CONVOLUTIONAL SEQUENCE TO SEQUENCE MODEL WITH NON-SEQUENTIAL GREEDY DECODING FOR GRAPHEME TO PHONEME CONVERSION

Moon-jung Chae[⋆]*, Kyubyong Park*[†]*, Jinhyun Bang*[⋆]*, Soobin Suh*[⋆]
Jonghyuk Park[⋆]*, Namju Kim*[†]*, Jonghun Park*[⋆]

[⋆]Department of Industrial Engineering & Center for Superintelligence, Seoul National University
[†]Kakao Brain Corporation

ABSTRACT

The greedy decoding method used in the conventional sequence-to-sequence models is prone to producing a model with a compounding of errors, mainly because it makes inferences in a fixed order, regardless of whether or not the model's previous guesses are correct. We propose a non-sequential greedy decoding method that generalizes the greedy decoding schemes proposed in the past. The proposed method determines not only which token to consider, but also which position in the output sequence to infer at each inference step. Specifically, it allows the model to consider easy parts first, helping the model infer hard parts more easily later by providing more information. We study a grapheme-to-phoneme conversion task with a fully convolutional encoder-decoder model that embeds the proposed decoding method. Experiment results show that our model shows better performance than that of the state-of-the-art model in terms of both phoneme error rate and word error rate.

Index Terms— Grapheme to phoneme conversion, Sequence to sequence model, Encoder-decoder model, Convolutional neural network, Decoding algorithm

1. INTRODUCTION

Grapheme-to-phoneme (G2P) conversion is a task of translating from letters (grapheme sequence) to their pronunciations (phoneme sequence). While G2P conversion can be done with dictionaries, the number of words they contain is limited, and new words cannot be converted by use of dictionaries [1]. For this reason, G2P models which generate proper pronunciations based on data-driven approach have been frequently employed in text-to-speech (TTS) and automatic speech recognition (ASR) systems [2, 3].

What makes G2P conversion challenging is that it does not always follow systematic rules, and similar graphemes can be pronounced differently [4]. Earlier work suggested joint sequence models for G2P conversion [5, 6, 7, 8], but these approaches require alignment between grapheme and phoneme sequences, which may not always be straightforward. With the introduction of the encoder-decoder model [9, 10] using recurrent neural networks (RNNs) [11] and attention mechanisms [12, 13] for machine translation, [14] applied them to the G2P task and achieved the state-of-the-art G2P performance without explicit alignments. More recently, encoder-decoder models using convolutional layers have been studied [15, 16, 17], and in particular [18] proposed a fully convolutional translation model using causal convolution for decoding, but these approaches have not been applied to G2P problem yet.

This work was supported by Kakao and Kakao Brain corporations.

Most encoder-decoder models proposed in the past carry out decoding sequentially, one step at a time from left to right, and use the outputs from the previous steps as decoder inputs. In some cases, however, the beginning of the sequence may be the most difficult part to infer, and inaccurate inference in the beginning can negatively affect what follows, which may result in serious compounding errors. On the other hand, if the decoding order is not fixed and a model starts decoding from the easiest part of the sequence to infer, then the model can utilize input with more information when it decodes difficult parts. For example, when a person plays a crossword puzzle, it is a good strategy to fill in the easiest blank first, referring to the hints given so far. According to this greedy strategy, the easy parts filled in earlier can be used as hints later.

A typical decoding method for sequence-to-sequence model is greedy decoding in which the token with the highest conditional probability at each step is selected from left to right. In order to improve performance, beam search that keeps several most likely hypotheses has been commonly used despite increased computational complexity. Unfortunately, while there is room for improvement in the decoding methods, there have only been a few studies focusing on them [19, 20]. [19] presented modified beam search algorithm which is more explorative than the typical ones. [20] proposed a decoding algorithm that trains a model to maximize a target decoding objective such as word error rate (WER) or BLEU using a deterministic policy gradient method. However, the proposed decoding algorithms in [19, 20] cannot allow flexible decoding order, and therefore cannot be free from compounding error problem.

Motivated by the above remarks, a non-sequential greedy decoding (NSGD) method proposed in this paper is a generalized version of the greedy decoding. Our method considers not only which token to select next but also which position to consider at each inference step. Since NSGD operates non-sequentially, convolutional neural networks (CNN) [21] appear to be more suitable for NSGD than RNN which has a serial structure. Inspired by the model structure of [18], we attempt to combine fully convolutional encoder-decoder model with NSGD. To the best of our knowledge, our work is the first study that applies fully convolutional encoder-decoder model on the G2P task and proposes a novel decoding algorithm by extending traditional greedy decoding for sequence-to-sequence encoder-decoder model.

In this paper, we present a new convolutional encoder-decoder model with NSGD. We verify the effectiveness of our model on G2P conversion task compared to the baseline models including the state-of-the-art model.

The rest of the paper is organized as follows: Section 2 describes a fully convolutional encoder-decoder model applying NSGD method and presents its inference and training procedures.

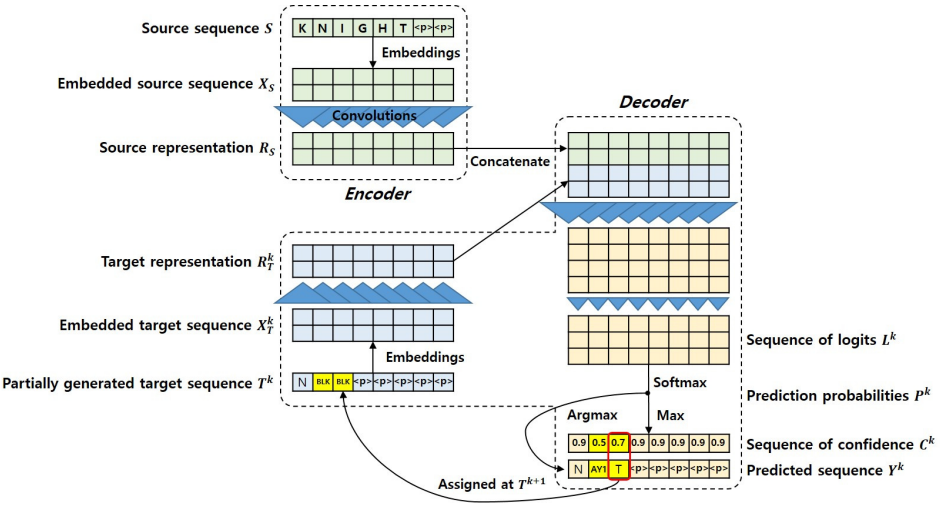

Fig. 1. The proposed convolutional encoder-decoder model with NSGD during test.

We show experimental results on G2P conversion task in Section 3. Section 4 presents conclusion and future work.

2. PROPOSED MODEL

2.1. Model Structure

Our convolutional encoder-decoder model uses two inputs, source sequence $S = (s_1, \ldots, s_N)$ and target sequence $T = (t_1, \ldots, t_N)$ of N tokens each, where subscript $n \in \{1, \ldots, N\}$ represents an index in a sequence. For clarity, we assume that the maximum lengths of source and target sequences are same. The source vocabulary set V_S and the target vocabulary set V_T contain candidate tokens s and t, respectively.

The encoder network consists of several number of residual blocks [22]. Each block consists of two rectifier linear units (ReLU) and convolutional layers. It also contains dropout [23] and normalization [24] layers as regularizers. Using embedded source sequence $X_S \in \mathbb{R}^{N \times d}$ generated from S where d is the embedding size, the encoder outputs source representation $R_S \in \mathbb{R}^{N \times d}$ through the residual blocks.

Using embedded target sequence $X_T \in \mathbb{R}^{N \times d}$ generated from T, the decoder network also builds target representation R_T with residual blocks of which structures are exactly same as those of the residual blocks in the encoder. The decoder then concatenates and decodes two representations R_S and R_T using residual blocks. Subsequently, a single convolutional layer with width-1 filter projects the representation to the sequence of logits $L = (L_1, \ldots, L_N) \in \mathbb{R}^{N \times |V_T|}$. Stacking a softmax layer on top of the network, the model finally outputs prediction probabilities $P = (P_1, \ldots, P_N)$ where $P_n = softmax(L_n)$.

2.2. Inference with Non-Sequential Greedy Decoding

With NSGD, the model iteratively infers the most likely part among the candidate positions of T that are not inferred yet. Superscript k denotes the number of positions of T predicted by the model so far. Specifically, we define T^k as the partially generated target sequence with the k positions already predicted by the model and the rest $N - k$ positions of blank tokens. The blank token, one of the elements in V_T, does not correspond to any phoneme, and is used to indicate that the positions filled with these tokens in T^k should be replaced by the model's predictions. We also define $I^k \subset \{1, \ldots, N\}$, where $|I^k| = k$, as the set containing indices of k positions predicted by the model so far. NSGD initially starts with T^0, and updates T^k to T^{k+1} until the fully generated target sequence T^N is obtained.

Algorithm 1 Inference procedure with NSGD

Require: sequence-to-sequence model G_θ with weights θ; source sequence S.

Initialize target sequence T^0 with blank tokens and set $I = \emptyset$.
for $k \in \{0, \ldots, N-1\}$ **do**
  $T^{k+1} \leftarrow T^k$.
  Generate P^k using G_θ with inputs S and T^k.
  Compute $Y^k = (y_1^k, \ldots, y_N^k)$ and $C^k = (c_1^k, \ldots, c_N^k)$ from P^k using argmax and max, respectively.
  for $n \in \{1, \ldots, N\}$ **do**
    if $n \in I$ **then**
      $c_n^k \leftarrow 0$
    end if
  end for
  $n^* \leftarrow \text{argmax}_n\,(C^k)$.
  $t_{n^*}^{k+1} \leftarrow y_{n^*}^k$.
  $I \leftarrow I \cup \{n^*\}$.
end for
return T^N.

The details of update procedure from T^k to T^{k+1} are as follows: The model produces P^k from given source sequence S and target sequence T^k. Predicted sequence $Y^k = (y_1^k, \ldots, y_N^k)$ where $y_n^k \in V_T$ and confidence sequence $C^k = (c_1^k, \ldots, c_N^k)$ are computed from P^k where $y_n^k = \text{argmax}_n(P_n^k)$ and $c_n^k = \max_n(P_n^k)$. We define the confidence value of prediction at the n-th position as the maximum value of P_n^k. The model chooses the position index n^* with the highest confidence value among the indices not included in I^k. The only update from T^k to T^{k+1} is replacing the blank token at the n^*-th position with $y_{n^*}^k$. Figure 1 shows an example of inference procedure of our model, generating the phoneme sequence from the grapheme sequence 'KNIGHT'. There are two positions with blank tokens, and the blank token at the third position is replaced with the

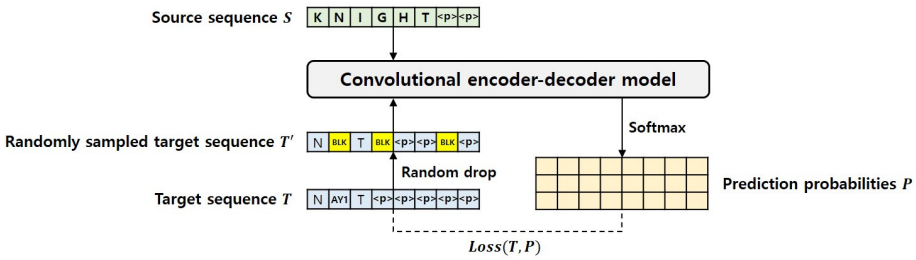

Fig. 2. The proposed convolutional encoder-decoder model with NSGD during training.

phoneme 'T'. Algorithm 1 shows detailed inference procedure with NSGD.

2.3. Training Procedure

Training procedure with NSGD is presented at Algorithm 2. During inference, the model with NSGD faces various cases of T^k. There are 2^N possible cases of T^k when assigning each position with either a blank token or the ground truth token. Since the sequence without blank tokens is not fed to the model during inference, this case can be excluded. During training, we sample an input from these $2^N - 1$ cases and feed it to the model (Figure 2).

We use random sampling function $r : T \rightarrow T'$ which replaces a random subset of ground truth tokens in T with blank tokens. Using r, we generate randomly sampled target sequence $T' = (t'_1, \ldots, t'_N)$ and feed it to the model. r is applied element-wise, i.e., $r(T) = (r_1(t_1), \ldots, r_N(t_N)) = (t'_1, \ldots, t'_N)$ such that:

$$r_n(t_n) = \begin{cases} t_n & \text{if } \varepsilon_n < \tau^2 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $i.i.d.$ random samples τ and ε_n for $n \in \{1, \ldots, N\}$ are drawn from the uniform distribution $u(0, 1)$. τ is sampled at each mini-batch, and ε_n is sampled for each input t_n. Here we assume that the value of a blank token is zero. The model outputs P from given source sequence S and randomly sampled target sequence T' and backpropagates the loss which is defined as follows using cross entropy loss function H:

$$Loss(T, P) = \sum_{n=1}^{N} H(t_n, P_n) \quad (2)$$

Algorithm 2 Training procedure with NSGD

Require: sequence-to-sequence model G_θ with weights θ; source sequence S; target sequence T

1: Initialize G_θ with random weights θ.
2: **while** not converged **do**
3: Generate T' from T with r following (1).
4: Generate P using G_θ with inputs S and T'.
5: Update θ to minimize (2)
6: **end while**

3. EXPERIMENTS

3.1. Dataset

We performed experiments on the latest released version of CMUDict US English dataset (0.7b, released at November 19, 2014) which is publicly available.[1] Recent studies [14, 25] on G2P conversion task also used CMUDict, and they mentioned that they used the partitions provided by the author of earlier work [5]. However, the datasets they used vary in the number of instances and their data preprocessing methods are not open to the public. For this reason, we constructed three versions of datasets to train and validate our model, addressing two preprocessing issues (Table 1). First, in the original CMUDict dataset, 6.5% of words have multiple pronunciations and preserving these words seems more realistic. On the other hand, words with multiple pronunciations given during training might confuse the model. Therefore we made two versions of datasets where one version includes those words, and the other excludes them. Second, we also constructed datasets with stress markings on phonemes, making the task of our study more realistic and difficult than those of previous studies, which removed the stress markings [5, 14, 25]. We partitioned the dataset with a fixed random seed into training, validation and test set, consisting of 85%, 5% and 10% of the preprocessed dataset, respectively. While partitioning the datasets with multiple pronunciations, all pronunciations of each word were assigned to the same partition. Hyperparameters of all models including ours and baseline models were selected using the validation set, not the test set.

Table 1. Preprocessed datasets.

Dataset	Multiple pronunciations	Stress markings	Number of instances
CMUDict-MS	Kept	Kept	133,853
CMUDict-M	Kept	Removed	133,853
CMUDict-S	Removed	Kept	116,919

3.2. Implementation

The proposed model had 15 residual blocks and a convolutional layer for projection. For building representations from source, target and concatenated sequences, 5 residual blocks were used, respectively. The sizes of embedding vectors for source and target sequences were both 256. We used Adam optimizer [26] with initial learning rate of 0.001 and mini-batch size of 256. For every 40 mini-batches, the average loss was calculated, and if it was greater than the maximum value of the previous three average losses, the learning rate was decayed by multiplicative factor of 0.98. For regularization, dropout with a keep probability of 80% was used at each residual block, and weight decay of 0.00001 is used. In order to verify the performance of the proposed model, we considered [12] and the state-of-the-art model for G2P conversion [14] as baselines. The hyperparameters

[1] http://svn.code.sf.net/p/cmusphinx/code/trunk/cmudict/

Table 2. Comparison of error rates on three versions of CMUDict datasets. $\pm$ indicates the standard deviation across 5 training runs of the model. The baseline model marked as * was reported as the state-of-the-art.

Model	CMUDict-MS		CMUDict-M		CMUDict-S	
	PER (%)	WER (%)	PER (%)	WER (%)	PER (%)	WER (%)
Encoder-decoder + attention [12]	8.00 ± 0.11	30.42 ± 0.48	5.91 ± 0.07	25.19 ± 0.20	8.00 ± 0.11	30.42 ± 0.48
Encoder-decoder + attention [14]*	7.63 ± 0.08	28.61 ± 0.37	5.72 ± 0.10	24.77 ± 0.38	7.88 ± 0.16	28.89 ± 0.41
Proposed model	$\mathbf{7.25 \pm 0.07}$	$\mathbf{28.42 \pm 0.22}$	$\mathbf{5.58 \pm 0.04}$	$\mathbf{24.10 \pm 0.19}$	$\mathbf{7.44 \pm 0.06}$	$\mathbf{28.87 \pm 0.26}$

Table 3. Comparison of decoding results of an example word 'entrap'. ○ and ● are symbols representing blank and padding tokens respectivly. Among the phonemes, some vowels include digits representing stress. Grey cells mean incorrect prediction results.

Source of sequence		Acquired phoneme sequence									
Our model with NSGD	$k=1$	○	○	○	○	○	○	○	○	○	●
	$k=2$	○	○	○	○	○	○	○	○	●	●
	$k=3$	○	○	t	○	○	○	○	○	●	●
	$k=4$	○	n	t	○	○	○	○	○	●	●
	$k=5$	○	n	t	r	○	○	○	○	●	●
	$k=6$	○	n	t	r	○	○	○	●	●	●
	$k=7$	○	n	t	r	○	p	○	●	●	●
	$k=8$	○	n	t	r	○	p	●	●	●	●
	$k=9$	ih0	n	t	r	○	p	●	●	●	●
	$k=10$	ih0	n	t	r	ae1	p	●	●	●	●
[14] with sequential greedy decoding		eh1	n	t	r	ah0	p	●	●	●	●
Ground truth		ih0	n	t	r	ae1	p	●	●	●	●

of each baseline model such as the number of layers, the number of RNN hidden units, embedding size, dropout probability and initial learning rate were tuned on the validation set. Each model was trained for 100 epochs and its best model in terms of WER on the validation set was selected. We compared test performance of all selected models, and beam search was not used. The entire code for the proposed model including the decoding method and dataset preprocessing are available online.[2]

3.3. Results

Table 2 presents the performances of our model on the test set, compared to the baselines. The proposed model shows the best performances in terms of both PER (phoneme error rate) and WER compared to the baselines including the state-of-the-art model. Furthermore, the proposed model shows more stable results with lower standard deviations of error rates than the baselines.

The performace of our model in terms of WER slightly decreases in some cases during hyperparameter tuning, but no significant performance degradation is found.

3.4. Decoding example

Table 3 shows the comparison of decoding results of an example word 'entrap'. When NSGD is used, the padding tokens located at the end of the target sequence is inferred first. After that, the consonants that are easier to infer than the vowels with stress are inferred, and then the rest of the paddings are inferred to determine the length of the sequence. It postpones the inference on vowel phonemes which is the most difficult part, and uses all the rest of the phonemes inferred so far as an input. The example demonstrates that utilizing phonemes predicted earlier allows our model with NSGD to infer the vowels with stress more accurately than [14] with sequential greedy decoding.

4. CONCLUSION

In this paper, we proposed a non-sequential greedy decoding method (NSGD) that generalizes traditional greedy decoding and two algorithms for inference and training. We also proposed a fully convolutional encoder-decoder model for NSGD. We were able to show the effectiveness of the proposed model and decoding method by achieving the state-of-the-art performances on the G2P task with various datasets and demonstrating the decoding results.

As future work, we plan to apply NSGD on various sequence-to-sequence domains such as text summarization and machine translation to examine the potential of NSGD. There is also room for improving the training and inference algorithms. It is necessary to modify the model so that parallel computation, which is an advatage of employing the convolutional encoder-decoder models, can be used. Random sampling approach used at the training procedure can be replaced with more effective approaches. Instead of using random drop sampling, we may apply the concept of replay buffer in reinforcement learning to sample inputs that a model is more likely to experience during inference [27].

[2]https://github.com/ctr4si/NSGD_G2P

5. REFERENCES

[1] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4225–4229.

[2] A.E.D. Mousa and B.W. Schuller, "Deep bidirectional long short-term memory recurrent neural networks for grapheme-to-phoneme conversion utilizing complex many-to-many alignments," in *Interspeech*, 2016, pp. 2836–2840.

[3] S. Arik, G. Diamos, A. Gibiansky, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," *arXiv preprint arXiv:1705.08947*, 2017.

[4] D. Alvarez-Melis and T.S. Jaakkola, "A causal framework for explaining the predictions of black-box sequence-to-sequence models," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing(EMNLP)*. 2017, pp. 412–421, Association for Computational Linguistics.

[5] S.F. Chen, "Conditional and joint models for grapheme-to-phoneme conversion.," in *8th European Conference on Speech Communication and Technology (Eurospeech)*, 2003, pp. 2033–2036.

[6] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech communication*, vol. 50, no. 5, pp. 434–451, 2008.

[7] S. Jiampojamarn, G. Kondrak, and T. Sherif, "Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion," in *HLT-NAACL*, 2007, vol. 7, pp. 372–379.

[8] L. Galescu and J. F. Allen, "Pronunciation of proper names with a joint n-gram model for bi-directional grapheme-to-phoneme conversion," in *Seventh International Conference on Spoken Language Processing*, 2002.

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1724–1734, Association for Computational Linguistics.

[10] I. Sutskever, O. Vinyals, and Q.V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*. 2014, pp. 3104–3112, Curran Associates, Inc.

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

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations (ICLR)*, 2015.

[13] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, "Grammar as a foreign language," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 2773–2781.

[14] S. Toshniwal and K. Livescu, "Jointly learning to align and convert graphemes to phonemes with neural attention models," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 76–82.

[15] J. Gehring, M. Auli, D. Grangier, and Y.N. Dauphin, "A convolutional encoder model for neural machine translation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017, pp. 123–135.

[16] J. Bradbury, S. Merity, C. Xiong, and R. Socher, "Quasi-recurrent neural networks," *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[17] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y.N. Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017, pp. 1243–1252.

[18] N. Kalchbrenner, L. Espeholt, K. Simonyan, A.V.D. Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," *arXiv preprint arXiv:1610.10099*, 2016.

[19] J. Li, W. Monroe, and D. Jurafsky, "A simple, fast diverse decoding algorithm for neural generation," *arXiv preprint arXiv:1611.08562*, 2016.

[20] J. Gu, K. Cho, and V.O. Li, "Trainable greedy decoding for neural machine translation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017, pp. 1958–1968.

[21] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[23] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[24] J.L. Ba, J.R. Kiros, and G.E. Hinton, "Layer normalization," in *NIPS 2016 Deep Learning Symposium*. 2016.

[25] K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," in *Interspeech*, 2015, pp. 3330–3334.

[26] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*. 2013.