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

# Char+CV-CTC: combining graphemes and consonant/vowel units for CTC-based ASR using Multitask Learning

*Abdelwahab Heba*[1,2]*, Thomas Pellegrini*[1]*, Jean-Pierre Lorré*[2]*, Régine Andre-Obrecht*[1]

[1]IRIT, Université Paul Sabatier, CNRS, France
[2]Linagora, France

{abdel.heba,thomas.pellegrini}@irit.fr, jplorre@linagora.fr, regine.obrecht@irit.fr

## Abstract

Previous work has shown that end-to-end neural-based speech recognition systems can be improved by adding auxiliary tasks at intermediate layers. In this paper, we report multitask learning (MTL) experiments in the context of connectionist temporal classification (CTC) based speech recognition at character level. We compare several MTL architectures that jointly learn to predict characters (sometimes called graphemes) and consonant/vowel (CV) binary labels. The best approach, which we call Char+CV-CTC, adds up the character and CV logits to obtain the final character predictions. The idea is to put more weight on the vowel (consonant) characters when the vowel (consonant) symbol 'V' ('C') is predicted in the auxiliary-task branch of the network. Experiments were carried out on the Wall Street Journal (WSJ) corpus. Char+CV-CTC achieved the best ASR results with a 2.2% Character Error Rate and a 6.1% Word Error Rate (WER) on the Eval92 evaluation subset. This model outperformed its monotask model counterpart by 0.7% absolute in WER and also achieved almost the same performance of 6.0% as a strong baseline phone-based Time Delay Neural Network ("TDNN-Phone+TR2") model.

**Index Terms**: automatic speech recognition, connectionist temporal classification, multi-task learning

## 1. Introduction

Recent advances in automatic speech recognition (ASR) systems have enabled training neural "end-to-end" architectures that attempt to map audio signals directly into text. A first step towards end-to-end ASR was made with the so-called Connectionist Temporal Classification (CTC) objective function introduced by Graves *et al.* [1, 2]. By contrast, sequence-to-sequence recurrent neural networks (RNNs), such as encoder-decoders [3, 4], process sequences of audio features through stacked layers. Higher-level representations are encoded and fed to decision layers that output labels in an end-to-end fashion for tasks that previously required significant human expertise.

There is strong evidence that end-to-end ASR systems implicitly learn linguistically meaningful representations at intermediate layers, between the acoustic input and the final symbolic output [5, 6]. The evidence is particularly strong for phonetic-related properties in ASR end-to-end intermediate representations [7, 8].

Multitask learning (MTL) approaches for end-to-end ASR systems have gained momentum in the last few years [9, 10]. Recent work introduced the use of hierarchical MTL in speech recognition with hierarchical CTC-based models [7, 11]. Performance gains have been obtained by combining phone-label predictions as an auxiliary task in training a spoken digit sequence recognizer, as in [12]. In [7], the authors proposed a hierarchical CTC model for a subword-based ASR model with an auxiliary phone-level CTC loss applied at an intermediate layer of a neural network.

The present work develops an MTL approach inspired by the ontology-based network architecture proposed for sound event detection (SED) in [13]. The primary SED task aims at classifying "low-level" sound events in categories such as "violin, piano", "eating, breathing", and "cat, dog". Jimenez *et al.* proposed a network that produces hierarchical outputs and multilevel predictions. It is based on a simple ontology-based layer defined with high-level classes such as "Music","Human", "Nature", etc. The layer consists of a fixed binary matrix $M$ which allows the conversion of low-level sound predictions into predictions of the high-level classes by simply summing up the probabilities of the low-level categories that share a common high-level category. This work showed that the higher-level recognition task improves the primary task performance by a large margin.

In this paper, we investigate the benefit of adding a consonant/vowel recognition auxiliary task as our high-level secondary task to train character-based CTC models. We propose and compare three MTL architectures involving the output of the secondary task in three ways: i) as an independent output head in the network (standard MTL), ii) on the top of the character recognition output head (similar to [13]), iii) as logits summed to the character-level logits to promote the prediction of a character being either a consonant or a vowel.

Since the CV and character logits are summed up, we call this last architecture "Char+CV-CTC". As we report in the paper, the experiments carried out on the WSJ 80-hour training set [14] and the Eval92 test set for evaluation show that Char+CV-CTC lead to the best results.

The paper is organized as follows. In Section 2, we begin by briefly introducing the CTC approach that is used to train our models at character and CV levels. We describe the various MTL model variants in Section 3. The decoding methods used in this work are described in Section 4. Finally, we evaluate our models on the Wall Street Journal dataset and compare the different multitask approaches in Section 5. As a side note, we use "characters" and "graphemes" interchangeably throughout the paper.

## 2. Connectionist Temporal Classification

Introduced in [1], CTC allows a system to automatically learn alignments between speech frames $X_{1,...,T}$ and their label sequences $K_{1,...,N}$ (where $N \leq T$, and $T$ and $N$ are the lengths of the speech frame sequence and of the label sequence, respectively). A CTC *path* is defined as a $T$-long sequence of output labels with probability $P^i = p(1, ..., T)$. In order to obtain CTC alignments and cope with the difference in lengths be-



http://dx.doi.org/10.21437/Interspeech.2019-1975

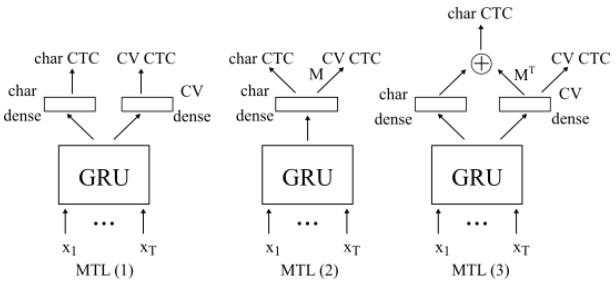


Figure 1: *Architecture of the three MTL variants tested in this work. "Char dense" and "CV dense": fully-connected dense layers with output dimensions equal to the number of target characters and CV, respectively. MTL (1): standard MTL with two heads, MTL (2): hierarchical chararcter to CV MTL, MTL (3): Char+CV-CTC MTL.*

tween the input and output sequences, Graves and colleagues proposed a dynamic constrained procedure that adds a blank label $\epsilon$ between each label in sequence $K$. This procedure can result in several possible paths (alignments) that yield to the expected label sequence $K$. For example, a sequence of three characters such as "ABC", along with a sequence of speech frames of length $T = 5$, can lead to paths such as "A $\epsilon$ B B C" or "A A B C $\epsilon$", etc.

During training, the CTC conditional probability function marginalizes over the set of valid alignments computing the probability for a single alignment step-by-step:

$$P(K|X_{1,...T}) = \sum_{i \in \theta_K} \prod_{t=1}^{T} P_t^i \tag{1}$$

where $\theta_K$ is the set of all possible paths that lead to the $K$ label sequence.

The CTC technique is generic and can be applied to phones, characters and all types of (sub-)word units. In the present work, we use it at character and consonant/vowel levels.

# 3. Proposed MTL approaches

In this work, we compare a standard character-based CTC model with variants that involve the auxiliary task of recognizing two higher-level categories: vowels and consonants, denoted as 'V' and 'C'. For the auxiliary task, we used five units: 'C' for consonants, 'V' for vowels, the quote tag, the $\epsilon$ (blank) symbol and the space label. The last three units are needed to perform CTC properly. The 'V' symbol was chosen to represent the letters 'a', 'e', 'i', 'o', 'u' and 'y' while 'C' represents all the other letters. In this setting, the semi-vowel 'w' is considered as a consonant. The character classification layer is comprised of 29 units to predict the 26 characters of the English alphabet, the quote tag, the $\epsilon$ symbol and the space label.

We compared three multitask variants, which are depicted in Fig. 1 and described in the sections here-after. In Fig. 1, "char CTC" and "CV CTC" indicate the placement of the two outputs layers used respectively for the character and CV tasks. These are the output layers on which the CTC loss objectives are optimized. Log-softmax was used as the activation function of these layers. "Char dense" and "CV dense" indicate fully-connected layers with a number of neuron units equal to the number of classes of characters (26+3) and CV (2+3), respectively.

In all the variants, we use the multitask loss defined in Eq. (2). It is the convex combination of two CTC-loss functions: $\mathcal{L}_1$, the CTC loss evaluated on the character predictions corresponding to $\mathbf{p}(\hat{y}_{\text{char}}|x)$, and $\mathcal{L}_2$, the CTC loss evaluated on the CV predictions corresponding to $\mathbf{p}(\hat{y}_{\text{cv}}|x)$, where $\hat{y}_{\text{char}}$ and $\hat{y}_{\text{cv}}$ denote the character and the CV sequence predictions for a given utterance, respectively. Here, $\lambda$ is a hyper parameter to be tuned to a real value within $[0, 1]$ that determines the weight of each task loss. When $\lambda = 1$ (0), the task is reduced to the monotask of character (CV) recognition.

$$\mathcal{L} = \lambda \mathcal{L}_{\text{char}} + (1 - \lambda)\mathcal{L}_{\text{cv}} \tag{2}$$

### 3.1. Standard MTL approach

The first variant, denoted MTL (1) in Fig. 1, is the standard MTL approach with two output heads: one for the character task (char) and one for the auxiliary task (CV).

### 3.2. Hierarchical MTL approach

The second variant, MTL (2), handles the two tasks in a hierarchical manner where we first output char logits at the level of the "char dense" layer depicted in Fig. 1. Then the logits corresponding to vowels and to consonants are accumulated together to get probabilities for 'C' and 'V', respectively. The relationship between the characters and the CV level logits is a simple matrix-vector product:

$$\hat{\mathbf{z}}_{\text{cv}} = M \cdot \hat{\mathbf{z}}_{\text{char}} \tag{3}$$

where $\hat{\mathbf{z}}_{\text{char}}$ and $\hat{\mathbf{z}}_{\text{cv}}$ are the character and CV logits outputted by the respective fully-connected layers, and $M$ is the mapping matrix.

The binary matrix $M$ of dimension $N_{\text{cv}} \times N_{\text{char}}$ maps characters to the 'C' and 'V' classes. $M$ is fixed and is not modified during training.

Char and CV probabilities are then obtained by applying the softmax function on the respective logits. To be more precise, the CTC loss implementation used in this work uses log-probabilities by applying the log-softmax activation function.

This variant was inspired by the so-called ontology-based networks for sound event detection proposed in [13]. Indeed, the CV categories can be viewed as a super class of the characters. We do not call this variant ontology-based MTL since there is no semantics involved in the char and CV categories.

### 3.3. Char+CV-CTC approach

The third variant MTL (3) is the main novel contribution of this work. We combine the char and CV outputs by simply summing up the outputs of the two fully-connected layers responsible for each task. We sum up the logits rather than the probabilities in order to keep the full range of real values possible and thereby avoid squashing the values. Log-softmax is applied afterwards. To do so, we replicate the CV probabilities with the transposed version of the $M$ matrix of variant (2):

$$\hat{\mathbf{z}}'_{\text{char}} = \hat{\mathbf{z}}_{\text{char}} + M^T \cdot \hat{\mathbf{z}}_{\text{cv}} \tag{4}$$

Variant (3) was designed based on the idea that if the model predicts that a certain character is a member of 'V', then we should put more weight on the characters that actually are vowels, and likewise for consonants ('C' class).

# 4. Decoding approaches

We used two decoding approaches: greedy decoding and Weighted Finite-State Transducer (WFST)-based decoding. We also tried the Prefix Beam Search algorithm [1] but as expected, we obtained performance values better than greedy search but worse than graph decoding so we chose to not report them.

Decoding algorithms are used to compute the final labels for an input sequence $X$ of length $T$. CTC outputs a set of probabilities $p(c|x_t), t = 1, ..., T$ over the set of all possible characters in the alphabet called $\Sigma$. $\Sigma$ is comprised of all the target characters plus a space, quote tag, and $\epsilon$.

### 4.1. Greedy Decoding

We use the simple greedy approach as a baseline decoding method. It does not use any language model or lexicon constraints. Instead, the best path is simply determined by selecting the character of highest probability at each frame to recover a character string. Then, the decoding procedure on the CTC sequence consists in collapsing repeated characters and removing the blanks to get the final label sequence.

### 4.2. WFST-based decoding

In this work, we used a WFST-based decoding, which is one of the most robust decoding techniques in ASR [15, 16]. More specifically, we used a generalized decoding implementation adapted to CTC outputs with a blank label proposed by Miao *et al.* [17]. The output transcription corresponds to the most likely path through the WFST, determined by the Viterbi algorithm. The WFST takes as input a sequence of symbols, namely the labels from a CTC model output, and emits a sequence of words (a transcript). CTC-WFSTs require three graphs: i) the set of *CTC Labels* called Tokens, ii) a *word lexicon* and iii) a *language model*, all encoded as separate WFSTs. The three WFSTs are merged, compiled and compressed into a single search graph.

# 5. Experimental setup

### 5.1. Speech material

The experiments were performed on the Wall Street Journal corpus (WSJ, LDC93S6A). We used the Kaldi data preparation WSJ recipe. The 81 hours of transcribed speech were split into train (95%) and development (5%) subsets and correpond to the subsets used in the Eesen article [17]. Evaluation was performed on the *Eval92* set. The *Dev93* set was used to fix the optimal value of the $\lambda$ mixing weight for MTL models.

Log-Mel filterbank coefficients with 40 frequency bins were extracted together with first and second derivatives. The features were then normalized via mean subtraction and variance normalization (CMVN) per-speaker. No speaker adaptation techniques were used.

### 5.2. Time Reduction (TR)

As in [7], pairs of consecutive input frames were concatenated for a reduction in time resolution at the input. This technique helped to speed up training. Using an Nvidia GTX-1080 TI, training took 34 hours instead of 58 hours without TR. We also observed that TR improved performance.

### 5.3. Model description

Compared to standard feed-forward networks, RNNs have the advantage of modeling temporal dynamics of sequences [2]. We chose to use recurrent layers, namely bidirectional Gated Recurrent Unit layers (BiGRU) [18].

After trial and error tests, we opted for a model with four BiGRU hidden layers with $2 \times 320$ cells in each layer, giving a total of 8.3M parameters.

Dropout (10% rate) [19] was applied on the output of each BiGRU layer. The output layer uses the log-softmax activation function. The network was trained on 100 epochs with a 32 batch size and the CTC objective function described in Section 2, the ADAM optimizer [20] with a 4e-5 learning rate and a 0.9 momentum value. For inference, we used greedy decoding with no language model, and WFST-based decoding as implemented in EESEN [17] with lexical expansion as proposed in the Kaldi WSJ recipe [16]. The language model, available within the WSJ release, is a trigram LM with a 20k word vocabulary. All the models were implemented with the PyTorch library [21].

# 6. Evaluation

### 6.1. Results

Table 1 gives the character and word error rates (CER, WER) obtained with our character-based CTC-GRU models using the greedy and WFST-based decoding algorithms. In the top part of the table, we report results from the literature on the same data using models that have a similar architecture as ours, with four layers and about 8 to 9 million learnable parameters.

Our baseline (character-based CTC-GRU) achieved a 7.4% WER, which is very close to that of the character-based CTC-LSTM models reported in [22, 17]. We also report the performance of 7.1% WER obtained by a phoneme-based HMM/DNN Kaldi system from [17]. This model is larger, being comprised of six hidden layers and 1024 units per layer (9.2M parameters).

Table 1: *Results on Eval92 in terms of character and word error rates (CER, WER). MTL (1), (2) and (3) correspond respectively to the standard MTL, hierarchical MTL and Char+CV-CTC models with the same architecture as the baseline monotask model CTC-GRU+TR2. TR2 ("Time-Reduction") refers to using acoustic feature frames concatenated every two frames.*

| Model | Greedy Search | | | WFST-based | |
|---|---|---|---|---|---|
| | CVER | CER | WER | CER | WER |
| TDNN-Phone [17] | - | - | - | - | 7.1 |
| CTC-LSTM [17] | - | - | - | - | 7,3 |
| CTC-LSTM [22] | - | 9.2 | 30.1 | - | 8.7 |
| TDNN-Phone+TR2 | - | - | - | - | 6.0 |
| CTC-GRU | - | 8.4 | 29.6 | 2.8 | 7.4 |
| CTC-GRU+TR2 | - | 8.0 | 28.8 | 2.4 | 6.8 |
| MTL (1) | 5.2 | 7.8 | 27.3 | 2.3 | 6.6 |
| MTL (2) | 5.2 | 8.3 | 28.2 | 2.8 | 7.5 |
| MTL (3) | 4.5 | 7.7 | 27.5 | 2.2 | 6.1 |

As a baseline, our CTC-GRU achieved better results compared to CTC-LSTM [22], with CER / WER of 8.4% / 29.6% respectively, by naively choosing the most likely label at each time step (greedy search). As expected, WFST-based decoding gave much better results than greedy search. Our baseline gave a 7.4% WER to be compared with the 29.6% WER with greedy search. In the remainder of the paper, we only consider the results obtained with WFST-based decoding.

We compare our baseline to a model trained using a Time Reduction of two frames: CTC-GRU+TR2. Improvements of 0.4% and 0.6% absolute in CER / WER respectively were obtained showing the positive impact of TR. This result was confirmed in our experiments with a hybrid HMM/TDNN phone-based system. Using TR of two frames, we obtained a 6.0% WER which is better than the 7.1% WER reported in [17]. These significant performance gains may be due to TR increasing the receptive field size at the input of the networks. Interestingly, Krishna *et al.* [17] only mentioned computation speed-ups but did not justify the use of TR for its performance gains. We used a TR of two frames in all our MTL models.

The main contribution of this work lies in the multitask learning (MTL) models. We compare CTC-GRU+TR2 to the three MTL models described in Section 3. The interpolation constant $\lambda$ was optimized on the dev set, the best value was $\lambda = 0.8$. The influence of $\lambda$ is discussed in Section 6.2.

MTL (1), which is the standard multitask model, performs slightly better than the monotask CTC-GRU+TR2 model. By contrast, the hierarchical variant MTL (2) performs slightly worse. It seems that putting the auxiliary task directly on top of the main task fully-connected layer reduces the main task performance, contrary to the results of the ontology-based networks for sound event detection reported in [13].

The best results were obtained with MTL (3): 2.2% CER and 6.1% WER. This corresponds to a 0.7% absolute (10% relative) gain in WER compared to CTC-GRU+TR2. Combining the two tasks by summing up their logits proved successful. This may be explained by the fact that this addition puts more weight on the vowel (consonant) characters when the vowel (consonant) symbol V (C) is predicted.

### 6.2. Effect of the Interpolation $\lambda$ Weight

We explored the effect of the mixing weight $\lambda$ constant as defined in Section 3. The closer the value is to 1, the closer the learning is to the monotask situation. Fig. 2 shows the evolution of the CER and WER on the development set Dev93 with respect to $\lambda$ for MTL (3). The reference values of 4.9% CER and 10.0% WER obtained with the monotask model counterpart CTC-GRU+TR2 are also plotted.

Values of $\lambda$ larger than 0.5 outperform the baseline and the best performance is obtained for $\lambda = 0.8$, the value that was used to report the results on the test subset in Table 1. This value is close to 1, showing that the additional supervision brought by the auxiliary task needs to be small to bring performance gains.

## 7. Conclusions

In this work, we presented a new multitask learning approach for character-based CTC ASR recognition. This approach, called Char+CV-CTC, consists in summing the logits of the primary and auxiliary tasks before outputting the primary task final character predictions. We proposed the auxiliary task of recognizing consonants (C) and vowels (V) based on a CTC objective function. Several architectures were tested: a standard two-head model, a hierarchical character-to-CV model and Char+CV-CTC.

Evaluation experiments were conducted on WSJ. All our models use BiGRU stacked layers fed with concatenated feature frames (referred to as time reduction, TR). TR brings performance gains and substantial computation time reductions. Char+CV-CTC achieved the best ASR results with a 2.2% CER and a 6.1% WER. This model outperformed its monotask model

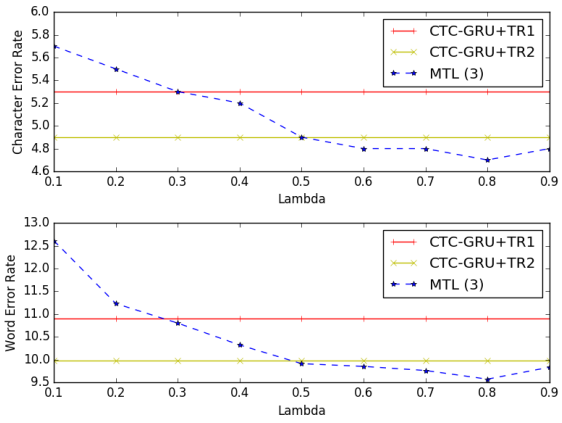


Figure 2: *Effect of varying the interpolation constant $\lambda$ on the WSJ dev93 set CER (%) and WER (%). For information, CTC-LSTM [17] achieved a 10.8% WER on this subset.*

counterpart by 0.7% absolute in WER. Another interesting result is that Char+CV-CTC achieves almost the same performance as a phone-based hybrid HMM-TDNN model.

We plan to confirm these results on larger-scale speech datasets such as Librispeech. Indeed, more experiments should be run to explore our char+CV-CTC approach. It would be interesting to try other ways of splitting the set of characters, including randomly, in order to explain the improvement obtained in the present work. It might come from the reduced number of classes in the auxiliary task and not from the semantic properties of the arbitrary split of the characters into consonant/vowel symbols.

Another future line of research will be to test Char+CV-CTC variants in which the auxiliary task is performed by an intermediate recurrent layer instead of the output layer. Indeed, in [7], improvements were obtained by doing so in a hierarchical MTL approach combining phone and sub-word CTC tasks, as the auxiliary and primary tasks, respectively.

## 8. Acknowledgements

This work was supported by Bpifrance within the LinTO project (*Programme des Investissements d'Avenir - Grands Défis du Numérique*, `https://linto.ai`) and by the Agence Nationale de la Recherche LUDAU project (Lightly-supervised and Unsupervised Discovery of Audio Units using Deep Learning, ANR-18-CE23-0005-01). The authors would like to thank the IRIT OSIRIM GPU cluster platform.

## 9. References

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*. Pittsburgh: ACM, 2006, pp. 369–376.

[2] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, Vancouver, 2013, pp. 6645–6649.

[3] L. Lu, X. Zhang, K. Cho, and S. Renals, "A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition," in *Proc. Interspeech*, 2015.

[4] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-

of-the-art speech recognition with sequence-to-sequence models," in *Proc. ICASSP*, 2018, pp. 4774–4778.

[5] T. Nagamine, M. L. Seltzer, and N. Mesgarani, "Exploring how deep neural networks form phonemic categories," in *Proc. Interspeech*, Dresden, 2015, pp. 1912–1916.

[6] T. Pellegrini and S. Mouysset, "Inferring phonemic classes from cnn activation maps using clustering techniques," in *Proc. Interspeech*, San Franscico, 2016, pp. 1287–1290.

[7] K. Krishna, S. Toshniwal, and K. Livescu, "Hierarchical multitask learning for ctc-based speech recognition," *arXiv preprint arXiv:1807.06234*, 2018.

[8] Y. Belinkov and J. Glass, "Analyzing hidden representations in end-to-end automatic speech recognition systems," in *Advances in Neural Information Processing Systems*, 2017, pp. 2441–2451.

[9] K. Rao and H. Sak, "Multi-accent speech recognition with hierarchical grapheme based models," in *Proc. ICASSP*, New Orleans, 2017, pp. 4815–4819.

[10] S. Kim, T. Hori, and S. Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*, New Orleans, 2017, pp. 4835–4839.

[11] R. Sanabria and F. Metze, "Hierarchical Multitask Learning With CTC," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 485–490.

[12] S. Fernández, A. Graves, and J. Schmidhuber, "Sequence labelling in structured domains with hierarchical recurrent neural networks," in *Proc. IJCAI*, Melbourne, 2007, pp. 774–779.

[13] A. Jimenez, B. Elizalde, and B. Raj, "Sound event classification using ontology-based neural networks," in *Proc. NeurIPS*, Montreal, 2018.

[14] J. Garofolo, D. Graff, D. Paul, and D. Pallett, "CSR-I (WSJ0) complete LDC93S6A," *Web Download. Philadelphia: Linguistic Data Consortium*, 1993.

[15] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[16] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *Proc. ASRU*, Hawaii, 2011.

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

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *Proc. ICML*, Lille, 2015, pp. 2067–2075.

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

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. NIPS*, Long Beach, 2017.

[22] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. ICML*, 2014, pp. 1764–1772.