

# Multitask Learning with Low-Level Auxiliary Tasks for Encoder-Decoder Based Speech Recognition

Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu

Toyota Technological Institute at Chicago

{shtoshni, haotang, llu, klivescu}@ttic.edu

## Abstract

End-to-end training of deep learning-based models allows for implicit learning of intermediate representations based on the final task loss. However, the end-to-end approach ignores the useful domain knowledge encoded in explicit intermediate-level supervision. We hypothesize that using intermediate representations as auxiliary supervision at lower levels of deep networks may be a good way of combining the advantages of end-to-end training and more traditional pipeline approaches. We present experiments on conversational speech recognition where we use lower-level tasks, such as phoneme recognition, in a multitask training approach with an encoder-decoder model for direct character transcription. We compare multiple types of lower-level tasks and analyze the effects of the auxiliary tasks. Our results on the Switchboard corpus show that this approach improves recognition accuracy over a standard encoder-decoder model on the Eval2000 test set.

**Index Terms:** speech recognition, multitask learning, encoder-decoder, CTC, LSTM

## 1. Introduction

Automatic speech recognition (ASR) has historically been addressed with modular approaches, in which multiple parts of the system are trained separately. For example, traditional ASR systems include components like frame classifiers, phonetic acoustic models, lexicons (which may or may not be learned from data), and language models [1]. These components typically correspond to different levels of representation, such as frame-level triphone states, phones, and words. Breaking up the task into such modules makes it easy to train each of them separately, possibly on different data sets, and to study the effect of modifying each component separately.

Over time, ASR research has moved increasingly toward training multiple components of ASR systems jointly. Typically, such approaches involve training initial separate modules, followed by joint fine-tuning using sequence-level losses [2, 3]. Recently, completely integrated end-to-end training approaches, where all parameters are learned jointly using a loss at the final output level, have become viable and popular. End-to-end training is especially natural for deep neural network-based models, where the final loss gradient can be backpropagated through all layers. Typical end-to-end models are based on recurrent neural network (RNN) encoder-decoders [4, 5, 6, 7] or connectionist temporal classification (CTC)-based models [8, 9].

End-to-end training is appealing because it is conceptually simple and allows all model parameters to contribute to the same final goal, and to do so in the context of all other model parameters. End-to-end approaches have also achieved impressive results in ASR [4, 9, 10] as well as other domains [11, 12, 13]. On the other hand, end-to-end training has some drawbacks: Optimization can be challenging; the intermediate learned repre-

sentations are not interpretable, making the system hard to debug; and the approach ignores potentially useful domain-specific information about intermediate representations, as well as existing intermediate levels of supervision.

Prior work on analyzing deep end-to-end models has found that different layers tend to specialize for different sub-tasks, with lower layers focusing on lower-level tasks and higher ones on higher-level tasks. This effect has been found in systems for speech processing [14, 15] as well as computer vision [16, 17].

We propose an approach for deep neural ASR that aims to maintain the advantages of end-to-end approaches, while also including the domain knowledge and intermediate supervision used in modular systems. We use a multitask learning approach that combines the final task loss (in our case, log loss on the output labels) with losses corresponding to lower-level tasks (such as phonetic recognition) applied on lower layers. This approach is intended to encapsulate the intuitive and empirical observation that different layers encode different levels of information, and to encourage this effect more explicitly. In other words, while we want the end-to-end system to take input acoustics and produce output text, we also believe that at some appropriate intermediate layer, the network should do a good job at distinguishing more basic units like states or phones. Similarly, while end-to-end training need not require supervision at intermediate (state/phone) levels, if they are available then our multitask approach can take advantage of them.

We demonstrate this approach on a neural attention-based encoder-decoder character-level ASR model. Our baseline model is inspired by prior work [18, 8, 19, 4, 7], and our lower-level auxiliary tasks are based on phonetic recognition and frame-level state classification. We find that applying an auxiliary loss at an appropriate intermediate layer of the encoder improves performance over the baseline.

## 2. Related Work

Multitask training has been studied extensively in the machine learning literature [21]. Its application to deep neural networks has been successful in a variety of settings in speech and language processing [22, 23, 24, 25, 26, 27]. Most prior work combines multiple losses applied at the final output layer of the model, such as joint Mandarin character and phonetic recognition in [26] and joint CTC and attention-based training for English ASR [25]. Our work differs from this prior work in that our losses relate to different types of supervision and are applied different levels of the model.

The idea of using low-level supervision at lower levels was, to our knowledge, first introduced by Søgaard & Goldberg [28] for natural language processing tasks, and has since been extended by [29]. The closest work to ours is the approach of Rao and Sak [30] using phoneme labels for training a multi-accent CTC-based ASR system in a multitask setting. Here we study

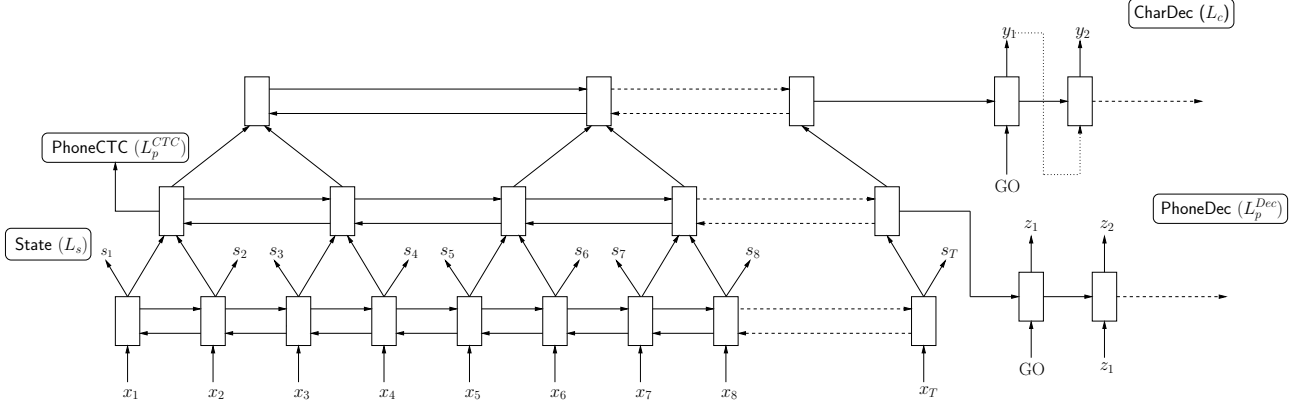


Figure 1: Sketch of our training-time model with multiple losses applied at different layers. The encoder is a pyramidal bidirectional LSTM (our experiments use 4 layers; we show 3 layers for simplicity). Different hidden state layers of this encoder are used for predicting HMM state label  $s_i$ , phone sequence  $\mathbf{z}$  (using either CTC or a LSTM decoder), and finally the output character sequence  $\mathbf{y}$  via a LSTM decoder. The dotted line in the character decoder denotes the use of (sampled) model predictions [20] during training (for the phone decoder only the ground-truth prior phone is used in training). At test time, only the character decoder is used for transcription.

the approach in the context of encoder-decoder models, and we compare a number of low-level auxiliary losses.

### 3. Models

The multitask approach we propose can in principle be applied to any type of deep end-to-end model. Here we study the approach in the context of attention-based deep RNNs. Below we describe the baseline model, followed by the auxiliary low-level training tasks.

#### 3.1. Baseline Model

The model is based on attention-enabled encoder-decoder RNNs, proposed by [19]. The *speech encoder* reads in acoustic features  $\mathbf{x} = (x_1, \dots, x_T)$  and outputs a sequence of high-level features (hidden states)  $\mathbf{h}$  which the *character decoder* attends to in generating the output character sequence  $\mathbf{y} = (y_1, \dots, y_K)$ , as shown in Figure 1 (the attention mechanism and a pyramidal LSTM layer are *not* shown in the figure for simplicity).

##### 3.1.1. Speech Encoder

The speech encoder is a deep pyramidal bidirectional Long Short-Term Memory [31] (BiLSTM) network [4]. In the first layer, a BiLSTM reads in acoustic features  $\mathbf{x}$  and outputs  $\mathbf{h}^{(1)} = (h_1^{(1)}, \dots, h_T^{(1)})$  given by:

$$\begin{aligned} \overrightarrow{h_i^{(1)}} &= f^{(1)}(x_i, \overrightarrow{h_{i-1}^{(1)}}) & \overleftarrow{h_i^{(1)}} &= b^{(1)}(x_i, \overleftarrow{h_{i+1}^{(1)}}) \\ h_i^{(1)} &= (\overrightarrow{h_i^{(1)}}; \overleftarrow{h_i^{(1)}}) \end{aligned}$$

where  $i \in \{1, \dots, T\}$  denotes the index of the timestep;  $f^{(1)}(\cdot)$  and  $b^{(1)}(\cdot)$  denote the first layer forward and backward LSTMs respectively<sup>1</sup>.

The first layer output  $\mathbf{h}^{(1)} = (h_1^{(1)}, \dots, h_T^{(1)})$  is then pro-

cessed as follows:

$$\left. \begin{aligned} \overrightarrow{h_i^{(j)}} &= f^{(j)}([h_{2i-1}^{(j-1)}; h_{2i}^{(j-1)}], \overrightarrow{h_{i-1}^{(j)}}) \\ \overleftarrow{h_i^{(j)}} &= b^{(j)}([h_{2i-1}^{(j-1)}; h_{2i}^{(j-1)}], \overleftarrow{h_{i+1}^{(j)}}) \\ h_i^{(j)} &= (\overrightarrow{h_i^{(j)}}; \overleftarrow{h_i^{(j)}}) \end{aligned} \right\} \text{ for } j = 2, 3, 4$$

where  $f^{(j)}$  and  $b^{(j)}$  denote the forward and backward running LSTMs at layer  $j$ . Following [4], we use pyramidal layers to reduce the time resolution of the final state sequence  $\mathbf{h}^{(4)}$  by a factor of  $2^3 = 8$ . This reduction brings down the input sequence length, initially  $T = |\mathbf{x}|$ , where  $|\cdot|$  denotes the length of a sequence of vectors, close to the output sequence length<sup>2</sup>,  $K = |\mathbf{y}|$ . For simplicity, we will refer to  $\mathbf{h}^{(4)}$  as  $\mathbf{h}$ .

##### 3.1.2. Character Decoder

The character decoder is a single-layer LSTM that predicts a sequence of characters  $\mathbf{y}$  as follows:

$$P(\mathbf{y}|\mathbf{x}) = P(\mathbf{y}|\mathbf{h}) = \prod_{t=1}^K P(y_t|\mathbf{h}, \mathbf{y}_{<t}).$$

The conditional dependence on the encoder state vectors  $\mathbf{h}$  is represented by context vector  $\mathbf{c}_t$ , which is a function of the current decoder hidden state and the encoder state sequence:

$$u_{it} = \mathbf{v}^\top \tanh(\mathbf{W}_1 \mathbf{h}_i + \mathbf{W}_2 \mathbf{d}_t + \mathbf{b}_a)$$

$$\alpha_t = \text{softmax}(\mathbf{u}_t) \quad \mathbf{c}_t = \sum_{i=1}^{|\mathbf{h}|} \alpha_{it} \mathbf{h}_i$$

where the vectors  $\mathbf{v}$ ,  $\mathbf{b}_a$  and the matrices  $\mathbf{W}_1$ ,  $\mathbf{W}_2$  are learnable parameters;  $\mathbf{d}_t$  is the hidden state of the decoder at time step  $t$ . The time complexity of calculating the context vector  $\mathbf{c}_t$  for every time step is  $O(|\mathbf{h}|)$ ; reducing the resolution on encoder side is crucial to reducing this runtime.

The hidden state of the decoder,  $\mathbf{d}_t$ , which captures the previous character context  $\mathbf{y}_{<t}$ , is given by:

$$\mathbf{d}_t = g(\tilde{\mathbf{y}}_{t-1}, \mathbf{d}_{t-1}, \mathbf{c}_{t-1})$$

<sup>1</sup>For brevity we exclude the LSTM equations. The details can be found, e.g., in Zaremba *et al.* [32].

<sup>2</sup>For Switchboard, the average of number of frames per character is about 7.

where  $g(\cdot)$  is the transformation of the single-layer LSTM,  $\mathbf{d}_{t-1}$  is the previous hidden state of the decoder, and  $\tilde{\mathbf{y}}_{t-1}$  is a character embedding vector for  $y_{t-1}$ , as is typical practice in RNN-based language models. Finally, the posterior distribution of the output at time step  $t$  is given by:

$$P(y_t|\mathbf{h}, \mathbf{y}_{<t}) = \text{softmax}(\mathbf{W}_s[\mathbf{c}_t; \mathbf{d}_t] + \mathbf{b}_s),$$

and the character decoder loss function is then defined as

$$L_c = -\log P(\mathbf{y}|\mathbf{x}).$$

### 3.2. Low-Level Auxiliary Tasks

As shown in Figure 1, we explore multiple types of auxiliary tasks in our multitask approach. We explore two types of auxiliary labels for multitask learning: phonemes and sub-phonetic states. We hypothesize that the intermediate representations needed for sub-phonetic state classification are learned at the lowest layers of the encoder, while representations for phonetic prediction may be learned at a somewhat higher level.

#### 3.2.1. Phoneme-Based Auxiliary Tasks

We use phoneme-level supervision obtained from the word-level transcriptions and pronunciation dictionary. We consider two types of phoneme transcription loss: *Phoneme Decoder Loss*: Similar to the character decoder described above, we can attach a phoneme decoder to the speech encoder as well. The phoneme decoder has exactly the same mathematical form as the character decoder, but with a phoneme label vocabulary at the output. Specifically, the phoneme decoder loss is defined as

$$L_p^{\text{Dec}} = -\log P(\mathbf{z}|\mathbf{x}),$$

where  $\mathbf{z}$  is the target phoneme sequence. Since this decoder can be attached at any depth of the four-layer encoder described above, we have four depths to choose from. We attach the phoneme decoder to layer 3 of the speech encoder, and also compare this choice to attaching it to layer 4 (the final layer) for comparison with a more typical multitask training approach.

*CTC Loss*: A CTC [33] output layer can also be added to various layers of the speech encoder [30]. This involves adding an extra softmax output layer on top of the chosen intermediate layer of the encoder, and applying the CTC loss to the output of this softmax layer. Specifically, let  $\mathbf{z}$  be the target phoneme sequence, and  $k$  be the speech encoder layer where the loss is applied. The probability of  $\mathbf{z}$  given the input sequence is

$$P(\mathbf{z}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{z})} P(\pi|\mathbf{h}^{(k)}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{z})} \prod_{j=1}^J P(\pi_j|\mathbf{h}_j^{(k)}),$$

where  $\mathcal{B}(\cdot)$  removes repetitive symbols and blank symbols,  $\mathcal{B}^{-1}$  is  $\mathcal{B}$ 's pre-image,  $J$  is the number of frames at layer  $k$  and  $P(\pi_j|\mathbf{h}_j^{(k)})$  is computed by a softmax function. The final CTC objective is

$$L_p^{\text{CTC}} = -\log P(\mathbf{z}|\mathbf{x}).$$

The CTC objective computation requires the output length to be less than the input length, i.e.,  $|\mathbf{z}| < J$ . In our case the encoder reduces the time resolution by a factor of 8 between the input and the top layer, making the top layer occasionally shorter than the number of phonemes in an utterance. We therefore cannot apply this loss to the topmost layer, and use it only at the third layer.<sup>3</sup>

<sup>3</sup>In fact, even at the third layer we find occasional instances (about 10 utterances in our training set) where the hidden state sequence is shorter

#### 3.2.2. State-Level Auxiliary Task

Sub-phonetic state labels provide another type of low-level supervision that can be borrowed from traditional modular HMM-based approaches. We apply this type of supervision at the frame level, as shown in Figure 1, using state alignments obtained from a standard HMM-based system. We apply this auxiliary task at layer 2 of the speech encoder. The probability of a sequence of states  $\mathbf{s}$  is defined as

$$P(\mathbf{s}|\mathbf{x}) = \prod_{m=1}^M P(s_m|\mathbf{x}) = \prod_{m=1}^M P(s_m|\mathbf{h}_m^{(2)}),$$

where  $P(s_m|\mathbf{h}_m^{(2)})$  is computed by a softmax function, and  $M$  is the number of frames at layer 2 (in this case  $\lceil T/2 \rceil$ ). Since we use this task at layer 2, we subsample the state labels to match the reduced resolution. The final state-level loss is

$$L_s = -\log P(\mathbf{s}|\mathbf{x}).$$

#### 3.2.3. Training Loss

The final loss function that we minimize is the average of the losses involved. For example, in the case where we use the character and phoneme decoder losses and the state-level loss, the loss would be

$$L = \frac{1}{3}(L_c + L_p^{\text{Dec}} + L_s).$$

## 4. Experiments

We use the Switchboard corpus (LDC97S62) [34], which contains roughly 300 hours of conversational telephone speech, as our training set. We reserve the first 4K utterances as a development set. Since the training set has a large number of repetitions of short utterances (“yeah”, “uh-huh”, etc.), we remove duplicates beyond a count threshold of 300. The final training set has about 192K utterances. For evaluation, we use the HUB5 Eval2000 data set (LDC2002S09), consisting of two subsets: Switchboard (SWB), which is similar in style to the training set, and CallHome (CHE), which contains unscripted conversations between close friends and family.

For input features, we use 40-dimensional log-mel filterbank features along with their deltas, normalized with per-speaker mean and variance normalization. The phoneme labels for the auxiliary task are generated by mapping words to their canonical pronunciations, using the lexicon in the Kaldi Switchboard training recipe. The HMM state labels were obtained via forced alignment using a baseline HMM/DNN hybrid system using the Kaldi NNet1 recipe. The HMM/DNN has 8396 tied states, which makes the frame-level softmax costly for multitask learning. We use the importance sampling technique described in [35] to reduce this cost.

### 4.1. Model Details and Inference

The speech encoder is a 4-layer pyramidal bidirectional LSTM, resulting in a 8-fold reduction in time resolution. We use 256 hidden units in each direction of each layer. The decoder for all tasks is a single-layer LSTM with 256 hidden units. We represent the decoders' output symbols (both characters and, at training time, phonemes) using 256-dimensional embedding vectors. At test time, we use a greedy decoder (beam size =

than the input sequence, due to sequences of phonemes of duration less than 4 frames each. Anecdotally, these examples appear to correspond to incorrect training utterance alignments

Table 1: Character error rate (CER, %) and word error rate (WER, %) results on development data.

Model	Dev CER	Dev WER
Enc-Dec (baseline)	14.6	26.0
Enc-Dec + PhoneDec-3	13.8	24.9
Enc-Dec + PhoneDec-4	14.5	25.9
Enc-Dec + PhoneCTC-3	14.0	25.3
Enc-Dec + State-2	13.6	24.1
<b>Enc-Dec + PhoneDec-3 + State-2</b>	<b>13.4</b>	<b>24.1</b>

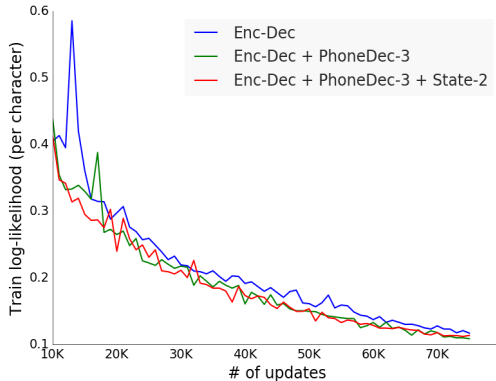


Figure 2: Log-likelihood of train data (per character) for different model variations.

1) to generate the character sequence. The character with the maximum posterior probability is chosen at every time step and fed as input into the next time step. The decoder stops after encountering the “EOS” (end-of-sentence) symbol. We use no explicit language model.

We train all models using Adam [36] with a minibatch size of 64 utterances. The initial learning rate is  $1e-3$  and is decayed by a factor of 0.95, whenever there is an increase in log-likelihood of the development data, calculated after every 1K updates, over its previous value. All models are trained for 75K gradient updates (about 25 epochs) and early stopping. To further control overfitting we: (a) use dropout [37] at a rate of 0.1 on the output of all LSTM layers (b) sample the previous step’s prediction [20] in the character decoder, with a constant probability of 0.1 as in [4].

## 4.2. Results

We evaluate performance using word error rate (WER). We report results on the combined Eval2000 test set as well as separately on the SWB and CHE subsets. We also report character error rates (CER) on the development set.

Development set results are shown in Table 1. We refer to the baseline model as “Enc-Dec” and the models with multitask training as “Enc-Dec + [auxiliary task]-[layer]”. Adding phoneme recognition as an auxiliary task at layer 3, either with a separate LSTM decoder or with CTC, reduces both the character error rates and the final word error rates.

In order to determine whether the improved performance is a basic multitask training effect or is specific to the low-level application of the loss, we compare these results to those of adding the phoneme decoder at the topmost layer (Enc-Dec + PhoneDec-4). The top-layer application of the phoneme loss produces worse performance than having the supervision at the lower (third) layer. Finally, we obtain the best results by adding both phoneme decoder supervision at the third layer and frame-level state supervision at the second layer (Enc-Dec + PhoneDec-3 + State-2). The results support the hypothesis that lower-level supervision is best provided at lower layers. Table 2 provides test set results, showing the same pattern of improvement on both

Table 2: WER (%) on Eval2000 for different end-to-end models. PhoneDec- $n$  refers to a phoneme decoder applied at layer  $n$  of the encoder. Similarly, PhoneCTC-3 means phoneme CTC loss applied at layer 3 and State-2 means state-label supervision applied at layer 2 of the encoder.

Model	SWB	CHE	Full
Our models			
Enc-Dec (baseline)	25.0	42.4	33.7
Enc-Dec + PhoneDec-3	24.5	40.6	32.6
Enc-Dec + PhoneDec-4	25.4	41.9	33.7
Enc-Dec + PhoneCTC-3	24.6	41.3	33.0
Enc-Dec + State-2	24.7	42.0	33.4
<b>Enc-Dec + PhoneDec-3 + State-2</b>	<b>23.1</b>	<b>40.8</b>	<b>32.0</b>
Lu et al. [7]			
Enc-Dec	27.3	48.2	37.8
Enc-Dec (word) + 3-gram	25.8	46.0	36.0
Maas et al. [8]			
CTC	38.0	56.1	47.1
CTC + 3-layer RNN LM	21.4	40.2	30.8
Zweig et al. [9]			
Iterated CTC	24.7	37.1	—
CTC + Char Ngram	19.8	32.1	—
CTC + Dictionary + Word Ngram	14.0	25.3	—

the SWB and CHE subsets. For comparison, we also include a variety of other recent results with neural end-to-end approaches on this task. Our baseline model has better performance than the most similar previous encoder-decoder result [7]. With the addition of the low-level auxiliary task training, our models are competitive with all of the previous end-to-end systems that do not use a language model.

Figure 2 shows the training set log-likelihood for the baseline model and two multitask variants. The plot suggests that multitask training helps with optimization (improving the training error). Training error is very similar for both multitask models, while the development set performance is better for one of them (see Table 1), suggesting that there may also be an improved generalization effect and not only improved optimization.

## 5. Conclusion

We have presented a multitask training approach for deep end-to-end ASR models in which lower-level task losses are applied at lower levels, and we have explored this approach in the context of attention-based encoder-decoder models. Results on Switchboard and CallHome show consistent improvements over baseline attention-based models and support the hypothesis that lower-level supervision is more effective when applied at lower layers of the deep model. We have compared several types of auxiliary tasks, obtaining the best performance with a combination of a phoneme decoder and frame-level state loss. Analysis of model training and performance suggests that the addition of auxiliary tasks can help in either optimization or generalization.

Future work includes studying a broader range of auxiliary tasks and model configurations. For example, it would be interesting to study even deeper models and word-level output, which would allow for more options of intermediate tasks and placements of the auxiliary losses. Viewing the approach more broadly, it may be fruitful to also consider higher-level task supervision, incorporating syntactic or semantic labels, and to view the ASR output as an intermediate output in a more general hierarchy of tasks.

## 6. Acknowledgements

We are grateful to William Chan for helpful discussions, and to the speech group at TTIC, especially Shane Settle, Herman Kamper, Qingming Tang, and Bowen Shi for sharing their data processing code. This research was supported by a Google faculty research award.

## 7. References

- [1] M. Gales and S. Young, "The application of hidden markov models in speech recognition," *Foundations and trends in signal processing*, vol. 1, 2008.
- [2] K. Vesely, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Interspeech*, 2013.
- [3] D. Povey and B. Kingsbury, "Evaluation of proposed modifications to mpe for large scale discriminative training," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007.
- [4] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.
- [5] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Neural Information Processing Systems (NIPS)*, 2015.
- [6] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.
- [7] L. Lu, X. Zhang, and S. Renals, "On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.
- [8] A. L. Maas, Z. Xie, D. Jurafsky, and A. Y. Ng, "Lexicon-free conversational speech recognition with neural networks," in *North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL HLT)*, 2015.
- [9] G. Zweig, C. Yu, J. Droppo, and A. Stolcke, "Advances in all-neural speech recognition," *CoRR*, vol. abs/1609.05935, 2016.
- [10] Y. Miao, M. Gowayyed, and F. Metze, "EESN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015.
- [11] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision (ECCV)*, 2016.
- [12] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. E. Hinton, "Grammar as a foreign language," in *Neural Information Processing Systems (NIPS)*, 2015.
- [13] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. B. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean, "Google's multilingual neural machine translation system: Enabling zero-shot translation," *CoRR*, vol. abs/1611.04558, 2016.
- [14] A.-r. Mohamed, G. E. Hinton, and G. Penn, "Understanding how deep belief networks perform acoustic modelling," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2012.
- [15] T. Nagamine, M. L. Seltzer, and N. Mesgarani, "On the role of nonlinear transformations in deep neural network acoustic models," *Interspeech*, 2016.
- [16] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision (ECCV)*, 2014.
- [17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [18] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Neural Information Processing Systems (NIPS)*, 2014.
- [19] 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.
- [20] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Neural Information Processing Systems (NIPS)*, 2015.
- [21] R. Caruana, "Multitask learning," *Machine Learning*, 1997.
- [22] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research (JMLR)*, 2011.
- [23] M. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task sequence to sequence learning," in *International Conference on Learning Representations (ICLR)*, 2016.
- [24] A. Eriguchi, Y. Tsuruoka, and K. Cho, "Learning to parse and translate improves neural machine translation," *CoRR*, vol. abs/1702.03525, 2017.
- [25] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," *CoRR*, vol. abs/1609.06773, 2016.
- [26] W. Chan and I. Lane, "On online attention-based speech recognition and joint Mandarin character-Pinyin training," in *Interspeech*, 2016.
- [27] Z. Wu, C. Valentini-Botinhao, O. Watts, and S. King, "Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.
- [28] A. Søgaard and Y. Goldberg, "Deep multi-task learning with low level tasks supervised at lower layers," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.
- [29] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher, "A joint many-task model: Growing a neural network for multiple NLP tasks," *CoRR*, vol. abs/1611.01587, 2016.
- [30] K. Rao and H. Sak, "Multi-accent speech recognition with hierarchical grapheme based models," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017.
- [31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, 1997.
- [32] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014.
- [33] A. Graves, S. Fernández, and F. Gomez, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2006.
- [34] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1992.
- [35] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, "On using very large target vocabulary for neural machine translation," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015.
- [36] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research (JMLR)*, vol. 12, 2011.
- [37] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.