Tiny Neural Network Pipeline for Vocal Commands Recognition @Edge

Ivana Guarneri[1], Alessandro Lauria[2], Giovanni Maria Farinella[2] and Corrado Santoro[2]
[1]*STMicroelectronics, System Research and Applications, Stradale Primosole 50, Catania, Italy*
[2]*Dipartimento di Matematica e Informatica, Università degli Studi di Catania, Catania, Italy*

Keywords: Speech Recognition, Deep Learning, Edge-AI.

Abstract: Vocal analysis and speech recognition have been a challenge for the research community for a long time. The widespread of deep learning approaches, the availability of big datasets and the increasing computational capabilities of processors, have contributed to achieve disruptive results in this field. Most of the high performing existing speech recognition systems are available as cloud services. Other systems are hybrid, with some parts on the cloud and some modules running on the microcontroller. One of the challenges is to realize high performing speech recognition systems running on the edge, where the edge is an integrated platform, composed by a processing unit, a bank of memory and a power unit. In this paper is proposed an end-to-end deep learning approach to recognize a set of vocal commands able to work on an edge IoT node. Tests have been performed on a tiny platform and the study with different settings is reported.

1 INTRODUCTION

Speech recognition is nowadays one of the most powerful and used application of artificial intelligence. Speech recognition engines reach high accuracy thanks to the exploitation of high performing hardware platforms coupled with complex deep neural architectures and the availability of large datasets. In the context of vocal analysis many improvements have been done thanks to the spread of new sophisticated deep learning techniques compared on the cloud. However, less attention has been given to automatic speech recognition (ASR) solutions to be employed on hardware platforms with limited capacities in terms of power processing and storage capacity. Deep neural network models need both a big amount of flash memory to store parameters and a high-power processing capability.

In this paper we focus on the recognition of a set of vocal commands with the goal of developing a deep learning solution to be executed directly on a microcontroller, i.e., a platform with limited resources in terms of memory and processing capability.

Most of the available works on vocal commands recognition with microcontrollers take inspiration from latest performing deep learning approaches used in the state of the art of ASR (Solovyev, 2018).

Being ASR on a microcontroller the main target, the classical approaches are usually resized and modified trying to develop tiny deep learning architectures able to guarantee good performances.

Interesting works are available in the literature. In (Zhang, 2018) the authors propose solutions for the keyword spotting problem. They trained different neural network architectures and compared the obtained results together with the memory and computing requirements. The study explains how to optimize the neural network architectures for a good fitting in the limited memory and how to obtain real time execution on the microcontroller without sacrificing the keyword spotting performances.

In (McGraw, 2016) the authors present a compact speech recognition system with large vocabulary able to run on mobile devices with low latency and good performances. To reach this result they used a Connectionist Temporal Classification (CTC) based Long Short Term Memory (LSTM) compressed model.

In (Sainath, 2015) the author explores a Convolutional Neural Network (CNN) for a small-footprint keyword spotting task. The effectiveness of the approach is demonstrated on two different use cases: in the first one the number of multiplications is

Guarneri, I., Lauria, A., Farinella, G. and Santoro, C.
Tiny Neural Network Pipeline for Vocal Commands Recognition @Edge.
DOI: 10.5220/0010908800003124
In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 2: HUCAPP*, pages 249-254
ISBN: 978-989-758-555-5; ISSN: 2184-4321

limited, and, in the second one, the number of parameters is limited. The result of the comparison shows that CNN architectures give an improvement in the range 27-44% relative to false reject rate compared to other Deep Neural Networks (DNN).

This paper proposes a solution for the automatic recognition of vocal commands to be executed on the edge. The considered edge platform is equipped with a processing unit, a bank of memory, a unit power, some sensors and connectivity. The developed solution runs directly on the microcontroller by processing data coming from the sensors which are also placed on the same board.

Hence, the edge computing processes data directly on the board where they are generated. This aspect is important also for a privacy point of view and for ensuring a lower power consumption.

It is worth noticing that when an algorithm runs on the cloud, data is usually transferred to and from a server. During the stream of data, the system is vulnerable, due to possible attacks. The typical data flow starts from an application running on a platform placed in the real world that captures data through its sensors. These data are used to take a decision or, in general, to generate a possible activation. Data is hence uploaded from sensors to the server. The output of the processing is sent back to the application. With this pipe, a processing unit running on the cloud can be attacked with possible private risks for the users.

An edge processing unit no need to transfer data. It ensures the privacy of data and the energy consumption is contained due to lower power consumption of an IoT node which is lower than the one of the Cloud.

We performed a study by means of a set of experiments that considered a deep learning algorithm able to classify 10 different speech commands. The solution has been tested on a very low constrained resourced IoT node named SensorTile (STMicroelectronics, 2019) produced by STMicroelectronics. This platform is equipped with a STM32L4 microcontroller running at 80MHz, 1MB of flash memory, 128KB of RAM and a digital microphone.

The paper is organized as follows. Section 2 describes the datasets used to train and test the proposed neural network system and reports the analysis done on the considered dataset. In Section 3, the details on the proposed method for short commands recognition are reported together with a summary of the results obtained by running the proposed algorithm on an STM32 microcontroller. Conclusions are given in Section 4.

2 DATASETS

The dataset used in this work has been populated by using data from two main repositories: Google Speech Commands Dataset (Warden, 2018) and Hey Snips Dataset (Coucke, 2019).

The Google dataset contains 30 different words for a total of 65.000 audio files of one-second each. Each word has about 2000 samples. The dataset is released under Creative Commons BY 4.0 and it is constantly updated by the community. Words are pronounced by different speakers (both male and female subjects), and using microphones placed at various distances.

The Snips dataset is a Natural Language benchmark dataset, it contains a large variety of English accents and recording environments. It is composed of about 11K wake-word utterances and 86.5K (~96 hours) negative examples.

The dataset used to train and test the proposed solution has been firstly arranged in four classes as reported in Table 1. The “Comms” class contains data from Google dataset and it is relative to ten vocal commands: go, stop, left, right, up, down, on, off, Marvin and Sheila. These commands are single words preceded and followed by silence. The other words available in the Google dataset have been used to represent the class “Words”.

Part of the 86.5K negative examples from Snips’ dataset has been used to populate the class “Talk” with natural language phrases. The class “Background” contains audio relative to not human speaking, such as silence and environmental noise taken from youtube. The set of “Comms” in Table1 is aligned with the one used in other works presented in literature as, for example, in (Zhang, 2018) or in (Gouda, 2020); differently to these works we substituted “yes” and “no” with “Marvin” and “Sheila” because, in our future works, these names will be used to wake up robots to be piloted with the proposed vocal commands recognition system.

Table 1: Dataset organized in four audio classes.

Classes	Description
Comms	10 vocal commands of Google dataset: go, stop, left, right, up, down, on, off, Marvin, Sheila
Words	Words different from the 10 selected commands of Google dataset
Talk	Continuous speaking sampled by Snips’ dataset
Background	Silence and noise environments sound sampled from youtube

The first three classes shown in Table 1, have been partitioned through a K-Means processing (Xu, 2020) as proposed in Figure 1.

K-Means is a classic algorithm to perform partition data from a semantic point of view. In this case it is used to perform a partition of the audio features for each class. We set K=2 to split in two clusters each class. The Background class has not been processed with K-Means because it includes homogeneous files. Figure 1 shows the dataset's folders organization before and after the K-mean processing.

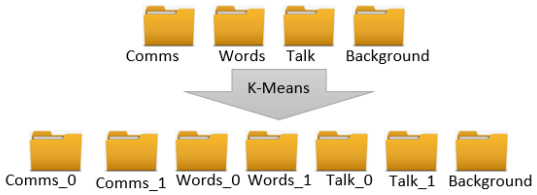

Figure 1: Dataset before and after the K-mean processing.

3 PROPOSED SYSTEM

3.1 Tiny Networks Pipeline

The proposed vocal command recognition system is configured as a cascade of two networks as shown in Figure 2.

The "First Net" is trained and tested with 7 classes reported in Figure 1: Background, Comms_0, Comms_1, Talk_0, Talk_1, Words_0 and Word_1.

The "Second Net" is trained and tested with 11 classes: ten vocal commands plus a filler class containing single words different from the ten commands.

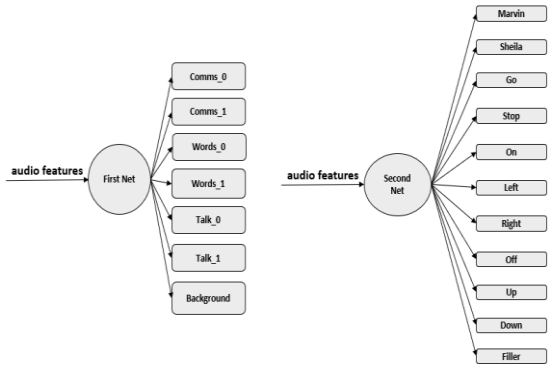

Figure 2: First and Second Network.

The selected 10 commands are: Marvin, Sheila, go, stop, left, right, up, down, off.

The interaction between the two nets is handled by a state machine. We consider two main states, S_0 and S_1. The execution of the First Net identifies the state. If the inference output of the First Net is the comms_0 or the comms_1 class, then the considered state is S_1. In all other cases the state S_0 is considered. The Second Net is executed only when the output of the First Net determines the state S_1. In state S_0 the MFCC matrix is updated by following the FIFO order and maintaining the system in a low power mode profile.

Figure 3 shows the general diagram of the entire system.

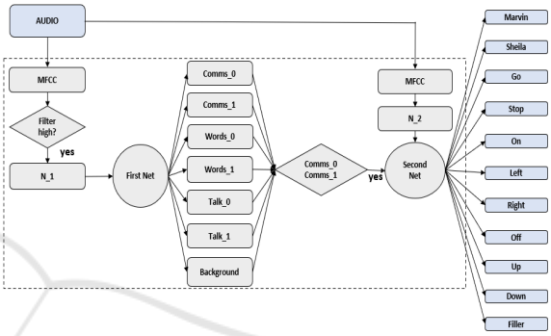

Figure 3: Vocal command recognition system diagram.

The input to the whole system is the audio stream captured by the microphone of the SensorTile. The output is one of the ten vocal commands or the Filler class.

The input of both nets are the Mel Filter Cepstral Coefficients (MFCC) that are classic audio features (Hasan, 2021) (Albadr, 2021). The MFCC coefficients are organized in a matrix 13x28 that is continuously updated with new columns that are added to the queue according to a FIFO (First In First Out) order.

Before the training step, the audio features are normalized according to the equation (1) to have a data distribution with zero mean and standard deviation equal to one.

$$x' = \frac{x - \mu}{\sigma} \quad (1)$$

were x is the original feature vector, μ and σ are the mean and the standard deviation of the training set respectively. Normalization is useful to speeds-up the convergence of the neural model during the training.

The computed (μ,σ) are used to normalize the audio features processed in the inference step. The proposed system includes two neural models trained on different datasets, consequently two different sets of parameters are used to normalize the input of the

two networks. The N_1 normalizes the MFCC in input of the First Net by applying the (μ_1, σ_1) values obtained from the dataset used to train the First Net.

Similarly, the N_2 normalizes the MFCC in input to the Second Net by using (μ_2, σ_2) obtained from the dataset used to train the Second Net.

The audio signal has high energy when people are speaking and it has low energy when there is silence, to discriminate these two different cases a filter is applied on audio features as pre-processing step.

This filter evaluates the temporal average value of MFCC zero coefficients (Guarneri, 2019) and this value is used as measure of the energy signal. The higher the energy, the higher the probability that a word has been pronounced.

When the output of the filter is high, the audio frames are firstly normalized by the N_1 and then are processed by the First Net. The Second Net is hence triggered and processes the audio after the normalization step with N_2 only if the output of the First Net is classified as Comms_0 or Comms_1 (state S_1).

The output obtained by the Second Net is the output of the overall system.

3.2 Embedding on Microcontroller

The proposed voice command recognition based on a cascade of networks has been selected after analysing different networks configurations and by taking in consideration the HW resources of the target edge platform. More specifically, the recognition engine has been designed to be executed on a SensorTile board (Figure 4) which is a tiny, square-shaped IoT module that packs processing capabilities, leveraging an 80 MHz STM32L476JGY microcontroller and Bluetooth low energy connectivity based on the BlueNRG network processor as well as a wide spectrum of motion and environmental MEMS sensors. The board includes a digital microphone. The SernsorTile memory capability is 1MB of FLASH and 256 KB RAM. SensorTile can fit snugly in the users IoT hub or sensor network node to become the core computing platform of the developed solution.

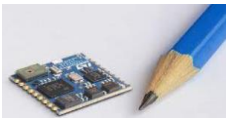

Figure 4: SensorTile IoT node.

In edge computing it is fundamental to find the right trade-off between the complexity of the solution and its acceptable performance. In this scenario the proposed system has been tuned through several tests by considering the memory and power processing limitations of the SensorTile module.

The two-nets final system, obtained after reducing its size and complexity, is constituted by two Recurrent Neural Networks (Cases, 2019) with a Long Short-Term Memory architecture (Palangi, 2016) (Sak, 2014).

The two LSTMs have different sizes, the first net includes two hidden layers each of 32 units, followed by a Fully Connected layer with 7 neurons.

The second network is quite bigger than the first one including two hidden layers each of 64 units, followed by a Fully Connected layer with 11 neurons.

The input of both nets is a 13x28 matrix of audio features.

The memory requirements and the complexity of the proposed cascade architecture has been estimated by using the X-CUBE-AI (STMicroelectronics, X-Cube-AI) expansion package part of the STM32Cube.MX ecosystem (STMicroelectronics, STM32Cube.MX).

This tool automatically optimizes pre-trained networks and integrates the generated optimized library into the user's project. The X-CUBE-AI Expansion Package also offers the possibility to validate artificial intelligence algorithms both on desktop PC and STM32 microcontrollers. It is also possible to measure performance on STM32 devices without user handmade ad-hoc C code. As reported in Table 2, the required memory of the proposed system is 0.08MB which fits in the SensorTile' FLASH.

Table 2: Memory requirements of the system and of the target edge platform.

Memory requirements of the cascade system	0,08 MB
SensorTile FLASH	1 MB

3.3 Tests on Microcontroller

To evaluate the performances in terms of accuracy, the proposed cascade system is compared with a baseline composed by a single network (Single Net). The single network is built trying to obtain a size comparable to the cascade network. In this way the two different systems, requiring the same amount of memory, can be properly compared in terms of accuracy and power saving capability. The Single Net is constituted by a LSTM architecture with 4 layers followed by a Fully Connected layer of 11 neurons. The first two layers are 16-units LSTM, while the other two are 32-units LSTM.

Details regarding memory requirements and complexity expressed in terms of multiply-accumulate (MAC) operations, are reported in Table 3 for the compared systems.

Data related to the proposed cascade approach are obtained by summing the relative MAC and FLASH values of the First and Second nets composing the system.

The proposed approach and the compared baseline based on a single net are very similar in terms of complexity and memory requirements.

Table 3: Complexity and memory requirements of the proposed cascade approach and of the Single Net.

APPROACH	MAC	FLASH
Cascade approach	515649	80,55 KB
Single Net	525795	79,68 KB

The two networks have been both trained on 11 vocal commands. The accuracy for each class is reported in Table 4

Table 4: Accuracy related to the 11 vocal commands.

	Cascade Net	Single Net
Marvin	85 %	64%
Sheila	93 %	81 %
Left	83 %	61%
Right	86 %	65 %
Up	71 %	50 %
Down	75 %	59 %
Go	80 %	48 %
On	78 %	63 %
Off	73 %	74 %
Stop	81%	67 %
Filler	78 %	76%
Avg	80,3 %	70,1 %

Results illustrate how the cascade-based system is more accurate than the Single Net.

The advantage of the proposed solution is not only relative to a higher accuracy but also to its contribution to the power saving aspect.

A first contribution comes from the front-end processing of the Cascade Net system. The neural network inferences are performed only if the filter on MFCC zero coefficients estimates a high energy in the audio signal; the filter switches its output from low to high level when a change from a noiseless environment to a noisy one is detected. This means that in a quiet environment the inferences are not executed, and the system stays in a low power consumption mode.

The second contribution comes from the handling of the two states S_0 and S_1 described in previous sections.

The state machine changes the system's status from S_0 to S_1 only if the output of the First Net of the cascade system is classified as a command, otherwise it remains in S_0 till the filter output is high. Considering that the audio stream content is mostly noise environment, continuous speaking or isolated words different from the few 10 selected commands, the system is for most of the time in the state S_0; in this state only the First Net (which is smaller than the Second Net) is executed and the MCU overall system workload is reduced.

The proposed cascade mechanism not only reaches a higher accuracy with respect to a Single Net architecture, but it is also a lower power solution.

Future tests will be oriented to increase the robustness of the system to false positives. To reach this goal a retraining can be done by including data taken from the heterogeneous Multilingual Spoken Words Corpus (MSWC) dataset. The MSWC is a speech dataset of over 340,000 spoken words in 50 languages, with over 23 million audio examples (Mazumder, 2021). The objective of the retraining is to obtain a model with higher capability to generalize and to handle differences in accents or dialects.

4 CONCLUSIONS

In this paper a tiny neural network pipeline able to classify vocal commands is proposed. The approach is based on a cascade pipeline which results in a power saving solution suitable to be executed on a tiny edge platform with low memory and power processing resources.

The solution has been mapped on the STM32L4 microcontroller and a comparison with a baseline architecture composed by a single network has been done. Results show that the proposed method obtains a higher accuracy.

Also, the cascade network allows to reach high accuracies by ensuring a lower MCU workload.

REFERENCES

Solovyev, A., R., Vakhrushev, m., Radionov, A., Aliev, V., Shvets, A., A. (2018). Deep Learning Approaches for Understanding Simple Speech Commands. In *arXiv 2018*

Zhang, Y., Naveen, S., Lai, L., Chandra, V. (2018). Hello Edge: Keyword Spotting on Microcontrollers. In *arXiv 14 Feb 2018*.

Gouda, S., K., Kanetkar, S., Harrison, V., Warmuth, M., K. (2020). Speech Recognition: Key Word Spotting through Image Recognition. In *arXiv 24 Nov 2020*.

McGraw, R., Prabhavalkar, R., Alvarez, R., Arenas, M. G., Rao, K., Rybach, D., Alsharif, O., Sak, H., Gruenstein, A., Beaufays, F., Parada, C. (2016). Personalized Speech Recognition on Mobile Device. In *arXiv 11 Mar 2016*.

Sainath, T. N., Parada, C. (2015). Convolutional Neural Networks for Small-footprint Keyword Spotting. In *Sixteenth Annual Conference of the International Speech Communication Association, 2015*.

STMicroelectronics (2019). STEVAL-STLKT01V1 *Data brief -SensorTile development kit*

Warden, P. (2018) Speech commands: A dataset for limited-vocabulary speech recognition. In *arXiv preprint arXiv:1804.03209, 2018*

Coucke, A., Chlieh, M., Gisselbrecht, T., Leroy, D., Poumeyrol, M., Lavril, Thibaut. (2019) Efficient Keyword Spotting Using Dilated Convolutions and Gating. In *arXiv, 18 Feb 2019.*

Muda, L., Begam, M., Elamvazuthi, I. (2010). Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques. In *Journal of Computing, volume 2, issue 3, March 2010, ISSN 2151-9617*

Xu, H., Yao, S., Li, Q., Ye, Z.. P. (2020). K-Means Clustering and Related Algorithms. In *IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS)*

Hasan, R., Hasan, M., Hossain, Z. (2021). How many Mel-frequency cepstral coefficients to be utilized inspeech recognition? A study with the Bengali language. In *The Journal of Engineering, 4 September 2021*

Albadr, M.A.A., Tiun, S., Ayob, M. et al. (2021) Mel-Frequency Cepstral Coefficient Features Based on Standard Deviation and Principal Component Analysis for Language Identification. (2021). In *Systems. Cognitive Computation 13, 1136–1153 16 July 2021*

Mazumder, M., Chitlangia, S., Banbury, C., Kang, Y., Ciro, J., Achorn, K., Galvez, D., Sabini, M., Mattson, P., Kanter, D., Diamos, G., Warden, P., Meyer, J., Reddi, V., J. (2021). Multilingual Spoken Words Corpus. In *35th Conference on NeurIPS Track on Datasets and Benchmarks, 2021*

Cases, I., Rosenbaum, C., Riemer, M., Geiger, A., Klinger, T., Tamkin, A., Li, O., Agarwal,S., Greene, J. D., Jurafsky, D., Potts, C., Karttunen, L. (2019). Recursive Routing Networks: Learning to Compose Modules for Language Understanding. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019*

Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X., Ward, R. (2016). Deep sentence embedding using long short-term memory networks: analysis and application to information retrieval. In *IEEE/ACM Transactions on Audio, Speech and Language ProcessingVolume 24Issue 4April 2016*

Sak, H., Senior, A., Beaufay, F. (2014). Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. In *Interspeech.2014-80*

Guarneri, N. I. (2019). Trigger to Keyword Spotting System. *United States Patent Application 20210065689.*

STMicroelectronics, X-Cube-AI. In *X-CUBE-AI - AI expansion pack for STM32CubeMX - STMicroelectronics*

STMicroelectronics, STM32Cube.MX. In *STM32CubeMX - STM32Cube initialization code generator - STMicroelectronics.*