Beating human performance at recognizing speech commands in temporal domain

Iván Vallés-Pérez, Fernando Mateo, Joan Vila-Francés, Antonio J. Serrano-López, Emilio Soria-Olivas

Escola Tècnica Superior d'Enginyeria, University of Valencia, Avenida de la Universitat s/n 46100 Burjassot, Valencia, Spain

The field of voice-activated virtual assistants is booming

- Several big companies like Amazon, Google, Baidu and Apple have already developed their version of virtual assistant
- In particular, **Deep Learning (DL) models** have revolutionized the field of automatic speech recognition[1], as **language features are highly hierarchical**.
- There are multiple **open research lines**.
  - Increasing the accuracy and relevance of the responses[2]
  - Reducing the answer delay[3]
  - Increasing their variability of the responses[4]
  - ...

---

[1] Ali Bou Nassif et al. "Speech recognition using deep neural networks: A systematic review". In: *IEEE Access* 7 (2019), pp. 19143–19165.

[2] Iulian Serban et al. "A Deep Reinforcement Learning Chatbot". In: *Proceedings of the Neural Information Processing Systems Conference.* 2017.

[3] Song Han et al. "ESE: Efficient Speech Recognition Engine with Sparse LSTM on FPGA". In: *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays.* FPGA. Monterey, California, USA, 2017, pp. 75–84. ISBN: 978-1-4503-4354-1. DOI: 10.1145/3020078.3021745.

[4] Jiwei Li et al. "Adversarial Learning for Neural Dialogue Generation". In: *Proceedings of the conference on Empirical Methods in Natural Language Processing.* 2017, 2157–2169.

This work focuses on increasing the accuracy of the voice commands recognition

- The **objective** of this project is to achieve the **best possible accuracy** on the recognition of speech commands under a **limited vocabulary setting**
- For that, we propose using Deep Learning techniques – more specifically: **convolutional neural networks**
- **No complex pre-processing** techniques (such as *FFT* and spectrograms) are intended to be applied to the audio clips: we are going to stay in the **temporal domain**
- To quantify the performance of the solution, we will not only measure against existing **benchmarks**, but also against manually measured **human accuracy**

The Google Tensorflow speech commands data set has been used to perform this study

- *Google Tensorflow speech commands data set*[5] is a collection of categorized audio utterances released by Google in 2018
- Noisy and low-quality audio clips, recoded in uncontrolled environments
- More than **100,000 1s-length audio clips** belonging to **35 different classes**[6]
- **Two versions** of the data set are available, where V2 is an extended and cleaned version of V1. Results have been reported on both versions of the data set

[left], [right], [yes], [no], [down], [up], [go], [stop], [on], [off], [zero], [one], [two], [three], [four], [five], [six], [seven], [eight], [nine], [dog], [cat], [wow], [house], [bird], [happy], [sheila], [marvin], [bed], [tree], [visual], ~~[follow], [learn], [forward], [backward]~~

[5] Pete Warden. "Speech Commands: A public dataset for single-word speech recognition.". In: *Datasets available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz and http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz* (2017).

[6] According to version 2 of the data set. Version 1 contains around 65,000 clips and 30 different classes

Classifying speech commands at temporal domain is not straightforward

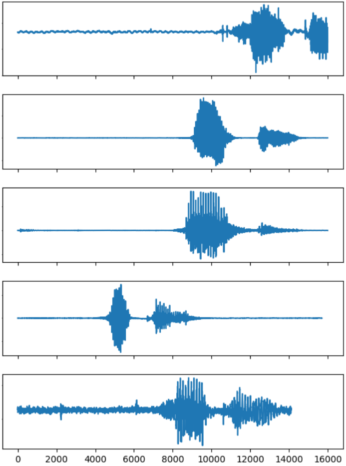


Figure: Happy: [1], [2], [3], [4], [5]

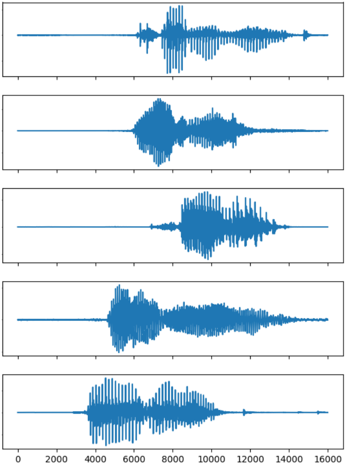


Figure: Forward: [1], [2], [3], [4], [5]

Several data augmentation techniques have been used to enhance generalization

The [original] data set has been augmented through the **application of 5 different distortions** with random intensities[7]:

- **Resampling**: expanding and contracting the audio clip + center-cropping. Examples: [1] [2] [3]
- **Pitch shift**: frequency increase/decrease to produce more acute/severe voices. Examples: [1] [2] [3]
- **Saturation**: amplitude increase until saturation. Examples: [1] [2] [3]
- **Time offset**: left or right zero padding. Examples: [1] [2] [3]
- **Noise addition**: sum of white noise on the temporal sequence. Examples: [1] [2] [3]

All the distortions are applied together with random intensities only over the training data, producing 5 new transformations of the original recordings [1] [2] [3] [4] [5]

[7] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing (4rd Ed.): Principles, Algorithms, and Applications*. 4th ed. Upper Saddle River, NJ, USA (2007): Prentice-Hall, Inc. ISBN: 978-8131710005.

Four different tasks have been defined with the available data for benchmarking purposes

- We defined a set of "tasks" by pre-selecting a subset of the classes and **assigning the others to a synthetic "unknown class"**.
- These groups have been established to be able to compare with the existing SOTA benchmarks[8][9][10][11].
  a. *35-words-recognition*
  b. *20-commands-recognition* + *unknown*
  c. *10-commands-recognition* + *unknown*
  d. *left-right* + *unknown*

---

[8] Douglas Coimbra de Andrade et al. "A neural attention model for speech command recognition". In: *Computing Research Repository CoRR*, arXiv:1808.08929 (2018). arXiv: 1808.08929 [eess.AS].

[9] Brian McMahan and Delip Rao. "Listening to the World Improves Speech Command Recognition". In: *Computing Research Repository CoRR* abs/1710.08377, arXiv:1710.08377 (2017). arXiv: 1710.08377.

[10] Pete Warden. "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition". In: *Computing Research Repository CoRR* abs/1804.03209, arXiv:1804.03209 (2018). arXiv: 1804.03209.

[11] Yundong Zhang et al. "Hello Edge: Keyword Spotting on Microcontrollers". In: *Computing Research Repository CoRR* abs/1711.07128, arXiv:1711.07128 (2017). arXiv: 1711.07128 [cs.SD].

Four different tasks have been defined with the available data for benchmarking purposes

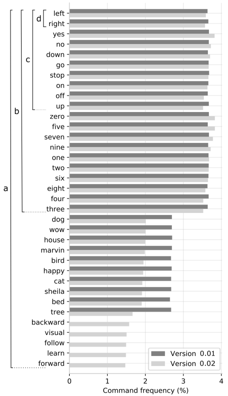


Figure: Command frequency distribution for both versions of the data set. As it can be noticed, the V2 is a refined and extended version of V1. In the left, the four different tasks that have been benchmarked in this work: (a) referred as *35-words-recognition* and comprising in both cases all the words for classification, (b) referred as *20-commands-recognition* (c) referred as *10-commands-recognition* (d) referred as *left-right* recognition.

As we group the unrecognized words under the "unknown" category, the class imbalance grows

- The **cost of a false positive** in a speech recognition agent is **higher** than that of a false negative
- The precision should be optimized at the expense of a worse recall
- Thus, having a positive imbalance towards the "unknown" class does not represent a very big inconvenience

Table: Percentage of words represented by the "unknown" category in each one of the proposed speech recognition tasks.

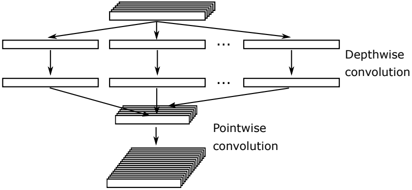
Data set version	*35-words*	*20-commands*	*10-commands*	*left-right*
V1	0.00%	26.84%	63.41%	92.71%
V2	0.00%	26.81%	63.58%	92.84%

A new CNN architecture based on the Xception network has been designed

- Xception[12] is a CNN-based deep learning architecture published by François Chollet (Google) in 2017 which recently achieved **SOTA results in multiple computer vision tasks**
- It uses *depthwise separable convolutions* and *residual connections*. Both together lead to a very **efficient yet deep** CNN.
- This algorithm is designed to work with images (2-D data). We have adapted it to work with sequences (1-D data): **Xception-1d**

[12] F. Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1800–1807. DOI: 10.1109/CVPR.2017.195.

The *depthwise separable convolution* operation is more efficient than the regular convolution (1/2)



Equation 1: regular convolution

$$\mathsf{G}_x = \sum_{s,m} \mathsf{K}_{s,m} \cdot \mathsf{F}_{x+s-\frac{S-1}{2},m} \quad (1)$$

Equation 2: depthwise convolution

$$\hat{\mathsf{G}}_{x,m} = \sum_{s} \mathsf{K}_{s,m} \cdot \mathsf{F}_{x+s-\frac{S-1}{2},m} \quad (2)$$

The *depthwise separable convolution* consists of two sequential steps

- *Depthwise convolution*[13]: **single filter per channel**. Modifies only spatial/temporal dimension(s). Number of channels remains intact. See equation 2
- *Pointwise convolution*[14]: **size-1 convolutions**. Modifies only the channels dimension. Spatial/temporal dimension(s) remain intact. See equation 1

---

[13] Yunhui Guo et al. "Depthwise Convolution is All You Need for Learning Multiple Visual Domains". In: *Association for the Advancement of Artificial Intelligence* (2019).

[14] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. "ChannelNets: Compact and Efficient Convolutional Neural Networks via Channel-Wise Convolutions". In: *Proceedings of Neural Information Processing Systems*. 2018.

The *depthwise separable convolution* operation is more efficient than the common convolution (2/2)

- The *depthwise separable convolution* receives its name because it **separates** the channel-wise and spatial/temporal-wise computations.
- The number of operations required by the depthwise-separable convolution is $\frac{1}{N} \cdot \frac{1}{S}$ **times** the number of operations required by a regular convolution[15] (where N is the number of output channels and S is the filter size), which represents a **meaningful performance improvement** for big networks.
- E.g. if we applied a size-5 convolution to generate a signal with 2 channels (e.g. a stereo audio signal), a regular convolution would need $5 * 2 = 10$ times more computation than a *depthwise separable convolution*.

[15] Andrew G. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.". In: *Computing Research Repository CoRR* abs/1704.04861, arXiv:1704.04861 (2017). arXiv: 1704.04861.

37 layers have been used to define the Xception-1d architecture, with a total of 23 million parameters

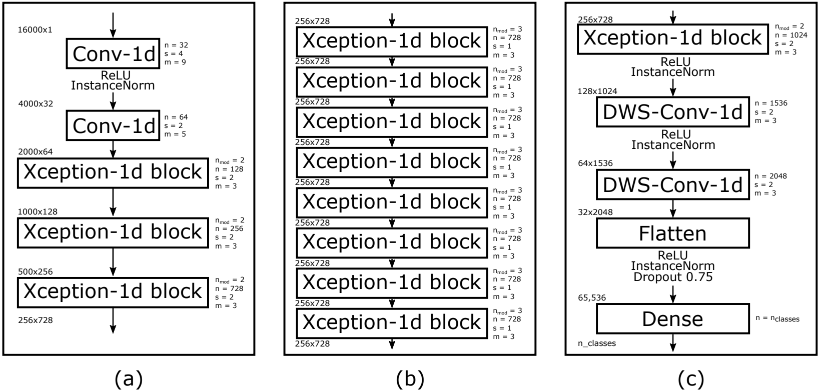
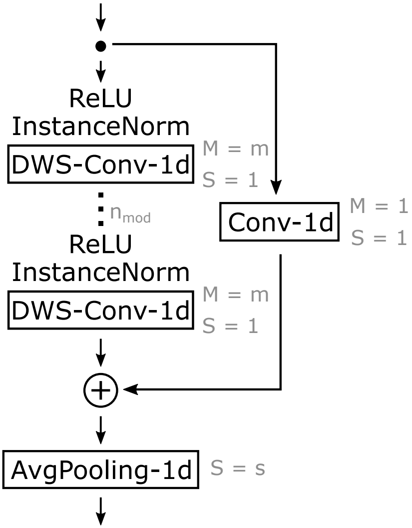


Figure: Diagram summarizing the architecture of *Xception-1d*. It's composed of three main modules: (a) **the entry module**, responsible for adapting the raw wave into a condensed representation, (b) the **middle module** responsible for learning the representation for extracting useful features from the raw data, (c) **the classification module**, responsible for mapping the learned representation into each output class

The building block of the architecture is composed of two conv layers, a skip connection and a pooling layer

- Each Xception-1d block contains n_{mod} **chained depth-wise separable convolutions** with a **residual connection**, followed by an **average pooling layer**
- **Instance Normalization** has been used as a way of reducing the covariance shift, and hence enhance generalization
- The **skip connections** of every block allows training deeper networks



The data has been divided in train/dev/test to enable model experimentation

- The **cross-validation split** has been provided by the authors of the data set (holding about 11,000 samples for development and other about 11,000 for test purposes)
- The models have been trained for **50 epochs** in each case with **early-stopping** and the parameters have been manually tuned
- **Five different models** have been trained for each task in order to explore and report the effect of different **random initializations** of the weights of the network
- With the aim of providing a baseline, **human performance** has been measured by **4 human subjects**, who manually labeled **1000 commands**
- The source code of this study is publicly available on **GitHub**: https://github.com/ivallesp/Xception1d

We achieved SOTA results in 3/4 tasks and beat the human performance in the 2 more complex ones

Table: Accuracy (in percentage points – mean $\pm$ standard deviation) obtained by the proposed solution on the different tasks compared to other benchmarks and compared to human accuracy. The results of best performing algorithms for each task have been highlighted in bold in each case. Results better than human performance (with statistical evidence at $\alpha = 0.05$) have been tagged with a star mark (*).

(a) Results for version 1 of the data set.

	Andrade et al.	McMahan et al.	Warden	*Xception-1d*	Human	p-value
35-words	94.30	84.35	-	**95.85 $\pm$ 0.12 ***	94.15 $\pm$ 1.03	$1.46 \cdot 10^{-2}$
20-commands	94.10	85.52	-	**95.89 $\pm$ 0.06 ***	94.56 $\pm$ 0.98	$3.14 \cdot 10^{-2}$
10-commands	95.60	-	85.40	**97.15 $\pm$ 0.03**	97.22 $\pm$ 0.85	$8.75 \cdot 10^{-1}$
left-right	**99.20**	95.32	-	98.96 $\pm$ 0.09	99.54 $\pm$ 0.16	$5.24 \cdot 10^{-4}$

(b) Results for version 2 of the data set.

	Andrade et al.	Zhang et al.	Warden	*Xception-1d*	Human	p-value
35-words	93.90	-	-	**95.85 $\pm$ 0.16 ***	94.15 $\pm$ 1.03	$1.50 \cdot 10^{-2}$
20-commands	94.50	-	-	**95.96 $\pm$ 0.16 ***	94.56 $\pm$ 0.98	$2.70 \cdot 10^{-2}$
10-commands	96.90	95.40	88.20	**97.54 $\pm$ 0.08**	97.22 $\pm$ 0.85	$4.84 \cdot 10^{-1}$
left-right	**99.40**	-	-	99.25 $\pm$ 0.07	99.54 $\pm$ 0.16	$1.27 \cdot 10^{-2}$

The precision and recall of 30 out of the 35 of the classes is always greater than 90%

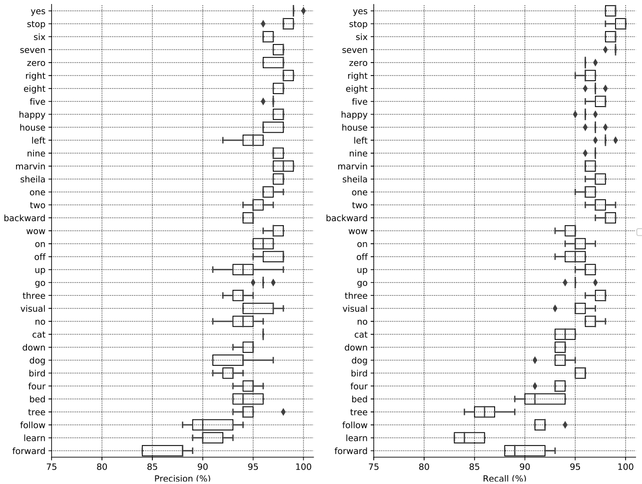


Figure: Precision and recall for each of the different classes using the *35-words-recognition* model trained with data version V2. Classes are sorted by descending f1-score.

We suggest Xception-1d as the default architecture for a speech commands classification problem

- The **proposed architecture** has around **20M parameters**. We should study the viability of implementing this model in an embedded device.
- We showed how a neural net that succeeded in the computer vision field can be **adapted** to the speech recognition field and achieve state of the art results
- We suggest *Xception-1d* as the *de facto* architecture when facing this kind of task

Backup

	precision	recall	f1-score	support
happy	98.00±0.89	97.80±0.75	98.00±0.63	180
cat	98.20±0.40	97.80±0.40	98.00±0.00	166
house	97.60±1.36	98.60±0.49	97.80±0.75	150
dog	97.80±0.75	98.00±0.00	97.80±0.40	180
marvin	99.00±0.63	96.80±0.75	97.80±0.40	162
stop	97.20±1.47	98.20±0.40	97.60±1.02	249
yes	98.60±1.02	96.40±0.80	97.40±0.49	256
sheila	97.20±1.33	97.00±0.63	97.20±0.75	186
wow	96.80±1.17	97.40±0.49	97.20±0.40	165
seven	95.80±1.47	98.40±0.80	97.20±0.40	239
four	96.40±0.80	97.20±0.75	97.00±0.63	253
two	95.80±1.33	97.60±0.49	96.80±0.75	264
nine	95.40±1.50	98.20±0.75	96.80±0.40	259
on	95.80±1.60	97.00±0.63	96.60±1.02	246
six	95.80±0.40	97.20±0.40	96.40±0.49	244
bird	95.80±0.98	96.40±0.49	96.20±0.75	158
eight	96.80±1.94	95.40±0.49	96.00±0.89	257
five	96.00±0.63	96.00±0.63	96.00±0.63	271
one	98.00±0.89	93.80±1.33	95.80±0.40	248
down	96.00±0.63	95.00±0.89	95.60±0.80	253
bed	95.00±1.67	96.20±1.47	95.40±1.02	176
left	93.00±1.67	97.20±0.40	95.20±0.75	267
off	96.80±1.47	94.00±1.26	95.20±0.40	262
right	97.40±1.20	92.80±0.40	95.20±0.40	259
zero	95.60±1.36	94.40±1.02	95.00±0.63	250
up	94.80±0.75	94.80±0.98	94.80±0.75	272
no	94.60±1.02	93.00±0.89	93.80±1.17	252
go	94.60±1.62	93.40±0.80	93.80±0.75	251
tree	92.40±1.50	90.60±1.36	91.60±0.49	193
three	89.60±0.49	92.80±0.75	91.20±0.40	267
avg/total	96.00±0.00	96.00±0.00	96.00±0.00	6835

	precision	recall	f1-score	support
yes	99.20±0.40	98.60±0.49	99.00±0.00	419
stop	98.00±1.10	99.40±0.80	98.60±0.49	411
seven	97.60±0.49	98.80±0.40	98.40±0.49	406
six	96.40±0.49	98.60±0.49	97.60±0.49	394
right	98.40±0.49	96.20±0.75	97.40±0.49	396
sheila	97.60±0.49	97.20±0.75	97.20±0.75	212
nine	97.40±0.49	96.80±0.40	97.20±0.40	408
eight	97.60±0.49	97.00±0.63	97.20±0.40	408
marvin	98.00±0.89	96.40±0.49	97.20±0.40	195
five	96.80±0.40	97.40±0.80	97.00±0.00	445
house	96.80±0.98	97.00±0.63	96.80±0.75	191
happy	97.60±0.49	96.00±0.63	96.80±0.40	203
zero	97.20±0.98	96.20±0.40	96.60±0.49	418
left	94.60±1.50	98.00±0.63	96.40±0.80	412
backward	94.60±0.49	98.20±0.75	96.40±0.49	165
one	96.60±0.80	96.20±0.75	96.40±0.49	399
two	95.40±1.02	97.40±1.02	96.20±0.75	424
wow	97.20±0.75	94.40±0.80	96.00±0.89	206
off	97.00±1.26	94.80±1.17	95.80±0.75	402
on	96.00±0.89	95.40±1.02	95.80±0.40	396
visual	96.00±1.67	95.20±1.33	95.60±0.80	165
go	96.00±0.63	95.20±0.98	95.40±0.49	402
no	93.80±1.72	96.80±0.75	95.40±0.49	405
up	94.20±2.32	96.20±0.75	95.20±0.98	425
cat	96.00±0.00	94.00±0.89	95.20±0.75	194
three	93.60±1.02	97.40±0.80	95.20±0.75	405
four	94.60±1.02	93.00±1.10	94.00±0.63	400
bird	92.60±1.02	95.60±0.49	94.00±0.63	185
down	94.40±0.80	93.60±0.49	94.00±0.00	406
dog	93.40±2.24	93.40±1.36	93.40±0.80	220
bed	94.40±1.36	91.60±2.06	93.20±1.17	207
follow	90.80±2.32	92.00±1.10	91.60±1.36	172
tree	94.80±1.72	86.20±1.72	90.40±1.20	193
forward	86.60±2.15	90.00±2.10	88.20±1.72	155
learn	90.80±1.47	84.40±1.36	87.60±1.02	161
avg/total	96.00±0.00	96.00±0.00	96.00±0.00	11005

Table: Detailed results for task *20-commands-recognition* and data version V1, sorted by decreasing f1-score order. The columns "precision", "recall" and "f1-score" have been represented as the mean $\pm$ the standard deviation in percentage scale.

	precision	recall	f1-score	support
nine	97.80±1.17	97.60±1.02	97.80±0.40	259
stop	97.00±1.79	98.00±0.63	97.60±1.02	249
yes	98.60±0.80	96.40±0.49	97.60±0.49	256
seven	96.40±0.49	98.20±0.75	97.40±0.49	239
six	97.20±0.75	97.40±0.49	97.20±0.40	244
unknown	96.60±0.49	97.00±0.00	97.00±0.00	1716
on	96.40±1.20	97.20±0.40	96.80±0.40	246
five	96.80±1.17	95.60±0.80	96.20±0.75	271
one	98.00±0.63	94.20±0.40	96.20±0.40	248
zero	96.60±1.50	94.60±1.02	95.80±0.98	250
four	94.00±1.10	97.60±0.49	95.80±0.75	253
two	94.80±1.60	96.40±0.80	95.60±1.02	264
left	93.60±1.02	97.20±0.75	95.40±0.80	267
eight	95.60±0.49	95.40±0.80	95.40±0.49	257
right	96.60±1.02	93.80±1.17	95.20±0.98	259
off	97.20±1.17	93.60±1.02	95.20±0.75	262
up	95.80±0.40	94.40±1.50	95.00±0.63	272
down	95.80±0.75	93.40±0.80	94.60±0.49	253
no	93.20±1.33	94.80±0.75	94.00±0.63	252
go	94.00±1.55	91.40±1.62	92.60±0.49	251
three	91.00±0.89	92.00±1.55	91.40±1.02	267
avg/total	96.00±0.00	96.00±0.00	96.00±0.00	6835

Table: Detailed results for task *20-commands-recognition* and data version V2, sorted by decreasing f1-score order. The columns "precision", "recall" and "f1-score" have been represented as the mean ± the standard deviation in percentage scale.

	precision	recall	f1-score	support
seven	98.80±0.40	98.60±0.49	98.80±0.40	406
yes	98.80±0.40	98.60±0.49	98.60±0.49	419
stop	98.80±0.40	99.00±0.63	98.60±0.49	411
six	97.60±1.02	98.20±0.75	97.60±0.49	394
eight	98.00±0.63	96.40±0.49	97.00±0.00	408
zero	97.80±0.75	96.40±0.49	96.80±0.40	418
nine	98.00±0.63	95.40±0.49	96.60±0.49	408
right	97.40±1.36	96.00±0.89	96.60±0.49	396
two	95.80±0.75	96.80±0.75	96.40±0.49	424
five	96.60±1.02	95.40±1.20	96.00±0.63	445
one	97.40±0.49	95.00±0.00	96.00±0.00	399
left	94.40±0.80	97.60±0.49	96.00±0.00	412
off	97.20±0.75	94.60±1.20	95.80±0.75	402
no	95.20±1.47	96.40±0.49	95.80±0.75	405
on	95.60±1.62	95.20±0.75	95.40±0.49	396
up	95.40±0.49	95.20±0.40	95.40±0.49	425
three	94.40±1.02	96.20±1.17	95.00±0.00	405
unknown	94.40±0.49	96.00±0.63	95.00±0.00	2824
go	95.00±1.41	94.80±0.75	94.80±0.40	402
down	94.80±0.40	93.40±0.49	94.20±0.40	406
four	95.20±0.98	92.00±1.67	93.60±0.49	400
avg/total	96.00±0.00	96.00±0.00	96.00±0.00	11005

Table: Detailed results for task *10-commands-recognition* and data version V1, sorted by decreasing f1-score order. The columns "precision", "recall" and "f1-score" have been represented as the mean ± the standard deviation in percentage scale.

	precision	recall	f1-score	support
unknown	97.60±0.49	99.00±0.00	98.20±0.40	4268
stop	98.20±1.17	97.60±0.80	97.80±0.40	249
yes	98.60±1.50	95.60±0.80	97.00±0.89	256
on	97.60±1.02	95.80±0.40	96.80±0.40	246
left	95.60±1.02	95.60±0.80	95.60±0.49	267
right	96.60±1.50	92.80±0.75	94.40±0.80	259
up	95.60±1.36	93.00±0.89	94.40±0.80	272
off	96.40±1.50	92.40±1.50	94.40±0.49	262
down	95.40±0.49	92.80±0.75	94.20±0.40	253
no	94.80±0.75	91.80±0.40	93.20±0.40	252
go	93.60±2.24	92.40±1.62	92.80±1.33	251
avg/total	97.00±0.00	97.00±0.00	97.00±0.00	6835

Table: Detailed results for task *10-commands-recognition* and data version V2, sorted by decreasing f1-score order. The columns "precision", "recall" and "f1-score" have been represented as the mean $\pm$ the standard deviation in percentage scale.

	precision	recall	f1-score	support
unknown	98.20±0.40	99.00±0.00	99.00±0.00	6931
yes	99.00±0.63	98.40±0.49	98.80±0.40	419
stop	98.40±0.49	98.60±0.49	98.40±0.49	411
right	97.80±0.75	95.60±1.02	96.80±0.75	396
left	94.40±1.02	97.40±0.49	95.80±0.40	412
go	95.40±1.02	94.80±0.75	95.00±0.63	402
up	96.20±0.75	93.60±0.49	95.00±0.00	425
no	93.80±1.33	95.20±1.33	94.80±0.75	405
off	95.40±0.80	93.80±0.40	94.60±0.49	402
on	95.60±1.02	93.80±0.75	94.40±0.49	396
down	94.80±0.98	93.00±0.89	94.00±0.63	406
avg/total	97.60±0.49	97.60±0.49	97.60±0.49	11005

Table: Detailed results for task *left-right* and data version V1, sorted by decreasing f1-score order. The columns "precision", "recall" and "f1-score" have been represented as the mean ± the standard deviation in percentage scale.

	precision	recall	f1-score	support
unknown	99.00±0.00	100.00±0.00	99.20±0.40	6309
left	95.40±1.96	92.00±0.89	93.60±0.80	267
right	96.20±1.94	87.60±1.85	91.80±0.75	259
avg/total	99.00±0.00	99.00±0.00	99.00±0.00	6835

Table: Detailed results for task *left-right* and data version V2, sorted by decreasing f1-score order. The columns "precision", "recall" and "f1-score" have been represented as the mean ± the standard deviation in percentage scale.

	precision	recall	f1-score	support
unknown	99.40±0.49	100.00±0.00	100.00±0.00	10197
left	95.80±0.98	94.00±0.89	95.00±0.63	412
right	98.20±1.47	90.60±2.65	94.00±1.10	396
avg/total	99.00±0.00	99.00±0.00	99.00±0.00	11005