating lattices using the latter decoder, and composing them with lattices from the former one.

Our work is similar to [17], where the speed-up in beam search is achieved by predicting ahead its future outputs. This technique could potentially be combined with our lattice decoding in order to further improve the results.

6. Conclusions

In this paper we have proposed a decoding algorithm for attention-based ASR models. The approach build on beam search and generates lattices through heuristical node contractions. we have replaced the recurrent network in a seq2seq decoder with a convolutional one whose outputs depend on a fixed size history. This allowed merging hypotheses by contracting nodes during beam search to generate lattices, which increases the actual number of considered decodings. Compared to regular beam search, our method can use smaller beams, while getting similar WER, speeding up the decoding. In addition, the method has a smaller oracle WER, meaning that the best transcripts in the lattice is consistently closer to the ground truth than in the case of regular beam search. This property might be useful when not a single transcript, but a wider set of hypotheses is desired, e.g. in audio indexing or translating systems.

7. Acknowledgments

The authors thank Polish National Science Center for funding under the Sonata 2014/15/D/ST6/04402 grant and to the PLGrid project for computational resources on the Prometheus cluster.

8. References

[1] J. Chorowski and N. Jaitly, "Towards better decoding and language model integration in sequence to sequence models," *arXiv:1612.02695 [cs, stat]*, Dec. 2016, arXiv: 1612.02695.

[2] A. Rosenberg, K. Audhkhasi, A. Sethy, B. Ramabhadran, and M. Picheny, "End-to-end speech recognition and keyword search on low-resource languages," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 5280–5284.

[3] G. Kumar, M. Post, D. Povey, and S. Khudanpur, "Some insights from translating conversational telephone speech," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 3231–3235.

[4] M. Gales and S. Young, "The Application of Hidden Markov Models in Speech Recognition," *Found. Trends Signal Process.*, vol. 1, no. 3, pp. 195–304, Jan. 2007.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112.

[6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.

[7] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-Based Models for Speech Recognition," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 577–585.

[8] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A Convolutional Neural Network for Modelling Sentences," *arXiv:1404.2188 [cs]*, Apr. 2014, arXiv: 1404.2188.

[9] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu, "Neural Machine Translation in Linear Time," *arXiv:1610.10099 [cs]*, Oct. 2016, arXiv: 1610.10099.

[10] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv:1803.01271 [cs]*, Mar. 2018, arXiv: 1803.01271.

[11] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding." in *Proceedings of ASRU*, 2015, pp. 167–174.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[13] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv:1502.03167 [cs]*, Feb. 2015, arXiv: 1502.03167.

[14] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language Modeling with Gated Convolutional Networks," *arXiv:1612.08083 [cs]*, Dec. 2016, arXiv: 1612.08083.

[15] T. Hori, S. Watanabe, and J. Hershey, "Joint CTC/attention decoding for end-to-end speech recognition," Jul. 2017, pp. 518–529.

[16] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 369–376.

[17] M. Stern, N. Shazeer, and J. Uszkoreit, "Blockwise Parallel Decoding for Deep Autoregressive Models," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 10 086–10 095.