TEMPORAL EARLY EXITING FOR STREAMING SPEECH COMMANDS RECOGNITION

Raphael Tang,[1] *Karun Kumar,*[1] *Ji Xin,*[2] *Piyush Vyas,*[*] *Wenyan Li,*[*] *Gefei Yang,*[1] *Yajie Mao,*[1]
Craig Murray,[1] *Jimmy Lin*[2]

[1]Comcast Applied AI [2]University of Waterloo

ABSTRACT

Limited-vocabulary speech commands recognition is the task of classifying a short utterance as one of several speech commands, for which neural networks obtain state-of-the-art results. In particular, recurrent neural networks represent a common approach for streaming commands recognition systems. In this paper, we explore resource-efficient methods to short-circuit such systems in the time domain when the model is confident in its prediction. We propose applying a frame-level labeling objective to further improve the efficiency–accuracy trade-off. On two datasets in limited-vocabulary commands recognition, our best method achieves an average time savings of 45% of the utterance without reducing the absolute accuracy by more than 0.6 points. We show that the per-instance savings depend on the length of the unique prefix in the phonemes across a dataset.

Index Terms— speech commands, early exiting, recurrent neural networks.

1. INTRODUCTION

Automatic speech recognition (ASR) systems power billions of devices, from the Amazon Echo to Google's voice assistant. Driven by advances in deep learning [1], they attain high precision and coverage of the English language, with the state of the art having a vocabulary size of thousands of wordpieces and a word error rate of 2.8% on LibriSpeech [2].

Why, then, is the limited-vocabulary case still interesting? For many voice-enabled platforms, queries follow a highly Zipfian distribution, where the top few dozen cover a large percentage of the total traffic. On the Comcast X1 entertainment system, for example, the top-twenty commands constitute around 30% of the traffic, amounting to a few million queries per day. Thus, the limited-vocabulary task remains relevant, as well as separate: clearly, using an ASR system is excessive for targeting phonetically distinct commands with a small vocabulary. The model complexity should match the task difficulty. Toward this limited-vocabulary task, researchers have developed lightweight, efficient neural models that fit on resource-constrained devices [3].

[*]Work done while employed by Comcast.

Even though these models are already low latency, the right kind of optimization is still meaningful. For example, we can drastically reduce the overall system latency by short-circuiting these models across time when they are confident in their predictions, since downstream components in query understanding systems often depend on their transcriptions. We can use the early-exited transcription and perform various downstream tasks before the user has finished speaking, thus saving time. In our television entertainment system domain, for instance, a downstream task is to check the availability of a desired channel for the user, requiring slow network calls and database queries.

In this paper, we precisely explore this temporal-early-exiting strategy for streaming, limited-vocabulary commands recognition. To further improve the savings–accuracy trade-off curves, we propose a simple yet novel frame-level labeling objective that encourages earlier exits. We evaluate our approaches on two datasets—one open dataset in simple commands recognition and another in the television entertainment system domain. Our main contributions are as follows: First, we are the first to elucidate and study early exiting *in the time domain* for speech commands recognition. Second, we propose a novel method for encouraging earlier exits and improving overall model quality. Finally, we provide insight into the early exit points, showing that the per-example savings depend on the length of the smallest unique prefix in the phonetic transcription. Our best proposed method achieves an average savings of 45% without hurting the absolute accuracy by more than 0.6 points.

2. OUR MODELING APPROACHES

We use a standard recurrent neural network (RNN) encoder architecture to model speech. Given 16kHz, 16-bit audio, we construct 60-dimensional log-Mel frames with a window of 30ms and frame shift of 10ms. Similar to previous work [4], we stack every three frames together for a downsampled frame rate of 30ms. We feed these superframes $\boldsymbol{x} := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ to a unidirectional RNN, which consists of l layers, h hidden units, and either long short-term memory (LSTM) [5] or gated recurrent unit (GRU) [6] cells, depending on the hyperparameters. Finally, we pass the RNN's hidden states to a pointwise, two-layer deep neural

network (DNN) with h hidden units, rectified linear unit activations, and $|V|$ output units, producing the final hidden states $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T \in \mathbb{R}^{|V|}$, where V is the vocabulary. To produce a probability distribution over the vocabulary, we use the softmax function, i.e.,

$$p(\boldsymbol{y}_i[k]|\boldsymbol{x}_{1:i}) := \frac{\exp(\boldsymbol{h}_i[k])}{\sum_j \exp(\boldsymbol{h}_i[j])} \tag{1}$$

for $1 \leq i \leq T$ and $1 \leq k \leq |V|$.

2.1. Early-Exiting Inference Criterion

Similar to DeeBERT [7], we use the entropy of the output distribution as the early-exiting criterion during inference. For each i^{th} frame, let the frame-level entropy $H(\boldsymbol{x}_i)$ be

$$H(\boldsymbol{x}_i) := -\sum_{j=1}^{|V|} p(\boldsymbol{y}_i[j]|\boldsymbol{x}_{1:i}) \log p(\boldsymbol{y}_i[j]|\boldsymbol{x}_{1:i}), \tag{2}$$

The lower the entropy, the higher the confidence. Then, define the early-exiting threshold function as

$$g(\tau, \boldsymbol{x}) := \min\left(\{i : H(\boldsymbol{x}_i) \leq \tau\} \cup \{T\}\right), \tag{3}$$

where $\tau \in \mathbb{R}$ is the confidence (entropy) threshold and T is the number of frames in $\boldsymbol{x}$. Put simply, $g(\tau, \boldsymbol{x})$ is the smallest frame index where the entropy is below τ; if none exist, it returns the final frame index.

2.2. Connectionist Temporal Classification Objective

Connectionist temporal classification (CTC) [8] is an objective for modeling sequential label distributions when the frame-level alignment is unknown. Concretely, it augments the vocabulary V with a blank label $\langle b \rangle$, i.e., $V' := V \cup \langle b \rangle$. Given some ground truth $\boldsymbol{y} := (y_1, \ldots, y_U)$, it defines a sequence generating function $\mathcal{B}(\boldsymbol{y})$ that produces all strings (alignments) $\hat{\boldsymbol{y}} \in V'^T$ of length T such that, if all consecutive nonblank symbols were joined and subsequent blanks removed, $\hat{\boldsymbol{y}}$ would equal $\boldsymbol{y}$, e.g., $cc\langle b\rangle aaa\langle b\rangle\langle b\rangle t \mapsto cat$. Given some input $\boldsymbol{x}$, CTC then models the conditional probability marginalized over all possible alignments as

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{y}' \in \mathcal{B}(\boldsymbol{y})} p(\boldsymbol{y}'|\boldsymbol{x}), \tag{4}$$

where the probability distribution on the right is represented by the RNN encoder. The authors [8] provide a forward–backward algorithm for computing the loss gradients for gradient-based training, which, upon convergence, yields probable alignments for a label sequence.

For our task, we define the vocabulary as the set of classification classes, with each example having a label sequence length of one (a single class). At first glance, this objective seems to be appropriate because we can pick an exit point using the alignment. However, in practice, CTC results in peaky, overconfident predictions and overly exploits a single path [9]. Additionally, its application is atypical for classification tasks such as ours. Nevertheless, it serves as a relevant baseline approach.

2.3. Last-Frame Cross Entropy Objective

The typical approach to limited-vocabulary speech commands recognition is to produce from the utterance a single probability distribution across the labels, and then minimize the cross entropy (CE) loss given the ground truth [10, 11, 12]. For streaming systems, researchers commonly use the final hidden state of the RNN (or RNN-DNN) as the fixed-length representation [12] and apply a softmax transformation across the labels. The last-frame CE $\mathcal{L}_{LF}$ for a single example is

$$\mathcal{L}_{LF} := -\log p(\boldsymbol{y}_T[c]|\boldsymbol{x}), \tag{5}$$

where c is the ground truth label index and $\boldsymbol{y}_T$ is the final hidden state. Although the intermediate output distributions $p(\boldsymbol{y}_i|\boldsymbol{x}_{1:i})$ for all $1 \leq i < T$ are not explicitly trained, we show experimentally that this popular method produces acceptable early exits.

2.4. All-Frame Cross Entropy Objective

For improved early exiting, we propose to apply the cross entropy objective to *all* frames instead of only the last. This way, all hidden states are encouraged to be discriminative. That is, for a single example, the all-frame objective is

$$\hat{\mathcal{L}}_{AF} := -\frac{1}{T}\sum_{i=1}^{T} \log p(\boldsymbol{y}_i[c]|\boldsymbol{x}_{1:i}). \tag{6}$$

To weigh the importance of the final frame versus all the frames, we add the last-frame loss to the all-frame loss with weight λ, for a final objective of $\mathcal{L}_{AF} := \mathcal{L}_{LF} + \lambda\hat{\mathcal{L}}_{AF}$. Although it seems like such a loss would hurt the original accuracy because early frames may not contain the label, we show that it in fact improves the quality for a wide range of λ.

3. EXPERIMENTS

3.1. Experimental Setup and Data

We implemented our models in PyTorch 1.6.0. We conducted all experiments on GPU-accelerated machines with Nvidia Tesla V100 and Titan RTX GPUs.

For evaluation, we first chose the Google Speech Commands dataset (GSC; v1) [11], which comprises 65,000 one-second utterances split evenly across 30 phonetically distinct words. Being open and licensed under Creative Commons BY 4.0, this dataset enables easy reproducibility. Following

Dataset	Training/Dev/Test Sizes	C	L	P	N
GSC	(51K, 6.8K, 6.8K)	11	1.00	23.7K	41.0K
CC20	(109.7K, 13.7K, 13.7K)	21	2.04	40K	97.2K

Table 1. Summary statistics of the datasets, where C denotes the total number of classes, L the average length of the utterances in seconds, and P and N the numbers of positive and negative examples, respectively.

the previous literature, we picked the 10 positive keywords "yes," "no," "up," "down," "left," "right," "on," "off," "stop," and "go." We collapsed the rest of the keywords into the negative class for a total of 11 distinct labels. The training, development, and test sets were, following the original paper [11], distinctly split into sizes of 80%, 10%, and 10% of the dataset—see Table 1 for a summary of the dataset.

We further curated a proprietary dataset for our television entertainment system domain. The dataset comprised 40K positive samples, split evenly across the top-20 commands, and 100K negative samples, divided evenly across 6670 commands. The top-twenty commands represent around 30% of our total traffic. We grouped the negative examples into a single class, for a total of 21 classes, all of which were phonetically unique between the classes. The dataset was collected with the help of an auto-annotation tool, which annotated transcriptions by analyzing subsequent user behaviors and identifying patterns for query reformulation within a given session [13, 14]. That is, we labeled transcriptions as correct when users provided positive implicit feedback (e.g., button click, user stayed on the program and continued watching). This auto-annotation process yields examples with very low word error rates [13], thus providing a reliable source for training data. We again bucketed the training, development, and test sets into 80%, 10%, and 10% of the dataset, respectively—Table 1 summarizes the dataset statistics. We name this dataset CC20.

For our hyperparameters, we chose a batch size of 64 and a learning rate of 5×10^{-4} using the Adam optimizer [15] with an exponential decay factor of 0.985 across a maximum of 40 epochs. We tuned the hidden size of the RNN and the number of layers across a grid of $\{384, 512\}$ hidden units and $\{1, 2\}$ layers. For efficiency, we performed this hyperparameter tuning on GSC's development set and used the same values on CC20. We then fixed the architecture and applied the all-frame objective with $\lambda \in \{0.1, 0.5, 2.5\}$. To improve the robustness of the model to noise, we follow the same procedure from the Howl keyword spotting toolkit [12] and randomly mix Gaussian noise $\sim$ NORMAL$(0, 0.02)$ with 0.2 probability at each training step, along with noise from MUSAN [16] and Microsoft SNSD [17], with a mixing factor of 0.1.

We assign meaningful names to the resulting models. The base model name is "{LSTM, GRU}-*h*-*l*," followed by the objective—CTC, LF (last-frame CE), or AF-λ (all-frame CE).

#	Model	GSC Dev/Test	CC20 Dev/Test
1	LSTM [12]	94.3/94.5	–
2	RNN [18]	–/95.6	–
3	GRU-384-1 (CTC)	92.8/93.1	71.5/71.6
4	GRU-384-1 (LF)	95.5/96.0	97.9/98.1
5	GRU-384-1 (AF-0.1)	95.8/96.7	98.1/98.2
6	GRU-384-1 (AF-0.5)	**96.4/96.7**	**98.3/98.4**
7	GRU-384-1 (AF-2.5)	96.2/96.5	98.1/98.1

Table 2. The model accuracy without early exiting.

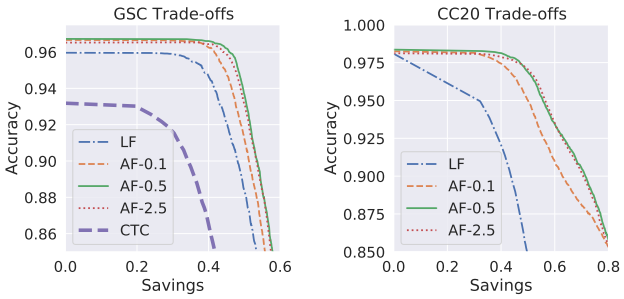

Fig. 1. The savings–accuracy trade-offs on the test sets.

3.2. Results

First, as a sanity check, we present our model results *without* early exiting in Table 2. All models have 670K parameters. Our models achieve comparable results to streaming state-of-the-art models (rows 1 and 2). We also try beam search decoding for the CTC model with a beam width of 100, but taking the strongest activation at each time step (i.e., greedy decoding) obtains the same quality. We note that our CTC model converges poorly on CC20, possibly because of more frame-level phonetic confusion between the examples (e.g., "recordings" and "recording pro"), which is a nonissue for GSC. Interestingly, the all-frame objective consistently outperforms LF and CTC by 0.1–0.7 points across a wide range of λ values—see rows 5–7 and 3 and 4. We conjecture that the all-frames objective serves as a form of regularization similar to deeply supervised networks [19], whose hidden features are explicitly trained to be discriminative. In our case, we can regard our objective as deep supervision on the final features across time, not the layers.

Next, we follow the early exiting inference strategy as described in Section 2.1, sweeping the entropy threshold τ from 0 to 1 in increments of $1/300$. For each operating point τ, we compute the time savings $\Theta(\tau)$ as the average proportion of the speech utterance that early exiting truncates, i.e.,

$$\Theta(\tau) = \frac{1}{n} \sum_{i=1}^{n} \frac{T_i - g(\tau, \boldsymbol{x}^{(i)})}{T_i}, \tag{7}$$

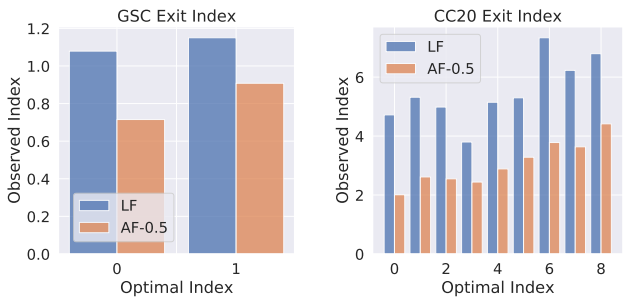

Fig. 2. The observed phonetic exit index against the optimal index. The confidence intervals are omitted because the estimates are precise.

where T_i is the length in frames of the i^{th} example $\boldsymbol{x}^{(i)}$, and $g(\tau, \boldsymbol{x}^{(i)}) \leq T_i$ is the exit time index given threshold τ and example $\boldsymbol{x}^{(i)}$, as defined in Eqn. (3). A time savings of 0.4 thus means that, on average, the model exits after observing the first 60% of the audio clip.

We plot the resulting time-savings-to-accuracy curves in Figure 1, with an accuracy cutoff of 85% on both datasets. The AF models achieve similar efficacy, with all of their curves being Pareto-better than the others. In particular, picking $\lambda = 0.5$ obtains the highest area under the curve (AUC) for both datasets; at a savings of 0.45, we lose only 0.5 and 0.6 points in absolute accuracy on GSC and CC20, respectively. The AF curves also fall off more gradually than the others, suggesting that the objective smooths the predictions across time.

3.3. Early-Exiting Analysis

We hypothesize that the exit index is smaller for instances with shorter unique phonetic prefixes in the dataset. For negative instances, the unique prefix is compared against the positive set only, because all of the negatives form a single class. For example, consider a dataset with the four transcriptions "Sophie," "Soho," "Ralph," and "Ralphie." Suppose Sophie and Soho are positives and the others negative. Intuitively, Sophie and Soho require more time to disambiguate than does Ralph and Ralphie, which can be classified as negative based on the first phoneme alone, since the negatives form a single class and hence do *not* require interclass distinction.

To test this hypothesis, we first build a pronunciation dictionary by applying the English pretrained finite state transducer–based grapheme-to-phoneme (G2P) model from the Montreal Forced Aligner (MFA) toolkit [20]. We additionally supplement the dictionary with the LibriSpeech lexicon. Next, we align the transcriptions with the audio using MFA, which outputs the most likely phoneme sequence and its constituent time intervals. For each transcription, we then compute the optimal exit point as the smallest unique phonetic prefix, taken across the entire dataset for positive examples and the positive set for negatives. In the aforementioned example, the optimal points would be "Soph·ie," "Soh·o," "R·alph,", and "R·alphie." Finally, we note the observed phonetic exit index for each clip in the test set.

For our analysis, we select the models trained on the LF and the AF-0.5 objectives, representing the vanilla method and our best approach, respectively. To enable a direct comparison, we pick the thresholds so that the accuracy between the two models are matched—93.6% on GSC and 95% on CC20. We plot the optimal index against the observed exit index in Figure 2. The observed exit index increases with the optimal exit point on both models and datasets, which supports our hypothesis. In agreement with the savings–accuracy curves in Figure 1, the AF-0.5 model consistently exits earlier than the LF one, as the lower bars show.

4. BACKGROUND AND RELATED WORK

Previous papers in the early exiting of neural networks perform layerwise exiting, not across time, as in our case. Final predictions are made based on intermediate hidden layers, with the help of "early exit modules" trained for each layer. If an intermediate exit module is certain enough in the final prediction, the model short-circuits and halts inference at that layer. Various examples include BranchyNet [21] for image classification, DeeBERT [7] for natural language processing, and Tyagi et al.'s work [22] for speech intent. Along a parallel direction, Tang et al. [23] explore predicting the final voice query from the intermediate ASR system outputs, an approach that we can readily combine with temporal early exiting for greater effectiveness.

A multitude of modeling approaches exist for limited-vocabulary speech commands recognition, such as convolutional neural networks [10], residual networks, and recurrent neural networks [18]. Since our focus is on streaming systems, we use RNNs as the target architecture. To compress these models further, researchers have explored neural architecture search [3] and dilated convolutions [24]. He et al. [4] also explore compressing an ASR model and achieve promising results, but their system is orders of magnitude larger than limited speech commands models.

5. CONCLUSIONS

In this paper, we explore early exiting across time for RNNs in speech commands recognition. We propose a simple objective for improving the efficiency–accuracy trade-off. On a few datasets in limited-vocabulary speech commands recognition, we obtain a time savings of 45% without dropping the accuracy by more than 0.6 points. We show that the savings correlate with how easy the phonetics are to disambiguate.

6. REFERENCES

[1] Dong Yu and Li Deng, *Automatic Speech Recognition: A Deep Learning Approach*, Springer Publishing Company, Incorporated, 2014.

[2] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proceedings of Interspeech*, 2019.

[3] Tom Véniat, Olivier Schwander, and Ludovic Denoyer, "Stochastic adaptive neural architecture search for keyword spotting," in *Proceedings of ICASSP*, 2019.

[4] Yanzhang He, Tara N. Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al., "Streaming end-to-end speech recognition for mobile devices," in *Proceedings of ICASSP*, 2019.

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

[6] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proceedings of the NIPS 2014 Workshop on Deep Learning*, 2014.

[7] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin, "DeeBERT: Dynamic early exiting for accelerating BERT inference," in *Proceedings of ACL*, 2020.

[8] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of ICML*, 2006.

[9] Hongzhu Li and Weiqiang Wang, "Reinterpreting CTC-based training as iterative fitting," *Pattern Recognition*, p. 107392, 2020.

[10] Mengjun Zeng and Nanfeng Xiao, "Effective combination of DenseNet and BiLSTM for keyword spotting," *IEEE Access*, vol. 7, pp. 10767–10775, 2019.

[11] Pete Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv:1804.03209*, 2018.

[12] Raphael Tang, Jaejun Lee, Afsaneh Razi, Julia Cambre, Ian Bicking, Jofish Kaye, and Jimmy Lin, "Howl: A deployed, open-source wake word detection system," *arXiv:2008.09606*, 2020.

[13] Wenyan Li and Ferhan Ture, "Auto-annotation for voice-enabled entertainment systems," in *Proceedings of SIGIR*, 2020.

[14] Raphael Tang, Ferhan Ture, and Jimmy Lin, "Yelling at your TV: An analysis of speech recognition errors and subsequent user behavior on entertainment systems," in *Proceedings of SIGIR*, 2019.

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

[16] David Snyder, Guoguo Chen, and Daniel Povey, "MUSAN: A music, speech, and noise corpus," *arXiv:1510.08484*, 2015.

[17] Chandan K. A. Reddy, Ebrahim Beyrami, Jamie Pool, Ross Cutler, Sriram Srinivasan, and Johannes Gehrke, "A scalable noisy speech dataset and online subjective test framework," *Proceedings of Interspeech*, 2019.

[18] Douglas Coimbra de Andrade, Sabato Leo, Martin Loesener Da Silva Viana, and Christoph Bernkopf, "A neural attention model for speech command recognition," *arXiv:1808.08929*, 2018.

[19] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu, "Deeply-supervised nets," in *Proceedings of AISTATS*, 2015.

[20] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger, "Montreal Forced Aligner: Trainable text-speech alignment using Kaldi.," in *Proceedings of Interspeech*, 2017.

[21] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proceedings of ICPR*, 2016.

[22] Akshit Tyagi, Varun Sharma, Rahul Gupta, Lynn Samson, Nan Zhuang, Zihang Wang, and Bill Campbell, "Fast intent classification for spoken language understanding systems," in *Proceedings of ICASSP*, 2020.

[23] Raphael Tang, Karun Kumar, Kendra Chalkley, Ji Xin, Liming Zhang, Wenyan Li, Gefei Yang, Yajie Mao, Junho Shin, G. Craig Murray, and Jimmy Lin, "Voice query auto completion," in *Proceedings of EMNLP*, 2021.

[24] Alice Coucke, Mohammed Chlieh, Thibault Gisselbrecht, David Leroy, Mathieu Poumeyrol, and Thibaut Lavril, "Efficient keyword spotting using dilated convolutions and gating," in *Proceedings of ICASSP*, 2019.