JOINT PHONEME-GRAPHEME MODEL FOR END-TO-END SPEECH RECOGNITION

Yotaro Kubo, Michiel Bacchiani

Google
Shibuya Stream, 3–21–3 Shibuya, Shibuya-ku, Tokyo.

ABSTRACT

This paper proposes methods to improve a commonly used end-to-end speech recognition model, Listen-Attend-Spell (LAS). The methods we propose use multi-task learning to improve generalization of the model by leveraging information from multiple labels. The focus in this paper is on multi-task models for simultaneous signal-to-grapheme and signal-to-phoneme conversions while sharing the encoder parameters. Since phonemes are designed to be a precise description of the linguistic aspects of the speech signal, using phoneme recognition as an auxiliary task can help guiding the early stages of training to be more stable. In addition to conventional multi-task learning, we obtain further improvements by introducing a method that can exploit dependencies between labels in different tasks. Specifically, the dependencies between phonemes and grapheme sequences are considered. In conventional multi-task learning these sequences are assumed to be independent. Instead, in this paper, a joint model is proposed based on "iterative refinement" where dependency modeling is achieved by a multi-pass strategy. The proposed method is evaluated on a 28000h corpus of Japanese speech data. Performance of a conventional multi-task approach is contrasted with that of the joint model with iterative refinement.

Index Terms— Automatic speech recognition, Listen-Attend-Spell, multi-task learning, iterative refinement

1. INTRODUCTION

Automatic speech recognition (ASR) technology has been rapidly expanding its application areas thanks to accurate modeling techniques empowered by deep learning. However, complicated building procedures for conventional acoustic and language models are costly and therefore simplicity in system building is now a focus.

End-to-end ASR is a generic term that refers to a series of attempts to represent all the ASR modules in a single neural network. Even though it only involves a single network to represent multiple components, it is shown that such a system is competitive with a system that uses independent components but significantly smaller in size [1]. Hence the benefit of such models is not only simplicity that support easier development, but also savings on model size.

In this paper, we focus on improving a commonly used end-to-end ASR model, Listen-Attend-Spell (LAS) [2]. Among several end-to-end architectures proposed in recent literature, LAS has been shown as a competitive architecture [3].

We aim at improving LAS by introducing modeling techniques that model speech-to-grapheme and speech-to-phoneme conversions jointly. The aim is to exploit the correlation between the grapheme and phoneme sequences to improve generalization and accuracy. Previous work in multi-task modeling has shown improved generalization in end-to-end models [4, 5] as well as in hybrid ASR systems [5].

The other focus in this paper is on advancing multi-task learning by modeling multiple label sequences with a joint-distribution. Previous work on multi-task learning resulted in better representation inside neural networks but it did not model dependencies between tasks. Specifically, previous multi-task learning maximized log-likelihood of labels while assuming independence between the task labels. However, this appears a poor assumption for modeling phoneme and grapheme sequences as they are clearly correlated in human language understanding. For example, the conventional ASR system with hidden Markov models typically employ a lexicon component that explicitly models phoneme and grapheme dependencies. Therefore, modeling these sequences with a joint distribution is important to fully leveraging the information between them.

The rest of the paper is organized as follows. In section 2 we introduce a multi-task model that has two independent decoders. In section 3 we introduce a joint model based on iterative refinement. The experimental setup and results are described in section 4, and section 5 concludes the paper.

2. MULTI-TASK MODEL

Let the training data set $\mathcal{D}$ be a set of input/output pairs as $\mathcal{D} = \{(\mathbf{X}_n, \mathcal{Y}_n) \mid n \in \{1, .., N\}\}$ where $\mathbf{X}_n$ is n-th feature sequence and $\mathcal{Y}_n$ is a set of label sequences corresponding to $\mathbf{X}_n$. $\mathcal{Y}_n$ and consists of several heterogeneous label sequences as $\mathcal{Y}_n = \left\{\boldsymbol{y}_n^{(1)}, \boldsymbol{y}_n^{(2)}, \cdots, \boldsymbol{y}_n^{(K)}\right\}$ where K is a number of annotation types. In our case, the label types are phoneme and grapheme and therefore $K = 2$. Each label type has a fixed token vocabulary $V^{(k)}$ and the label sequences are made up out of tokens from the corresponding vocabularies, i.e. $\boldsymbol{y}^{(k)} \in \left(V^{(k)}\right)^*$. It should be noted that each label sequence $\boldsymbol{y}_k$ may have a different length. Even though the experimental objective in this paper is to model a grapheme and

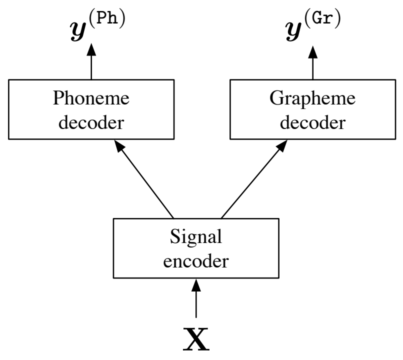
$\boldsymbol{y}^{(\mathtt{Ph})}$
$\boldsymbol{y}^{(\mathtt{Gr})}$
Phoneme decoder
Grapheme decoder
Signal encoder
$\mathbf{X}$

Fig. 1. Block diagram of multi-task sequence-to-sequence model

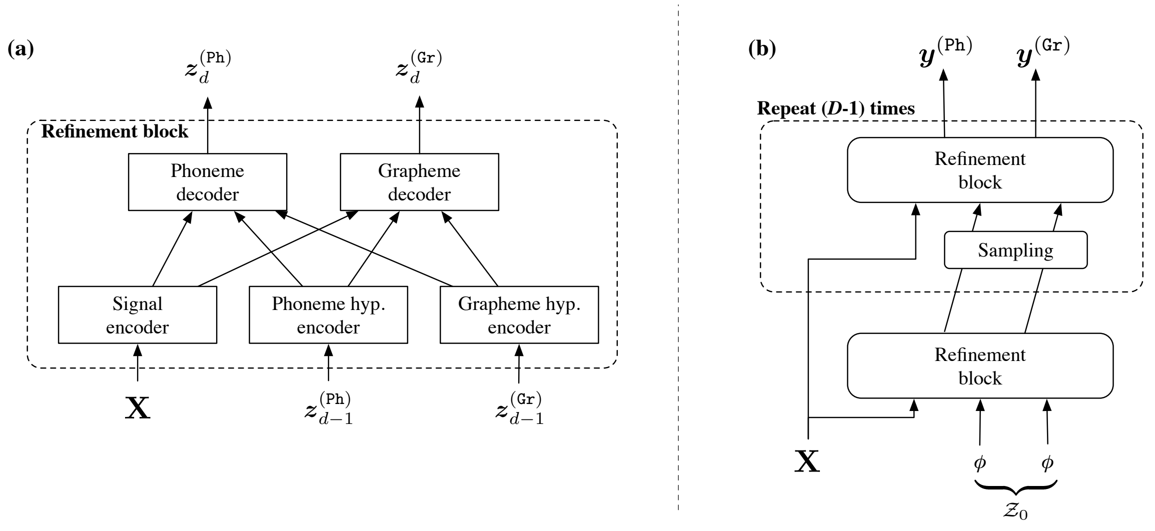

Fig. 2. Block diagram of multi-sequence iterative refinement network. Left: Block diagram of each refinement block, right: block diagram connecting multiple refinement steps.

phoneme sequence jointly, the algorithm presented is more general and could jointly model any number of sequences.

Multi-task training is a well-known method to leverage information from multiple labels. For the sequence-to-sequence encoder-decoder framework, [6] introduced several architectures for multi-task training. In this paper, the "one-to-many" approach is adopted to train a shared encoder with separate, independent phoneme and grapheme decoders. A block diagram for the "one-to-many" encoder-decoder architecture is shown in Fig. 1.

The loss function $L(\Theta)$ for training the multi-task model is defined as a sum of cross-entropy optimization problems as follows [1]:

$$L(\Theta) = -\sum_n \underbrace{\sum_k \log p\left(\boldsymbol{y}_n^{(k)} \mid \mathbf{X}_n, \Theta\right)}_{\simeq \log p(\mathcal{Y}_n \mid \mathbf{X}_n, \Theta)}. \qquad (1)$$

The minimization of this loss function corresponds a maximum-likelihood estimation assuming independence between the output sequences, i.e. the model cannot learn the dependencies between the phoneme and grapheme sequences. However, since there is a shared encoder, the parameters of the encoder are optimized for accuracy of both decoders. Thus, with this training criterion, the encoder is trained for multiple targets even though the decoders model those targets independently.

3. JOINT MODELING VIA ITERATIVE REFINEMENT

Going beyond multi-task models, we adapt the iterative refinement method proposed in [7] for capturing dependencies among heterogeneous target labels.

Iterative refinement is a method that uses noisy hypotheses as a latent variable. Let the latent variable be denoted as $\mathcal{Z}$, then the joint distribution of multiple labels can be defined as a marginal distribution,

$$p(\mathcal{Y}_n \mid \mathbf{X}_n, \Theta) = \sum_{\mathcal{Z}} p(\mathcal{Y}_n \mid \mathcal{Z}, \mathbf{X}_n, \Theta) p(\mathcal{Z} \mid \mathbf{X}_n, \Theta). \qquad (2)$$

Making an independence assumption we define the output probability of the label sequences as a product of independent decoders,

$$p(\mathcal{Y}_n \mid \mathcal{Z}, \mathbf{X}_n, \Theta) \overset{\text{def}}{=} \prod_k p\left(\boldsymbol{y}_n^{(k)} \mid \mathcal{Z}, \mathbf{X}_n, \Theta\right). \qquad (3)$$

Note that even with this independence assumption in the output distribution, the marginal distribution in Eq. (2) still models the joint distribution without an independence assumption through the use of the latent variable $\mathcal{Z}$.

Iterative refinement defines the latent variable to have the same structure as in the target label space. $\mathcal{Z}$ is assumed to be a set of sequences as $\mathcal{Z} \overset{\text{def}}{=} \left\{\boldsymbol{z}^{(k)} \mid k \in \{1,..,K\}\right\}$, and each $\boldsymbol{z}^{(k)}$ is a sequence of output tokens $\boldsymbol{z}^{(k)} \in \left(V^{(k)}\right)^*$. In our case the latent variables are the pair of phoneme and grapheme sequences.

Fig. 2 shows a block diagram for our application of iterative refinement method. Since the latent variables introduced are variable length sequences, two additional encoders are used for representing Eq. (3) in an encoder-decoder architecture. As shown in Fig 2-(a), our encoder-decoder architecture has three encoders for handling three kinds of inputs: the acoustic feature sequence $\mathbf{X}_n$ and the latent variables $\mathcal{Z}$ for phoneme and grapheme sequences. The acoustic feature sequences are provided to the "Signal encoder" block in the figure, and the phoneme and grapheme parts of the latent variables are provided to the "Phoneme hyp. encoder" and "Grapheme hyp. encoder" blocks in the figure, respectively. Since this encoder-decoder neural net can be viewed as performing refinement of the previous phoneme and grapheme hypotheses represented in $\mathcal{Z}$, this part is called a "refinement block."

A similar encoder-decoder architecture can also be used as a definition of a prior probability $p(\mathcal{Z} \mid \mathbf{X})$ of the latent variable. Similar to the definition of $p(\mathcal{Y}_n \mid \mathcal{Z}, \mathbf{X})$, the prior distribution of $\mathcal{Z}$ can

[1]Task weights can also be introduced to the loss function for representing relative importance of the tasks. However, in this paper, we omit those from our formulation for simplicity. The discussions here can also be applied to the weighted loss functions.

also be defined using independent decoders, as follows:

$$p(\mathcal{Z} \mid \mathbf{X}_n, \Theta) \stackrel{\text{def}}{=} \prod_k p(\boldsymbol{z}^{(k)} \mid \mathbf{X}_n, \Theta). \quad (4)$$

This probability function can also be implemented by using the encoder-decoder network with the input $\mathbf{X}_n$. It should be noted again that even with the independence assumption of the prior probabilities (Eq. (4)) and the output probability (Eq. (3)), the marginal distribution (Eq. (2) will model the joint.

Since encoders with empty input sequences do not provide any additional information to the decoders, the prior probability (Eq. (4)) can be viewed as another refinement block where each hypothesis encoder gets an empty sequence (denoted as ϕ) as an input. Fig 2-(b) shows the block diagram of the neural net that provides the samples from the prior distribution to the output distribution both represented as refinement blocks.

Generalizing the idea of refinement to have multiple latent variables $\mathcal{Z}_d$ where $d \in \{1, .., D\}$, deep iterative refinement can be implemented. Extending Eq (2), the marginal probability of deep iterative refinement can be defined as follows:

$$p(\mathcal{Y}_n \mid \mathbf{X}_n) \stackrel{\text{def}}{=} \sum_{\mathcal{Z}_1 \cdots \mathcal{Z}_D} p(\mathcal{Z}_1 \mid \mathbf{X}_n, \Theta) \prod_{d=2}^{D-1} p(\mathcal{Z}_d \mid \mathbf{X}_n, \mathcal{Z}_{d-1}, \Theta) \times p(\mathcal{Y}_n \mid \mathcal{Z}_{D-1}, \mathbf{X}, \Theta). \quad (5)$$

Similar to the shallow version of iterative refinement, each latent variable generation can be modeled using an encoder-decoder architecture. Since the same architecture is used for all the refinement blocks in the system, parameters can be tied among refinement blocks. In Fig. 2, it is shown that the first refinement block handles the empty hypotheses ϕ, and the refinement process is repeated $(D-1)$ times with performing sampling of each hypotheses.

Unlike typical methods for training latent variable model, training of iterative refinement models is performed with supervision over the latent variable. This only applies because the latent variables are designed to have the same structure as the output variables, and the model assumes that the latent variables are distributed around the correct label $\mathcal{Y}_n$. In our specific case, the latent variables are designed to be hypotheses of a phoneme and grapheme sequence for which we have supervised label information. With supervised label information, every refinement block is trained to minimize the cross-entropy between the correct label distribution and the model prediction.

The loss function L for training can therefore be expressed as

$$L(\Theta) = \sum_n \ell(\mathbf{X}_n, \mathcal{Y}_n; \Theta)$$
$$\ell(\mathbf{X}_n, \mathcal{Y}_n; \Theta) = -\sum_d \log p(\mathcal{Y}_n \mid \mathbf{X}_n, \mathcal{Z}'_{d-1}, \Theta), \quad (6)$$

where $\mathcal{Z}'_{d-1}$ is a sample drawn from the previous refinement step. i.e.

$$\mathcal{Z}'_d \sim p(\mathcal{Z}_d \mid \mathbf{X}_n, \mathcal{Z}'_{d-1}, \Theta), \quad (7)$$

where the first latent variable $\mathcal{Z}'_0$ is defined as a set of empty sequences. This loss function corresponds to performing cross-entropy training of each refinement block using the sampled latent variable as the input hypotheses sequences.

With this definition of the loss function, the training process can be implemented by adopting the conventional training method for the LAS model where the input hypotheses $\mathcal{Z}_d$ are generated sequentially by running feed-forward of the network with sampling. However, even though the hypotheses are sampled depending on the output of the previous refinement blocks, the gradients do not propagate to the previous blocks. This is consistent since the model assumes the latent variables to be visible in the training phase.

In decoding, in order to reduce estimation noise, one best hypotheses obtained by beam search are used as inputs of the succeeding refinement block, instead of sampled hypotheses. In this regard, the proposed method can be considered as a multi-pass decoding techniques, and therefore, it can be seen as a variant of Deliberation Networks [8]. It should also be noted that our iterative refinement model here uses auto-regressive recurrent neural net (RNN) decoders whereas the original work on iterative refinement [7] aimed at using non-auto-regressive models. Unlike the original paper where iterative refinement is introduced to capture the mutual dependency between sequence elements, this paper introduce it for capturing the mutual dependency between different sequences in multi-task training.

4. EXPERIMENTS

To assess the effectiveness of independent, multi-task and joint modeling through iterative refinement, we conducted experiments on a Japanese data set. Japanese has a complex relationship between the graphemic and phonetic representations and hence is expected to benefit from a joint modeling approach.

The training data we used consists of 12 million anonymized utterances with a total duration of 28000 hours. For each utterance, 128-dimensional Mel-filterbank outputs are computed from a 32ms window with a 10ms frame shift. The network input signal $\mathbf{X}$ is 512 dimensional and obtained by stacking 4 consecutive Mel-filterbank frames. The graphemic sequences were obtained from a manual transcription. The corresponding phonemic strings where obtained from a forced alignment with the Google hybrid HMM-based system. The vocabulary sizes $V^{(k)}$ for the grapheme and phonetic sequences were 4976 and 34 respectively. The development and evaluation data sets consist of 50 thousand anonymized utterances. The token error rates of the development set were used to optimize hyper parameters.

For all experiments, the encoder consisted of 5 long short-term memory (LSTM) [9] layers each with 1024 cells. The decoders consisted of 2 LSTM layers also with 1024 cells each. In addition, the iterative refinement models used 2 layer LSTM encoders with 1024 cells for re-encoding grapheme and phoneme hypotheses. All models used additive attention as described in [10]. For iterative refinement models, we computed three different attentions for the signal, phoneme and grapheme sequences, each with different parameters. The computed attention vectors were stacked to form 3072 dimensional context vector provided to the subsequent decoders.

For optimization, we used Adam [11] with learning rate $\alpha = 0.0001$ and the smoothing parameters $\beta_1 = 0.9, \beta_2 = 0.999$. The parameter trajectory obtained from the stochastic optimization is smoothed with exponential moving averaging (EMA) [12] with decaying factor 0.999. Training used synchronous updates from 64 tensor processing units (TPUs) and used a minibatch size of 1024.

4.1. Model accuracy comparison

Table 1 shows the token error rates of the various systems. The first two rows correspond to the single task models, i.e. the error rates in the Grapheme and Phoneme columns were computed by separate models without any shared parameters. The last three rows show the results of multi-task and joint models with iterative refinement. It

Table 1. Comparison between single task models, multi-task model, and joint model. The numbers in the parentheses are relative error reduction from the baseline (separate model) results.

	Token error rates [%]	
Models	**Grapheme**	**Phoneme**
Separate model	18.6 (–)	8.8 (–)
Separate model + IR ($D = 2$)	18.6 (+0.0%)	9.6 (+9.1%)
Multi-task	17.8 (-4.3%)	8.4 (-4.5%)
Joint model with IR ($D = 2$)	17.2 (-7.5%)	8.0 (-9.1%)
Joint model with IR ($D = 3$)	16.6 (-10.8%)	8.3 (-5.7%)

should be noted that a multi-task model is equivalent to an iterative refinement model with $D = 1$.

First, we discuss the effects of iterative refinement for our tasks by applying it to the single task models. For both grapheme and phoneme tasks, no significant improvements were obtained by applying iterative refinement. Since the main advantage of iterative refinement is to enable joint modeling, the results were unsurprising since auto-regressive decoders are able to model a joint distributions of label sequences. On the other hand, it was also expected that a refinement strategy could be beneficial to alleviate the effect of using a left-to-right auto regressive decoder. However, the results show that the effect of the left-to-right assumption on results is negligible.

Second, we discuss the advantages of constructing multi-task or joint models. Comparing "multi-task" and "separate" models, one can observe improvement for both the main (grapheme) and auxiliary (phoneme) tasks. Considering that phonemes are designed to represent the linguistic information in the speech signal, the auxiliary phoneme recognition supports the signal representation in the encoder optimization. Thus, the use of a phoneme auxiliary recognition task was shown to be effective for LAS model training.

Joint modeling via iterative refinement further reduced the error rates for the both tasks. This gap suggests that the conventional multi-task learning with the task independence assumption might not be optimal especially for the tasks where the label sequences are highly correlated as can be observed in the error rate reduction of the phoneme recognition task. This suggested that the phoneme sequences are also dependent on the grapheme sequences even though the conventional ASR models typically only model a uni-directional dependency from phoneme to grapheme using the pronunciation lexicon. It was also observed that the deeper refinement models were more capable in representing complicated dependencies between tasks, and it resulted in a lower error rate of both the main and auxiliary tasks. On the other hand, training of deeper refinement models is time consuming due to the fact that the computational cost for gradient computation scales proportionally with increasing depth D. In the next section, we discuss on the possibility of using smaller D for training.

4.2. Refinement with adaptive depth

We further investigated on the number of refinement steps in the decoding phase. It is beneficial if the depth D could be set independently in the decoding phase depending on the computational environment. Table 2 shows the token error rates obtained by varying D in the decoding phase for a model that was trained with $D = 2$.

The first row of the table 2 shows the results when the iterative refinement model ($D = 2$) was used as a multi-task model ($D = 1$) in decoding. This is equivalent to using the first intermediate hypotheses of the iterative refinement model as the final output. We observe a slight degradation in the token error rates compared to "multi-task" in Table 1. This might be due to the negative effect of parameter sharing. The first refinement block is special since it performs speech recognition without any previous hypotheses, parameter sharing might reduce the effective number of parameters used for the first step. This might be mitigated by separating the parameters for the first stage from the parameters for the subsequent stages. The effectiveness of parameter untying will be addressed in future work.

Table 2. Token error rates varying the run-time number of refinement in the model trained with $D = 2$.

	Token error rates [%]	
Depth in decoding	**Grapheme**	**Phoneme**
$D = 1$	18.0	8.8
$D = 2$	17.2	8.0
$D = 3$	16.8	7.9
$D = 4$	16.8	8.0

The third and fourth rows of table 2 show the effect of running the refinement process repeatedly, i.e. running more refinement stages than were used in training. We confirm that it is beneficial to repeat the refinement process.

Comparing Table 2 with Table 1, the effect of using mismatched condition in decoding is assessed. Overall, the use of different depth D in decoding have a negative effect. For example, comparing $D = 3$ rows in Table 2 and Table 1, we confirm slight degradation in the main (grapheme) task. Even though the use of mismatched depth might not show gains, the flexibility in allocating computational resources as a trade-off is beneficial. If computational resources are limited, this model can fallback to a shallower model ($D = 1$) at some accuracy loss, or it can allocate more computational resources for an accuracy gain without retraining of the models.

5. CONCLUSION

In this paper, we confirmed that joint modeling of phoneme and grapheme sequences is beneficial for improving the performance of attention-based end-to-end speech recognition. Overall, the experimental results show that the performance improvements originate from two factors: One is by robust encoder obtained by multi-task training consisting of speech-to-phoneme and speech-to-grapheme conversion tasks. The other is a mutually dependent, joint decoder model obtained by the iterative refinement technique. The first factor was confirmed by comparing the multi-task model with the single task models. From Table 1, we observed 4.3% of relative error reduction from our multi-task model compared to the performance of the single task models. The second factor was confirmed by comparing the iterative refinement models with the multi-task models. The error reduction rate for the iterative refinement models was 10.8% compared to the single task models. This corresponds to 6.7% error reduction from the multi-task models that models phoneme and grapheme decoders independently. In addition, we showed the computation and accuracy trade-off of changing the number of refinement stages.

Future work will focus on training of iterative refinement models that handle the latent variable without explicit supervision, or with weak-supervision. The current algorithm is derived by assuming the latent variable to be hypotheses of each label sequence which might be too restrictive. In addition it seems worth to extend the proposed method to different end-to-end architectures, such as RNN transducers.

6. REFERENCES

[1] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al., "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[2] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 4960–4964.

[3] Rohit Prabhavalkar, Kanishka Rao, Tara N Sainath, Bo Li, Leif Johnson, and Navdeep Jaitly, "A comparison of sequence-to-sequence models for speech recognition.," in *Interspeech*, 2017, pp. 939–943.

[4] Rich Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[5] D. Chen, B. Mak, C. Leung, and S. Sivadas, "Joint acoustic modeling of triphones and trigraphemes by multi-task learning deep neural networks for low-resource speech recognition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 5592–5596.

[6] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser, "Multi-task sequence to sequence learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[7] Jason Lee, Elman Mansimov, and Kyunghyun Cho, "Deterministic non-autoregressive neural sequence modeling by iterative refinement," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, Oct.-Nov. 2018, pp. 1173–1182, Association for Computational Linguistics.

[8] Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu, "Deliberation networks: Sequence generation beyond one-pass decoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1784–1794.

[9] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[11] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[12] Wei Xu, "Towards optimal one pass large scale learning with averaged stochastic gradient descent," *arXiv preprint arXiv:1107.2490*, 2011.