Raise to Speak: An Accurate, Low-power Detector for Activating Voice Assistants on Smartwatches

Shiwen Zhao
Apple Inc
shiwen_zhao@apple.com

Brandt Westing
Apple Inc
bwesting@apple.com

Shawn Scully
Apple Inc
sscully@apple.com

Heri Nieto
Apple Inc
hnieto@apple.com

Roman Holenstein
Apple Inc
rholenstein@apple.com

Minwoo Jeong
Apple Inc
mjeong@apple.com

Krishna Sridhar
Apple Inc
srikrishna_sridhar@apple.com

Brandon Newendorp
Apple Inc
bnewendorp@apple.com

Mike Bastian
Apple Inc
mbastian@apple.com

Sethu Raman
Apple Inc
sethur@apple.com

Tim Paek
Apple Inc
timpaek@apple.com

Kevin Lynch
Apple Inc
kevinlynch@apple.com

Carlos Guestrin
Apple Inc
guestrin@apple.com

ABSTRACT

The two most common ways to activate intelligent voice assistants (IVAs) are button presses and trigger phrases. This paper describes a new way to invoke IVAs on smartwatches: simply raise your hand and speak naturally. To achieve this experience, we designed an accurate, low-power detector that works on a wide range of environments and activity scenarios with minimal impact to battery life, memory footprint, and processor utilization.

The raise to speak (RTS) detector consists of four main components: an on-device gesture convolutional neural network (CNN) that uses accelerometer data to detect specific poses; an on-device speech CNN to detect proximal human speech; a policy model to combine signals from the motion and speech detector; and an off-device false trigger mitigation (FTM) system to reduce unintentional invocations trigged by the on-device detector. Majority of the components of the detector run on-device to preserve user privacy.

The RTS detector was released in watchOS 5.0 and is running on millions of devices worldwide.

CCS CONCEPTS

• **Human-centered computing → Gestural input**; • **Computing methodologies → Speech recognition**; **Neural networks**; *Supervised learning by classification.*

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330761

KEYWORDS

Gesture recognition, speech detection, neural network, multimodal

ACM Reference Format:
Shiwen Zhao, Brandt Westing, Shawn Scully, Heri Nieto, Roman Holenstein, Minwoo Jeong, Krishna Sridhar, Brandon Newendorp, Mike Bastian, Sethu Raman, Tim Paek, Kevin Lynch, and Carlos Guestrin. 2019. Raise to Speak: An Accurate, Low-power Detector for Activating Voice Assistants on Smartwatches. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19), August 4–8, 2019, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330761

1 INTRODUCTION

Intelligent voice assistants (IVA) have become ubiquitous [15]. The two most common ways to invoke IVAs are using physical buttons or issuing specific trigger phrases such as "Hey Siri". Neither of these invocation methods are natural on smartwatches; the button invocation is a two-handed invocation while trigger phrases are unnatural when used frequently, especially in public environments [14]. In this paper we propose a more natural invocation for IVAs on smartwatches; simply raise and speak to the device. To enable this interaction, we designed an accurate, low-power detector (called "Raise to Speak" or RTS) that uses only an accelerometer and a microphone. The detector is designed to run mostly on-device to preserve user privacy and provide a low-latency experience for users.

There are two main contributions of this work. First, we propose a new invocation method for IVAs on smartwatches that is both natural as well as specific enough for an accurate detector to identify. Second, we design a four component detector that works on a wide range of environments with minimal impact to battery life, memory footprint, and processor utilization.

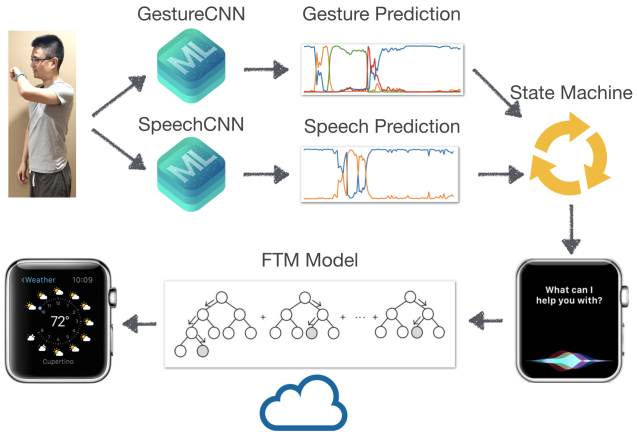

Figure 1: Overview of RTS pipeline.

In designing the RTS detector, there are two main challenges to address. First, the detector must be accurate in a wide range of environments from quiet offices to busy streets. Furthermore, the invocation is expected to feel natural in a wide range of activity scenarios such as sitting, standing, walking etc. for a varied distribution of users. Thus, the already known challenges of user-independent gesture recognition problems [1] are amplified by additional variability in environment and user activity. Second, the entire detector must run in an extremely resource constrained environment such as a smartwatch. Running the detector on-device is crucial to selectively allow audio that is intended for the IVA off device while keeping the audio that is not intended private, maximally preserving user privacy. The key challenge here is to optimize accuracy of the detector with other system constraints such as memory footprint, disk size, processor utilization, and battery usage. Creating neural network models with these tradeoffs in mind is an active area of research [7, 9, 19].

Figure 1 provides an overview of the RTS system. There are four main components; An on-device gesture detector (GestureCNN) that uses accelerometer data to detect specific stages of hand movement; an on-device proximal-speech detector (SpeechCNN) to detect proximate human speech; a policy model to combine signals from the motion and speech detector; and an off-device false trigger mitigation (FTM) system to reduce unintentional invocations trigged by the on-device detector. The system is deployed on Apple's watchOS 5.0 and used by millions of customers. The on-device detector only starts to operate once Apple Watch face is wakened to minimize the usage of private signals of users.

This paper is organized as follows. In Section 2, we briefly highlight related work. In Section 3, we define the details of the user experience to invoke the IVA. Section 4 introduces the details of the RTS detector. In Section 5, we evaluate the detector on real-world data set collected from a user study.

2 RELATED WORK

Combining gesture and speech signals to facilitate human computer interaction is an area of active research [2, 16, 20]. [20] provides a more detailed review on this topic. The two most relevant systems are from [16] and [2]. [16] describes a system that combines gesture as captured by a camera with speech as captured by a head-mounted microphone to estimate cues in conversational interactions. *Shake2Talk* is another experience where users can send audio messages via simple gestures [2]. These gestures were captured using a SHAKE device with multiple motion capturing sensors such as accelerometers, gyroscopes and capacitive sensors. One of the key differences between these systems and RTS is that RTS is deployed on a resource constrained embedded device using only sensors that are available today (accelerometer and microphone) on almost all smartwatches.

3 USER EXPERIENCE

The goal of the RTS system is to enable a gestural activation method whereby the IVA on smartwatches can be activated without trigger phrase or button press. The system is designed to trigger on a gesture where the user raises their smartwatch to their mouth and speaks into the watch to converse with the IVA. A conversation can include issuing commands (e.g. *Set a timer for* 2 *min*), asking questions (e.g. *what is the weather today?*), replying to messages (e.g. *Reply: I am running late*) or performing any action that you could otherwise do without needing to press buttons or mention trigger phases.

There are four distinct stages related to an RTS gesture; raising, raised, dropping, and dropped (see Figure 2 for a detailed breakdown of the gesture). In each of these stages, we identify constraints on the gesture that can help improve the accuracy of the detector while still ensuring that the gesture feels natural to most users. These constraints are:

- During the raising, the increase in movement of the device must be at least 4 inches vertically with respect to ground and the absolute acceleration must be at least 0.5g. This prevents the detector from triggering on gestures that imply little positional change of the device or that are too slow.
- The raised stage cannot start less than 0.5 seconds from the start of the raising so as to prevent triggering on gestures that are a result of high velocity positional changes.
- On completion of the raising stage, the watch must be within 4 inches of the mouth. This serves to filter out gestures where the user is not clearly speaking into the watch.
- During the raised stage, the watch must point towards the user's face while the query is being issued to the IVA. In addition the forearm should be approximately parallel with respect to ground.

In addition to gesture constraints, we also require the speech issued to the IVA must be at normal conversational loudness.

The RTS detector is expected to work for a wide distribution of users performing different activities and within most acoustic environments. The primary user activity scenarios that are targeted are sitting, standing, walking, running and driving. For acoustic environments, a spectrum of noise levels ranging from quiet places (e.g. office/home) to moderate and even loud environments (e.g. coffeeshops/bars) are supported.

To evaluate the performance of the detector as it relates to the user experience, we define the following metrics:

Table 1: Details of 31 gesture temporal features

Name	Num of features
raw accelerometer x, y, z	3
moving averages of window size 10	3
moving averages of window size 20	3
moving averages of window size 50	3
moving standard deviations of window size 10	3
moving standard deviations of window size 20	3
moving standard deviations of window size 50	3
20 step difference of moving average of 10	3
50 step difference of moving average of 10	3
40 step difference of moving average of 20	3
absolute value of acceleration vector	1

- *False reject rate* (FRR). This is defined as the proportion of RTS attempts that do not trigger the IVA over the total number of attempts. It measures the type two error. A large FRR suggests that users are unable to freely trigger the IVA using RTS.
- *False accepts per week per user* (FAWU). This metric measures the frequency that RTS falsely triggers the IVA. For the purposes of the RTS system, this is considered the type one error. A large FAWU leads to an annoying user experience and poor privacy preservation even for users that may not wish to use the IVA normally.

High quality detectors have a low FRR while maintaining a small number of FAWU for supported activity scenarios and acoustic environments.

4 THE RTS DETECTOR

In this section we detail each of the four components of the RTS detector; the on-device gesture detector (GestureCNN), the on-device proximal-speech detector (SpeechCNN), the on-device policy model, and the off-device false trigger mitigator (FTM) system.

4.1 GestureCNN

The gesture detector is designed to trigger each time the user's physical gesture satisfies the constraints set in Section 3. The GestureCNN operates on the accelerometer sensor input. Each physical gesture is broken down into four stages; raising, raised, dropping, and dropped (labelled using color strips in Figure 2B). Using this methodology, we turn the gesture detection problem into a four-class timeseries classification problem. GestureCNN is a compact convolutional neural network (CNN) trained to solve this problem.

GestureCNN operates on derived temporal features from the raw accelerometer data. After experimentation, 28 temporal features (see Table 1) were identified as salient to the problem. These features include time-windowed aggregates such as moving averages over different window sizes; distant differences of moving averages and absolute acceleration. In addition to the 28 derived features, GestureCNN also uses the three raw accelerometer signals (x, y, z).

The GestureCNN architecture is described in Figure 3. The first layer is a 1D convolution layer with 20 filters of size one. The convolutions are along the temporal dimension to create a feature map

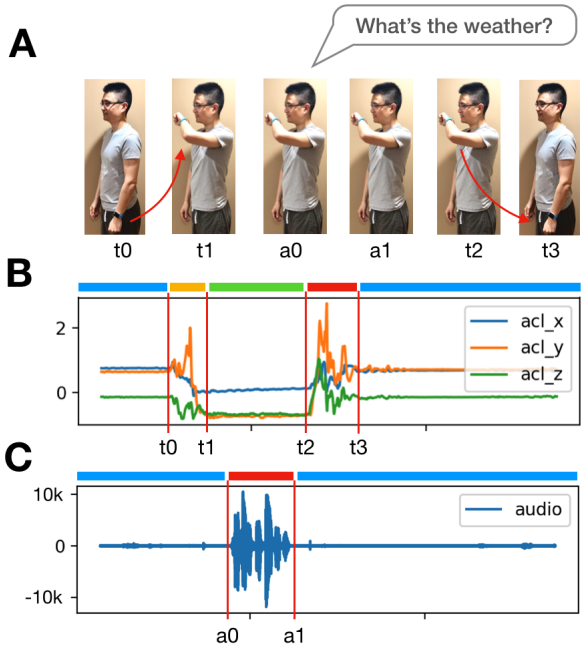

Figure 2: Input signals. A: Six timepoints are labeled, respectively representing the start of raising (t0), end of raising (t1), start of speech (a0), end of speech (a1), start of dropping (t2) and end of dropping (t3). B: Accelerometer timeseries data for gesture detection. The timeseries is divided into four classes as indicated by the top strips: raising class (yellow), raised class (green), dropping class (red) and the dropped (blue). C: Raw speech signal for voice activity detection. The speech is also divided into two classes including speech class (red strip) and non-speech class (blue strip).

of size 50 for each filter. The output of the convolutional layer is flattened and stacked as input into two fully connected layers. The sizes of the two fully connected layers are 128 and 32 respectively. The final layer is a softmax layer. For each fully connected layer (including the flatten layer from the convolution), batch normalization [10] and dropout [18] is applied to the respective output. The dropout probability was set to 0.5. The model is trained for 50 epochs with a batch size of 1024 and constant learning rate of 0.001. The optimization algorithm chosen was Adam [12] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The number of parameters in the model sums to $133K$ and the size of the model loaded into memory is approximately $500KB$.

4.2 SpeechCNN

The proximal-speech detector is designed to trigger each time the user invokes the IVA by speaking into the watch in a way that satisfies our speech constraint in Section 3. SpeechCNN operates on audio signals from the microphone captured at 16kHz. Figure 2C shows voice activity during a typical command issued to an IVA. In each session, the start and end of the speech query divides the audio into two separate parts; speech and non-speech. With this

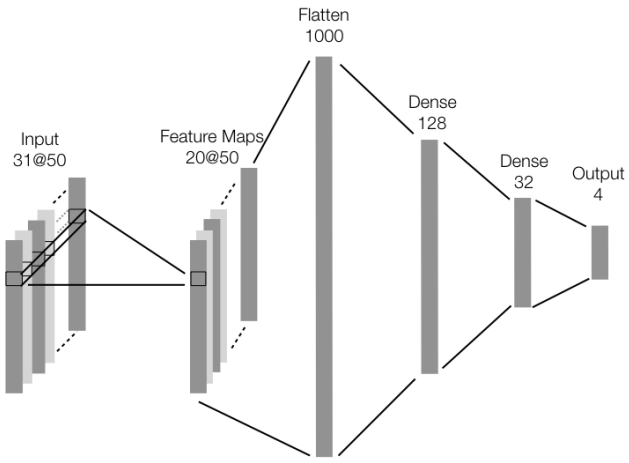

Figure 3: The architecture of the GestureCNN. The CNN consists of a convolutional layer and two dense layers.

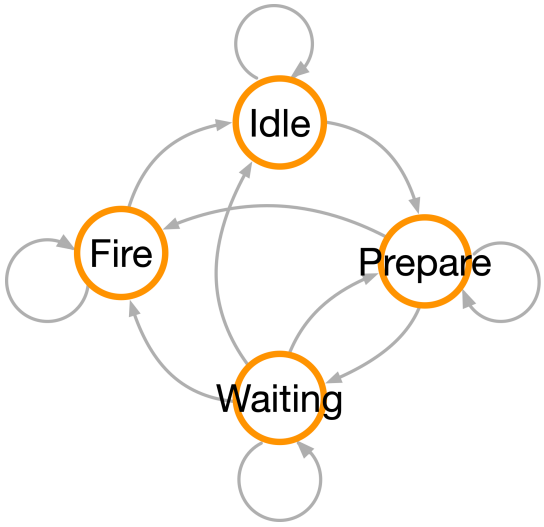

Figure 4: The gesture state machines implemented in policy model.

approach, we were able to formulate this problem as a proximal voice activity detection (VAD) [6, 17, 21]. Designing accurate low-power VADs on embedded systems is an area of active research [4, 8]

SpeechCNN operates on 40 mel-frequency filter banks derived from the audio signals. The 40 filter banks are calculated in each frame, resulting in a 40 dimensional time series at a rate of 100Hz. The input to the SpeechCNN is a historical buffer of filter banks from 50 frames. SpeechCNN is of the same architecture as GestureCNN model, except an input size of 50 by 40 and the output layer had two nodes (Figure 3). The training parameters are set to the same values as the GestureCNN model. The number of parameters in SpeechCNN and its size are similar to the counterparts of the GestureCNN model (133K parameters and 497KB in size).

4.3 Policy Model

While GestureCNN and SpeechCNN serve as independent components, the on-device detector requires a combination of coordinated gesture and speech. To take this into account, a light-weight policy model that combines the outputs from the GestureCNN and SpeechCNN is used to make the final decision for on-device triggering of the IVA. This policy model consists of two state machines; a gesture state machine and a speech state machine. The gesture state machine contains four states: `Idle`, `Prepare`, `Waiting`, and `Fire` (see Figure 4) that are determined based on the predictions from GestureCNN. Appendix A details the transition logics between the various states of the gesture state machine. The speech state machine consists of two stages: `Idle` and `Fire` that are determined based on the predictions from SpeechCNN. When both gesture and speech state machines are in `Fire` state, the policy model triggers the IVA on the device.

Across both state machines, there are three free parameters; the threshold at which the GestureCNN makes a raising prediction, the threshold for raised stage detection; and a speech threshold for SpeechCNN. These three free parameters were tuned on validation data and were chosen to optimize the quality metrics (defined in Section 3) that directly impact the user experience.

4.4 False Trigger Mitigation Model

Once the IVA is triggered on-device, there is an opportunity to reduce accidental invocations by inspecting the speech and language content of the utterance for mitigation. During this stage, the off-device FTM (false trigger mitigator) can send back a mitigation signal to the device when an accidental invocation is detected. The FTM system is designed to be a language-agnostic model that takes as input features such as utterance duration, SNR, speech confidence scores, per phoneme duration, phoneme count, and other salient features.

5 EXPERIMENTS

One of the advantages of the RTS detector is that the four loosely coupled components (GestureCNN, SpeechCNN, Policy Model, and FTM) can mostly be trained independently. In this section, we outline the experiments that led to the final architecture of the RTS detector by going through each of the components individually.

5.1 Data Collection

During data collection, controlled user studies are conducted to collect data directly from the device. The data collected includes co-ordinated accelerometer data at 100Hz and audio data at 16KHz. Both positive (RTS attempts) and negative sessions are collected. The distributions of signal lengths of all collected sessions are shown in Figure 6. The data collection process is divided into three stages. The first stage (S1) consists of data collected in the most ideal environments. In this stage, the majority (53.2%) of sessions consist of users that are either sitting or standing (see Table 2). The second stage (S2) consists of users walking while performing RTS. The third stage includes users that are primarily running and cycling. In total we collected 4228 sessions from 92 different users.

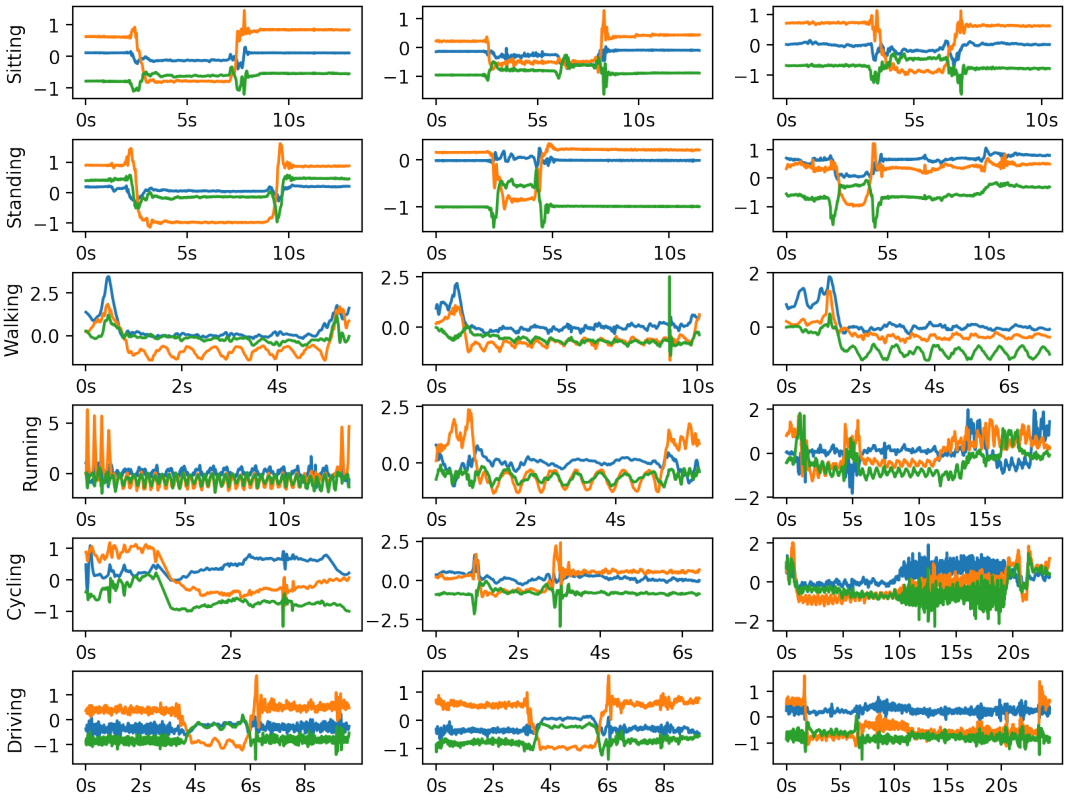

Figure 5: Visualization of three motion signals in each targeted scenario. Each row represents an activity scenario, and each column indicates a sample gesture. Blue: *acl_x*, Orange: *acl_y*, Green: *acl_z*.

All the accelerometer data from these sessions are used to train the gesture detection model. All data is collected internally at Apple on Apple employees. Figure 5 shows three gesture examples for each of the targeted scenarios.

In addition to the above activity scenarios, the data collected also include a wide range of acoustic environments, including quiet office, coffee shops, and even bars. The speech data from each of the 4228 sessions were split into 3 categories based ambient noise level (quiet, moderate, and loud were the three chosen segments). In S1, the majority of the data was in a quiet acoustic environment. In S2 and S3 more sessions in a moderate and loud acoustic environment are present. In addition, 1998 background sessions containing no speech as negative sessions are added in S3. The total number of sessions to train SpeechCNN is 6226. The distribution of data collected in these three acoustic environments is illustrated in Table 3.

In addition to controlled user studies, both positive and negative sessions for the 92 users during a course of three weeks are collected. These users wear the device naturally and are not part of any guided study. This data collection results in 1259 positive sessions and 21937 negative sessions. The positive sessions cover a wide range of activity scenarios and acoustic environments as shown in Table 2 and 3. The data collected in the live setting are used as the validation dataset.

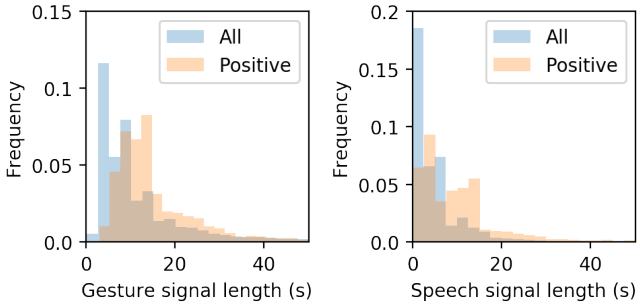

Figure 6: The distribution of signal lengths.

5.2 Gesture Detection

We trained the GestureCNN models respectively using data collected in the three stages. Those models are applied to the positive sessions in the validation data set. The accuracy and loss are reported in Table 4. The loss reported is cross entropy loss. We observe that with increase of training data, the model performance increases accordingly.

Table 2: Distributions of scenarios in positive sessions

Scenarios	Sitting	Standing	Walking	Running	Cycling	Driving	Unknown	Total
Training (S1)	372/37.4%	157/15.8%	133/13.4%	0	18/1.8%	43/4.3%	271/27.3%	994
Training (S2)	373/14.9%	173/6.9%	595/23.8%	111/4.4%	42/1.7%	92/3.7%	1112/44.5%	2498
Training (S3)	373/8.8%	173/4.1%	1167/27.6%	693/16.4%	152/3.6%	92/2.2%	1578/37.3%	4228
Validation	180/14.3%	145/11.5%	327/26.0%	142/11.3%	46/3.7%	22/1.7%	397/31.5%	1259

Table 3: Distributions of acoustic environment

Environment	Quiet	Moderate	Loud	Total
Training (S1)	440/44.3%	293/29.5%	261/26.2%	994
Training (S2)	968/38.8%	842/33.7%	688/27.5%	2498
Training (S3)	1463/23.5%	1679/27.0%	3084/49.5%	6226
Validation	422/33.5%	385/30.6%	452/35.9%	1259

We consider three different alternative baseline approaches of performing gesture classification: gradient boosting tree (GBT), dynamic time wrapping (DTW), and a hidden Markov model (HMM). Each of the models are trained on the entire training set (S3) and evaluated on the validation set. Table 4 compares the performance of the various approaches to solve the gesture detection problem. Figure 7 provides a visual comparison of predictions from each of these models on a sample session in the validation set.

Gradient Boosting Tree. A GBT model is trained (using XGboost [3]) to classify the gesture classes using the same 31 features used to train GestureCNN. During training, all the parameters are set to default except the number of trees, which is set to 50, 100, 500, 1000.

Dynamic Time Warping. DTW has been widely used for gesture detection [1, 13]. DTW uses a dynamic programing algorithm to align two timeseries and then calculates a distance measurement given to the alignment. The computational complexity is quadratic due to the alignment step. The high computational complexity of DTW makes it unsuitable to run on an embedded system using the specific timeseries data relevant to this problem. To reduce computational complexity, an approximation algorithm by calculating a lower bound of the optimal distance in linear time is chosen [11].

A template library consisting of raw acceleration timeseries for each gesture class is constructed. All the sessions in the training data are used to construct the library. The number of templates in the library are 4226, 4218, 2687, 6858 for raising, raised, dropping and dropped classes respectively. During the validation and test stages, the new timeseries is first divided into chunks of 50 with overlap of 40. A k-nearest-neighbor (KNN) classifier compared each chunk to the gestures in the library. The KNN algorithm is implemented using a priority heap [11]. The number of neighbors in the KNN is set to 10, 50, 100 and 500.

To further reduce computational complexity, a clustering method named affinity propagation [5] is used to find representative gestures in each gesture class. These gestures are used to build gesture templates. We experimented with different levels of clustering strengths, but observed a significant drop in classification accuracy. Therefore, during evaluation all gestures are used to build the dictionary.

Table 4: Model performance

Modality	Model	Accuracy	Loss	Latency	Size
Gesture	DTW_KNN10	0.7422	3.4538	2.52s	101M
	DTW_KNN50	0.7806	1.9390	2.60s	101M
	DTW_KNN100	0.7878	1.4752	2.65s	101M
	DTW_KNN500	0.7855	0.9419	2.65s	101M
	HMM_k3	0.6660	15.112	11.5ms	5K
	HMM_k5	0.6614	14.281	13.0ms	6K
	HMM_k10	0.6651	14.036	19.4ms	9K
	GBT50	0.7559	0.6312	0.66ms	51K
	GBT100	0.7776	0.5859	0.73ms	103K
	GBT500	0.8068	0.5243	1.31ms	523kK
	GBT1000	0.8135	0.5103	2.43ms	1.05M
	GestureCNN(S1)	0.7033	0.9953	7.86ms	504K
	GestureCNN(S2)	0.7328	0.9410	7.86ms	504K
	GestureCNN(S3)	**0.8722**	**0.4274**	7.86ms	504K
Speech	GBT50	0.8836	0.3198	0.65ms	13K
	GBT100	0.8895	0.3098	0.65ms	26K
	GBT500	0.8979	0.2911	0.79ms	128K
	GBT1000	0.9007	0.2837	0.91ms	257K
	SpeechCNN(S1)	0.9183	0.4461	8.40ms	497K
	SpeechCNN(S2)	0.9426	0.1823	8.40ms	497K
	SpeechCNN(S3)	**0.9412**	**0.1555**	8.40ms	497K

Hidden Markov Model. Four diagonal Gaussian HMM models, one for each gesture class, using the raw acceleration signals are examined. We set the number of latent states to different numbers (3, 5, 10) to evaluate overall performance. During the validation and test stages, similar to DTW, the new time series is divided into chunks of 50 with overlap of 40. The likelihood of the chunk given each of the trained Gaussian HMM model is calculated. The likelihood is then converted to class posterior probability using Bayes rule.

Each of the models in the above three approaches are trained using all training data. The comparison of the GestureCNN with the three baseline models is shown in Table 4. GestureCNN performs best among the methods. In addition to model accuracy, the prediction latency (averaged over 1000 runs) and disk-footprint are compared.

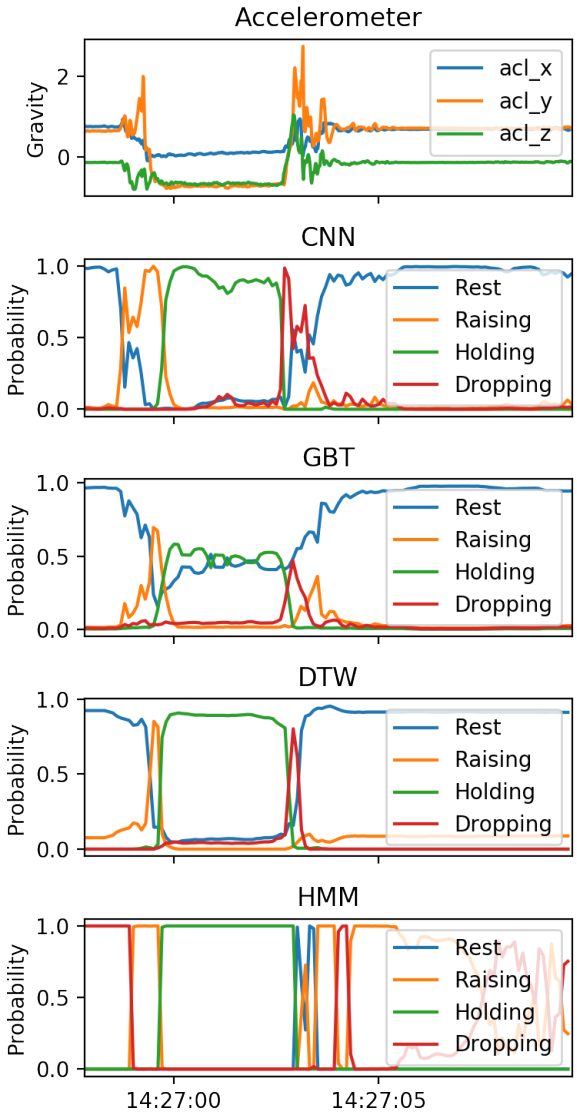

Figure 7: Visualization of predictions from different gesture models.

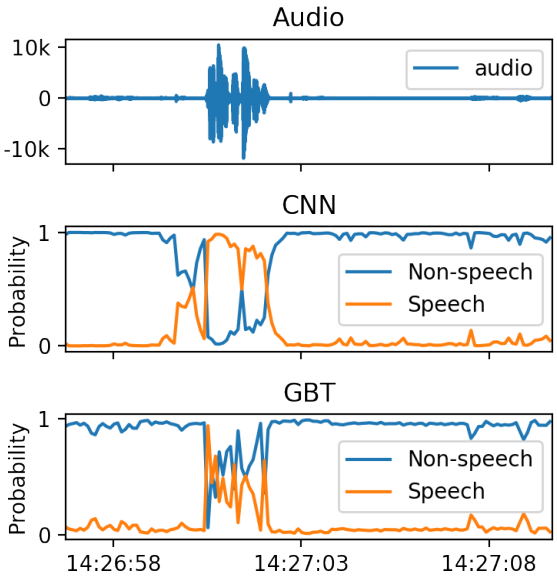

Figure 8: Visualization of predictions from different speech models.

5.3 Speech Detection

The SpeechCNN model was trained using the data collected in the three stages respectively. The resulting models were applied to the positive sessions in the validation data set. Table 4 shows the performance of the three models. We observed that adding more data, especially data collected in loud environment, significantly improves the performance of the model in detecting proximal-speech.

We compare SpeechCNN with a baseline model trained using GBT using all data (S3). Similar to the gesture GBT model, the speech GBT model is trained using XGboost. The number of trees is set to 50, 100, 500, 1000. The comparison of SpeechCNN with GBT is shown in Table 4. The SpeechCNN outperforms the GBT models by a large margin. Figure 8 visually compares predictions from the two models.

In addition to classifying speech vs. non-speech for this application, it also becomes necessary to correctly identify the start of speech. The system must accurately establish the beginning point of the user's utterance so that it can start speech recognition and language understanding from that time point. In button presses and trigger phrases, the invocation method itself can serve as the correct starting point. For RTS, it is necessary to estimate the start of the speech query to ensure part of the query is not clipped in later processing. This estimation is performed by understanding the relationship between labeled speech starting points and the time at which the SpeechCNN model triggers RTS. Given that time span, an estimate of the appropriate time in which the system must look back from when the SpeechCNN detects speech is determined. This time is determined to be approximately 0.75 seconds according to our experiments. That is, the system will start the IVA request 0.75 seconds in the past with respect to the speech detection timepoint.

5.4 Policy Evaluation

In this section, various combinations of speech and gesture models are evaluated using the user metrics (FRR and FAWU). The values of the three free parameters in the policy model are varied on a [0.5, 1.0) grid with step size of 0.05. The resulting FAWU and FRR metrics provide an operation curve for the model combination.

Figure 9A shows the resulting metrics of using GestureCNN and SpeechCNN trained in different data collection stages. A clear trend that adding more data improves the quality of the resulting models can be observed. Additionally, as would be expected, the policy models using individual modality perform worse than the one using both gesture and speech (Figure 9B). Figure 9C shows the results

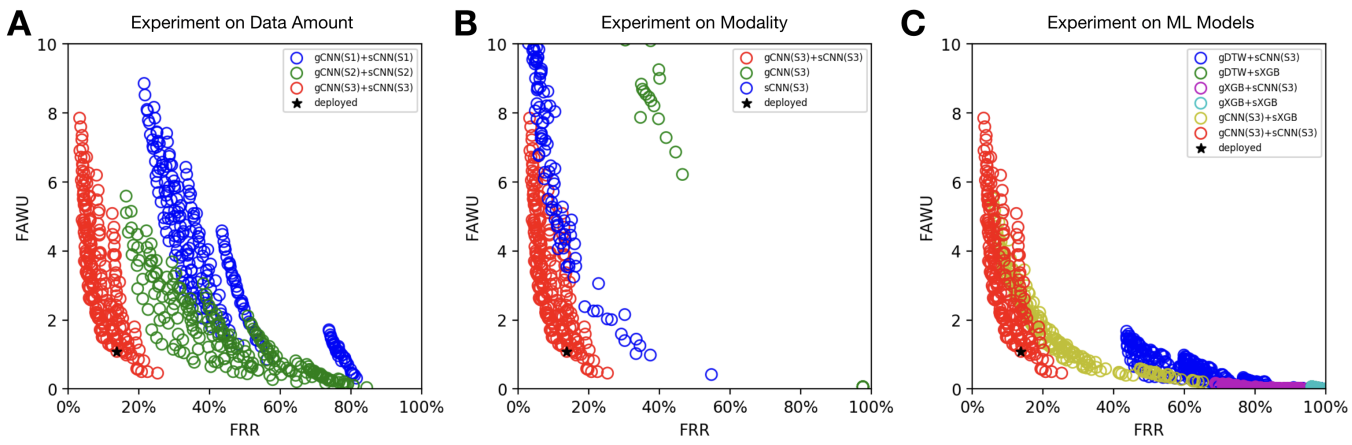

Figure 9: Evaluation of different gesture and speech models. A: Evaluations of GestureCNN and SpeechCNN models trained during the three data collection stages (S1, S2, S3). B: Evaluations using only one modality trained with S3 data. C: The comparisons of different gesture and speech model combinations. "GBT" in the legend represents the GBT model trained using 1000 trees. "DTW" in the legend represents the DTW model with KNN500. "g" stands for gesture. "s" stands for speech.

of using different baseline models compared with GestureCNN and SpeechCNN. The GestureCNN and SpeechCNN pair has the lowest FRR and FAWU. From the operation curve we set the raising threshold to 0.8, holding threshold to 0.9 and speech threshold to 0.95 as the policy model parameters in the deployed model. These parameters gives a 1.09 FAWU at a FRR of 13.8% on the validation data set (The black star in Figure 9). Table 5 shows the FRR values under each of the supported scenario.

5.5 FTM Model

The data collected in subsection 5.1 is used to train the on-device models. To train the FTM models, audio data (total 23064 audio clips) is collected from Apple employees (from 2130 users). The dataset is split into 8052 audio sessions for training and 15012 audio sessions for testing. The audio samples are also annotated with labels (intended for IVA or not). The FTM model is trained using XGboost and cross-validated on a held-out validation set. On the test set, the FTM model brought down the FAWU from 1.09 to 0.15 for the three weeks with an increase of FRR to 16.0%.

6 CONCLUSION

In this paper we present an accurate and low power detector to facilitate interacting with IVAs on smartwatches. A four component detector is presented consisting of an on-device gesture detector (GestureCNN), an on-device speech detector (SpeechCNN), a policy model, and an off-device false trigger mitigator (FTM). In Section 5.2 we evaluate three other baseline approaches for gesture detection in addition to GestureCNN. Experimentation shows that the closest match to the GestureCNN in model performance are the GBT models. However, adding more trees showed diminishing returns when a large number of trees are reached (Table 4). Other common approaches are either too computationally expensive for an embedded system (e.g DTW) or aren't complex enough to capture the non-linearity in the data. For speech detector, SpeechCNN outperforms the GBT model by a large margin (See Section 5.3).

The system was launched to millions of devices with the public release of Apple's WatchOS 5.0. It achieves negligible impact on battery life and system latency.

7 APPENDIX

A DETAILS OF GESTURE STATE MACHINE

The gesture state machine contained four states: `Idle`, `Prepare`, `Waiting` and `Fire`. The transition logics are shown below.

(1) The state machine is initialized with `Idle` state
(2) In `Idle` state, the probability of raising class is monitored. When this probability surpasses 0.8, transition to `Prepare` state. Otherwise stay in `Idle`
(3) In `Prepare` state, the probabilities of both raising and holding are monitored.
   - If the probability of holding is greater than 0.9, transition to `Fire` state
   - Else if the probability of raising is greater than 0.8, stay in `Prepare` state
   - Else, transition to `Waiting` state
(4) In `Waiting` state, a timer of 1.2s is set. All the prediction probilities are monitored in this state
   - If the probability of holding is greater than 0.9, transition to `Fire` state
   - Else if the probility of raising is greater than 0.8, transition to `Prepare` state
   - Else if the sum of the probilities of dropping and rest is greater than 0.3, transition to `Idle` state
   - Else if the timer is not end, stay in `Waiting`
   - Else transition to `Idle` state

Table 5: FRR measured under different scenarios in validation data set

Scenarios	Sitting	Standing	Walking	Running	Cycling	Driving	Unknown	Total
FRR	7.2%	4.8%	12.2%	17.6%	39.1%	13.6%	17.1%	13.8%

(5) In `Fire` state, the prediction of holding is monitered. If the holding prediction is above 0.9, stay in `Fire` state. Otherwise transition to `Idle`

The three parameters in the policy model were determined with an exhaustive grid search on a validation data set based on our metrics as described in Section 3.

ACKNOWLEDGMENTS

The authors would like to thank Jeffrey Bigham and Nick Foti for helpful suggestions and comments. The authors also would like thank Karla Vega for coordinating and proofreading the paper.

REFERENCES

[1] Ahmad Akl and Shahrokh Valaee. 2010. Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2270–2273.

[2] Lorna M. Brown and John Williamson. 2007. Shake2Talk: multimodal messaging for interpersonal communication. In *International Workshop on Haptic and Audio Interaction Design*. Springer, 44–55.

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[4] Florian Eyben, Felix Weninger, Stefano Squartini, and Björn Schuller. 2013. Real-life voice activity detection with lstm recurrent neural networks and an application to hollywood movies. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 483–487.

[5] Brendan J. Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *science* 315, 5814 (2007), 972–976.

[6] Eleftheria Georganti, Tobias May, Steven van de Par, Aki Harma, and John Mourjopoulos. 2011. Speaker distance detection using a single microphone. *IEEE Transactions on Audio, Speech, and Language Processing* 19, 7 (2011), 1949–1961.

[7] Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter Jin, Sicheng Zhao, and Kurt Keutzer. 2018. SqueezeNext: Hardware-Aware Neural Network Design. *arXiv preprint arXiv:1803.10615* (2018).

[8] Thad Hughes and Keir Mierle. 2013. Recurrent neural networks for voice activity detection. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 7378–7382.

[9] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360* (2016).

[10] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*. 448–456. http://proceedings.mlr.press/v37/ioffe15.html

[11] Eamonn Keogh and Chotirat Ann Ratanamahatana. 2005. Exact indexing of dynamic time warping. *Knowledge and information systems* 7, 3 (2005), 358–386.

[12] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. (Dec. 2014). https://arxiv.org/abs/1412.6980

[13] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2009. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing* 5, 6 (2009), 657–675.

[14] Ewa Luger and Abigail Sellen. 2016. Like having a really bad PA: the gulf between user expectation and experience of conversational agents. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 5286–5297.

[15] Michael McTear, Zoraida Callejas, and David Griol. 2016. *The Conversational Interface: Talking to Smart Devices*. Springer. Google-Books-ID: X_w0DAAAQBAJ.

[16] Francis Quek, David McNeill, Robert Bryll, Susan Duncan, Xin-Feng Ma, Cemil Kirbas, Karl E. McCullough, and Rashid Ansari. 2002. Multimodal human discourse: gesture and speech. *ACM Transactions on Computer-Human Interaction (TOCHI)* 9, 3 (2002), 171–193.

[17] Javier Ramirez, Juan Manuel Górriz, and José Carlos Segura. 2007. Voice activity detection. fundamentals and speech recognition system robustness. In *Robust speech recognition and understanding*. InTech.

[18] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[19] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[20] Matthew Turk. 2014. Multimodal interaction: A review. *Pattern Recognition Letters* 36 (2014), 189–195.

[21] Xiao-Lei Zhang and Ji Wu. 2013. Deep belief networks based voice activity detection. *IEEE Transactions on Audio, Speech, and Language Processing* 21, 4 (2013), 697–710.