An Audio-Based Wakeword-Independent Verification System

Joe Wang[1], *Rajath Kumar*[1], *Mike Rodehorst*[1], *Brian Kulis*[1,2†], *Shiv Vitaladevuni*[1]

[1]Amazon.com, Cambridge, MA, USA
[2]Boston University, Boston, MA, USA

{wangjose, rajathku, mikerode}@amazon.com, bkulis@bu.edu, shivnaga@amazon.com

Abstract

We propose an audio-based wakeword-independent verification model to determine whether a wakeword spotting model correctly woke and should respond or incorrectly woke and should not respond. Our proposed model works on any wakeword-initiated audio, independent of the wakeword by operating only on the audio surrounding the wakeword, yielding a wakeword agnostic model. This model is based on two key assumptions: that audio surrounding the wakeword is informative to determine if the user intended to wake the device and that this audio is independent of the wakeword itself. We show experimentally that on new wakewords not included in the training set, our model trained without training examples or knowledge of the wakeword is able to achieve verification performance comparable to models trained on 5,000 to 10,000 annotated examples of the new wakeword.

Index Terms: speech recognition, keyword spotting

1. Introduction

Voice-activated devices are designed to activate upon hearing a specific wakeword[1] and reply to the following command with either a spoken response or an action (e.g. playing music, controlling a smart home device, etc.). Generally, the system is structured where an efficient, small-footprint model is running locally on a device [1, 2]. When this small-footprint wakeword-spotting model hears the specified wakeword, audio is sent to a cloud-based system, allowing for more complex automatic speech recognition (ASR) and natural language understanding (NLU) models to attempt to interpret the command. Based on the command, the cloud-based system can then issue a response, with the ability to access content (e.g. music, live broadcasts, etc.) that is not stored on the device and interact with smart home devices over the internet.

Although wakeword detection models are capable of high-degrees of accuracy, there is a risk of false wakes occurring due to incorrect detections by the small-footprint model running locally on the device. In the event of a false wake, devices may speak, play music, or interact with smart home devices without prompting from the user. One approach to avoiding this is through wakeword verification [3]. Wakeword verification is the problem of confirming whether audio was correctly sent to the cloud by a device-directed wakeword or incorrectly sent due to a false identification of a wakeword by the model running locally on a device.

Wakeword verification systems improve upon on-device wakeword-spotting models due to two main factors: the ability to use audio content following the wakeword and to run larger/significantly more complex models than possible as a detection model. Wakeword-spotting models are unable to fully leverage audio following the wakeword, as waiting for this audio to arrive before making a decision would significantly increase the latency of the overall end-to-end system. Additionally, the need to run continually streaming detection presents limits on the inference complexity that can be run by the wakeword-spotting model.

For a known wakeword, one natural approach to building a wakeword verification system is to build a model that takes audio (or audio-based features) as an input and attempts to predict whether the user intended to wake the device based on annotated training examples.

A wakeword agnostic system is necessary for many applications. For example, in voice interoperability, the goal is for users to be able to directly interact with various cloud-based systems by using distinct wakewords. In this setting, there is a need for a wakeword verification system that scales to many continually changing and growing wakewords while avoiding the need to have a distinct model for each wakeword. Another potential application is highly personalized wakewords, where a large number of wakewords are required with limited or no training data provided. These applications motivate the problem of wakeword-independent verification, where we seek to build a system that identifies whether a wakeword was directed at the device for any unknown wakeword.

For this problem, we focus on constructing a single model that operates on any incoming wakeword. We assume the system is wakeword agnostic, that is it has no prior knowledge of the wakeword used by the device. Finally, training examples are provided only for a small subset of wakewords, preventing use of few-shot or query-by-example approaches that would allow explicit modeling of the individually chosen wakewords.

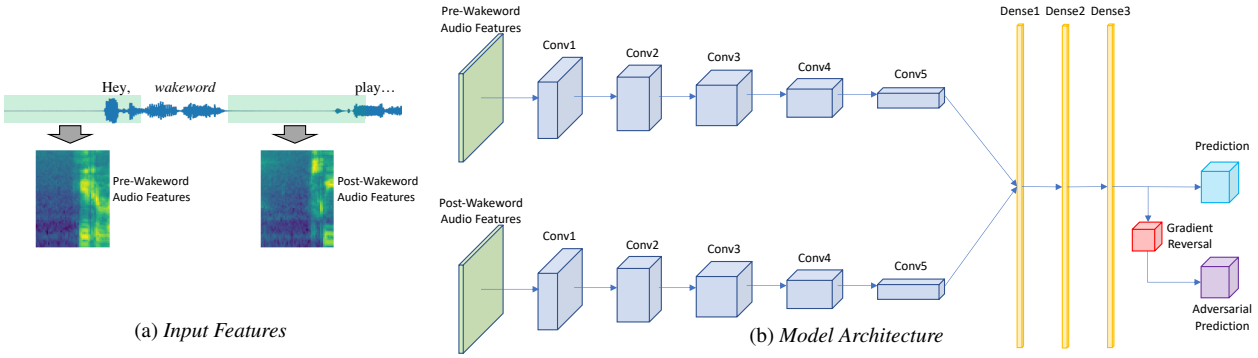
We propose an approach to solving the wakeword-independent verification problem that operates only on audio before and after the wakeword. We assume the audio content surrounding the wakeword is informative in determining whether the wakeword was user-generated and device-directed. Additionally, we assume this behavior generalizes across wakewords, allowing for models trained on surrounding audio from one set of wakewords to generalize to new, unseen wakewords.

To leverage this assumption, we propose a novel convolutional neural network architecture with two independent sets of convolutional filters, one set operating on the audio preceding the wakeword and one set operating on the audio following the wakeword. We introduce an adversarial loss during training to encourage generalization across wakewords and avoid overfitting on specific on training wakewords.

Empirically, we show that the proposed model trained on data from two wakewords is capable of generalizing to new, unknown wakewords with performance comparable to wakeword-specific models built with knowledge of the wakeword and

†Work conducted while the author was at Amazon.com

[1]The terms wakeword and keyword are often used interchangeable in literature. For consistency we use the term wakeword throughout when referring to a word chosen to wake a voice-activated device.



(a) *Input Features* (b) *Model Architecture*

Figure 1: ***(a)*** *Features used in by the wakeword-independent model. The audio associated with the wakeword is ignored, with audio in the preceding 0.5 seconds and following 0.5 seconds cropped out and LFBE features are extracted for each cropped segment.* ***(b)*** *Architecture of our wakeword-independent verification CNN model.*

trained on 5,000 to 10,000 annotated examples.

1.1. Related Work

As far as we know, [3] is the only existing published paper on wakeword verification. In [3], verification is performed using an Automatic Speech Recognition (ASR) model. In this setting, audio that is streamed to the cloud is processed by an ASR system modified to improve wakeword spotting accuracy. This approach is inapplicable for a wakeword agnostic system, where the wakeword chosen to wake the device is unknown or may not be in the ASR lexicon (e.g. foreign language phrases).

Related past approaches have utilized a fusion of a low-footprint detection model with a verification model running on the device [4, 5]. These approaches rely on knowledge of the chosen wakeword and access to annotated training examples.

Our approach is related to zero-shot learning [6, 7, 8], where models are built to extend to unseen classes during test time. One major difference between our approach and the standard zero-shot learning setting is that we do not receive any semantic descriptors of the unknown wakewords and are unaware of the selected wakeword during test time. As such, approaches based on semantic descriptions of unseen classes are not easily applied to our problem setting.

Similarly, exemplar-based learning [9, 10, 11, 12], few-shot learning [13], metric-based approaches [14], and transfer learning [15] are also closely related, however generally are not applicable to this problem as we do not have access to examples (annotated or unannotated) for the unknown selected wakeword.

2. Wakeword-Independent Verification

Our proposed wakeword-independent verification system operates on any wakeword-initiated stream, with the goal of determining whether the model on the device falsely woke.

2.1. Input

Our system is designed to work on audio sent to the cloud by devices running a wakeword-spotting model. On these devices, a model is continually running to identify wakewords. When this model believes a wakeword has been spoken, the device wakes and audio is sent from the device to the cloud. We assume that when audio is sent to the cloud, an estimate of where the wakeword begins and ends in the audio stream is provided, but no additional information is sent (e.g. what wakeword was used or a score/confidence from the device-side model).

Once the device-side wakeword model detects a wakeword and estimated location of the wakeword are sent to the cloud. Note that only 0.5 seconds of audio preceding the wakeword is sent to the cloud where it is used to initialize feature extraction for the ASR system. Using this estimated location of the wakeword, our verification model extracts audio from the 0.5 seconds preceding and 0.5 seconds following the wakeword, ignoring the audio during the wakeword itself as shown in Figure 1a. For each audio segment, log filterbank energies (LFBE) are extracted for 64 frequency bins between the frequencies 80 and 7,200 Hz over a 25 msec window with a 10 msec stride, yielding two sets of (48×64)-dimensional features.

2.2. Model Architecture

We propose a convolutional neural network (CNN) model with two sets of convolutional layers, one taking the pre-wakeword audio features as inputs and the other taking the post-wakeword audio features as inputs, as shown in Fig. 1b. These convolutional layers have the same structure, however independent filters are learned in each set of convolutional layers, allowing for different types of embedding to efficiently be learned for the pre-wakeword and post-wakeword audio segment.

Layer Name	Parameters	Output Dim.
Inputs		$48 \times 64 \times 1$
Conv1	96 @ 7×5 Filters 2×3 Max Pooling	$21 \times 20 \times 96$
Conv2	128 @ 5×3 Filters 2×1 Striding 1×2 Max Pooling	$9 \times 9 \times 128$
Conv3	128 @ 3×3 Filters	$7 \times 7 \times 128$
Conv4	160 @ 2×3 Filters	$6 \times 5 \times 160$
Conv5	160 @ 2×3 Filters	$5 \times 3 \times 160$
Flatten/Merge		4800
Dense1	500 Nodes	500
Dense2	500 Nodes	500
Dense3	500 Nodes	500

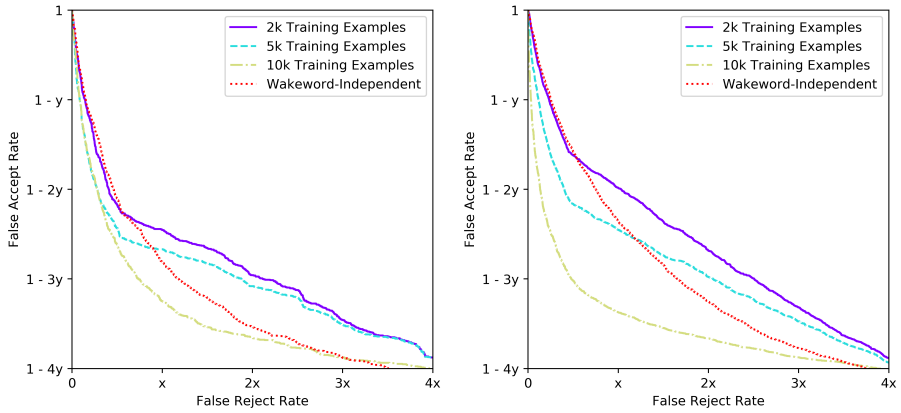
Table 1: *Layer Parameters*

The embedding generated from the pre-wakeword and post-wakeword audio features is flattened and concatenated, then sent to dense layers. Following these dense layers, a binary prediction of whether the audio was from a device-directed utterance of a wakeword or the device incorrectly woke is output. See Table 1 for details on the parameters for each layer.

During training, a gradient reversal layer [16] is attached to the output of the dense layers followed by a wakeword prediction head. Details about this are described in Section 2.3 below.

2.3. Adversarial Training

To train the wakeword-independent verification model, we use annotated data collected from a set of different wakewords, training the model to identify whether a user intentionally woke the device or whether the device falsely woke.



(a) *"Amazon" wakeword* (b) *"Computer" wakeword*

Figure 2: *Comparison of the wakeword-independent model with wakeword-specific models trained using 2k, 5k, and 10k annotated examples for the wakewords "Amazon" and "Computer" (trained separately for each wakeword). Note that the relative axes are independent across graphs.*

Training data is collected for a small subset of commonly used wakewords, where the data is labeled by human annotators as either a true wake, where a user correctly pronounced the selected wakeword and intended to wake the device, or as a false wake, where the user did not intend to wake the device, the device woke to media, or the user spoke a different/mispronounced wakeword.

Although the audio associated with the wakeword is removed from the features input into the network, the model can potentially overfit to the wakewords in the training set. Overfitting risks include bias in words used around a specific wakeword in conversation as well as potential "leakage" of the start and end of the wakeword into the pre-wakeword and post-wakeword audio segments due to inaccurate estimation of the start and end points of the wakeword audio. Additionally, the difference in wakeword detection models used on device to generate the training data can cause the model to overfit to specific environment/noise conditions, leading to poor generalization to new wakewords and device-side wakeword detection models.

To overcome this, we use an adversarial strategy [16] across wakewords during training. The goal of the adversarial prediction task is to ensure that the embedding used for prediction is agnostic to the initiated wakeword. To this end, we define the objective function as

$$E\left(\theta,\gamma_y,\gamma_w\right)=\sum_{i=1}^{n}\Big[L\left(\gamma_y^T f\left(x_i,\theta\right),y_i\right) - \lambda L\left(\gamma_w^T f\left(x_i,\theta\right),w_i\right)\Big], \quad (1)$$

where θ are the set of parameters for the network excluding the prediction and adversarial prediction layers, $f(x,\theta)$ is the embedding of the network for example x and parameters θ, γ_y is the set of weights in the prediction layer (predicting whether the user intended to wake the device), γ_w is the set of weights in the adversarial prediction layer (predicting the wakeword detected correctly or incorrectly by the device), L is a loss function (we use cross entropy), λ is an adversarial weight, x_i is the set of LFBE features for the i^{th} example, y_i is the label of whether the i^{th} example was device directed, and w_i is the wakeword the device generating the i^{th} example was attempting to detect.

During training, we seek the parameter values at the saddle point where θ and γ_y minimize the objective while γ_w maximizes the objective. To implement this, we attach to the final embedding of the network a gradient reversal layer and second prediction head that attempts to predict the wakeword.

In practice, we sweep over the weight assigned to the adversarial loss (along with other hyperparameters such as learning rate and regularizaiton) and select the set of values with the highest validation performance on a held-out set of annotated data from a held out set of examples that used the same wakewords as in the training set.

2.4. Unlabeled Data

One additional benefit of adversarial training is the ability to leverage unannotated data where the wakeword detected by the device is known, but the audio has not been annotated to determine whether the device falsely woke. Given this data, the training data set can be augmented with these examples, with no weight put on the verification task for these examples, but a non-zero adversarial weight on the wakeword prediction task. This potentially improves performance of the model across a wider range of noise conditions and allows better generalization across the selection of wakewords. We do not explore this empirically in this paper and leave this as an area of future work.

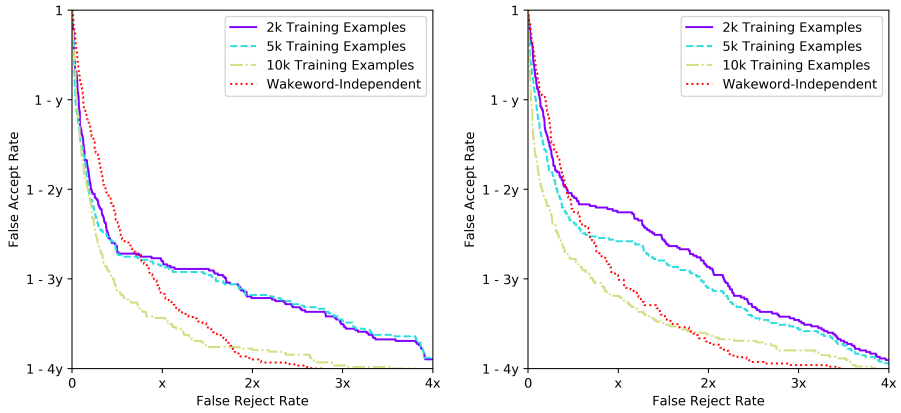
3. Experiments

We train our wakeword-independent model on annotated training data for two known wakewords, "Alexa" and "Echo", with hyper-parameters optimized over a validation split of data from these two wakewords. This annotated data set consists of millions of annotated examples combined from these two wakewords. For evaluation, we apply our model on two new, unseen wakeword test data sets for the wakeword "Amazon" and "Computer", each containing tens of thousands of evaluation examples per wakeword. All datasets have a significant bias towards true wakes, with a similar distribution between true and false wakes across training and evaluations datasets.

We compare performance of our model to wakeword-specific CNN models trained over the entire audio segment (including the wakeword itself). Each wakeword-specific model is trained using annotated data for the associated test wakeword for both training and hyperparameter optimization. We train these models using varying volumes of annotated data and compare performance to our wakeword-independent models.

In order to evaluate the impact of using different audio context, the wakeword-specific models have a nearly identical architecture as the wakeword-independent model, with the exception of a larger input context (a fixed length of 2 seconds of input audio) and the use of a single set of convolutional layers operating on this context.

A comparison of performance between the wakeword-independent and wakeword-specific models trained with differing volumes of training data is shown in Figure 2. Over the en-



(a) *"Amazon" wakeword* (b) *"Computer" wakeword*

Figure 3: *Comparison of the wakeword-independent model with wakeword-specific models trained using 2k, 5k, and 10k annotated examples for the wakewords "Amazon" and "Computer" (trained separately for each wakeword), where the negative set is limited to examples annotated as non-device directed or media-sourced wakeword. Note that the relative axes are independent across graphs.*

tire evaluation datasets, we see the performance on both "Amazon" and "Computer" falls between the wakeword-specific models trained using 5,000 and 10,000 annotated examples with a strong bias towards true wakes.

Note that we do not compare the performance of our wakeword-independent model to that of a wakeword-spotting model due to the fact that at test-time, it is unclear as to the best approach to normalize scores across different wakewords. In particular, for an examplar-based approach, two users who have chosen the same wakeword and provided examples will end up with different models, making comparison of scores difficult. Furthermore, as the wakeword-specific baseline verification models operate on the same audio as a device-model with additional context, these baseline models provide an upper-bound on the performance expected of device-side models of similar or smaller architectures.

The baseline models we compare against have a significant advantage over the wakeword-independent model in two ways.

First, these models are wakeword aware and specific, being able to look for the exact wakeword as opposed to being wakeword agnostic. This allows them to reject examples where the user intended to wake the device, but used an incorrect wakeword. Our wakeword-independent model is inherently agnostic to this type of false-wake, as it instead attempts to predict whether the command was a user-generated, device-directed command and not whether the wakeword was correctly spoken.

To examine the impact of this, we restrict the negative set to be examples marked by annotators as non-device directed or media-generated wakewords, with results shown in Figure 3. We see a significant improvement in the performance of the wakeword-independent model relative to the baseline models, with performance much closer to the wakeword-specific model trained on 10,000 annotated examples, in particular for the second test wakeword. This difference in behavior demonstrates the efficacy of our model in identifying false wakes that contain the wakeword but were not intended for the device.

Second, these models have the advantage of observing examples (positive and negative) drawn from the test distribution during training, whereas the wakeword-independent model sees no examples drawn from this distribution during training. Given the observed distribution during test-time is dependent on the detection model running on device, this can cause the wakeword-independent verification model to be forced to act on types of examples never seen before, e.g. novel noise patterns consistently rejected by the device-wakeword models used to create the training data set.

We observe that the wakeword-independent model appears to have worse performance at low false reject rates. Anecdotally we observe that a majority of the randomly sampled examples incorrectly classified by the wakeword-independent model as non-device directed speech generally fall into two categories. First, these examples tend to have a long pause after wakeword and no speech before the wakeword, providing no audio content of use to the wakeword-independent model. Second, we observe that many of these examples contain background media or conversation, with the context of the wakeword-independent model containing this audio rather than speech from the user. This implies to us that the performance of the system in the low false reject rate regime could be improved using a longer audio context to ensure we capture speech directed at the device (if present) as well as fusion with source separation or speaker identification systems in order to remove background audio from the hypothesized wakeword audio source.

Interestingly, we also anecdotally observe that the wakeword-independent model appears to outperform the baseline wakeword-dependent models in identifying non-device directed audio where the wakeword is present but not directed at the device. We hypothesize this is due to the baseline models over-indexing on the presence of the wakeword, whereas the wakeword-independent model must rely on the surrounding context, allowing it to correctly infer in these cases that the audio is not intended for the device.

4. Conclusions

We present an approach to performing wakeword-independent verification based on audio surrounding a chosen wakeword. Our proposed system leverages the assumption that for a device-directed command, audio surrounding the wakeword is independent of the wakeword itself, providing an approach to reducing false responses from voice-activated systems for any chosen wakeword. We demonstrate the efficacy of this approach by evaluating our proposed system on two unseen wakewords, with performance comparable to task-specific models trained using thousands of annotated examples.

5. References

[1] T. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Interspeech*, 2015.

[2] M. Sun, D. Snyder, Y. Gao, V. K. Nagaraja, M. Rodehorst, S. Panchapagesan, N. Strom, S. Matsoukas, and S. Vitaladevuni, "Compressed time delay neural network for small-footprint keyword spotting," in *INTERSPEECH*, 2017.

[3] A. Michaely, C. Parada, F. Zhang, G. Simko, and P. Aleksic, "Key-

word spotting for google assistant using contextual speech recognition," in *ASRU 2017*, 2017.

[4] M. Sun, V. Nagaraja, B. Hoffmeister, and S. Vitaladevuni, "Model shrinking for embedded keyword spotting," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 369–374.

[5] S. Sigtia, P. Clark, R. Haynes, H. Richards, and J. Bridle, "Multi-task learning for voice trigger detection," *ArXiv*, vol. abs/2001.09519, 2020.

[6] H. Larochelle, D. Erhan, and Y. Bengio, "Zero-data learning of new tasks." in *AAAI*, vol. 1, no. 2, 2008, p. 3.

[7] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Advances in neural information processing systems*, 2009, pp. 1410–1418.

[8] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2251–2265, 2018.

[9] G. Chen, C. Parada, and T. N. Sainath, "Query-by-example keyword spotting using long short-term memory networks," in *ICASSP*, 2015.

[10] Y. He, R. Prabhavalkar, K. Rao, W. Li, A. Bakhtin, and I. McGraw, "Streaming small-footprint keyword spotting using sequence-to-sequence models," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 474–481.

[11] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, "End-to-end asr-free keyword search from speech," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1351–1359, 2017.

[12] B. Kim, M. Lee, J. Lee, Y. Kim, and K. Hwang, "Query-by-example on-device keyword spotting," *arXiv preprint arXiv:1910.05171*, 2019.

[13] M. Fink, "Object classification from a single example utilizing class relevance metrics," in *Advances in neural information processing systems*, 2005, pp. 449–456.

[14] B. Kulis, "Metric learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2013.

[15] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1126–1135.

[16] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," vol. 17, no. 1. JMLR.org, Jan. 2016, p. 2096–2030.