AUTOMATIC GAIN CONTROL AND MULTI-STYLE TRAINING FOR ROBUST SMALL-FOOTPRINT KEYWORD SPOTTING WITH DEEP NEURAL NETWORKS

Rohit Prabhavalkar[1], *Raziel Alvarez*[1], *Carolina Parada*[1], *Preetum Nakkiran*[2*], *Tara N. Sainath*[1]

[1]Google Inc., Mountain View, USA; [2]University of California, Berkeley, Department of EECS, USA
{prabhavalkar, raziel, carolinap, tsainath}@google.com preetum@berkeley.edu

ABSTRACT

We explore techniques to improve the robustness of small-footprint keyword spotting models based on deep neural networks (DNNs) in the presence of background noise and in far-field conditions. We find that system performance can be improved significantly, with relative improvements up to 75% in far-field conditions, by employing a combination of multi-style training and a proposed novel formulation of automatic gain control (AGC) that estimates the levels of both speech and background noise. Further, we find that these techniques allow us to achieve competitive performance, even when applied to DNNs with an order of magnitude fewer parameters than our baseline.

Index Terms— keyword spotting, automatic gain control, multi-style training, small-footprint models

1. INTRODUCTION

With the proliferation of mobile devices, speech-enabled technologies are becoming increasingly widespread. Services such as Google voice search [1] and conversational assistants such as Google Now, Apple's Siri and Microsoft's Cortana prominently feature speech technologies as a means of interacting with and controlling devices. Improving speech interfaces in these systems is critical to ensure their widespread adoption.

In the present work, we consider the problem of developing a keyword spotting (KWS) system that can run on mobile devices and respond appropriately when a user utters a specific keyword [2]. Due to resource constraints imposed by mobile devices, the proposed KWS system must have a small memory and CPU footprint, while simultaneously providing high performance in terms of false alarm (FA) and false reject (FR) rates. Further, since the system is likely to be used in various situations (e.g., while driving, in a crowded restaurant, etc.) we require that the system perform well in a variety of conditions.

Traditional approaches to KWS rely on the use of large vocabulary continuous speech recognition (LVCSR) systems (e.g., [3, 4]). In these systems, the task of KWS is reduced to searching for keywords in utterances that have already been recognized and indexed by the LVCSR system, and as such are unfeasible for on-device KWS due to memory and power constraints. Examples of alternative approaches to KWS, which avoid a full Viterbi decoding of the audio, include dynamic time warping (DTW) on phone posteriorgrams [5], training large-margin classifiers [6, 7] and point process models [8].

In previous work, we presented a KWS system [2] based on deep neural networks (DNNs) trained to identify word targets. This method was shown to significantly outperform a baseline keyword-filler system. This system is appealing for our task because it can be implemented very efficiently to run in real-time on devices, and memory and power consumption can be easily adjusted by changing the number of parameters in the DNN. Although the proposed system works extremely well in clean conditions, performance degrades significantly when speech is corrupted by noise, or when the distance between the speaker and the microphone increases (i.e., in far-field conditions). In the present work, we explore techniques to improve system robustness in these two conditions.

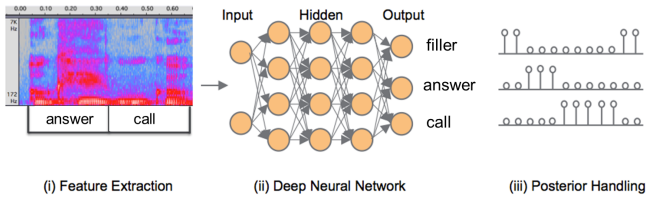


Fig. 1: Block diagram of DNN-based KWS system proposed in [2].

In order to improve robustness to background noise, we explore the use of multi-style training, i.e., creating training instances by artificially adding in noise to simulate expected evaluation conditions, which has been shown to result in large improvements while recognizing noisy speech [9, 10, 11]. To improve performance in far-field conditions, we develop a novel formulation of automatic gain control (AGC) [12, 13, 14] that attempts to selectively boost signal levels based on estimating whether the signal contains speech or not.

The rest of the paper is structured as follows: In Section 2 we review the details of the proposed DNN-based KWS system as described in [2]. In Section 2.2, we present details of our proposed AGC approach. We describe our experimental setup in Section 3, and evaluate the effectiveness of our proposed system in Sections 4 and 5 where we demonstrate that the combination of AGC and multi-style training improves false reject rates by up to 75%, relative, in noisy far-field conditions. We also consider the effectiveness of these techniques when applied to a DNN system with an order of magnitude fewer parameters than the system presented in [2], where we find that using a combination of multi-style training and AGC allow us to obtain competitive performance with extremely small models. We conclude with a discussion of our findings in Section 6.

2. DNN-BASED KEYWORD SPOTTING SYSTEM

Our KWS system is an extension of our previous work [2], a block diagram of which is shown in Figure 1. Conceptually, our system consists of three components: (i) a feature extraction module which extracts acoustic features which are input to a neural network, (ii) a DNN, which computes posterior probabilities of the individual

*This work was performed as part of a summer internship at Google.

words in the keyword phrase, and (iii) a posterior handling module which combines the individual frame-level posterior scores into a single score corresponding to the keyword. Feature extraction and DNN topology are described in Section 3 where we describe our experimental setup; the posterior handling module in described below.

2.1. Detecting Keywords using DNN Posteriors

In order to detect keywords in the incoming speech at run time, we run our keyword detection algorithm repeatedly over sliding windows of length T_s of the input speech. We denote $\mathbf{x} = \{x_1, x_2, \cdots, x_{T_s}\}$ as one such input window over the utterance, consisting of individual frames $x_t \in \mathbb{R}^n$ (in our experiments, these correspond to log-mel-filterbank energies, stacked together with adjacent left and right context frames). We assume that the keyword to be detected, $\mathbf{w}$, consists of M words, $\mathbf{w} = \{w_1, w_2, \cdots, w_M\}$. For each frame, t, in the input speech, we denote the posterior probability of the k-th word in the keyword by $y_t(w_k)$. We compute smoothed posterior values, $s_t(w_i)$, by averaging the posteriors over the previous L frames, which in [2] are then used to define the keyword score, $\widehat{h}(\mathbf{x}, \mathbf{w})$, as follows:

$$s_t(w_i) = \frac{1}{L} \sum_{j=t-L+1}^{t} y_j(w_i); \quad \widehat{h}(\mathbf{x}, \mathbf{w}) = \left[\prod_{i=1}^{M} \max_{1 \le t \le T_s} s_t(w_i) \right]^{\frac{1}{M}} \tag{1}$$

The chief advantage of the keyword score in Equation 1 is its simplicity: the score can be computed in $\Theta(MT)$ time, and has been shown to achieve good KWS performance [2]. However, this score does not account for the *relative order in which the keyword targets 'fire'*. Therefore, in the present work, we define an alternative keyword score, $h(\mathbf{x}, \mathbf{w})$, as the largest product of the smoothed posteriors in the input sliding window, *subject to the constraint that the individual words 'fire' in the same order as specified in the keyword*,

$$h(\mathbf{x}, \mathbf{w}) = \left[\max_{1 \le t_1 \le \cdots \le t_M \le T_s} \prod_{i=1}^{M} s_{t_i}(w_i) \right]^{\frac{1}{M}} \tag{2}$$

Although the keyword score in Equation 2 contains additional constraints, it can still be computed in $\Theta(MT)$ time using dynamic programming. In pilot experiments, we found that imposing the ordering constraint in Equation 2 significantly reduces FAs relative to the keyword score in Equation 1. All results in this work are therefore reported using the keyword score in Equation 2.

2.2. Automatic Gain Control

In order to improve the KWS system's performance in far-field conditions, where the input signal is attenuated because of distance, we propose to use automatic gain control (AGC) to normalize the signal level. The core assumption in traditional AGC systems is that the signal to be boosted is present most of the time in the incoming audio [12]. In our application, this tends to be the opposite: most of the incoming audio does not contain speech, and thus it is undesirable to boost it. This necessitates a more dynamic AGC (cf., [15]).

In order to distinguish portions of the sound signal corresponding to input speech, we estimate two probabilistic classes which model peak levels of the time-domain audio samples: the signal, $\mathcal{S}$, corresponding to input speech, and the non-signal floor, $\mathcal{B}$, corresponding to background where no speech is present. This allows us to selectively gain up only those speech samples that are likely to contain speech, without boosting background noise. Our approach is similar, in spirit, to previous work (e.g., [13, 14]), except that our AGC implementation is designed to have a small footprint, introduce minimal latency and be efficient in terms of its power consumption.

2.2.1. A Generative Model of Peak Signal Levels

We process the input time-domain signal by segmenting it into 100 ms non-overlapping chunks of audio samples. We then compute the peak signal level, l, of the audio samples in each of these chunks, where $0 \le l \le 1$. The peak-level of audio chunks is modeled as being generated from a mixture of two Gaussians: corresponding to speech $l_{\mathcal{S}} \sim \mathcal{N}(\mu_{\mathcal{S}}, \sigma_{\mathcal{S}})$ and non-speech background $l_{\mathcal{B}} \sim \mathcal{N}(\mu_{\mathcal{B}}, \sigma_{\mathcal{B}})$, with $\mu_{\mathcal{S}} > \mu_{\mathcal{B}}$.[1] Using relatively long non-overlapping chunks allows us to assume that individual peak chunk-levels are independent. Peak levels are used instead of softer norms (mean, RMS, etc.) because we desire an indicator of the strength of the dominant source in the chunk (e.g., 90ms of quiet speech and 10ms of loud speech should still be identified as containing loud speech).

2.2.2. Parameter Estimation

We estimate the unknown model parameters: the means $(\mu_{\mathcal{S}}, \mu_{\mathcal{B}})$ and standard deviations $(\sigma_{\mathcal{S}}, \sigma_{\mathcal{B}})$ using the Expectation-Maximization (EM) algorithm (specifically, "hard"-EM), with modifications for efficient real-time updates:

1. Given current model estimates, we classify a new signal level, l, as either $\mathcal{S}$ or $\mathcal{B}$, using the simplified maximum-likelihood hypothesis testing rule described in Section 2.2.3.
2. Once the chunk has been classified as either $\mathcal{S}$ or $\mathcal{B}$, we update model parameters for the corresponding class. For GMMs, this requires the computation of sample means and variances for each class. To do this efficiently, in real-time, without using additional memory, we recursively compute "moving averages" of the sample means and variances.[2]

2.2.3. Maximum-Likelihood Hypothesis Testing Rule

In order to classify the measured peak signal level, l, we compute the likelihood ratio, R, to compare likelihoods of it belonging to the two classes, namely speech or background noise:

$$R = \frac{p(l|\mathcal{S})}{p(l|\mathcal{B})} = \frac{\sigma_{\mathcal{B}}}{\sigma_{\mathcal{S}}} \exp\left(\frac{-0.5}{z_{\mathcal{S}}^2 - z_{\mathcal{B}}^2} \right) \tag{3}$$

where, $z_{\mathcal{S}}$ and $z_{\mathcal{B}}$ are z-scores,

$$z_{\mathcal{S}} = \frac{l - \mu_{\mathcal{S}}}{\sigma_{\mathcal{S}}} \qquad z_{\mathcal{B}} = \frac{l - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} \tag{4}$$

Thus, $R > 1 \iff z_{\mathcal{S}}^2 - z_{\mathcal{B}}^2 < -2\ln(\frac{\sigma_{\mathcal{S}}}{\sigma_{\mathcal{B}}})$. We make the further simplifying approximation that $\sigma_{\mathcal{S}} \approx \sigma_{\mathcal{B}}$, so that, our final classification rule can be written as:[3]

$$\text{classification}(l) = \begin{cases} \mathcal{S}, & \text{if } z_{\mathcal{S}}^2 < z_{\mathcal{B}}^2 \\ \mathcal{B}, & \text{otherwise} \end{cases} \tag{5}$$

[1] The mixture component weights are both assumed to be 0.5, and are not learned in our model.

[2] $(\mu \leftarrow \kappa_\mu l + (1 - \kappa_\mu)\mu)$ and $\left(\sigma^2 \leftarrow \kappa_\sigma (l - \mu)^2 + (1 - \kappa_\sigma)\sigma^2\right)$, with $\kappa_\mu = 0.5$, and $\kappa_\sigma = 0.33$ in our experiments, determined by tuning on the development set.

[3] This is reasonable because we expect the signal and noise variances to be of roughly the same order, and in fact our 'decay' logic weakly enforces this (see Section 2.2.4). Avoiding the computation of $\ln(\frac{\sigma_{\mathcal{S}}}{\sigma_{\mathcal{B}}})$ allows for significantly reduced power consumption when running on the device.

2.2.4. Model Decay and Gain Strategy

We 'decay' model estimates, to mediate certain undesirable effects of incorrect classification: if either $\sigma_{\mathcal{S}}$ or $\sigma_{\mathcal{B}}$ (denoted by $\sigma_{\mathcal{X}}$ in Equation 6, below) becomes too concentrated, new chunks will likely not be classified into that class, and the model will not adapt. Therefore, we slowly increase the variance of both models with each iteration, to model growth of uncertainty over time.[4]

$$\text{if } (\sigma_{\mathcal{X}}^2 < \tau^2) \text{ then } \sigma_{\mathcal{X}}^2 \leftarrow \sigma_{\mathcal{X}}^2 + \frac{\sigma_{\mathcal{S}}^2 + \sigma_{\mathcal{B}}^2}{2\delta} \tag{6}$$

If a chunk is classified as $\mathcal{S}$, and if we are confident in our estimates, i.e., if the signal and noise models are well-separated ($\mu_{\mathcal{S}} - \mu_B > \sigma_{\mathcal{S}} + \sigma_{\mathcal{B}}$), then we gain the input signal in order to normalize the signal level; however, if the signal and noise estimates are not well-separated, then we use a conservative gain strategy,[5]

$$\text{gain} = \begin{cases} \frac{\theta}{\mu_{\mathcal{S}} + \sigma_{\mathcal{S}}}, & \text{if } \mu_{\mathcal{S}} - \mu_{\mathcal{B}} > \sigma_{\mathcal{S}} + \sigma_{\mathcal{B}} \\ \frac{\theta'}{\min(\mu_{\mathcal{S}} + \sigma_{\mathcal{S}}, \mu_{\mathcal{B}} + \sigma_{\mathcal{B}})}, & \text{otherwise} \end{cases} \tag{7}$$

The gain is applied to scale up the input samples smoothly across chunks (we do not attenuate the signal), while ensuring that no clipping occurs.

3. EXPERIMENTAL SETUP

Our DNN systems are standard feed-forward, fully connected neural networks, with three hidden layers and a softmax output layer. The softmax output layer contains one output target for each of the words in the keyword phrase to be detected, plus a single additional output target which represents all frames that do not belong to any of the words in the keyword (denoted as 'filler' in Figure 1). We determine (word) labels for each input acoustic frame of the training utterances by performing a forced-alignment using a large LVCSR system [16]. We use rectified linear unit (ReLU) activation functions for the hidden layers [17]. The input to the DNN consists of log-mel-filterbank energies (computed over 25ms of speech, with a 10ms frame-shift), stacked together with left and right context frames. Since each additional frame of right context adds an additional 10ms of latency to the system, we use a larger number of left contextual frames than right contextual frames. Our acoustic feature extraction module and the DNN runtime engine are both implemented using fixed-point arithmetic in order to minimize power consumption [18]. The network weights and biases are trained to optimize a cross-entropy criterion using distributed asynchronous gradient descent [19].

3.1. Datasets

In order to validate the proposed approach, we selected fourteen phrases[6] and collected about 10K–15K utterances containing each of these phrases. We also collected a much larger set of approximately 396K utterances which do not contain any of the keywords and are thus used as 'negative' training data. The utterances were then randomly split into training, development, and evaluation sets in the ratio of 80:5:15, respectively. We also collected a much larger set of approximately 100K speech utterances from our anonymized voice-search logs (dev-voicesearch) to use as an additional development set; we select the system threshold to correspond to 1 FA per hour of speech on this set.[7] We further collected two types of additional noisy data to represent two common use-cases for our proposed system: cafeteria noise, consisting mostly of background speech, occasionally mixed in with some music, and car noise collected in various conditions (e.g., window cracked open, radio playing, air-conditioner on, etc.). The collected noise sources were partitioned into separate training/development and evaluation portions.

We created noisy training and evaluation sets by artificially adding in car and cafeteria noise at various SNRs. Noisy training data was created by adding a random snippet of car or cafeteria noise to training set utterances at an SNR randomly sampled between [-5dB, +10dB]. In addition to a clean evaluation set, consisting of the utterances containing a given keyword and the set of 'negative' utterances, we also created noisy versions of the clean evaluation set by adding in car noise at -5dB (car_-5db), and cafeteria noise at +5dB (cafe_5db), respectively. Since the most common far-field use case for our application is one in which the user is driving, we created far-field versions of the clean and car_-5db evaluations sets by simulating a distance of 100 cm between the speaker and microphone (clean_100cm, car_-5db_100cm, respectively).

4. EXPERIMENTS I: IMPACT OF MULTI-STYLE TRAINING AND AGC

Our first set of experiments is aimed at determining the impact of multi-style training and AGC on system performance. Following [2], our DNN baseline system (baseline) consists of 3 hidden layers of 128 nodes each. The input to the net consists of 40 dimensional log-mel-filterbank energies with 30 frames of left-context and 10 frames of right-context. We run the keyword detection algorithm over sliding windows of 100 frames ($T_s = 100$), with posteriors smoothed over 30 frames ($L = 30$).

We compare performance of the baseline system against (i) a system trained with multi-style training (MS), (ii) with AGC turned on during evaluation (AGC) or (iii) both (MS+AGC). Receiver operating characteristic (ROC) curves comparing the systems are presented in Figure 2. We present results at the operating point of 1 false alarm (FA) per hour on the dev-voicesearch set in Table 1. Since average FA rates are consistently low across all of our systems and evaluation sets, ranging from 0.03%–0.10%, we only report false reject (FR) rates in Table 1. As can be seen in the table, performance degrades significantly on the noisy and far-field sets, relative to the clean set. The use of multi-style training significantly improves performance over the baseline system on the noisy sets, with relative improvements of 11.1% (car) and 27.3% (cafe) in FR rates, although it does not produce gains on the far-field sets. Using AGC alone, produces large gains (approximately 75% relative) in FR rates of both far-field datasets, but produces worse performance in the noisy evaluation sets with significant degradation on cafe_5db. We hypothesize that this is the result of errors in the AGC's modeling of speech and background levels. We note, however, that removing this probabilistic model of speech and background results in a much larger performance degradation.[8] Using a combination of AGC and multi-style

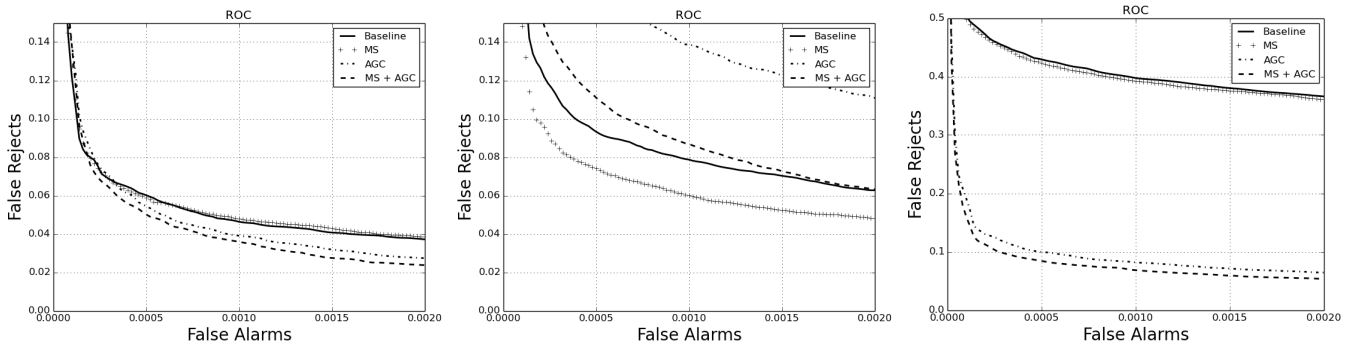
[4] In our experiments, we set $\delta = 16$, and $\tau = 0.5 * (\text{max\_signal\_level})$ in Equation 6, determined by tuning on the development set.

[5] We set $\theta = 0.8$ and $\theta' = 0.1$ in our experiments, determined by tuning on a development set.

[6] The keyword phrases are: 'answer call', 'decline call', 'email guests', 'fast forward', 'next playlist', 'next song', 'next track', 'pause music', 'pause this', 'play music', 'set clock', 'set time', 'start timer', and 'take note'.

[7] At this operating point, our baseline system in Section 4 achieves 1 FA per 25.6 hours of music and 1 FA per 45.7 hours of background noise.

[8] We compared our AGC implementation, to one in which there was no probabilistic modeling of speech and background: each 100ms chunk was smoothly gained towards $\frac{\theta}{l}$, where l is the peak chunk-level. At an operating point of 1 FA per hour on the dev-voicesearch set, the alternative AGC



(a) Results on clean (b) Results on cafe_5db (c) Resuts on car_-5db_100cm

Fig. 2: ROC curves comparing performance of the baseline system (baseline) against a system that employs multi-style training (MS), or AGC during evaluation (AGC) or both (MS+AGC), for three of the evaluation sets: clean, cafe_5db and car_-5db_100cm. ROC curves are plotted by averaging performance over all of the keywords for each evaluation set. Curves closer to the origin are better.

training, improves performance over using AGC alone, achieving large gains in clean and far-field conditions ranging from 22.2%–78.1%, relative, albeit some performance degradation in noisy sets. However, this difference is not significant ($p \geq 0.07$)[9] due to the variance in FR rates for this set across the fourteen keywords. Overall, using a combination of multi-style training and AGC allows for the best combination of performance averaged over all evaluation sets. In particular, we note that performance of the MS+AGC system does not change dramatically across all of the sets, unlike the baseline where performance varied significantly across all of the evaluation conditions.

System	baseline	MS	AGC	MS+AGC
clean	6.34%	6.48%	5.56%	4.93%†
car_-5dB	8.85%	7.87%†	12.41%	9.79%
cafe_5db	11.24%	8.83%†	22.02%†	16.04%
clean_100cm	46.36%	46.61%	10.72%†	10.98%†
car_-5db_100cm	47.00%	46.97%	11.97%†	10.31%†
average	23.96%	23.35%	12.54%	10.41%

Table 1: FR rates averaged across all keyword phrases, corresponding to an operating point of 1 FA per hour on the dev-voicesearch set. (†) indicates a significant difference ($p \leq 0.05$) in FR rates, relative to the baseline system.

5. EXPERIMENTS II: REDUCING MODEL SIZE

In order to satisfy memory constraints of mobile devices, our DNN models need to be small; generally speaking, CPU usage and thus power consumption are directly proportional to the number of parameters. In our second set of experiments, we examine how model performance varies as a function of model size. In particular, we are interested in determining whether we can achieve competitive performance while reducing model size by an order of magnitude.

We reduce the number of model parameters in the system, relative to the baseline presented in Section 4, by using fewer mel-filterbanks (15 instead of 40), reducing left and right context frames (25 and 5, instead of 30 and 10, respectively), and using fewer nodes in each of the three hidden layers (64 instead of 128), so that the system has about 40K parameters (baseline-40k).

In Table 2, we report performance of the KWS systems averaged across all of the keyword phrases, corresponding to an operating point of 1 FA per hour on the dev-voicesearch set. We report performance of the baseline system (baseline-40k), a system trained with multi-style training (baseline-40k+MS), and a system with both multi-style training and AGC (baseline-40k+MS+AGC). Since average FA rates across all systems and evaluation sets are similar, ranging from 0.03%–0.10%, we only report FR rates.

System	baseline-40k	baseline-40k +MS	baseline-40k +MS+AGC
clean	9.58%	10.06%	8.12%†
car_-5dB	13.41%	12.39%†	16.76%
cafe_5db	16.61%	14.12%†	25.25%†
clean_100cm	56.01%	56.62%	16.60%†
car_-5db_100cm	56.78%	57.12%	16.78%†
average	30.48%	30.06%	16.70%

Table 2: FR rates averaged across all keyword phrases, corresponding to an operating point of 1 FA per hour on the dev-voicesearch set. (†) indicates a significant difference ($p \leq 0.05$) in FR rates, relative to the baseline-40k system.

As seen in Table 2, although performance of the 40K parameter baseline is worse than the 240K parameter baseline, trends are similar as before: performance improves significantly with multi-style training and AGC. In particular, multi-style training alone results in significant improvements on the two noisy sets, with relative improvements of 7.6% (car) and 15.0% (cafe) in FR rates. Similarly, the combination of multi-style training and AGC obtains large gains in FR rates in the far-field condition (approximately 70% relative), albeit a significant degradation in the noisy sets.

6. CONCLUSIONS

We examined techniques to improve the robustness of our previously proposed system for small-footprint KWS [2], where we found that the use of multi-style training, coupled with a novel "speech-aware" AGC formulation allowed us to significantly improve KWS performance in noisy, far-field conditions. Using both techniques, we found that we were able to train a system with 6x fewer parameters than our baseline, that outperformed a much larger baseline without these techniques, and was only about 1.6x worse in terms of FR rates than the larger baseline with these techniques applied.

performed well on clean sets (e.g., 10.6% FR @ 0.1% FA on clean), but it performed significantly worse than our "speech-aware" AGC system (AGC) on noisy datasets, in terms of both FR and FA (e.g., 18.9% FR @ 1.1% FA on car_-5db and 33.5% FR @ 0.3% FA on cafe_5db).

[9]We use a two-tailed Wilcoxon signed-ranks test for all significance testing in this work.

7. REFERENCES

[1] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope, ""Your word is my command": Google search by voice: A case study," in *Advances in Speech Recognition*, Amy Neustein, Ed., pp. 61–90. Springer US, 2010.

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

[3] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddingtion, "Results of the 2006 spoken term detection evaluation," in *Proceedings of Special Interest Group on Information Retrieval (SIGIR)*, 2007, vol. 7, pp. 51–57.

[4] J. Cui, X. Cui, B. Ramabhadran, J. Kim, B. Kingsbury, J. Mamou, L. Mangu, M. Picheny, T. N. Sainath, and A. Sethy, "Developing speech recognition systems for corpus indexing under the IARPA Babel program," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6753–6757.

[5] T. J. Hazen, W. Shen, and C. M. White, "Query-by-example spoken term detection using phonetic posteriorgram templates," in *IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU*, 2009, pp. 421–426.

[6] J. Keshet, D. Grangier, and S. Bengio, "Discriminative keyword spotting," *Speech Communication*, vol. 51, no. 4, pp. 317–329, 2009.

[7] R. Prabhavalkar, K. Livescu, E. Fosler-Lussier, and J. Keshet, "Discriminative articulatory models for spoken term detection in low-resource conversational settings," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 8287–8291.

[8] K. Kintzley, A. Jansen, K. Church, and H. Hermansky, "Inverting the point process model for fast phonetic keyword search," in *Proceedings of Annual Conference of the International Speech Communication Association (Interspeech)*, 2012, pp. 2348–2441.

[9] M. L. Seltzer, "Acoustic model training for robust speech recognition," in *Techniques for Noise Robustness in Automatic Speech Recognition*, T. Virtanen, R. Singh, and B. Raj, Eds., pp. 347–368. John Wiley & Sons, 2012.

[10] A. Narayanan and D. L. Wang, "Investigation of speech separation as a front-end for noise robust speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 4, pp. 826–835, 2014.

[11] D. Yu, M. Seltzer, J. Li, J. T. Huang, and F. Seide, "Feature learning in deep neural networks - studies on speech recognition," in *Proceedings of International Conference on Learning Representations (ICLR)*, May 2013.

[12] A. Perez, J. P. Pueyo, S. Celma, and B. Calvo, *Automatic Gain Control: Techniques and Architectures for RF Receivers*, Springer-Verlag, 2011.

[13] P. L. Chu, "Voice-activated AGC for teleconferencing," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1996, vol. 2, pp. 929–932.

[14] F. J. Archibald, "Software implementation of automatic gain controller for speech signal," Available Online: `www.ti.com/lit/wp/spraal1/spraal1.pdf`, 2008.

[15] R. F. Lyon, "Automatic gain control in cochlear mechanics," in *The Mechanics and Biophysics of Hearing*, P. Dallos, C. D. Geisler, J. W. Matthews, M. A. Ruggero, and C. R. Steele, Eds., vol. 87, pp. 395–402. Springer, 1990.

[16] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proceedings of Annual Conference of the International Speech Communication Association (Interspeech)*, 2012.

[17] M. D. Zeiler, M. A. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, "On rectified linear units for speech processing," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[18] X. Lei, A. Senior, A. Gruenstein, and J. Sorensen, "Accurate and compact large vocabulary speech recognition on mobile devices," in *Proceedings of Annual Conference of the International Speech Communication Association (Interspeech)*, 2013, pp. 662–665.

[19] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng, "Large scale distributed deep networks," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1223–1231.