INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2026, VOL. 72, NO. 3, PP. 1–8
Manuscript received March 12, 2026; revised July 2026. doi: 10.24425/ijet.2026.157946

Google Speech Commands benchmarks tests with new dataset extension

Mateusz Kucharski, Radosław Idzikowski, and Wiktoria Sarnicka

***Abstract*—The Google Speech Commands dataset remains one of the most widely used benchmarks for evaluating keyword-spotting and limited-vocabulary speech recognition systems. However, the discontinuation and partial loss of functionality of the Papers with Code platform has created gaps in the accessibility, transparency, and reproducibility of previously published benchmark results. This paper addresses this issue by reconstructing, validating, and comparing the performance of state-of-the-art keyword-spotting models whose code repositories remain publicly available. Four major frameworks—ML-KWS-for-MCU, Keyword Transformer (KWT), Howl, and SpeechCmdRecognition—were tested using both the original Google Speech Commands dataset and a newly developed extension recorded at Wrocław University of Science and Technology. The extension provides 3,710 high-quality recordings from 106 non-native English speakers, captured under controlled acoustic conditions. Experimental results show that most architectures generalize well across datasets, with KWT models achieving the highest robustness and accuracy, including perfect classification under certain conditions. Conversely, classical DNN-based models exhibit significant performance degradation, demonstrating their sensitivity to acoustic variability. The study underscores the need for reliable benchmark repositories and standardized evaluation environments, particularly as emerging paradigms—such as quantum machine learning—begin to influence research in speech recognition. The findings provide a consolidated, verifiable comparison of contemporary models and lay groundwork for future benchmarking efforts in both classical and quantum-enhanced approaches to speech processing.**

***Keywords*—machine learning; neural networks; speech recognition; keyword spotting; benchmarking; classification**

I. Introduction

SPEECH recognition in recent decades became an indispensable part of our daily lives with implementation ranging from smartphones to cars and even fridges [1] with artificial neural networks being the methodology yielding by far the best results. Researchers continue to improve this technology by developing new algorithms and network architectures. With this development comes the need for testing, benchmarking and consequently comparing the efficiency. To achieve results that can be fairly compared to the previously developed algorithms, it is important to utilize the same data. Thus, many datasets are being created and published, enabling fair testing and allowing researchers to focus on their new algorithms rather than gathering new data, which is a very time-consuming task. The most common and balanced dataset used in speech recognition is the Google Speech Commands dataset created by Pete Warden [2], whose popularity is well demonstrated by its high number of citations – 2295 according to Google Scholar at the time of writing.

Another tool needed for clear and transparent comparison of researchers achievement claims is a benchmark hub that allows to present results in a verifiable way with the ability to replicate the process. One such tool used to be a website 'papers with code', which allowed users to upload research results together with a scientific paper describing them and a GitHub repository link for code access. Unfortunately, at the time of writing, the website is no longer available in the same way as before. It has been replaced by another site 'huggingface', that has clearly lost some of the previously uploaded results and still seems under construction. The 'papers with code' website is still partially available with the internet archive 'wayback machine', and this fact has become one of the reasons for this article – to compile as much of researchers' achievements, while it is still possible.

The second reason for this research is the development of the Google Speech Commands dataset extension at Wrocław University of Science and Technology, which has been recorded in the fall of 2024 with the help of university students. This extension allows for, in the spirit of fair research comparison, testing models and algorithms presented by scientists, ensuring that all testing data is new and has never been used in the training process, which is paramount in neural network research.

The structure of this article is as follows. Section I introduces the goal of the research behind this paper, the motivations of the authors as well as briefly summarizes the landscape of machine learning benchmarks with focus on the task of speech recognition with the most popular dataset. Then Section II presents the current state of research as well as introduces all papers that were published on the 'papers with code' website before its shutdown. Next, Section III gives the overview of the dataset used by researchers in the chosen benchmarks, as well as its extension provided by Wroclaw University of Science and Technology. In Section IV the research methodology used to test the benchmarks is

Mateusz Kucharski is with Department of Control and Quantum Computing, Wrocław University of Science and Technology, Poland (e-mail: mateusz.kucharski@pwr.edu.pl, ORCiD: 0000-0002-1838-5901).

Radosław Idzikowski is with Department of Control and Quantum Computing, Wrocław University of Science and Technology, Poland (e-mail: radoslaw.idzikowski@pwr.edu.pl, ORCiD: 0000-0001-5232-1998).

Wiktoria Sarnicka Wrocław University of Science and Technology, Poland (e-mail: sarnickawiktoria@gmail.com).

described. The Section V describes the achieved results and compares them based on several factors. Finally, Section VII summarizes the findings, presents conclusions and presents potential research interests for the future.

II. Literature Review

Many scientific papers tackling the speech recognition problem have been published, and since 2018, a sizable portion of them have used the Speech Commands dataset. However, the number of entries provided on the 'papers with code' website was just 42, and out of them, multiple entries can be attributed to a single paper. Of them, only 32 original papers in total can be distinguished, as some of them describe multiple models that create separate entries. Despite the website name and purpose, 13 entries did not provide any code repository; it was also not found in the paper itself, nor could the team find it anywhere through search engines. After rejecting the papers without code, this number shrinks further to 21. It must be addressed here, that it may be possible that the changes to the website and the use of internet archive system may have an impact on the code visibility. A good example is the paper 'EdgeCRNN: an edge-computing oriented model of acoustic feature enhancement for keyword spotting' [3], which seems well-crafted and well-explained but lacks a code repository. Similar case appears for the 'Effective combination of DenseNet and BiLSTM for keyword spotting' [4]. The first article in the leaderboard, 'Learning efficient representations for keyword spotting with triplet loss' [5], does have an active repository and it is well described in the paper, however the team had trouble utilizing it due to the complexity of the input and the need to train the whole model again.

Another popular category of solutions consists of approaches based on the ResNet model [6], [7] and its different variants. This family of models is widely used in image analysis, and due to the use of residual blocks and skip connections, the training process is highly efficient, which enables their deployment on mobile devices [8]. In all ResNet cases, no pretrained model was available. A very recent and emerging approach to audio analysis involves hybrid methods that leverage quantum computers in the NISQ era. In this setting, audio signals can be represented as Mel-spectrograms, which are first encoded using convolutional layers on a classical CPU to reduce the dimensionality of the problem. The processed features are then passed to quantum-circuit–based layers (Ansatz) for classification. Among current hybrid architectures, Quantum Convolutional Neural Networks (QCNNs) [9] have attracted significant attention, primarily because of the limited size of contemporary quantum processors. The authors employed the PennyLane framework to simulate and interface with quantum processing units. Using a QPU (Quantum Processor Unit) simulator, they achieved an accuracy of 95.12 ± 0.18.

As observed, a wide range of approaches can be applied to the speech recognition problem—from transformer-based architectures, through CNNs, general DNN or RNN models, to hybrid methods involving QPUs. Most studies evaluate and compare models using the Google Speech Commands dataset, either selecting subsets or using the full set of commands. Even within a single publication, authors often propose multiple models and benchmark them against established baselines. However, none of the analyzed works attempted to test the trained models on newly created samples (recording according to the original authors' guidelines) to verify the generalization capabilities of the models.

III. Dataset description

The task of creating a good dataset for any classification problem is difficult, mainly because gathering the material itself is time-consuming. The data has to be well balanced and appear in many repetitions, which are still different enough from each other as to allow the machine learning algorithm to gather enough information from the data, so that it would be able to recognize it later from a source it has never seen before. In terms of speech recognition tasks, this is problematic, as it requires not only multiple phrases, words, or phonemes, but also multiple speakers, so as not to narrow the learning algorithm to a single person. There are still many good datasets created that try to circumvent the issue of recording, the most common dataset, Librispeech, uses audiobooks as data [10], some 'in-the-wild' datasets also use recordings of politician speeches or movies [11].

One dataset that does however use dedicated recordings is Google Speech Commands [2]. This dataset has two main versions. First one contains 30 common english words, including numbers 0-9, words like 'yes'-'no', 'up'-'down', 'cat', 'dog' and english names 'Marvin' and 'Sheila'. The second one is an enlargement of the first, which expands the dataset by 5 new words: 'backward', 'forward', 'follow', 'learn', and 'visual'. The recordings are exactly 1 second in length, even when the duration of the spoken word is less – thus containing silence at the beginning and end. This helps in fitting the input sizes for machine learning algorithms. The files have 16 kHz sampling frequency and are monophonic. They contain multiple speakers, who are anonymized and only utter several words, thus this dataset is not suitable for speaker recognition. The dataset is mostly balanced – ranging in 1500 to 2500 recordings per word and having 64 727 entries in total in version 1 and 105 829 in version 2.

Based on the second version of the Google Speech Commands dataset, an extension was developed. Recordings were conducted with help of 106 students at Wrocław University of Science and Technology, Faculty of Information and Communication Technology, uttering single repetition of all 35 words from datasets second version, totaling 3710 new audio files. The recordings were conducted in a room adhering to ITU-T P.800 [12] standards using an MXL770 condenser microphone and a Focusrite Scarlett Solo sound card. The recordings were originally done with a 512 kHz sampling frequency and, for this research, were downsampled to 16 kHz. They were also trimmed to match the original datasets' 1 second in length. Noteworthy aspect about the speakers in this extension is the fact that, unlike the original, none of them is an English native. Most are Polish and some Ukrainian and while all of them know English, their accent noticeability varies significantly.

IV. METHODS

Four repositories, together with their corresponding reference articles, were subjected to detailed analysis. The basic information was collected and presented in Table I. To improve the clarity and readability of the work, each repository was assigned a distinct color, which is consistently used throughout the subsequent sections to identify its results. Some repositories contained multiple models; in such cases, all available models were included in the analysis.

TABLE I
LIST OF GITHUB REPOSITORIES USED FOR THIS RESEARCH

Architecture	Repository link	Reference
ML KWS for MCU	https://github.com/ARM-software/ML-KWS-for-MCU	[13]
Keyword Transformer	https://github.com/ARM-software/keyword-transformer	[14]
Howl	https://github.com/castorini/howl/	[15]
SpeechCmdRecognition	https://github.com/douglas125/SpeechCmdRecognition	[16]

A. ML KWS for MCU

The first repository selected for the experiments was ML-KWS-for-MCU. This repository includes seven different neural network architectures trained on the Google Speech Commands dataset. The implemented models comprise a deep neural network (DNN), a convolutional neural network (CNN), a recurrent neural network (RNN), a convolutional recurrent neural network (CRNN), and a depthwise separable convolutional neural network (DS-CNN).

Each of these architectures was trained in multiple model size configurations: S (80 KB, 6 MOPS), M (200 KB, 20 MOPS), and L (500 KB, 80 MOPS), allowing for systematic evaluation under varying memory and computational constraints.

B. Keyword Transformer

The next repository used in this study, which already provides pretrained models, is the Keyword Transformer project. This project implements the Keyword Transformer (KWT) model, designed for keyword spotting, i.e., the recognition of specific spoken keywords from short audio recordings. In contrast to conventional architectures employed in ML-KWS-for-MCU, such as CNN, RNN, or hybrid models the KWT architecture is based entirely on the self-attention mechanism, following the Transformer paradigm.

The repository includes three principal variants of the model that differ in size and computational complexity: KWT-1, KWT-2, and KWT-3. These variants enable systematic evaluation of trade-offs between accuracy, model capacity, and resource requirements. In addition to the Transformer-based KWT models, another architecture referenced in the literature and relevant to keyword spotting is the MHAtt-RNN (Multi-Head Attention Recurrent Neural Network). This model combines the strengths of recurrent neural networks with the expressive power of multi-head attention, providing an alternative to both purely convolutional and purely self-attention-based approaches. In the MHAtt-RNN architecture, an RNN backbone (typically based on GRU or LSTM units) is augmented with a multi-head attention mechanism that operates on the sequence of hidden states. This design allows the model to capture long-range temporal dependencies more effectively than standard RNNs, while still maintaining a relatively compact structure suitable for real-time keyword spotting.

The multi-head attention module plays a crucial role in selectively focusing on the most informative time steps in the audio sequence, enabling the model to better discriminate between acoustically similar keywords and remain robust to temporal variations, such as speaking rate and background noise. Compared with conventional RNN-based architectures, MHAtt-RNN has been shown to deliver improved accuracy while preserving efficiency, making it a competitive option for embedded keyword-spotting systems. The paper demonstrates that this hybrid attention-RNN formulation achieves a favorable balance between model complexity and performance, positioning it between lightweight CNN/RNN models and more computationally expensive Transformer-based solutions such as KWT.

TABLE II
MODEL PARAMETERS FOR THE KWT ARCHITECTURE

Model	dim	mlp-dim	heads	layers	# parameters
KWT-1	64	256	1	12	607K
KWT-2	128	512	2	12	2,394K
KWT-3	192	768	3	12	5,261K

C. Howl

In this project, an acoustic model from the HOWL framework was employed, based on the lightweight convolutional Res8 architecture optimized for classifying short spoken commands from the Google Speech Commands dataset. The res8 model is a variant of a deep convolutional neural network from the ResNet family, consisting of eight convolutional layers interconnected through residual blocks. The model was trained to recognize ten core speech commands (yes, no, up, down, left, right, on, off, stop, go) according to the predefined vocabulary (VOCAB). Classification is performed using the CrossEntropyLoss function, which compares the predicted probability distribution with the ground-truth label. The optimization process follows the AdamW algorithm, combining adaptive learning rates with weight-decay regularization. The initial learning rate was set to 0.01, and after each epoch, it was reduced by a factor of 0.8 to gradually stabilize the training process. Training was conducted over 20 epochs using a batch size of 64. The project provided by the original authors did not include a pretrained model. Therefore, the model was trained and tested independently as part of this work, using both the

official test split of the Google Speech Commands dataset as well as an additional dataset created for the experiment.

D. SpeechCmdRecognition

One of the analyzed models was the Attention RNN, proposed in the SpeechCmdRecognition project. It represents an efficient and compact architecture designed for recognizing short voice commands from the Google Speech Commands dataset. The model's design was developed with deployment on devices with limited computational resources in mind, while maintaining high classification performance. In the feature extraction stage, one-dimensional temporal convolutions were applied, enabling the capture of local temporal patterns in the sequence of spectrogram frames. The acoustic representations processed in this manner are subsequently passed to the recurrent layers. The architecture includes two layers of bidirectional LSTM networks, allowing modeling of long-term dependencies in both forward and backward directions. This enables the integration of acoustic information while accounting for the full temporal context of the recording. The model was trained using the Adam optimizer with an initial learning rate of 0.001. The learning schedule reduces the learning rate by a factor of 0.4 every 10 epochs, helping stabilize the optimization process in later stages of training. The training procedure used minibatches of 64 samples. The maximum number of epochs was set to 40; however, an early-stopping mechanism was employed, terminating training if no improvement in validation performance was observed for 10 consecutive epochs.

V. Experiments

All experiments, including model execution and training, were performed on an NVIDIA DGX Station A100 equipped with an AMD EPYC 7742 processor (64 cores, 128 threads, 1.5–2.25 GHz), 512 GB DDR4 ECC RAM, and four NVIDIA A100 SXM4 80 GB accelerators (CUDA 12.4, driver 550.xx). The system operated under a DGX-compatible Ubuntu distribution and featured a 4-node NUMA topology. This hardware configuration provided a high-performance environment for parallel and GPU-accelerated computing, enabling efficient training of large-scale models and fast data processing.

Model Keyword Transformer evaluation was performed with the use of a Python testing script that integrates the TensorFlow Lite interface engine. The script allows for automated evaluation of a model on a entire directory containing audio recordings in the .wav format. The model is provided as a TFLite file, "non_stream.tflite", which is loaded by the TFLite interpreter. Class labels are read from a text file into the LABELS list, where each line corresponds to a single class (e.g., yes, no, down). This list is subsequently used to map model output indices to their corresponding class names. Each audio file is processed using the soudfile library, which loads the waveform and its sampling rate. The script verifies that the sampling rate is 16kHz, as required by the model. If the input contains multiple channels, it is converted to a mono signal by averaging the channels. The waveform is then adjusted to a duration of exactly one second by zero-padding shorter recordings or truncating longer ones. The resulting preprocessed audio vector serves as the input to the TensorFlow Lite model during inference. Model in the ML-KWS-fo-MCU repository are stored az frozen TensorFlow graphs (.pd) and therefore a Python evaluation script based on the TensorFlow library is used to load and run them.

To evaluate the performance of the analyzed keyword-spotting models, two types of metric ware collected: the classification report and the confusion matrix. Combining these two source of information makes it possible to assess both the overall effectiveness of the model and to identify its weaknesses at the level of individual classes. The classification report consisted of the following metrics: Precision - a measure indicating what proportion of the samples predicted as belonging to a given class actually belonged to the class, Recall - a measure showing what proportion of the samples from a given class were correctly detected by the model, F1-score the harmonic mean of precision and recall, providing a balanced assessment of classification performance for each class, Support - the number of test samples belonging to each class. The confusion matrix makes it possible to reveal specific cases in which the model fails, wihich are not always visible in aggregated metrics. The diagonal elements correspond to correct classifications, while the off-diagonal elements represent misclassifications. This facilitates the identification of classes for which the model achieves low recall despite high overall accuracy.

VI. Results and discussion

The evaluation was performed separately for each repository. In three cases, the models provided by the original authors were used. Only one repository required training a model from scratch; therefore, a detailed description of this model and training process is included.

A. ML KWS for MCU

For some models, especially simple architectures such as DNN, the accuracy obtained on our dataset is noticeably lower than the values reported both in the ML-KWS-for-MCU documentation and in our Google reference benchmarks. This indicates that the DNN model is highly sensitive to differences between the reference dataset and our recording collection, demonstrating low robustness to acoustic variability. Across all three size categories the CNN, LSTM, GRU, CRNN, and DS-CNN models achieve significantly better results than the classical DNN architecture. These models exhibit strong resilience to variations in the input data. In many cases, the models achieved accuracies higher than those reported by the original authors as well as those observed in our tests on the Google dataset. A clear trend is visible: the more advanced the architecture, the more stable and consistent the results. In each group (S/M/L) recurrent models obtain the highest performance. An interesting observation is that several Small variants (e.g., CRNN-S, GRU-S) reach accuracies comparable to, or even exceeding, their Medium and Large counterparts. This suggests that certain low-complexity models

can be equally effective, making them attractive candidates for deployment on resource-constrained devices.

TABLE III
ACCURACY COMPARISON FOR SMALL (S) MODELS

Model (S)	Acc in doc	Acc on Google	Acc on our dataset
DNN	0.846	0.8525	0.5670
CNN	0.916	0.9158	0.8774
Basic LSTM	0.920	0.9277	0.9453
LSTM	0.929	0.9329	0.9509
GRU	0.935	0.9362	0.9670
CRNN	0.940	0.9345	0.9726
DS-CNN	0.944	0.9385	0.9519

TABLE IV
ACCURACY COMPARISON FOR MEDIUM (M) MODELS

Model (M)	Acc in doc	Acc on Google	Acc on our dataset
DNN	0.864	0.8801	0.5943
CNN	0.922	0.9297	0.9377
Basic LSTM	0.930	0.9355	0.9575
LSTM	0.939	0.9406	0.9623
GRU	0.942	0.9453	0.9642
CRNN	0.944	0.9377	0.9670
DS-CNN	0.949	0.9424	0.9500

TABLE V
ACCURACY COMPARISON FOR LARGE (L) MODELS

Model (L)	Acc in doc	Acc on Google	Acc on our dataset
DNN	0.867	0.8967	0.6509
CNN	0.927	0.9332	0.9500
Basic LSTM	0.934	0.9394	0.9538
LSTM	0.948	0.9454	0.9538
GRU	0.947	0.9477	0.9736
CRNN	0.950	0.9449	0.9764
DS-CNN	0.954	0.9473	0.9632

The analysis of the confusion matrix for the DNN model reveals numerous difficulties of this architecture in correctly recognizing keyword classes. This is particularly evident for classes that exhibit strong acoustic or temporal similarity, such as (yes/no), (up/down), or (left/right). The model produces a substantial number of misclassifications in both directions, as evidenced by the high values outside the diagonal of the matrix.

TABLE VI
CONFUSION MATRIX FOR THE TESTED MODEL DNN

	silence	_unknown_	yes	no	up	down	left	right	on	off	stop	go
silence	0	0	0	0	0	0	0	0	0	0	0	0
unknown	0	0	0	0	0	0	0	0	0	0	0	0
yes	0	22	68	0	0	1	13	2	0	0	0	0
no	0	21	0	78	0	3	0	0	0	0	0	4
up	0	19	0	8	22	23	1	0	5	12	16	0
down	0	29	0	28	0	46	0	0	2	0	0	1
left	0	24	1	5	0	2	63	10	0	1	0	0
right	0	15	0	0	0	2	0	88	1	0	0	0
on	0	28	0	2	0	10	0	0	65	0	1	0
off	0	22	1	3	0	1	3	0	7	67	1	1
stop	0	16	0	2	9	3	2	0	0	1	71	2
go	0	21	0	44	0	8	0	0	0	0	0	33

The analysis of the confusion matrix shows that the CRNN model achieves very high classification accuracy across all evaluated keyword classes. In contrast to the classical DNN, the model makes only a few mistakes and maintains exceptional prediction stability. The only notable error occurs between the ”down” and ”no” classes, which may be attributed to their shared temporal dynamics. Nevertheless, the magnitude of this error is small and no comparable issues are observed for the remaining classes.

TABLE VII
CONFUSION MATRIX FOR THE TESTED MODEL CRNN

	silence	_unknown_	yes	no	up	down	left	right	on	off	stop	go
silence	0	0	0	0	0	0	0	0	0	0	0	0
unknown	0	0	0	0	0	0	0	0	0	0	0	0
yes	0	2	104	0	0	0	0	0	0	0	0	0
no	0	1	0	105	0	0	0	0	0	0	0	0
up	0	0	0	0	106	0	0	0	0	0	0	0
down	0	0	0	12	0	94	0	0	0	0	0	0
left	0	0	0	0	0	0	106	0	0	0	0	0
right	0	3	0	0	0	0	0	103	0	0	0	0
on	0	0	0	0	0	0	0	0	106	0	0	0
off	0	0	0	0	6	0	0	0	0	100	0	0
stop	0	0	0	0	4	0	0	0	0	0	102	0
go	0	1	0	5	0	1	0	0	0	0	0	99

B. *Keyword Transformer*

The analysis of the results presented for the KWT-1, KWT-2, KWT-3 and MHAtt-RNN models clearly demonstrates that all variants of the Keyword Transformer architecture achieve very high accuracy, both in the values reported in the documentation and in our evaluation on the custom test dataset. In every case, the accuracy exceeds 0.98, confirming the architecture's exceptional effectiveness for keyword spotting tasks. In particular, the KWT-2 model achieved perfect accuracy (100%), substantially exceeding the results reported by the original authors. Such performance may indicate a particularly strong alignment between the model and the characteristics of our test dataset, or a high degree of robustness to acoustic variability. Despite differences in the number of parameters, model size, and architectural depth, the performance of all three models remains highly comparable, with accuracy variations in our data set limited to fractions of a percentage point. Even the smallest variant, KWT-1, achieves performance comparable to that of the largest model, KWT-3. At this stage, it becomes evident that the KWT models exhibit greater stability and robustness than the architectures provided in the ML-KWS-for-MCU repository.

TABLE VIII
ACCURACY COMPARISON OF KWT MODELS

Model	Acc on Google	Acc. in doc	Acc. on our dataset
KWT-1	0.9922	0.9856	0.9972
KWT-2	0.9988	0.9843	1.0000
KWT-3	0.9995	0.9808	0.9991
MHAtt-RNN	0.9947	0.9836	0.9981

The confusion matrix for KWT-1 shows that the model achieves nearly perfect classification performance, with the only noticeable errors being a few sporadic misclassifications between certain class pairs, such as (down-no). A similar issue was previously observed in the CRNN model, likely due to similarities in the temporal dynamics of these words, as discussed earlier.

TABLE IX
CONFUSION MATRIX FOR THE TESTED MODEL KWT-1

	silence	_unknown_	yes	no	up	down	left	right	on	off	stop	go
silence	0	0	0	0	0	0	0	0	0	0	0	0
unknown	0	0	0	0	0	0	0	0	0	0	0	0
yes	0	0	106	0	0	0	0	0	0	0	0	0
no	0	0	0	106	0	0	0	0	0	0	0	0
up	0	0	0	0	105	0	0	0	0	1	0	0
down	0	1	0	0	0	104	0	0	0	0	0	1
left	0	0	0	0	0	0	106	0	0	0	0	0
right	0	0	0	0	0	0	0	106	0	0	0	0
on	0	0	0	0	0	0	0	0	106	0	0	0
off	0	0	0	0	0	0	0	0	0	106	0	0
stop	0	0	0	0	0	0	0	0	0	0	106	0
go	0	0	0	0	0	0	0	0	0	0	0	106

The analysis of the confusion matrix for KWT-2 demonstrates that the model achieved perfect classification on our dataset. This result surpasses both the accuracy reported by the original authors in the documentation and our own evaluation conducted on the Google Speech Commands dataset.

TABLE X
CONFUSION MATRIX FOR THE TESTED MODEL KWT-2

	silence	_unknown_	yes	no	up	down	left	right	on	off	stop	go
silence	0	0	0	0	0	0	0	0	0	0	0	0
unknown	0	0	0	0	0	0	0	0	0	0	0	0
yes	0	0	106	0	0	0	0	0	0	0	0	0
no	0	0	0	106	0	0	0	0	0	0	0	0
up	0	0	0	0	106	0	0	0	0	0	0	0
down	0	0	0	0	0	106	0	0	0	0	0	0
left	0	0	0	0	0	0	106	0	0	0	0	0
right	0	0	0	0	0	0	0	106	0	0	0	0
on	0	0	0	0	0	0	0	0	106	0	0	0
off	0	0	0	0	0	0	0	0	0	106	0	0
stop	0	0	0	0	0	0	0	0	0	0	106	0
go	0	0	0	0	0	0	0	0	0	0	0	106

TABLE XI
CONFUSION MATRIX FOR THE TESTED MODEL KWT-3

	silence	_unknown_	yes	no	up	down	left	right	on	off	stop	go
silence	0	0	0	0	0	0	0	0	0	0	0	0
unknown	0	0	0	0	0	0	0	0	0	0	0	0
yes	0	0	106	0	0	0	0	0	0	0	0	0
no	0	0	0	106	0	0	0	0	0	0	0	0
up	0	0	0	0	106	0	0	0	0	0	0	0
down	0	0	0	0	0	106	0	0	0	0	0	0
left	0	0	0	0	0	0	106	0	0	0	0	0
right	0	0	0	0	0	0	0	106	0	0	0	0
on	0	0	0	0	0	0	0	0	106	0	0	0
off	0	0	0	0	0	0	0	0	0	106	0	0
stop	0	0	0	0	0	0	0	0	0	0	106	0
go	0	0	0	0	0	0	0	0	1	0	0	105

TABLE XII
CONFUSION MATRIX FOR THE TESTED MODEL MHATT-RNN

	silence	_unknown_	yes	no	up	down	left	right	on	off	stop	go
silence	0	0	0	0	0	0	0	0	0	0	0	0
unknown	0	0	0	0	0	0	0	0	0	0	0	0
yes	0	0	106	0	0	0	0	0	0	0	0	0
no	0	0	0	106	0	0	0	0	0	0	0	0
up	0	0	0	0	106	0	0	0	0	0	0	0
down	0	1	0	1	0	104	0	0	0	0	0	0
left	0	0	0	0	0	0	106	0	0	0	0	0
right	0	0	0	0	0	0	0	106	0	0	0	0
on	0	0	0	0	0	0	0	0	106	0	0	0
off	0	0	0	0	0	0	0	0	0	106	0	0
stop	0	0	0	0	0	0	0	0	0	0	106	0
go	0	0	0	0	0	0	0	0	0	0	0	106

C. Howl

Based on the plots illustrating the training process of the Howl model, it can be observed that the training progresses in a stable and efficient manner. The training loss curve drops rapidly during the first epochs, indicating that the model effectively adapts to the input data. At the same time, the validation loss decreases at a similar rate and remains close to the training loss. The absence of significant divergence between the two curves suggests that the model does not suffer from overfitting it generalizes well to the validation data without excessively fitting the training set.

The accuracy plots further confirm the stability of the learning process. The training accuracy increases quickly and exceeds 0.95 after just a few epochs, gradually approaching approximately 0.97-0.98. The validation accuracy follows a similar trend, indicating a high level of consistency between the training and validation sets. Notably, from around the fifth epoch onward, the accuracy curves nearly overlap, demonstrating a strong level of convergence and the absence of noticeable fluctuations that could suggest instability or optimization issues.

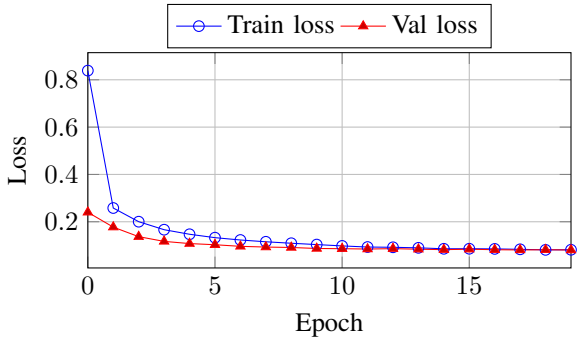

Fig. 1. Training and validation loss for the Howl model.

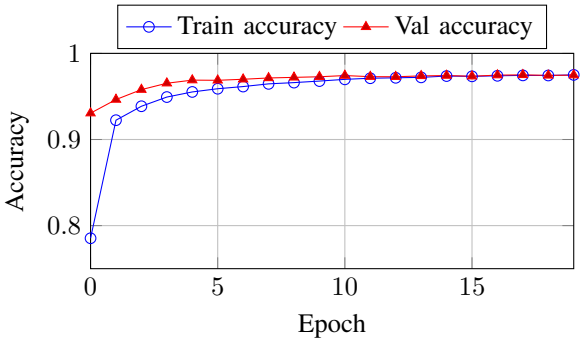

Fig. 2. Training and validation accuracy for the Howl model.

Analysis of confusion matrices provides a more detailed assessment of the model's performance. In the case of our own dataset, the model achieves almost perfect classification all commands such as (yes, no, left, right, down or stop) are recognized correctly in nearly 100% of cases, and the only errors that occur are sporadic single misclassifications. This indicates that the model represents very well the distribution of the data on which it was trained, and that neither the quality nor the homogeneity of the recordings poses a challenge.

More interesting conclusions emerge from the analysis of the Google Speech Commands dataset, which is characterized by greater variability, a larger number of samples, and different

recording conditions. Despite this, the Howl model still maintains very high effectiveness the values along the diagonal of the confusion matrix reach 2800-3200 correct classifications for each of the main commands. Most errors are rare (single misclassifications), while the few larger deviations, such as confusing the command no with left (81 cases), are related to the similar acoustic structure of certain words and the high diversity of the GSC recordings. Despite these differences, the proportion of errors remains very small compared to the number of correct predictions, indicating good generalization of the model beyond the training dataset.

TABLE XIII
CONFUSION MATRIX FOR THE TESTED MODEL HOWL ON OUR DATASET

	silence	_unknown_	yes	no	up	down	left	right	on	off	stop	go
silence	0	0	0	0	0	0	0	0	0	0	0	0
unknown	0	0	0	0	0	0	0	0	0	0	0	0
yes	0	0	106	0	0	0	0	0	0	0	0	0
no	0	0	0	106	0	0	0	0	0	0	0	0
up	0	0	0	0	104	0	0	0	0	1	0	0
down	0	0	0	0	0	104	0	0	0	0	0	0
left	0	0	0	0	0	0	106	0	0	0	0	0
right	0	0	0	0	0	0	0	103	0	0	0	3
on	0	0	0	0	0	0	0	0	105	0	0	1
off	0	0	0	0	0	0	0	0	0	104	0	1
stop	0	0	0	0	0	0	0	0	0	0	105	1
go	0	0	0	0	0	1	0	0	0	0	0	104

TABLE XIV
CONFUSION MATRIX FOR THE TESTED MODEL HOWL ON GOOGLE SPEECH COMMANDS

	silence	_unknown_	yes	no	up	down	left	right	on	off	stop	go
silence	0	0	0	0	0	0	0	0	0	0	0	0
unknown	0	0	0	0	0	0	0	0	0	0	0	0
yes	0	0	3186	0	0	1	3	0	0	0	0	1
no	0	0	3	3034	0	2	81	0	3	0	0	6
up	0	0	0	0	2846	0	4	0	10	20	5	0
down	0	0	1	8	0	3041	0	0	0	0	0	4
left	0	0	3	0	0	0	2955	3	0	0	1	1
right	0	0	0	1	0	0	1	2942	0	0	0	2
on	0	0	0	0	9	18	0	0	2989	23	0	0
off	0	0	0	0	25	0	0	0	23	2873	0	0
stop	0	0	0	3	3	1	0	0	0	1	3058	0
go	0	0	0	10	1	0	0	1	2	0	1	3018

D. SpeechCmdRecognition

The analyzed Attention RNN model exhibits significant variability in performance depending on the dataset on which it is evaluated. As shown in Table XV, the model achieves high accuracy both on the reference datasets described in the original documentation (93.9%) and on the dataset prepared for this study (93.53%). At the same time, a noticeably lower accuracy was obtained on the original Google Speech Commands dataset (78.45%). This discrepancy indicates that although the model demonstrates strong generalization, its performance is sensitive to the data-acquisition method and the acoustic characteristics of the recordings.

TABLE XV
ACCURACY COMPARISON OF ATTENTION RNN MODELS

Model	**Acc on Google**	**Acc. in doc**	**Acc. on our dataset**
Attention RNN	0.7845	0.9390	0.9353

The confusion matrix (Table XVI) clearly demonstrates that the model performs very well in recognizing commands with distinct and easily distinguishable acoustic signatures, particularly those such as yes, no, stop, go, on, off, left, and down. These classes exhibit very high values along the diagonal of the matrix (exceeding 360 correct predictions, and even surpassing 400 for stop and yes), which indicates that the attention mechanism is effectively utilized to capture their characteristic phonetic features. The fact that the model achieves high accuracy on both the documentation-based datasets and our custom dataset, yet performs worse on the Google dataset, suggests that acoustic consistency between the training and testing data plays a significant role. The original Google dataset is characterized by substantial variability in recordings, including differences in microphones, recording environments, and speaker accents. In contrast, the data used for retraining were considerably more homogeneous. This outcome indicates that the Attention RNN model may require additional generalization techniques, such as more extensive data augmentation or transfer learning, to achieve consistently high performance across a broader range of acoustic conditions.

TABLE XVI
CONFUSION MATRIX FOR THE TESTED ATTENTION RNN MODEL (SELECTED CLASSES)

	yes	no	up	down	left	right	on	off	stop	go
yes	409	0	0	0	0	0	0	0	1	0
no	0	379	0	0	1	0	0	0	1	0
up	0	0	386	0	0	0	0	17	3	0
down	0	2	0	378	0	0	1	5	0	7
left	6	1	0	0	388	1	0	0	0	2
right	0	2	0	0	2	377	1	1	2	1
on	0	1	4	1	0	0	368	6	0	3
off	0	1	8	0	0	0	6	373	0	3
stop	0	0	1	0	1	1	1	1	404	0
go	0	4	3	6	0	1	1	0	0	374

TABLE XVII
CONFUSION MATRIX FOR THE TESTED MODEL ON GOOGLE SPEECH COMANDS (SELECTED CLASSES)

	yes	no	up	down	left	right	on	off	stop	go
yes	409	0	0	0	0	0	0	0	1	0
no	0	379	0	0	1	0	0	0	1	0
up	0	0	386	0	0	0	0	17	3	0
down	0	2	0	378	0	0	1	5	0	7
left	6	1	0	0	388	1	0	0	0	2
right	0	2	0	0	2	377	1	1	2	1
on	0	1	4	1	0	0	368	6	0	3
off	0	1	8	0	0	0	6	373	0	3
stop	0	0	1	0	1	1	1	1	404	0
go	0	4	3	6	0	1	1	0	0	374

VII. CONCLUSIONS

The primary goal of this research was to gather, compare, and evaluate papers describing the achievement of the best results with the use of the Google Speech Commands dataset, which also share their repositories with the general public. A task the authors once thought to be easy, proved much more challenging at the end. A valuable resource that was the 'papers with code' website ceased operations couple of months before the start of this project. With the help of the 'wayback machine' internet archive the team managed to recover some data, yet its quality still left a lot to be desired. A total of 42 entries used to exist claiming their achievements on the website, yet at the end only 12 of them proved operational.

Additionally, some of those led to a single repository and research paper, thus narrowing the list of frameworks to 4 ([14], [13], [15], [16]. Among the papers recovered that the team did not manage to test, there are many reasons for the inability to verify the authors claims: the lack of a linked repository, wrong or not archived link, or simply the repository not working – either due to errors or even being unrelated to the task.

There was a possibility, that the models presented in the ranking based on accuracy may turn out to be overfitted and yield low results when tested on a dataset different to the one used in training process. This for the most part was not the case, as test results presented similar or sometimes even higher accuracy to the one claimed in the ranking. Nevertheless, the results were not completely devoid of anomalies. All frameworks were tested and compared with both the original Google Speech Commands dataset and an extension prepared by the authors of this paper.

The first framework, 'ML KWS for MCU' [13], included 7 different network architectures with 3 sizes (small, medium and large). All of them yielded very similar results on the original dataset, with a 1 percentage point difference between the test and the claim. Interestingly, the tests using the extension were, for the most part, better than with the original by 2-3 points, with one notable exception of the DNN model, which yielded much lower accuracy, by almost 30 percentage points. This suggests overfitting and disqualifies the model.

The second model, 'Keyword Transformer' [14], was the most stable one. It consisted of 4 models, 3 versions of KWT and MHAtt-RNN. All of them claimed around 98% accuracy in the documentation, and all yielded above 99% accuracy with both the original Google Speech Commands and the extension, with the KWT-2 model reaching 100% accuracy on the extension. This proved to be the best resource, corroborating its high ranking on the 'papers with code' list (fifth on the leaderboard).

The third model, 'Howl' [15], did not provide a pretrained model and the team needed to train the model. It was trained only using the original dataset and ti did in fact learn very well. The validation results on both the original and extension datasets had nearly 100% accuracy. This confirmed the quality of the proposed framework and its high ranking – fourth on the leaderboard, and the highest among those tested in this project.

The final model, 'SpeechCmdRecognition' [16], was the most odd among those tested within this project. It claimed around 94% accuracy, but while it yielded a very similar result on the teams extension, it did much poorer on the original, yielding only around 78%.

The project described in this paper highlights the importance of proper benchmarking and the need for dedicated spaces to compare research results. As new classification methods continue to emerge —- and as entirely new paradigms appear -— there must be a framework for evaluating their performance against existing approaches. One such paradigm is quantum computing, a rapidly developing technology that has given rise to quantum machine learning algorithms. These algorithms have already attracted interest in the niche area of speech recognition [17]. In the future, the team intends to explore this growing field as well, which will require reliable comparisons with current state-of-the-art. Several initial approaches have already been investigated, including a formant-based quantum neural network for vowel recognition, in which formant features were extracted using a quantum annealing framework. In future work, the team plans to explore hybrid quantum–classical architectures, particularly models combining classical convolutional layers with quantum kernel methods, as well as classical autoencoders for dimensionality reduction prior to quantum processing. Considering the initial findings reported in the literature, it is also essential to evaluate the proposed approaches on real quantum hardware, as well as in simulated environments that accurately emulate the noise characteristics of current NISQ-era machines.

REFERENCES

[1] "Talk to your samsung family hub with voice assistants," https://www.samsung.com/us/support/answer/ANS00062803/, accesed: 14.10.2025.

[2] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[3] Y. Wei, Z. Gong, S. Yang, K. Ye, and Y. Wen, "Edgecrnn: an edge-computing oriented model of acoustic feature enhancement for keyword spotting," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 3, pp. 1525–1535, 2022.

[4] M. Zeng and N. Xiao, "Effective combination of densenet and bilstm for keyword spotting," *IEEE Access*, vol. 7, pp. 10 767–10 775, 2019.

[5] R. Vygon and N. Mikhaylovskiy, "Learning efficient representations for keyword spotting with triplet loss," in *International Conference on Speech and Computer*. Springer, 2021, pp. 773–785.

[6] B. Kim, S. Chang, J. Lee, and D. Sung, "Broadcasted residual learning for efficient keyword spotting," *arXiv preprint arXiv:2106.04140*, 2021.

[7] S. Chang, H. Park, J. Cho, H. Park, S. Yun, and K. Hwang, "Subspectral normalization for neural audio data processing," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 850–854.

[8] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha, "Temporal convolution for real-time keyword spotting on mobile devices," *arXiv preprint arXiv:1904.03814*, 2019.

[9] C.-H. H. Yang, J. Qi, S. Y.-C. Chen, P.-Y. Chen, S. M. Siniscalchi, X. Ma, and C.-H. Lee, "Decentralizing feature extraction with quantum convolutional neural network for automatic speech recognition," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6523–6527.

[10] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[11] N. M. Müller, P. Czempin, F. Dieckmann, A. Froghyar, and K. Böttinger, "Does audio deepfake detection generalize?" *Interspeech*, 2022.

[12] "P.800 : Methods for subjective determination of transmission quality," https://www.itu.int/rec/t-rec-p.800-199608-i, accesed: september 2024.

[13] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.

[14] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword transformer: A self-attention model for keyword spotting," *arXiv preprint arXiv:2104.00769*, 2021.

[15] R. Tang, J. Lee, A. Razi, J. Cambre, I. Bicking, J. Kaye, and J. Lin, "Howl: A deployed, open-source wake word detection system," *arXiv preprint arXiv:2008.09606*, 2020, arXiv:2008.09606 [cs.CL].

[16] D. C. de Andrade, S. Leo, M. L. Da Silva Viana, and C. Bernkopf, "A neural attention model for speech command recognition," *arXiv preprint arXiv:1808.08929*, 2018.

[17] M. A. Kucharski, "A survey on quantum machine learning in speech acoustics," *IEEE 25th International Symposium on Computational Intelligence and Informatics (CINTI)*, 2025, accepted.