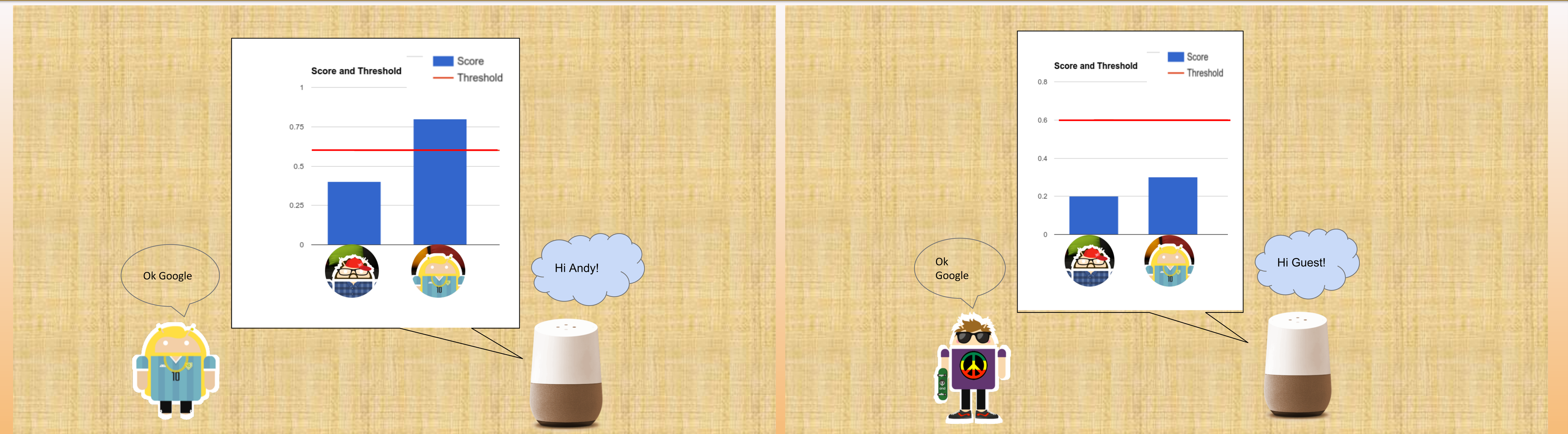
ATTENTION-BASED MODELS FOR TEXT-DEPENDENT SPEAKER VERIFICATION

F A Rezaur Rahman Chowdhury, Quan Wang, Ignacio Lopez Moreno, Li Wan*
Washington State University, Google Inc., USA*
fchowdhu@eecs.wsu.edu {quanw, elnota, liwan}@google.com

Speaker Identification

Problem and Application



➡Speaker recognition can be used to provide secured personalized interactions to systems controlled by voice.
➡Global-password text dependent speaker recognition aims to distinguish among speakers using fixed phrases like **"Ok Google"** or **"Hey Google"**.
➡Since 2014 end2end neural network architectures for speaker recognition have shown to outperform traditional approaches. [Variani, Ehsan, et al. "Deep neural networks for small footprint text-dependent speaker verification." Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on. IEEE, 2014.]
➡This work is a study over alternative attention mechanisms to further improve the standard end2end architectures for text-dependent speaker recognition.

End to End Architecture

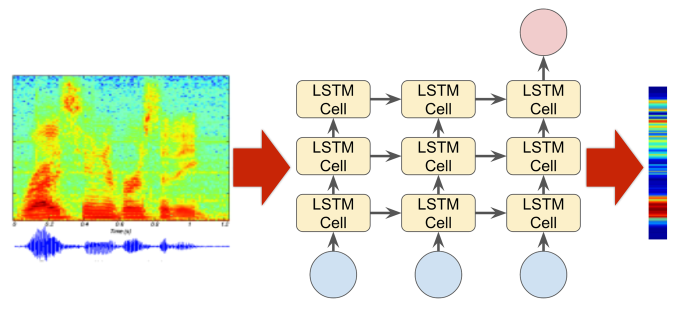


Fig. We use LSTM as audio feature extractor.

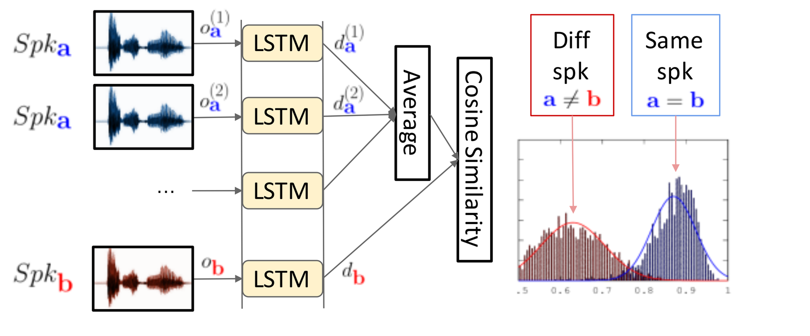


Fig. Tuple E2E loss: Speaker verification as a binary classification problem.

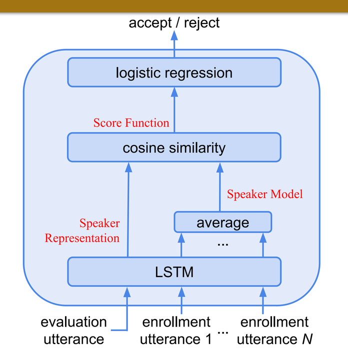


Fig. Our baseline end-to-end training architecture

➡For each **training** step, a tuple of **one evaluation utterance** and **N enrollment utterances** is fed into our LSTM network
➡Features are extracted using **log-mel-filterbank** energies from a fixed-length segment
➡We use LSTM model to calculate the **d-vector**. We average the d-vectors of the enrollment utterances
➡The similarity of the utterances are defined using the **cosine similarity** function of their d-vectors

Baseline LSTM Model

- We use **3-Layer LSTM** in our baseline LSTM model
- Dimension of each layer - **128**
- **Projection layer** in each layer with dimension - **64**
- On top of the LSTM layers, a **linear layer** of dimension - **64**
- The acoustic parametrization consists of **40-dimensional** log-mel-filterbank coefficients computed over a window of 25ms with 15ms of overlap

- **Problems -**
  - **Silence** and **background noise** are NOT being well captured in this system
  - The phonemes are usually surrounded by **frames of silence and background noise**. Ideally, the speaker embedding should be built only using the frames corresponding to phonemes.

- Thus, we propose to use an **attention layer** as a soft mechanism to **emphasize** the most relevant elements of the input sequence.

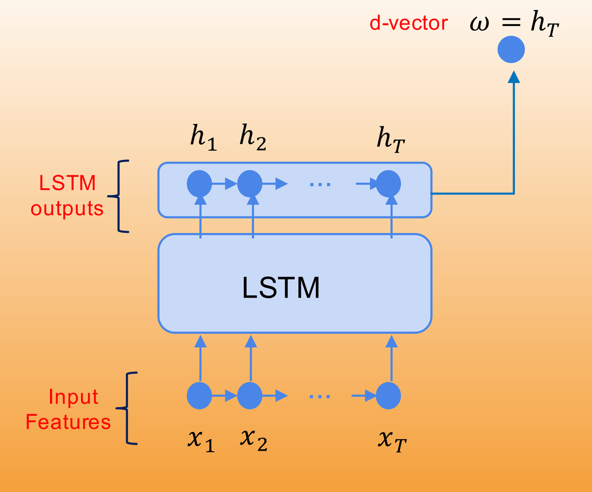


Fig. LSTM-based d-vector baseline

Attention-based models on LSTM

Basic Idea

➡**Motivation**: A **weighted combination** of all hidden layer outputs to learn the **more important** parts of the input
➡In our basic attention model computation of the weights is done on the **final hidden layer outputs**
➡We use a **scoring function** $e_t = f(h_t)$ to compute weights based on the hidden status
➡Then we get the **normalized weights** using $\alpha_t = \frac{\exp(e_t)}{\sum_j \exp(e_j)}$
➡Finally we compute a **weighted combination** of the weights as,

$$\omega = \sum_t \alpha_t h_t \quad \text{where,} \quad \sum_t \alpha_t = 1$$

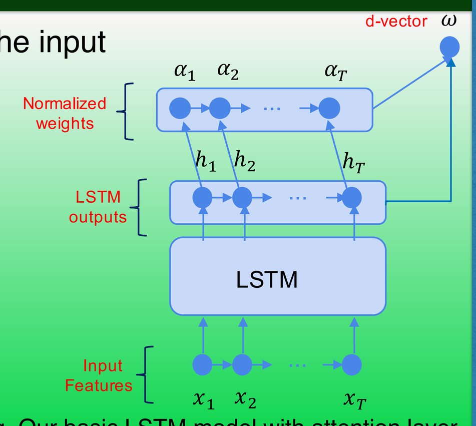


Fig. Our basic LSTM model with attention layer

Different Scoring Functions

➡We experimented using **different types of scoring functions** for computation of weights of the attention layer.
➡**Bias only attention** - It does NOT depend on the LSTM output and is scalar
➡**Linear and non-linear attention** - We call attention **linear** and **non-linear** based on the function used to calculate the attention
➡**Shared-parameters attention** - We experimented using **shared parameters** through **all time steps** for linear and non linear attentions

Bias only	Linear	Shared-parameter linear	Non-linear	Shared-parameter non-linear
$e_t = f_{BO}(h_t) = b_t$	$e_t = f_L(h_t) = w_t^T h_t + b_t$	$e_t = f_{SL}(h_t) = w^T h_t + b$	$e_t = f_{NL}(h_t) = v_t^T \tanh(W_t h_t + b_t)$	$e_t = f_{SNL}(h_t) = v^T \tanh(W h_t + b)$

Attention Layer Variants

➡We introduce two variants of the attention layer - **cross-layer attention** and **divided-layer attention**

Cross-layer attention

➡**Motivation** - Using **same** layer for weight computation and d-vector computation is NOT very **informative**
➡We calculate weights from an **intermediate** LSTM layer
➡We change our scoring function to $e_t = f(h'_t)$ where, h'_t is an intermediate LSTM layer (e.g. **second-to-last layer**)
➡We calculate the d-vector from the weighted average of the **last layer**, h_t as before

$$\omega = \sum_t \alpha_t h_t$$

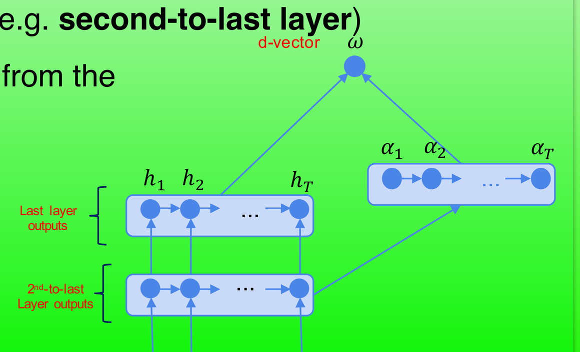


Fig. LSTM model with cross-layer attention

Divided-layer attention

➡**Motivation** - Using **independent** layers for weight computation and d-vector computation
➡We double the size of the **final** LSTM layer and then **divide** it into two equal sized part-a h_t^a and part-b h_t^b
➡We compute the weights from **part-b** by the scoring function $e_t = f(h_t^b)$
➡We calculate the d-vector from the weighted average from **part-a**,

$$\omega = \sum_t \alpha_t h_t^a$$

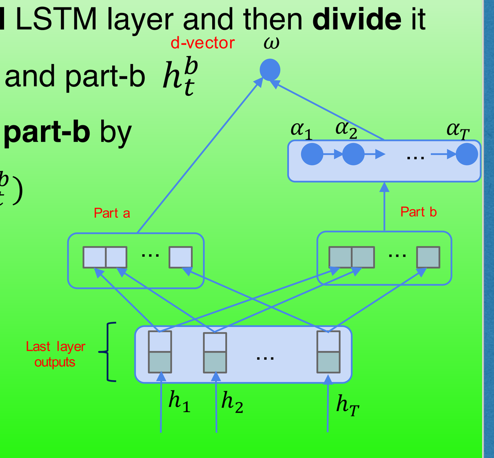


Fig. LSTM model with divided-layer attention

Weights Pooling Idea and Variants

➡**Motivation** - To **make the weights sparse.** Sparse weights can focus more on the most important parts with **temporal variation** in speech
➡We used two variants for the weight pooling idea in our design
➡**Sliding window maxpooling** - We run a sliding window on the weights.
  ➡For each window, only keep the **largest value** and set others to 0.
➡**Global top-K maxpooling** - We only keep the **largest K values** in the weights, and set all other values to 0.

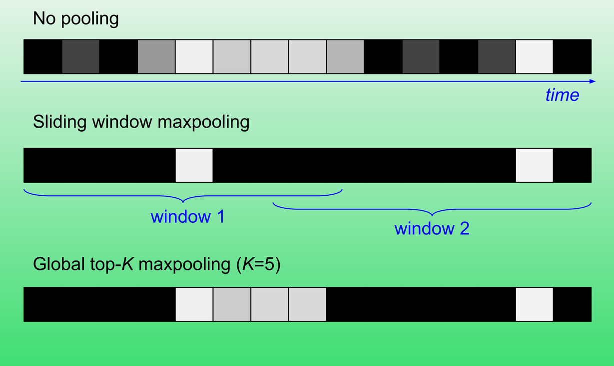


Fig. Different pooling methods on attention weights. The tth pixel corresponds to the weight α_t and a brighter intensity means a larger value of the weight

Experiments

Datasets and Evaluation

Training set
- Anonymized voice queries
- **150M** utterances, **630K** speakers
- Mixture of "**OK Google**" and "**Hey Google**"

Testing set
- Manual collection of **665** speakers
- For each "OK Google" and "Hey Google" - **Two** enrollment sets, **Two** verification sets
- Enrollment set: **~4.5** utterances per speaker; Verification set: **~10** utterances per speaker.

Evaluation
- We report the speaker verification **Equal Error Rate (EER)** on the four combinations of enrollment set and verification set.

Results

Test data Enroll → Verify	Non-attention Baseline	Basic Attention f_{BO}	f_L	f_{SL}	f_{NL}	f_{SNL}
OK Google → OK Google	0.88	0.85	0.81	0.8	0.79	0.78
OK Google → Hey Google	2.77	2.97	2.74	2.75	2.69	2.66
Hey Google → OK Google	2.19	2.3	2.28	2.23	2.14	2.08
Hey Google → Hey Google	1.05	1.04	1.03	1.03	1.00	1.01
Average	1.72	1.79	1.72	1.70	1.66	1.63

Table. Evaluation EER(%): Non-attention baseline model vs. basic attention layer using different scoring functions.

Test data	Basic f_{SNL}	Cross-layer	Divided-layer
OK → OK	0.78	0.81	0.75
OK → Hey	2.66	2.61	2.44
Hey → OK	2.08	2.03	2.07
Hey → Hey	1.01	0.97	0.99
Average	1.63	1.61	1.56

Table.Evaluation EER(%): Basic attention layer vs. variants - all using f_{SNL} as scoring function

Test data	No-pooling	Sliding window	Top-K
OK → OK	0.75	0.72	0.72
OK → Hey	2.44	2.37	2.63
Hey → OK	2.07	1.88	1.99
Hey → Hey	0.99	0.95	0.94
Average	1.56	1.48	1.57

Table. Evaluation EER(%): Different pooling methods for attention weights - all using f_{SNL} and divided-layer

➡First, we compare the baseline model with basic attention layer using **different scoring functions**
  ➡Performance wise **non-linear with shared parameter is better than others.**
➡We compare **basic attention** with **cross-layer** and **divided-layer** attentions fixing the best scoring function from previous experiment
  ➡Performance wise **divided-layer is better than other two variants**
➡We compare **different pooling strategies** fixing the best setting from previous experiment
  ➡Performance wise **sliding window maxpooling better than other two variants**

Attention Visualization

- We **visualize** the **attention weights** of a training batch for different pooling methods.
- **Interesting observation** - when there's no pooling, we see a clear **4-strand** or **3-strand** pattern in the batch. This pattern corresponds to the "**O-kay-Goo-gle**" 4-phoneme or "**Hey-Goo-gle**" 3-phoneme structure of the keywords
- When we apply **sliding window maxpooling** or **global top-K maxpooling**, the attention weights are much larger at the **near-end** of the utterance, The LSTM has accumulated more information at the **near-end** than at the beginning, thus is **more confident** to produce the d-vector

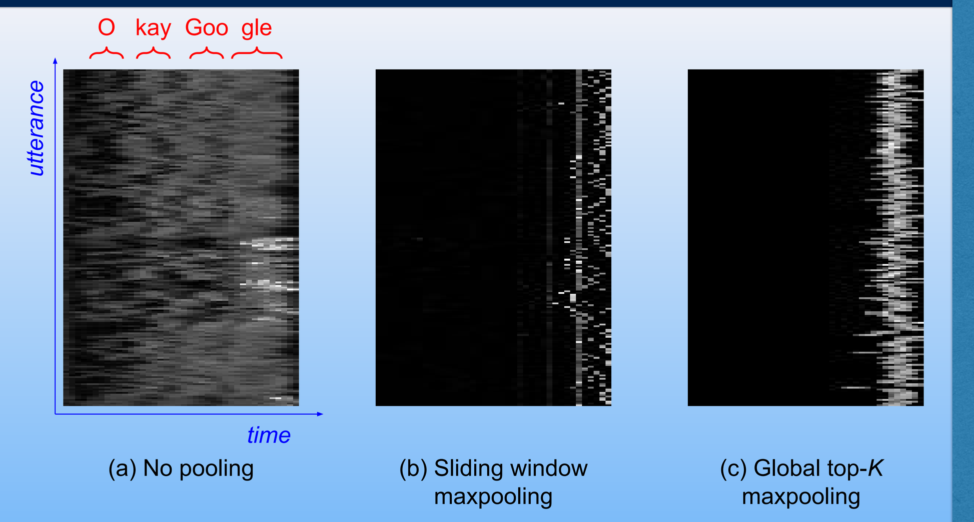


Fig. Visualized attention weights for different pooling methods. In each image, x-axis is **time**, and y-axis is attention weights (**brighter intensity is larger weight**) for different utterances in a training batch. (a) No pooling; (b) Sliding window maxpooling, where window size is 10, and step is 5; (c) Global top-K maxpooling, where K = 5.

ICASSP 2018
The 43rd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), April 15 - 20, 2018, Calgary, Alberta, Canada
IEEE Signal Processing Society
IEEE