ICASSP 2022

Ruihai Hou

2022/07/08

DEEP PIECEWISE HASHING FOR EFFICIENT HAMMING SPACE RETRIEVAL

* Chinese Academy of Sciences

- Motivation
  - consider both discrimination and robustness for the dissimilar points inside the Hamming ball
- Datasets
  - MS-COCO
  - NUS-WIDE
- Methods
  - Deep Piecewise Hashing
  - Piecewise loss
- Experiment
  - evaluate the retrieval performance of DPH with eight state-of-the-art methods

DEEP PIECEWISE HASHING FOR EFFICIENT HAMMING SPACE RETRIEVAL

* Chinese Academy of Sciences

- Methods
  - consists of two parts, including feature extractor and piecewise
  - use AlexNet as the base network to obtain representative features of images, and use the tanh function to transform the feature representation into continuous code.
  - piecewise loss solves the problem of data misclassification around Hamming ball, but also keeps the model's discrimination and robustness

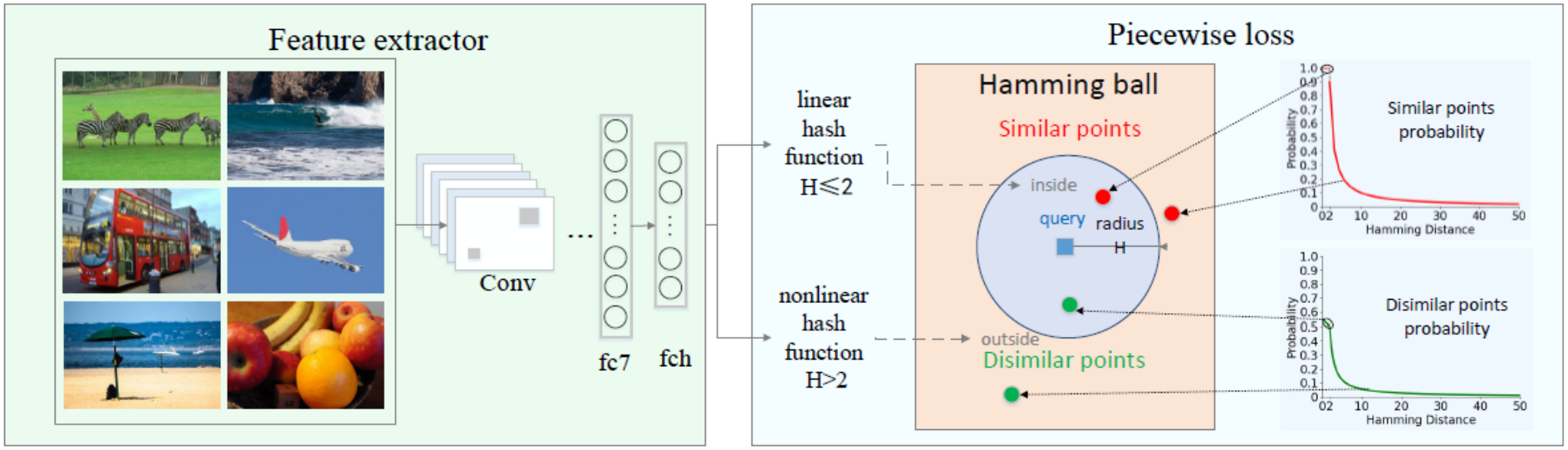


Fig. 2. The framework of DPH. The entire framework consists of two parts: feature extraction and piecewise loss.

* Chinese Academy of Sciences

- Methods
  - The piecewise loss function is designedas follows:

$$L = \begin{cases} \sum_{s_{ij} \in S} w_{ij}((1 - s_{ij}) \log \frac{1}{1-(kD_H(i,j)+b)} \\ +s_{ij} \log 1) \ , \ H \leq 2 \\ \\ \sum_{s_{ij} \in S} w_{ij}((1 - s_{ij}) \log \frac{r+(D_H(i,j)-H)}{((1-\alpha_2)r+(D_H(i,j)-H))} \\ +s_{ij} \log \frac{r+(D_H(i,j)-H)}{\alpha_1 r}) \ , \ H>2 \end{cases} . \tag{5}$$

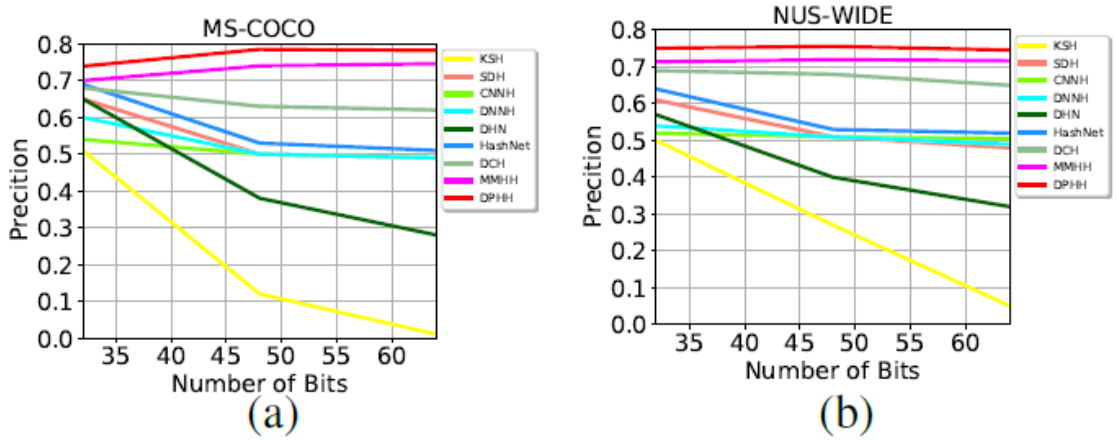
DEEP PIECEWISE HASHING FOR EFFICIENT HAMMING SPACE RETRIEVAL

* Chinese Academy of Sciences

- Experiments

Table 1. The **MAP@H $\leq$ 2** for different bits of DPH and baselines on MS-COCO and NUS-WIDE image datasets.

Method	MS-COCO			NUS-WIDE		
	32	48	64	32	48	64
KSH	0.513	0.244	0.025	0.563	0.422	0.063
SDH	0.663	0.526	0.513	0.671	0.592	0.453
CNNH	0.562	0.532	0.510	0.590	0.576	0.574
DNNH	0.609	0.522	0.510	0.624	0.592	0.563
DHN	0.663	0.513	0.421	0.710	0.674	0.564
HashNet	0.689	0.562	0.536	0.724	0.680	0.612
DCH	0.755	0.729	0.709	0.776	0.758	0.713
MMHH	0.765	0.797	0.809	0.793	0.801	0.799
DPH	**0.796**	**0.841**	**0.842**	**0.817**	**0.829**	**0.817**



DEEP HASHING WITH HASH CENTER UPDATE FOR EFFICIENT IMAGE RETRIEVAL

* RWTH Aachen University

- Motivation
  - Canonical Correlation Analysis (CCA) was used such that the correlation between the feature vectors and label vectors was maximized
- Datasets
  - MS-COCO
  - NUS-WIDE
- Methods
  - DeepCentral Similarity Hashing (DCSH) method
  - A novel weighted mean and thresholding-based hash center update scheme is proposed.
  - CCA-based loss formulation
- Experiment
  - Two multi-labeled datasets, were used for evaluating the final retrieval performance of DCSH.

DEEP HASHING WITH HASH CENTER UPDATE FOR EFFICIENT IMAGE RETRIEVAL

* RWTH Aachen University

- Methods
  - The residual network is used as the basic feature extractor which was pre-trained on ImageNet.
  - hashing layer is used to generate hashes
  - intermediate layer is used to generate high output dimensionality
  - a linear combination of the two losses as, $L_{DCSH} = L_{hash} + \alpha L_{class}$

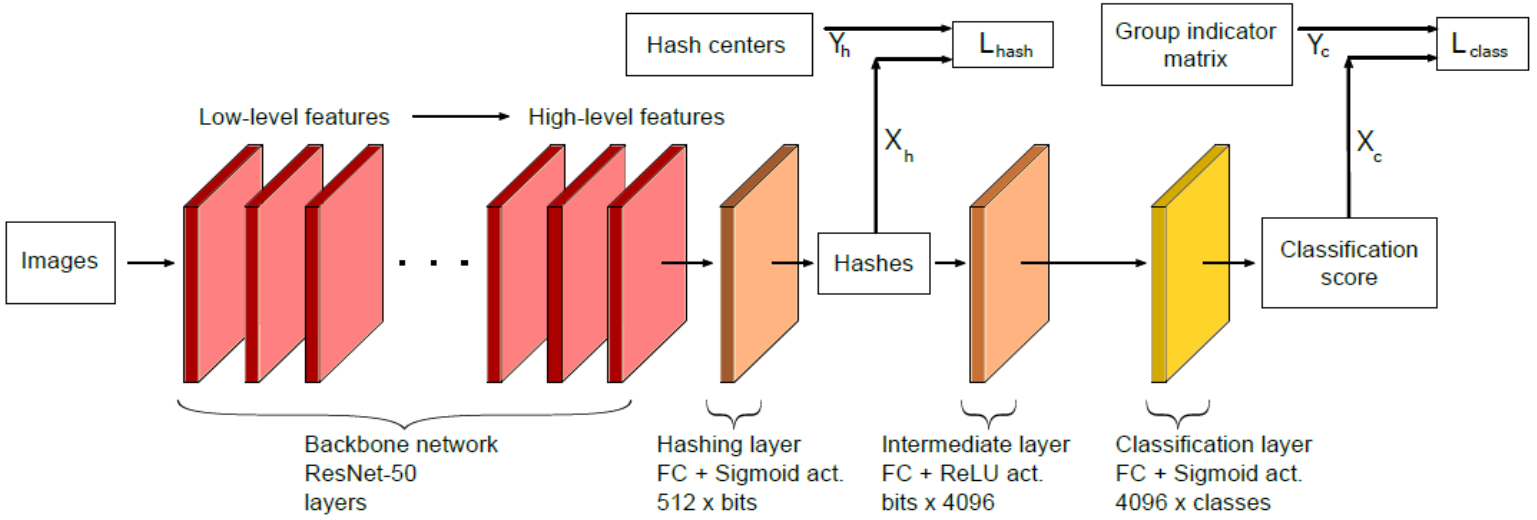


Fig. 1: Overview of the proposed Deep Central Similarity Hashing network architecture. ResNet layers according to [31] is used as the backbone network for basic feature extraction. Both hashing and classification layer, consist of a fully connected (FC) layer with subsequent sigmoid activation. The intermediate layer comprises a fully connected layer with subsequent ReLU activation. Bits indicate the bit length of the hash code. Classes indicate the number of categories in the dataset.

DEEP HASHING WITH HASH CENTER UPDATE FOR EFFICIENT IMAGE RETRIEVAL

* RWTH Aachen University

- Experiments

MS-COCO					NUS-WIDE				
Method	16 bits	32 bits	48 bits	64 bits	Method	12 bits	24 bits	32 bits	48 bits
DCSH (ours)	**0.805**	**0.847**	**0.859**	**0.861**	**DCSH (Ours)**	**0.823**	**0.833**	**0.841**	**0.857**
CSQ [21]	0.796	0.838	-	**0.861**	DPSH [15]	0.794	0.822	0.838	0.851
DCCH [19]	0.659	0.729	0.731	0.739	DCCH [19]	0.782	0.814	0.825	0.834
HashNet [3]	0.687	0.718	0.730	0.736	CSQ [21]	-	-	0.825	-
DHN [28]	0.677	0.701	0.695	0.694	DSDH [26]	0.776	0.808	0.820	0.829
DNNH [29]	0.593	0.603	0.604	0.610	DDSH [17]	0.791	0.815	0.821	0.827
CNNH [30]	0.564	0.574	0.571	0.567	DTSH [27]	0.773	0.808	0.812	0.814

Table 1: MAP on MS-COCO and NUS-WIDE for different approaches.

GRAPH CONVOLUTIONAL NETWORK BASED SEMI-SUPERVISED LEARNING ON MULTI-SPEAKER MEETING DATA

* Xiamen University

- Motivation
  - graph convolutional networks(GCN) exploit the connectivity patterns between nodes to improve learning performance
- Datasets
  - Voxceleb1
  - Voxceleb2
- Methods
  - present a GCNbased approach for semi-supervised learning
  - present a self-correcting training mechanism
- Experiment
  - After a network is trained, the clustering algorithms are evaluated in a simulated meeting scenario.
  - run ten tests on each group
  - evaluate the performance in terms of the average precision, recall,and F-score

* Xiamen University

- Methods
  - **Affinity Graph Construction**: based on the embeddings, each sample is regarded as a vertex and the cosine similarity is used to find K nearest neighbors for each sample
  - `Cluster Proposal Generation:` setting various thresholds on the edge weights of this graph, a set of super-vertices is generated, and then a higher level graph based on a super-vertex is constructed
  - **Cluster Detection:** A graph convolutional network is used to extract features for each proposal, and high-quality clusters are selected from the generated cluster proposals.

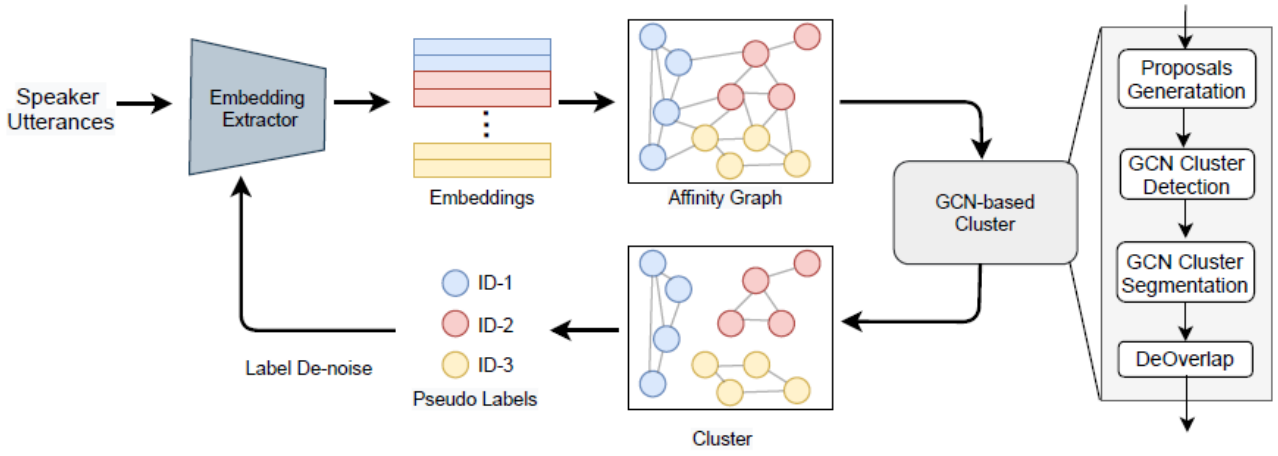


Fig. 1: The pipeline of our structure. Utterances are first fed into feature extractors to obtain speaker embeddings. An affinity graph is constructed to perform clustering. The cluster results with pseudo-labels are applied to re-training the deep embedding extractor.

* Xiamen University

- Methods
  - **Cluster Segmentation:** Another similar GCN is developed to exclude outliers from the proposal, in the model predictions, the outliers are removed from the proposals.
  - `De-Overlapping:` de-overlap procedure uses the predicted GCN scores and sorts them in descending order. The highest GCN score is selected for the proposals to partition the unlabeled dataset into a proper cluster.

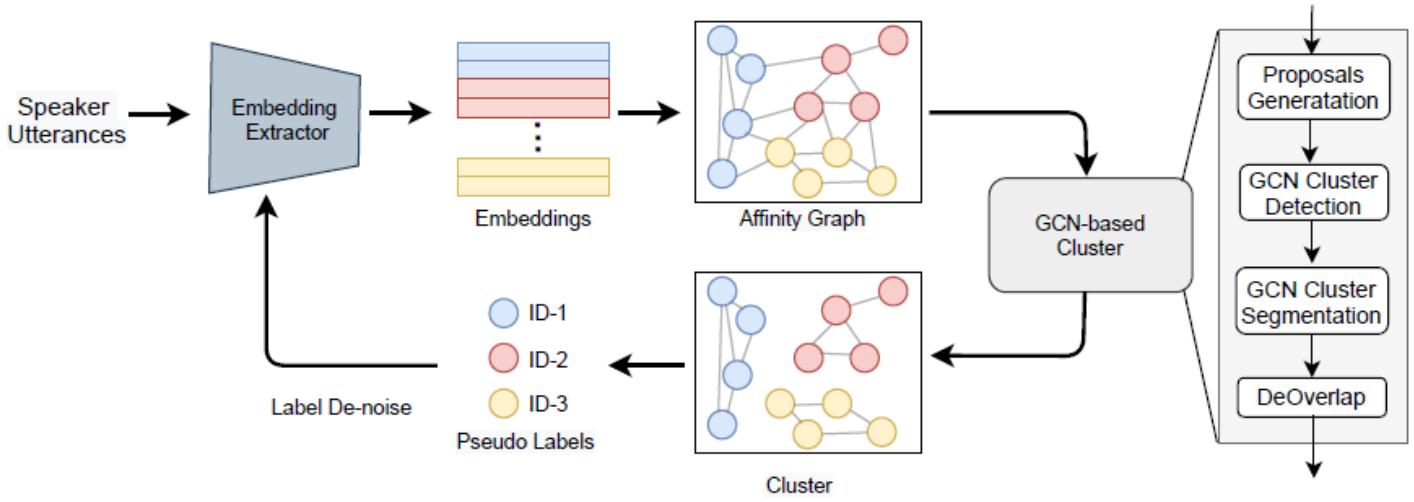


Fig. 1: The pipeline of our structure. Utterances are first fed into feature extractors to obtain speaker embeddings. An affinity graph is constructed to perform clustering. The cluster results with pseudo-labels are applied to re-training the deep embedding extractor.

* Xiamen University

- Experiments

Table 2: Comparison of speaker clustering when the number of clusters is 3, 6, and 9. The results are the average of the clustering results on 10 different sets of testing data.

#	Methods	Precision	Recall	F-score
3	K-means	0.80	0.52	0.63
	SC	0.76	0.68	0.71
	AHC	0.75	0.77	0.75
	GCN	0.82	0.79	0.80
6	K-means	0.78	0.56	0.65
	SC	0.71	0.65	0.67
	AHC	0.77	0.79	0.78
	GCN	0.84	0.78	0.81
9	K-means	0.77	0.53	0.63
	SC	0.73	0.66	0.69
	AHC	0.82	0.76	0.78
	GCN	0.85	0.80	0.82

Table 3: Performance comparisons of clustering and speaker recognition results using models trained with different clustering pseudo-labels. The * symbol indicates that label de-noising was employed.

Model	Precision	Recall	F-score	EER	minDCF
Baseline	-	-	-	3.34	0.384
+ K-means	0.78	0.54	0.64	2.04	0.255
+ SC	0.74	0.67	0.70	1.73	0.213
+ AHC	0.79	0.77	0.77	1.51	0.186
+ GCN	0.83	0.79	0.81	1.43	0.174
+ GCN*	0.83	0.79	0.81	1.30	0.152
Oracle	-	-	-	1.28	0.165

TURN-TO-DIARIZE: ONLINE SPEAKER DIARIZATION CONSTRAINED BY TRANSFORMER TRANSDUCER SPEAKER TURN DETECTION

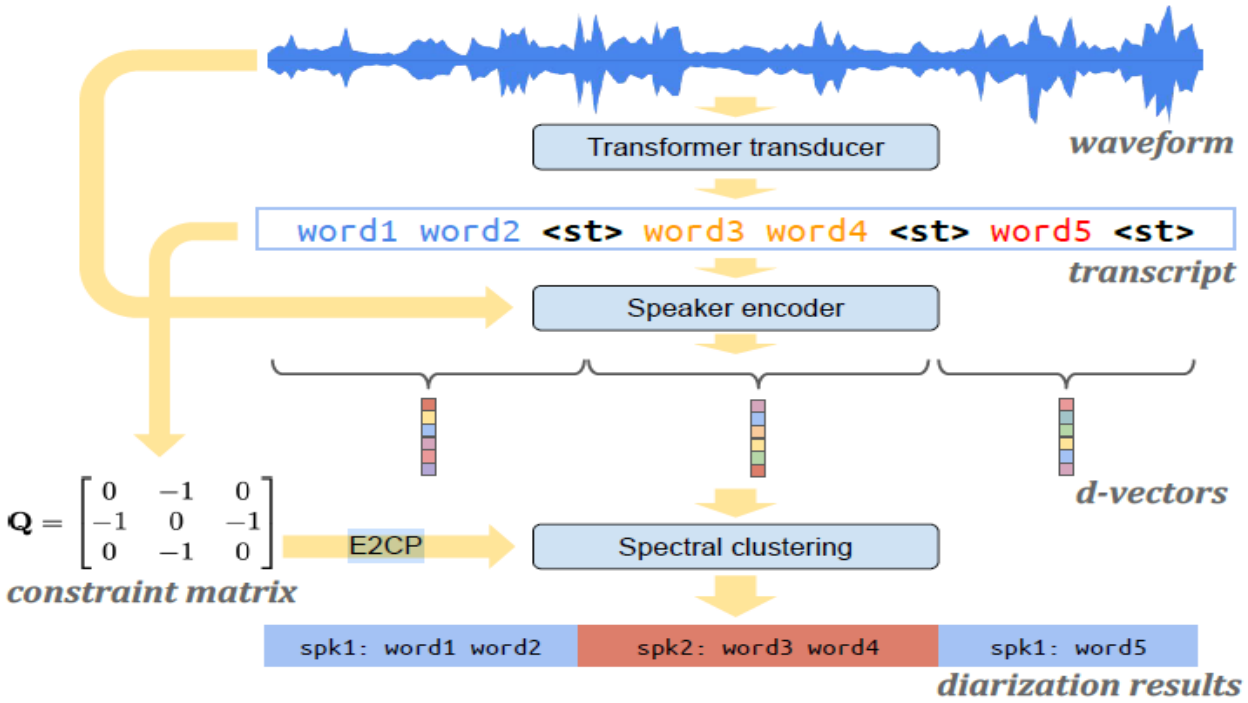
* Google LLC

- Motivation
  - the system only requires including speaker turn tokens during the transcribing process, which largely reduces the human efforts involved in data collection.
- Datasets
  - internal call center domain dataset
  - Callhome American English corpus
- Methods
  - transformer transducer-based model for joint ASR and speaker turn detection
  - a constrained spectral clustering algorithm
  - speaker diarization system for streaming on-device applications
- Experiment
  - show the experimental results of the “dense d-vector” and the proposed “turn-to-diarize” systems on the Internal Inbound,Outbound datasets, as well as the publicly available Callhome evaluation set.

TURN-TO-DIARIZE: ONLINE SPEAKER DIARIZATION CONSTRAINED BY TRANSFORMER TRANSDUCER SPEAKER TURN DETECTION

* Google LLC

- Methods
  - The input utterance is first fed into a transformer transducer model for joint ASR and speaker turn detection.
  - Then the utterance is segmented into speaker turns, and each turn is fed into an LSTM based speaker encoder to extract a d-vector embedding.
  - use a spectral clustering algorithm to cluster these turn-wise d-vectors, but with constraints from the detected speaker turns.



* Google LLC

- Datasets
  - The “Outbound”, which includes 450 conversations initiated by the call center. This dataset has approximately 35 hours of speech in total. Each utterance has 2 speakers.
  - The “Inbound” subset, which includes 250 conversations initiated by customers. This dataset has approximately 22 hours of speech in total. Each utterance has 2 to 10 speakers.
  - the train subset has been used for training the speaker turn detection model, we report the diarization results on the eval set of 20 utterances, which is about 1.7 hours of recordings in total.

TURN-TO-DIARIZE: ONLINE SPEAKER DIARIZATION CONSTRAINED BY TRANSFORMER TRANSDUCER SPEAKER TURN DETECTION

* Google LLC

- Experiments

Table 2. Confusion (%), total DER (%) and GFLOPS/s on three datasets for different embeddings and methods.

System	Method	Inbound		Outbound		Callhome Eval		GFLOP/s at 10min	GFLOP/s at 1h
		Conf.	DER	Conf.	DER	Conf.	DER		
Dense d-vector	Dense	17.98	22.13	10.66	15.97	5.39	7.76	0.85	36.54
	Dense + Auto-tune	14.09	18.24	9.56	14.88	5.42	7.79	4.76	361.37
Turn-to-diarize	Turn	17.87	19.43	8.41	10.34	8.23	10.08	1.00	1.18
	Turn + E2CP	17.21	18.77	7.94	9.86	3.56	5.41	1.00	1.18
	Turn + Auto-tune	13.83	15.39	7.01	8.93	5.11	6.95	1.02	2.81
	Turn + E2CP + Auto-tune	**13.66**	**15.22**	**6.86**	**8.78**	**3.49**	**5.33**	1.02	2.81

Thanks