Efficient Ensemble Architecture for Multimodal Acoustic and Textual Embeddings in Punctuation Restoration using Time-Delay Neural Networks

Xing Yi Liu[1] *and Homayoon Beigi*[1,2]

[1]Columbia University, New York, USA
[2]Recognition Technologies, Inc., New York, USA
`liu.peter@columbia.edu, beigi@recotechnologies.com`

Abstract

Punctuation restoration plays an essential role in the post-processing procedure of automatic speech recognition, but model efficiency is a key requirement for this task. To that end, we present EfficientPunct, an ensemble method with a multimodal time-delay neural network that outperforms the current best model by 1.0 F1 points, using less than a tenth of its parameters to process embeddings. We streamline a speech recognizer to efficiently output hidden layer latent vectors as acoustic embeddings for punctuation restoration, as well as BERT to extract meaningful text embeddings. By using forced alignment and temporal convolutions, we eliminate the need for multi-head attention-based fusion, greatly increasing computational efficiency but also raising performance. EfficientPunct sets a new state of the art, in terms of both performance and efficiency, with an ensemble that weights BERT's purely language-based predictions slightly more than the multimodal network's predictions.

Index Terms: speech recognition, punctuation restoration, multimodal learning

1. Introduction

Automatic speech recognition (ASR) systems' transformation of audio into text opens up possibilities for a wide range of downstream tasks. With natural language text, applications like machine translation and voice assistants are enabled. However, raw ASR outputs lack punctuation and hence the full meaning of texts, which must be restored for usage by the aforementioned tasks. To illustrate the importance of punctuation, consider how the meaning of the sentence, "I have a favorite, family," differs drastically from the unpunctuated version, "I have a favorite family." Punctuation restoration is therefore also important for readability of transcribed speech and accuracy of conveyed message.

Following the standard of the punctuation restoration task, we focus on three key punctuation marks which most commonly occur and play critical roles in language: commas (,), full stops (.), and question marks (?). We also consider no punctuation (NP) as a fourth class in need of our model's consideration.

1.1. Related work

Many works and proposed architectures have been devoted to restoring punctuation, and two main research categories have emerged: (1) considering only text output from ASR, and (2) considering both text output from ASR and the original audio.

Most consider text only, effectively forming a natural language processing task. They usually train and evaluate on the benchmark, textual datasets from IWSLT 2011 and 2012. Researchers have studied a wide variety of methods, including n-gram models [1], recurrent neural networks [2, 3, 4], adversarial models [5], contrastive learning [6], and transformers [7, 8]. Conditional random fields [9, 10, 11, 12] had particularly notable success. Direct fine-tuning of BERT [13] has also proven effective, which we demonstrate in Section 4.1.

In the other category, both audio and text modalities are considered. Earlier techniques involved statistical models like finite state machines [14], but unsurprisingly, more recently we see the exploration of neural networks [15, 16] and re-purposing existing models to take audio-based input and predict punctuation [17, 18]. Current state of the art models begin in separate branches: one to tokenize and process text and the other to process raw audio waveforms. They then use the attention mechanism [19] to fuse text and acoustic embeddings [20, 21].

1.2. Significance of multimodal approach

Despite research in multimodal punctuation restoration being far less numerous than the text-only category, [17] explicitly demonstrated the value of added acoustic information. Intuitively, audio provides more diverse features from which models may learn [22]. As a simple example, long pauses in speech are definitive indicators of a full stop's (.) occurrence. Similarly, shorter pauses may indicate a comma (,), and rising pitch is often associated with question marks (?).

The substantial benefit of involving both the transcribed text and original speech audio is that, in practical applications, we can design a highly streamlined system for restoring punctuation. Speech can first be transcribed into text by forward-passing audio signals through an ASR network, but one may preserve a hidden layer's latent representation for further usage as input (along with the transcribed text's embeddings) to a separate punctuation model. Then, the concatenated input would embed not only textual information, but also acoustics and prosody.

Our work is precisely motivated by this potential for high-speed punctuation labeling after receiving ASR output. We present EfficientPunct, a model that surpasses state of the art performance while requiring far fewer parameters, enabling practical usage.

2. Method

We formulate the problem as follows. We are given spoken audio signal $\mathbf{a} = (a_1, a_2, \ldots, a_S)$ and transcription words $\mathbf{t} = (t_1, t_2, \ldots, t_W)$. Here, S is the number of samples in the audio, and W is the number of words. The goal is to predict punctuation labels $\mathbf{y} = (y_1, y_2, \ldots, y_W)$ that follow each word, where each $y_i \in \{\texttt{","}, \texttt{"."}, \texttt{"?"}, \texttt{NP}\}$.

As illustrated in Figure 1, EfficientPunct begins in two branches which separately process the audio signal $\mathbf{a}$ and transcription text $\mathbf{t}$. Their details are as follows.

2.1. Text encoder

First, the text sequence $\mathbf{t}$ is passed through the default WordPiece tokenizer used by BERT. Then, using a pre-trained BERT model which we have fine-tuned for predicting the four previously described punctuation classes, we obtain final hidden layer text embeddings

$$H_t = \text{BERT}(\mathbf{t}). \tag{1}$$

H_t is a matrix whose columns are 768-dimensional vectors and represent embeddings of tokens. These text embeddings contain each token's context-aware information about grammar and linguistics.

2.2. Audio encoder

To process raw spoken audio waveforms and obtain meaningful acoustic embeddings, we use a pre-trained model built using the Kaldi speech recognition toolkit [23]. This is directly analogous to previous works' usage of wav2vec 2.0 [24] as their pre-trained audio encoder. Kaldi's TED-LIUM 3 [25] framework first extracts Mel frequency cepstral coefficients (MFCCs) [22] and i-vectors, which are then passed to a time-delay neural network for speech recognition. We extract the 12th layer's representation of the input audio for further usage in the punctuation model:

$$H_a = \text{KaldiTedlium12}(\mathbf{a}). \tag{2}$$

H_a is a matrix whose columns are 1024-dimensional embedding vectors. The number of columns is equal to the number of frames in the original audio.

2.3. Alignment and fusion

The first step of fusing the 768-dimensional embedding vectors from H_t and the 1024-dimensional embedding vectors from H_a is to find correspondences between columns in each matrix. In other words, we must determine the text token being spoken during each frame of audio. This is performed through forced alignment. According to columns matched between the two modalities' embeddings, we concatenate them into columns of 1792-dimensional embedding vectors. To fuse the two concatenated portions of each vector, we use a linear layer to learn affine transformations of embeddings which may be useful to punctuation restoration.

Many related works opt for attention-based fusion of the two modalities, but we found forced alignment and a simple linear layer to be the most parameter-efficient and competitive approach. Through experiments, we determined that more sophisticated fusion methods were counterproductive.

2.4. Time-delay neural network

Next, the fused embeddings are passed through a time-delay neural network (TDNN) [26]. It contains a series of 1D convolution layers to capture temporal properties of the features, with a gradually decreasing number of channels. At the last convolution layer, there are 4 channels, with each one corresponding to a punctuation class. The channels are passed through two linear layers with weights and biases shared among the channels to output 4 values for softmax activation.

2.5. Ensemble method

To complete EfficientPunct, we create an ensemble of the main TDNN and predictions using BERT's text embeddings only. We pre-trained BERT using the dark- and light-blue modules in Figure 1, which can still be used at inference time to obtain a set of predictions that only consider text, grammar, and linguistics. The other set of predictions obtained from the TDNN consider both text and audio.

Let $\alpha \in [0, 1]$ be the weight assigned to the TDNN's predictions and $1 - \alpha$ be the weight assigned to BERT's predictions. Our final predicted punctuation will be

$$f(\mathbf{a}, \mathbf{t}, \alpha) = \arg\max \left[\alpha y_a + (1 - \alpha) y_t \right], \tag{3}$$

where y_a is the TDNN's softmax values and y_t is BERT's softmax values. Essentially, if either the TDNN or BERT outputs a maximum class probability much lower than 1, then the other model may help resolve the ambiguity in predicting a punctuation mark.
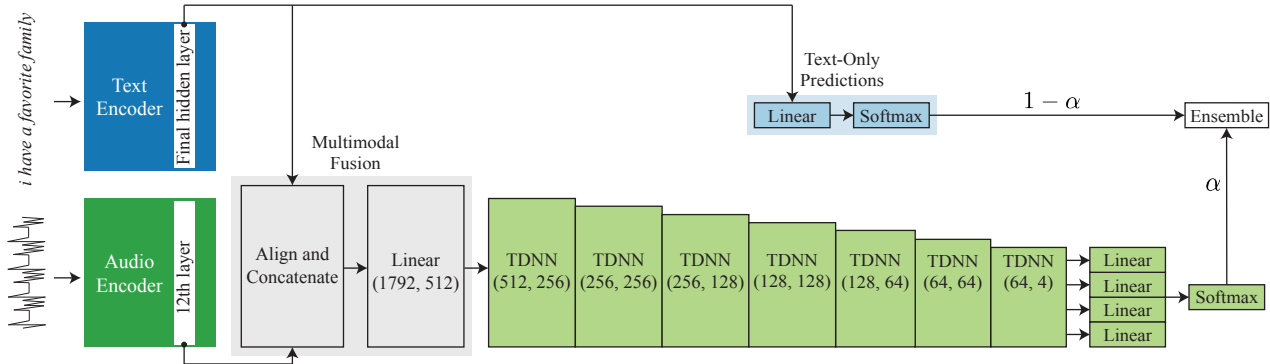
Figure 1: *The EfficientPunct framework. The top branch predicts using text only, while the bottom branch predicts using text and audio.*

3. Experiments

3.1. Data

Our primary dataset is the publicly available MuST-C version 1 [27], the same as that used by UniPunc [21] for sake of fair comparison. This dataset was compiled using TED talks. We also use same training and test set splits as the original authors, whose information is available on GitHub. We further split the original training set into 90% for training and 10% for validation. Please see Table 1 for full information.

Table 1: *Training, validation, and test set information*

Set	Number of samples	Total duration (h)
Training	92,723	392.0
Validation	10,301	43.5
Test	490	2.8

Each sample is an English audio piece of approximately $10\,\text{s}$ to $30\,\text{s}$ with the corresponding transcription text. In Kaldi, we use a frame duration of $10\,\text{ms}$ for MFCCs, i-vectors, and 12th layer acoustic embeddings. We follow the procedure described in Section 2.3 to generate a matrix of aligned embeddings for each data sample. Then, to obtain examples for training and inference, we consider segments of 301 frames, or $3\,\text{s}$, wherein the exact middle frame is the point of transition from one text token to the next. The resulting example will thus be labeled with the punctuation following the prior token and occurring at the middle frame. We use a context window of $3\,\text{s}$, because this duration should be sufficient to capture all acoustic and prosodic information relevant to a punctuation mark, such as pauses and pitch rises. At the same time, this duration is not so long as to include much unnecessary information, such as extensions into adjacent words.

For the entire dataset, punctuation label distributions were as follows. Due to the highly imbalanced nature of the dataset, we sampled less occurring classes more frequently for training so that, in effect, all class counts are equal, and the network avoids learning only the prior probability distribution.

Table 2: *Punctuation label distributions*

Label	Number of examples	% of total
No punctuation (NP)	3,567,572	86.9%
Comma (,)	280,446	6.8%
Full stop (.)	238,213	5.8%
Question mark (?)	20,897	0.5%

Moreover, since BERT was already pre-trained on massive corpora, we fine-tune it for punctuation prediction using the National Speech Corpus [28] of Singaporean English, in addition to MuST-C.

3.2. Training

To fine-tune BERT and pre-train the text encoder, we place two linear layers on top of the base, uncased BERT's last hidden layer for four-way classification. For the pre-trained audio encoder, we use the TED-LIUM 3 [25] framework in Kaldi.

Our main TDNN module for punctuation restoration comprises seven 1-dimensional convolution layers, with said dimension spanning across time. Figure 1 shows the number of input and output channels of each layer. The kernel sizes used are, in order: $9, 9, 5, 5, 7, 7, 5$, alternating between no dilation and a dilation of 2. The stride was kept at 1 in all layers. Additionally, we apply ReLU activation and batch normalization [29] to the output of each layer. We trained using stochastic gradient descent [30] with learning rate 0.00001 and momentum 0.9, instead of the typically used Adam optimizer [31]. This allowed for greater generalizability but still reasonable training speed [32].

To experiment with our ensemble, we explored the effect of varying α, the weight assigned to the TDNN for final predictions. $1 - \alpha$ is the weight assigned to BERT.

In Section 4, we report results for $\alpha = 0.3$ to $\alpha = 0.7$ in 0.1 increments.

We used a standard Linux computing environment hosted on Google Cloud Platform with a single NVIDIA Tesla P100 GPU. Training took roughly 2 days, and inference can be performed on CPU-only machines 50 times faster than real time, or in about 0.02 seconds per second of audio.

4. Results

Our results reported in Table 3 includes a comparison with current state of the art (SOTA) and best-performing models, MuSe [20] and UniPunc [21]. We also divide the reporting of EfficientPunct's results into three categories:

1. *EfficientPunct-BERT* considers text only, which is equivalent to the fine-tuned BERT model.

2. *EfficientPunct-TDNN* considers text and audio via our TDNN.

3. *EfficientPunct* is an ensemble of predictions from categories (1) and (2) with $\alpha = 0.4$, the best performing weight as reported in Section 4.2.

Categories (1), (2), and (3) are reported in the third, fourth, and fifth rows of Table 3, respectively.

As is standard in punctuation restoration research, we report the F1 scores of commas, full stops, and question marks. The "overall" F1 score aggregates these while considering the imbalanced classes' varying numbers of examples. We also state each model's number of parameters to provide an indication of computational efficiency.

4.1. EfficientPunct and submodules

Our main EfficientPunct model achieves an overall F1 score of 79.5, outperforming all current state of the art frameworks by 1.0 or more points. We also achieve highest F1 scores for each individual punctuation mark, with the most significant improvement occurring for question marks. These were accomplished with EfficientPunct using less than half of UniPunc's total number of parameters, which achieved the previous best results. The significant improvement in recognizing question marks may be attributed to our audio encoder, Kaldi's TED-LIUM 3 framework, aiming explicitly at phone recognition. In this process, the acoustics surrounding question marks may be more pronounced in the embedding representation than other acoustics models.

Even more lightweight models are EfficientPunct-BERT and EfficientPunct-TDNN. EfficientPunct-BERT is simply a concatenation of two linear layers and a softmax layer on top of BERT. With the incorporation of audio features, we observe that EfficientPunct-TDNN indeed performs slightly better.

These results validate the strength of TDNNs, traditionally used in speech and speaker recognition, in punctuation restoration. UniPunc and MuSe both used attention-based mechanisms for fusing text and acoustic embeddings, but alignments learned as such rely on trainable attention weights. Our forced alignment strategy likely generated more precise temporal matches between text and audio. Combined with a TDNN architecture, we achieved a significantly more efficient model.

4.2. Ensemble weights

In this section, we observe the effect of ensemble weights on EfficientPunct's performance. Equation 3 details the role of α in weighting predictions made by the TDNN and BERT, with $\alpha = 0$ meaning pure consideration of BERT, and $\alpha = 1$ meaning pure consideration of the TDNN.

Table 4 reports the effect of α on model performance. When both BERT and the TDNN play an approximately equal role in the ensemble, a fair voting mechanism is enabled, and we achieve the highest F1 scores. However, notice that $\alpha = 0.4$, a weight that considers BERT slightly more strongly than the TDNN, achieves the maximum overall F1. This gain comes mostly from sharper comma predictions, which present notorious difficulties due to varying grammatical and (transcription) writing styles. We reason that $\alpha = 0.4$ excels, because a stronger reliance on BERT's language modeling perspective yields more linguistically correct punctuation, as agreed upon by countless writers' contributions to BERT's training corpora.

The strength of our ensemble method is that, in cases of uncertain predictions by either party, i.e. approximately equal softmax probabilities over all classes, the other can provide guidance to clarify the ambiguity. This process demands very little additional parameters through which the input must be passed, as shown by the last two rows of Table 3, but greatly advances state of the art performance.

4.3. Parameter Breakdown

In order to show the specific modules in which we attain superior efficiency, we further break down the parameters count from the last column of Table 3. In Table 5, we detail the number of parameters devoted by each model to extracting embeddings and inferring those embeddings to make punctuation decisions.

EfficientPunct requires much less computational cost in both the embedding extraction and inference stages. Our usage of Kaldi's TED-LIUM 3 model brought massive efficiency gains compared to MuSe and UniPunc's usage of wav2vec 2.0. Moreover, our inference module uses less than a tenth of UniPunc's parameters in the same stage, which achieved the previous best results.

---

[1]Statistics taken directly from UniPunc paper due to public inaccessibility of certain models hindering our ability to run them. Fairness of comparison is ensured, since we use the exact same training and test sets as the UniPunc authors.

[2]Number of parameters in MuSe was conservatively estimated from information provided in the original paper.

Table 3: *F1 scores of EfficientPunct and its various submodules on each punctuation type, compared against existing state of the art (SOTA) models. EfficientPunct-BERT considers text only, EfficientPunct-TDNN considers text and audio, and EfficientPunct predicts using an ensemble of the prior two.*

	Model	Embedding Type(s) Used	Comma	Full Stop	Question	Overall	Number of Parameters
SOTA	MuSe[1,2]	BERT, wav2vec 2.0	73.2	83.6	79.4	77.9	1.7×10^8
	UniPunc[1]	BERT, wav2vec 2.0	74.2	83.7	80.8	78.5	2.5×10^8
Ours	EfficientPunct-BERT	BERT	73.4	83.9	84.7	78.4	1.1×10^8
	EfficientPunct-TDNN	BERT, TED-LIUM 3	74.3	83.6	85.8	78.5	1.2×10^8
	EfficientPunct (Ensemble)	BERT, TED-LIUM 3	**75.4**	**84.3**	**86.5**	**79.5**	1.2×10^8

Table 4: *F1 scores for different α weights*

α	Comma	Full Stop	Question	Overall
0.3	75.0	84.1	86.3	79.2
0.4	75.4	84.3	86.5	**79.5**
0.5	75.0	84.0	86.5	79.1
0.6	75.0	83.8	86.2	79.0
0.7	74.8	83.8	85.8	78.9

Table 5: *Number of parameters required in various stages of each model*

Model	Embedding Network	Inference Network	Total
MuSe	1.6×10^8	4.3×10^6	1.7×10^8
UniPunc	2.0×10^8	4.8×10^7	2.5×10^8
EfficientPunct	$\mathbf{1.1 \times 10^8}$	$\mathbf{3.0 \times 10^6}$	$\mathbf{1.2 \times 10^8}$

5. Conclusion

In this paper, we explored the application of time-delay neural networks in punctuation restoration, which proved to be more computationally efficient than and as effective as previous approaches. Combined with BERT in an ensemble, EfficientPunct establishes a strong, new state of the art with a fraction of previous approaches' number of parameters. A key factor of our model's success is removing the need for attention-based fusion of text and audio features. In previous approaches, multiple attention heads added extraordinary overhead in the punctuation prediction stage. We demonstrated that forced alignment of text and acoustic embeddings, in conjunction with temporal convolutions, rendered attention unnecessary.

Additionally, we studied the effect of different weights assigned to members of the ensemble. We found that a slightly stronger weighting of BERT against the multimodal TDNN optimized performance by emphasizing language rules associated with punctuation.

In future works, the effectiveness of jointly training ensemble weights and the TDNN may be examined, which could allow the learning of an optimal ensemble.

Jointly training with the text and audio encoders may also be considered, but this procedure should not inhibit the encoders' generalizability for purposes other than punctuation restoration. Finally, we would like to explore the applicability of EfficientPunct in more languages and a similar framework for other post-processing tasks of speech recognition.

6. References

[1] A. Gravano, M. Jansche, and M. Bacchiani, "Restoring punctuation and capitalization in transcribed speech," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 4741–4744.

[2] O. Tilk and T. Alumäe, "Bidirectional recurrent neural network with attention mechanism for punctuation restoration," in *Interspeech*, 2016, pp. 3047–3051.

[3] S. Kim, "Deep recurrent neural networks with layer-wise multi-head attentions for punctuation restoration," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 7280–7284.

[4] W. Salloum, G. Finley, E. Edwards, M. Miller, and D. Suendermann-Oeft, "Deep learning for punctuation restoration in medical reports," in *BioNLP 2017*, 2017, pp. 159–164.

[5] W. Wang, Y. Liu, W. Jiang, and Y. Ren, "Making punctuation restoration robust with disfluency detection," in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design*, 2022, pp. 395–399.

[6] Q. Huang, T. Ko, H. L. Tang, X. Liu, and B. Wu, "Token-level supervised contrastive learning for punctuation restoration," in *Interspeech*, 2021, pp. 2012–2016.

[7] M. Courtland, A. Faulkner, and G. McElvain, "Efficient automatic punctuation restoration using bidirectional transformers with robust inference," in *Proceedings of the 17th International Conference on Spoken Language Translation*, 2020, pp. 272–279.

[8] T. Alam, A. Khan, and F. Alam, "Punctuation restoration using transformer models for high-and low-resource languages," in *Proceedings of the 2020 EMNLP Workshop W-NUT: The Sixth Workshop on Noisy User-generated Text*, 2020, pp. 132–142.

[9] W. Lu and H. T. Ng, "Better punctuation prediction with dynamic conditional random fields," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010, pp. 177–186.

[10] T. B. Nguyen, Q. M. Nguyen, T. T. H. Nguyen, Q. T. Do, and C. M. Luong, "Improving vietnamese named entity recognition from speech using word capitalization and punctuation recovery models," in *Interspeech*, 2020, pp. 4263–4267.

[11] H. T. T. Uyen, N. A. Tu, and T. D. Huy, "Vietnamese capitalization and punctuation recovery models," in *Interspeech*, 2022, pp. 3884–3888.

[12] N. Ueffing, M. Bisani, and P. Vozila, "Improved models for automatic punctuation prediction for spoken and written text," in *Interspeech*, 2013, pp. 3097–3101.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT 2019*, 2019, pp. 4171–4186.

[14] H. Christensen, Y. Gotoh, and S. Renals, "Punctuation annotation using statistical prosody models," in *ITRW on Prosody in Speech Recognition and Understanding*, 2001.

[15] O. Tilk and T. Alumäe, "Lstm for punctuation restoration in speech transcripts," in *Interspeech*, 2015, pp. 683–687.

[16] O. Klejch, P. Bell, and S. Renals, "Sequence-to-sequence models for punctuated transcription combining lexical and acoustic features," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5700–5704.

[17] J. Yi and J. Tao, "Self-attention based model for punctuation prediction using word and speech embeddings," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 7270–7274.

[18] O. Klejch, P. Bell, and S. Renals, "Punctuated transcription of multi-genre broadcasts using acoustic and lexical approaches," in *2016 IEEE Spoken Language Technology Workshop*, 2016, pp. 433–440.

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

[20] M. Sunkara, S. Ronanki, D. Bekal, S. Bodapati, and K. Kirchhoff, "Multimodal semi-supervised learning framework for punctuation prediction in conversational speech," in *Interspeech*, 2020, pp. 4911–4915.

[21] Y. Zhu, L. Wu, S. Cheng, and M. Wang, "Unified multimodal punctuation restoration framework for mixed-modality corpus," in *2022 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2022, pp. 7272–7276.

[22] H. Beigi, *Fundamentals of Speaker Recognition*. New York, NY, USA: Springer, 2011.

[23] 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 *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, 2011.

[24] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems*, 2020, pp. 12 449–12 460.

[25] F. Hernandez, V. Nguyen, S. Ghannay, N. Tomashenko, and Y. Esteve, "Ted-lium 3: Twice as much data and corpus repartition for experiments on speaker adaptation," in *20th International Conference on Speech and Computer*, 2018, pp. 198–208.

[26] "Phoneme recognition using time-delay neural networks," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1989, pp. 328–339.

[27] R. Cattoni, M. A. Di Gangi, L. Bentivogli, M. Negri, and M. Turchi, "Must-c: A multilingual corpus for end-to-end speech translation," in *Computer Speech & Language Journal*, 2020.

[28] J. X. Koh, A. Mislan, K. Khoo, B. Ang, W. Ang, C. Ng, and Y.-Y. Tan, "Building the singapore english national speech corpus," in *Interspeech*, 2019, pp. 321–325.

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[30] H. Robbins and S. Monro, "A stochastic approximation method," in *The Annals of Mathematical Statistics*, 1951, pp. 400–407.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2015.

[32] A. Gupta, R. Ramanath, J. Shi, and S. S. Keerthi, "Adam vs. sgd: Closing the generalization gap on image classification," in *OPT2021: 13th Annual Workshop on Optimization for Machine Learning*, 2021.