UNIFIED SPECULATION, DETECTION, AND VERIFICATION KEYWORD SPOTTING

Geng-shen Fu, Thibaud Senechal, Aaron Challenner, Tao Zhang

Amazon Alexa Science, Cambridge, MA, USA

ABSTRACT

Accurate and timely recognition of the trigger keyword is vital for a good customer experience on smart devices. In the traditional keyword spotting task, there is typically a trade-off needed between accuracy and latency, where higher accuracy can be achieved by waiting for more context. In this paper, we propose a deep learning model that separates the keyword spotting task into three phases in order to further optimize both accuracy and latency of the overall system. These three tasks are: Speculation, Detection, and Verification. Speculation makes an early decision, which can be used to give a head-start to downstream processes on the device such as local speech recognition. Next, Detection mimics the traditional keyword trigger task and gives a more accurate decision by observing the full keyword context. Finally, Verification verifies previous decision by observing even more audio after the keyword span. We propose a latency-aware max-pooling loss function that can train a unified model for these three tasks by tuning for different latency targets within the same model. In addition, we empirically show that the resultant unified model can accommodate these tasks with desirable performance and without requiring additional compute or memory resources.

Index Terms— keyword spotting, accuracy latency trade-off, convolutional recurrent neural network, max-pooling loss, multi-task learning

1. INTRODUCTION

Keyword detection is a key feature for hands-free digital assistants. Different approaches have been proposed for the keyword spotting problem. A traditional approach employs a hybrid Deep Neural Network (DNN)-Hidden Markov Model (HMM) decoding framework, where a DNN is used as an acoustic model (AM) and the HMM models both keyword and background speech [1]. In recent years, we have seen increasing research on DNN-based end-to-end keyword spotting system [2, 3, 4, 5, 6, 7, 8, 9], which predicts the posterior probability of the keyword directly.

A successful keyword spotting system is expected to have both high accuracy and low latency. Latency is a key issue for customer experience, since a system that lags behind user speech feels sluggish. However, most of existing work only focuses on the accuracy. Among the few works that explore latency, most of them [2, 6, 10, 11, 12] study computational latency, i.e., inference time, rather than observational latency, which is caused when the keyword spotting system must wait for certain acoustic signals that clearly indicate the existence of the keyword. In [13], authors propose a two-stage progressive voice trigger detection system. The first-stage is an always-on low-power detector, and the second-stage is a larger verification model that re-scores the keyword segment marked by the first-stage. However, additional device resources are needed to run inference of the second-stage. In addition, more sophisticated training are needed, since these are two separate models to train, and the second-stage model training depends on segmentation results from the first-stage.

In this paper, we fill the gap and study the observational latency. We propose a unified speculation, detection, and verification model (USDV) using a convolutional recurrent neural network (CRNN) architecture [7, 8, 9]. Negligible additional resources are needed to achieve these three tasks simultaneously. The Speculation task is trained to trigger on the first part of the keyword, before the end of the word is observed. This output can be used to give a head-start downstream speech processing components on the device in order to reduce end-to-end system latency. The Detection task is trained to detect a keyword immediately after the full word is observed. By seeing the full keyword duration, the model makes a more accurate decision without lagging behind user speech. The Verification task is intended to wait longer for more right-hand context, in order to correct mistakes made by the Speculation and Detection tasks. We propose a latency-aware max-pooling loss that maximizes accuracy under the latency constraints we apply. The USDV model is trained in multi-task learning (MTL) [14] fashion by minimizing the latency-aware max-pooling losses on these three tasks with different target latencies. Results show that the unified model achieves different accuracy and latency trade-off with the new training loss. In addition, without increasing the size for the unified model, it performs on the same level as single task baseline models, indicating the proposed CRNN architecture have enough capacity to achieve these three tasks simultaneously.

978-1-6654-0540-9/22/$31.00 ©2022 IEEE

2. MODEL ARCHITECTURE

We design a model with convolutional neural network front-end that has a receptive field of 34 frames and has a stride of 6 frames. In the CNN front-end, each layer is composed of a convolution layer, a rectified linear unit activation layer, an optional max pooling layer, a batch normalization layer and a drop out layer. This behaves as an efficient feature extractor to model local temporal and spectral dependencies. Its outputs are vectorized and fed to a long short-term memory (LSTM) layer, which capture dependencies among different frames using "gating" mechanism. After the LSTM layer, a full-connected (FC) layer is used to further transform features before Softmax output. Due to the similarity of speculation, detection, and verification tasks, i.e., all of them try to detect the same word from audio, we share the same convolutional front-end, LSTM, and FC layer for them to reduce model size. We only add three output heads with separate linear layers for dimension reduction and Softmax outputs, as shown in Fig. 1. The exact layer settings are shown in Table 1. The additional computations to achieve these three tasks simultaneously are only introduced by these small output heads, hence are negligible. With the layer setting in Table 1, each output head only requires 200 multiplications, since Softmax layer outputs a two dimensional score vector for background and keyword.

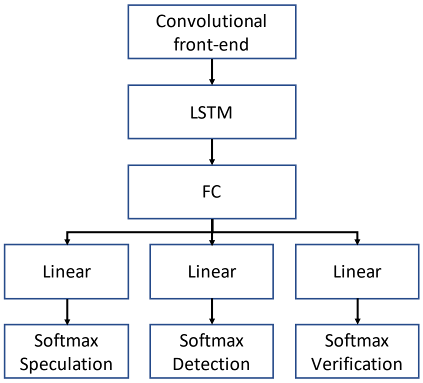

Fig. 1. CRNN-based unified speculation, detection, and verification model architecture, which uses a shared CRNN model and three output heads for these tasks.

layer-type	CONV							LSTM	FC
filter-heights	7	5	2	2	2	1	1		
filter-widths	5	3	4	3	4	1	1		
stride-heights	1	3	1	1	1	1	1		
stride-widths	1	1	1	1	1	1	1		
pooling-heights	2	1	1	1	1	1	1		
pooling-widths	3	2	1	1	1	1	1		
num-filters/units	96	128	128	160	160	500	100	100	100

Table 1. CRNN-based USDV model layer settings, where each column represents settings for a layer. Different number of output heads can be added after the FC layer to achieve different tasks.

3. TRAINING LOSS

In [15], we proposed a max-pooling loss for training RNN models. Rather than minimizing cross-entropy loss on a human selected frame that requires a lot of domain knowledge and likely to be suboptimal, the max-pooling loss selects a frame automatically, on which it minimizes the cross-entropy loss. For a positive example, it selects the frame that has the maximum score on the corresponding class, since we only need the highest score of the class to pass a threshold. For a negative example, it selects the frame that has the lowest score on negative class, or highest score on positive class for binary classification problem, to encourage low positive scores across all frames. Hence, max-pooling loss tries to maximize the lower bound of negative scores so that negative scores are high on all frames. But, this maxpooling loss tends to delay the detection, since it understands that a more accurate detection, hence lower loss, can be achieved by exploring more contextual information. In this paper, we propose a latency-aware max-pooling loss by adding a constraint to original one as shown in (1).

$$\mathcal{L}(t_l) = -log(p_{yt}), \tag{1}$$

$$t = \begin{cases} argmin_i \; (p_{0i}) & y = 0 \\ argmax_{i \in (i-e<=t_l)} \; (p_{yi}) & y \neq 0 \end{cases},$$

where t and i are frame indices, y is class label, $y = 0$ is the negative class, e is the end of the keyword, p_{yi} is the output score of yth class on the frame i, and t_l is the target latency. On negative examples, this loss still try to maximize the lower bound of negative score. But, on positive examples, as shown in Fig. 2, it will discard frames that do not meet latency requirement. Hence, a model learns to detect a keyword within the target latency, since a later detection won't reduce loss.

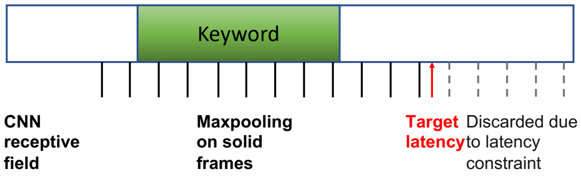
Keyword
CNN receptive field
Maxpooling on solid frames
Target latency
Discarded due to latency constraint

Fig. 2. Latency-aware max-pooling loss on a positive example. By discarding frames that have higher latency, the model is encouraged to learn to detect within the target latency.

To show the effectiveness of this loss function, we train three CRNN models using this loss with different target latency, $[10, 40, 70]$. As shown in Fig. 3, all models learn to delay the detection. However, their latencies converge to different values that are determined by their target latencies, which shows that the proposed loss can effectively control the latency of the model. In addition, we can see that model with larger latency achieve lower error rate, which validates the trade-off between accuracy and latency.

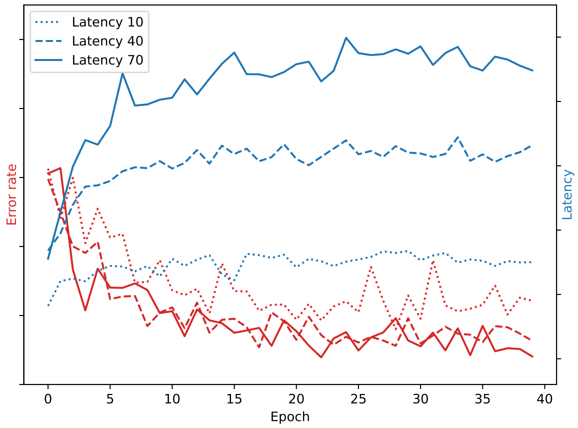

Fig. 3. Compare error rate and latency over different training epochs for three models with target latencies, 10, 40, 70. Over training epochs, models learn to trade latency for accuracy. The proposed latency-aware max-pooling loss can control latency accuracy trade-off effectively.

With the proposed latency-aware max-pooling loss, we train USDV model in a MTL fashion as shown in (2).

$$\mathcal{L}_{mtl} = w_s\mathcal{L}(t_s) + w_d\mathcal{L}(t_d) + w_v\mathcal{L}(t_v), \quad (2)$$

where (w_s, t_s), (w_d, t_d), (w_v, t_v) are weights and target latencies for speculation, detection, and verification tasks, respectively. By tuning (t_s, t_d, t_v), we can achieve different accuracy and latency trade-off for these three tasks.

4. EXPERIMENTS

In this section, we study USDV performance by comparing against single task baseline models. We also compare the performance of USDV model with different target latency setups and with different model architectures. In the following experiments, we always control the false reject rate (FRR) to a constant value, C_{frr}, and compare false accept rate (FAR) and latency. Latency is calculated as the time difference between the frame that a model emits a detection and the end frame of the keyword. We add constant offsets to FAR and latency to avoid showing absolute performance value, while still preserving the relative change across all models.

4.1. Data sets

We chose ALEXA as the keyword in all following experiments. We used a de-identified far-field corpus that was collected under different acoustic conditions in real households. The audio data was sampled with a 16-bit resolution at a 16 kHz sampling frequency. This dataset was further divided into 39 thousands of hours of training data and 1.4 thousands of hours of testing data through random sampling. A random fraction of data from training partition are kept for hyperparameter tuning.

4.2. Training setup

We use synchronous stochastic gradient descent for training all models. Two GPUs are used with each mini-batch per GPU containing 1000 training examples. All models are trained by 30 epochs, where each epoch has $10,000$ steps. Model input is 64-dimensional Log Filter Bank Energies (LFBE) acoustic features computed every 10ms over a window of 25ms. We use the Adam optimizer with learning rate of 0.001. Batch normalization is used after every convolutional layers. In addition, we have a global normalization layer that normalizes training data with mean and variance of the whole training dataset. From our previous study, dropout improves model performance, and the best dropout rates are 0.3 for the CNN front-end, 0.1 for LSTM recurrent state, 0 for LSTM input, and 0.1 for the FC layer after the LSTM layer. Moving average with decay value 0.99 is used to smooth the model weights when exporting a model after its training. Equal weights are used on losses of three tasks.

4.3. USDV vs. single task baseline models

USDV model tries to achieve three tasks without increasing model size. It is natural to ask whether the CRNN model has enough capacity to achieve desirable performance on all three tasks simultaneously; whether the MTL loss can train the model to achieve these three task effectively; whether there is interference among these tasks which could affect model performance. To answer these questions, we compare USDV against single task baseline models. All baseline models use the same layer setup as shown in Table 1, except that they only have one output head rather than three. The baseline speculation, detection, and verification models are trained using loss (1) with target latency -10, 10, 70 frames, respectively. The USDV model has the exact same layer setup as baseline models except that it has three output heads as explained in section 2. It is trained with the MTL loss (2) with the same target latency as the baseline models training.

Model	FAR (%) @ constant FRR	Latency (s)
Speculation	1.75a	b-0.15
Detection	a	b
Verification	0.80a	b+0.3
USDV-speculation	1.75a	b-0.14
USDV-detection	1.03a	b-0.01
USDV-verification	0.82a	b+0.27

Table 2. Performance comparison shows that USDV achieves same level of accuracy compared to single task baselines.

Results are shown in Table 2. As expected, speculation models from both the single task baseline and USDV have the earliest detection, which brings downstream modules a 150ms early start. With more right context, verification models achieves the lowest FAR, hence can be used to correct mistakes made by speculation and detection models. The fact that the USDV model is able to achieve three tasks with different accuracy and latency trade-off validates the effectiveness of the MTL training loss (2). Furthermore, the USDV model achieves same level of performance as baseline models on all three tasks, which shows that the CRNN architecture has enough capacity to perform all three task simultaneously. Small degradation on the performance of certain task can be addressed by different weighting in the MTL loss (2). In addition, we may also branch out these output heads earlier to achieve better performance with additional computation. For example, the USDV model can transform embedding from LSTM differently for these three tasks, hence, likely to achieve better performance, if we branch out before the FC and after the LSTM layer.

4.4. USDV with different target latency

In practice, we normally have a target latency for the detection task to avoid sluggish response. But, the desired latency for the speculation model can be very different in different scenarios. We may want earlier start for downstream modules by tolerating worse performance, and vice versa. In this experiment, we study the impact of different target latency. All models are trained with the same target latency for detection task, but different target latencies for speculation and verification tasks. As shown in Fig. 4, all models retain the same accuracy and latency trade-off for detection tasks, though they achieve very different latency for speculation task due to different target latency settings. Hence, different accuracy and latency trade-off can be tuned independently across these three tasks with our CRNN architecture and latency-aware max-pooling loss in a MTL fashion.

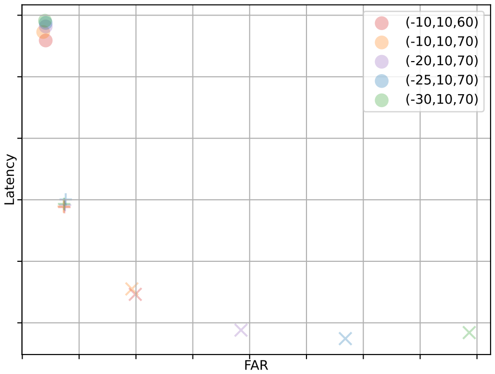

Fig. 4. Scatter plot that shows the performance of USDV models trained with different target latencies for speculation, detection, and verification as shown by the legend. The performance speculation, detection, and verification are indicated by x, +, and o marker, respectively.

4.5. USDV with different model architectures

In this experiment, we study the impact of model size. We design four variations of the model shown in Table 1 by freezing kernel sizes and changing the number of channels for the CNN front-end, and the number of units for the LSTM and FC layers as shown in Table 3. As shown in Fig. 5, in general, larger model is able to achieve better performance. We expect the verification needs larger capacity due to more sophisticated right context to model.

layer-type	CONV							LSTM	FC
model1	80	96	112	128	160	400	40	40	40
model2	96	128	128	160	160	500	100	100	100
model3	96	128	128	160	160	500	100	200	200
model4	96	128	160	192	240	500	200	200	200

Table 3. Number of filters for CNN front-end and number of nodes for LSTM and FC layers for different model variations.

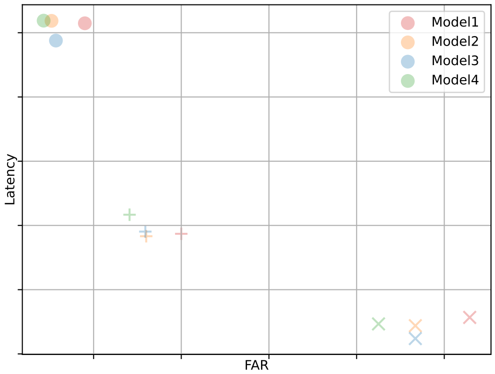

Fig. 5. Scatter plot that shows the performance of USDV models trained with different model sizes. The performance speculation, detection, and verification are indicated by x, +, and o marker, respectively.

5. CONCLUSION

In this paper, we propose an CRNN-based unified speculation, detection, and verification keyword detection model. With negligible additional resources, this model is able to achieve three tasks simultaneously. We propose a latency-aware max-pooling loss, and show empirically that it teaches a model to maximize accuracy under the latency constraint. With the new training loss, a USDV model can be trained in a MTL fashion and achieves different accuracy and latency trade-off across these three tasks. We study the performance of USDV model against single task baseline models, and results show that it can achieve same level of performance. Results also show that we can easily tune the accuracy and latency trade-off for a task without impacting other tasks. In addition, we show that larger model size leads to better performance, especially on verification task.

6. REFERENCES

[1] S. Panchapagesan, Ming Sun, Aparna Khare, S. Matsoukas, Arindam Mandal, Björn Hoffmeister, and S. Vitaladevuni, "Multi-task learning and weighted cross-entropy for dnn-based keyword spotting," in *INTERSPEECH*, 2016.

[2] Guoguo Chen, Carolina Parada, and Georg Heigold, "Small-footprint keyword spotting using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 4087–4091.

[3] Ming Sun Sankaran Panchapagesan Geng-Shen Fu George Tucker, Minhua Wu and Shiv Vitaladevuni, "Model compression applied to small-footprint keyword spotting," in *Interspeech*, 2016.

[4] Tara Sainath and Carolina Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Interspeech*, 2015.

[5] Raphael Tang and Jimmy J. Lin, "Deep residual learning for small-footprint keyword spotting," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5484–5488, 2018.

[6] Seungwoo Choi, Seokjun Seo, Beomjun Shin, Hyeongmin Byun, Martin Kersner, Beomsu Kim, Dongyoung Kim, and S. Ha, "Temporal convolution for real-time keyword spotting on mobile devices," in *INTERSPEECH*, 2019.

[7] R. Kumar, Vaishnavi Yeruva, and Sriram Ganapathy, "On convolutional lstm modeling for joint wake-word detection and text dependent speaker verification," in *INTERSPEECH*, 2018.

[8] Sercan Ömer Arik, Markus Kliegl, Rewon Child, Joel Hestness, Andrew Gibiansky, Christopher Fougner, Ryan Prenger, and Adam Coates, "Convolutional recurrent neural networks for small-footprint keyword spotting," *CoRR*, vol. abs/1703.05390, 2017.

[9] Taiki Yamamoto, Ryota Nishimura, M. Misaki, and N. Kitaoka, "Small-footprint magic word detection method using convolutional lstm neural network," in *INTERSPEECH*, 2019.

[10] Enea Ceolini, Jithendar Anumula, Stefan Braun, and Shih-Chii Liu, "Event-driven pipeline for low-latency low-compute keyword spotting and speaker verification system," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 7953–7957.

[11] Hu Du, Ruohan Li, Donggyun Kim, Kaoru Hirota, and Yaping Dai, "Low-latency convolutional recurrent neural network for keyword spotting," in *2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*, 2018, pp. 802–807.

[12] Bo Zhang, Wenfeng Li, Qingyuan Li, Weiji Zhuang, Xiangxiang Chu, and Yujun Wang, "Autokws: Keyword spotting with differentiable architecture search," *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2830–2834, 2021.

[13] Siddharth Sigtia, John Bridle, Hywel Richards, Pascal Clark, Erik Marchi, and Vineet Garg, "Progressive voice trigger detection: Accuracy vs latency," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 6843–6847.

[14] Rich Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 4175, 1997.

[15] Ming Sun, Anirudh Raju, George Tucker, Sankaran Panchapagesan, Geng-Shen Fu, Arindam Mandal, Spyros Matsoukas, Nikko Strom, and Shiv Vitaladevuni, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 474–480.