Small footprint keyword detection: study and extention of the neural attention model

Andrea Sipka[†], Nicoló Ruggeri[‡]

Abstract—**Keyword spotting has been an increasingly studied problem in the recent years due to the growing number of applications. The use of keyword spotting in portable devices developed by most major technology companies (Google, Amazon, Apple, amongst others) incentivised the contribution of academic and technical community. In addition, a benchmark dataset was recently open sourced by Google, making results comparison standardised through different papers and architectures.**
A major difficulty in developing models appropriate for this task is the need to keep the computational and memory cost appropriately low, due to the requirement to process data in place, within a low memory setting. Even though other approaches are possible, like in-device sound detection with cloud processing after activation, the need for compact, low footprint architecture is compelling in many use cases.
In line with some of the recent advances in the field, this paper aims to study one of the architectures proposed recently, trying to improve it from a memory and computational footprint and classification performance points of view. We succesfully improved in both directions and all of our models are available on our github page. [1]
Beyond its state-of-the-art performances, the selection of the model has been based on the versatility of it, as well as for the two main ideas that it exploits: recurrent neural networks and attention mechanism. For this reason, by methodically studying effects of architectural modifications on this model, we attempt to draw some functional conclusions about the role of layers in the network itself. This is in line with attempt of the technical community to tackle the problem of the black-box nature of some machine learning models, present within majority of deep learning approaches, which is considered one of the major drawbacks in regards to the usage of neural networks, and go beyond just prediction and towards understanding of the models.

Index Terms—**keywork detection, speech detection, recurrent networks, attention, low cost models**

I. Introduction

Small footprint keyword spotting is a task that has seen a growing importance and interest within technical studies in recent times. In addition to the growing use of mobile devices beyond smartphones (such as smart watches) and increasing use of small devices within the environment (in context such as smart homes, cars, offices), the field of human-computer interaction has indicated that the addition of alternative sensory channels such as voice not only enables the use of technology within constrained environments or in contexts where alternative inputs aren't possible, but it also creates a richer experience for the user in general.

While applications and use cases may seem obvious, the development of a small footprint architecture is not as much. It is due to this that decrease in computational and memory footprint, and not just increase in performance, is of particular interest.

The main approaches that have been used lately to tackle these issues have some common characteristics, both from a feature extraction and learning algorithm point of view. Our work focuses on building upon previous architectures to improve them in either direction. In particular, we focused our study on variants of the architecture proposed in [1].

Previous noted works in the area of deploying small footprint architectures have addressed the problem using Deep Neural Networks [2], or alternatively by developing the approach by introducing Convolutional Neural Networks [3]. The latest developments in the direction of improving the performances of the model can be found in [4], presenting the introduction of residual networks and dilated convolutions, and in [1], with the deployment of an attention mechanism combined with CNNs.

The advantage of the attention mechanism deployment goes beyond a purely technical perspective, as it is in line with current efforts of the technical community to improve the transparency of how learning models arrive to their decisions by gaining insights into which aspects of the data (in this case, the raw audio file) are considered, and to which extent. In this paper, we aim to further study and extend the capabilities of the architecture presented in [1], in three directions:

- Firstly, we study the tradeoff between the reduction of the number of parameters/flops and the accuracy achieved, in line with the reasoning of [2] and [3].
- Secondly, we attempt to improve the performance against the benchmark dataset by tweaking the (hyper)-parameters of the network.
- Finally, by analyzing a number of networks that we trained, we try to also give a functional interpretation of the various layers in the net, carrying on the inspection from the previous papers, which allowed the study of the attention mechanism by means of partially non black-box weights in the attention region.

This paper is organized as follows: in Section 2 we delve deeper into the previous approaches which relate to our study, in Section 3 we present a technical specification of our learning pipeline, and in Section 4 we present comparative results and give some final remarks.

[‡]Padua University, email: nicolo.ruggeri.1@studenti.unipd.it
[†]Padua University, email: andrea.sipka@studenti.unipd.it
[1]https://github.com/NRuggeriProjects

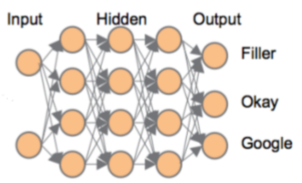
II. Previous Work

The three main approaches we are analysing and building upon are Deep neural network approach presented in [2], Convolutional neural network approach as presented in [3] and Recursive layers and attention mechanism approach as developed by [1].

In regards to feature extraction, all three follow a line of utilising log-mel filterbanks followed by frame stacking on the left and right of the frame being processed. Different sizes and windows have been utilised, with tendency to tailor the approach to the best trade off in regards to accuracy, latency and computational expense.

A. Deep Neural Network Approach

Fig. 1: Simple DNN



The first natural approach for audio file classification is the one presented in [2], using a simple Deep neural network for keyword spotting with a very low computational and memory cost.

It is to be noted that the task in hand was classification of continuous sound, which does not have an impact on the actual architecture, but is relevant for the preprocessing of the posterior probabilities, which are smoothed.

The approach, while showing considerable improvement comparing to the previous, Hidden Markov Model based solution, does not exploit any of the recursive and organised structure characterising the data, which is what much of further work in the area has built upon.

B. Convolutional Neural Network Approach

The consequent work was the deployment of convolutional layers from [3], which are able to capture the correlations among different features, while still preserving a very low cost in terms of computational and memory resources. The study from the reference paper focuses on deploying various architectures, all following the same structure: one or two convolutional layers, one or two fully connected layers and a softmax activation. The advantage of this approach is the ability to exploit the structured nature of the data, which is reflected in the improvement of the classification score.

C. Recursive layers and attention mechanism approach

The final previous approach considered here, and presented in [1], employs, after the extraction of information by convolutional layers, two major additions: the use of recursive layers, that further exploit the sequential structure of the data, and the introduction of an attention mechanism after all the information extraction. The general architecture can be seen on Figure 2.

This model presents the following advantages:

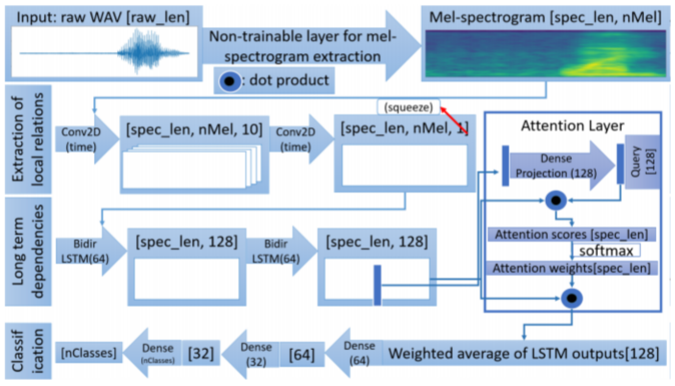


Fig. 2: Architecture presented in [1]

- An ordered structure with logical groupings, which allows for experimenting on different parts of the net with possible functional interpretations. Moreover, many adjustments of subcomponents are possible, allowing room for further improvements.
- The attention mechanism is easily interpreted by plotting the weights in correspondence of the mel-spectrogram. This is a good step forward towards the elimination of completely black-box models, which is always an issue when interpreting results of and attempting to improve any neural network architecture that doesn't allow this type of insight.
- The model embeds the convolutions used in previous ones, allowing for improvements and higher level representation of the input data.

III. Learning Framework

A. Approach

All of the models presented begin with the same preprocessing of the data, consisting of a 40 dimensional log-mel filterbanks computed every 10ms over a rolling window of 25ms. Frames are then stacked on the left and right of the actual one, although number of frames to be stacked varies between implementations. In all of our architectures we are stacking 23 frames on the left and 8 on the right, as was the approach taken by [1].

In regards to the processing stages following feature extraction, we focused on fully studying the architecture proposed in the same paper, and while keeping the pre-processing untoched, we explored alternative solutions and parameters in the following aspects:

- Within the local relations extraction part: the number of convolutions and the number of parameters for each convolution (seen in green in Figure 3).
- For part relating to the long term dependencies detection: the type, number and parameters of the recurrent layers used (seen in yellow in Figure 3).
- The type of information fed into the attention mechanism (seen in red in Figure 3).

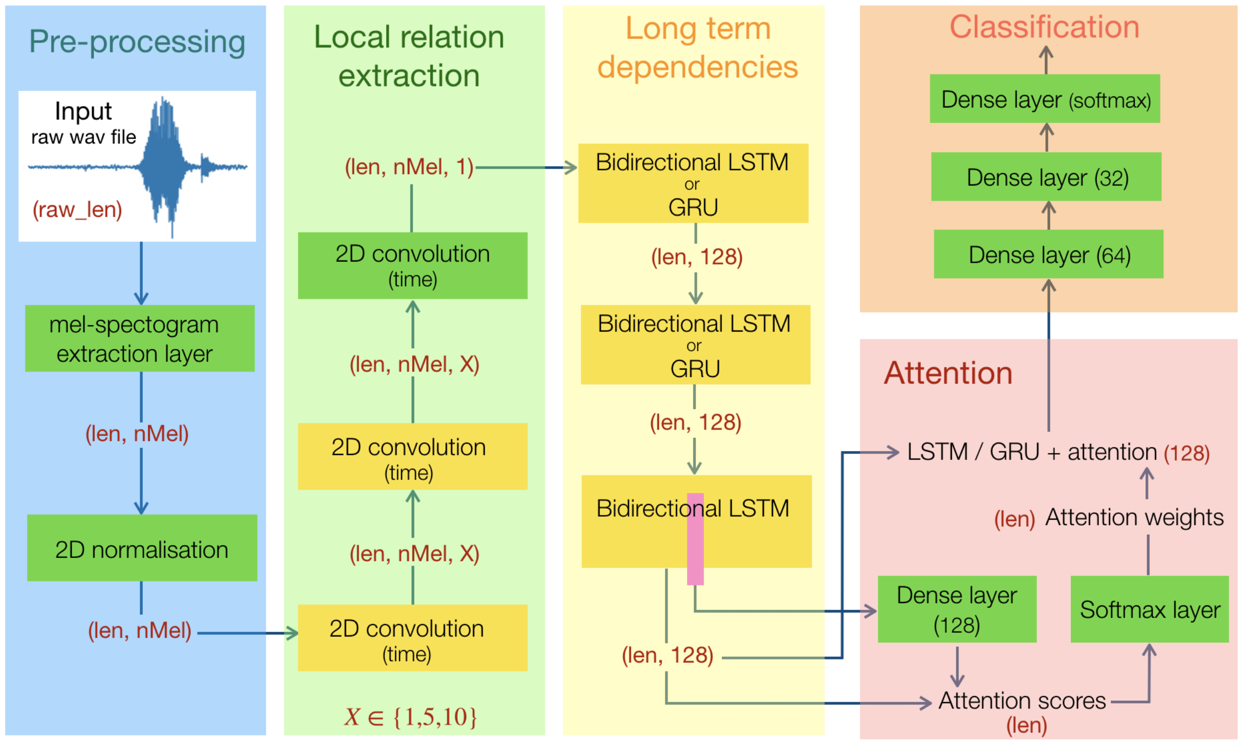


Fig. 3: Overview of the learning architecture proposed

The aim is to reduce the number of parameters and operations and to achieve performance enahncement.
The reasoning behind choice of these three aspects will be explained in detail in the following sections.

B. Complexity Reduction Architectures

1) Recurrent Layer Type: While in the reference paper no mention is given to the recurrent layer type, we believe that this could be of contribution in order to better understand the necessity to exploit long term relations in the data. What we did was to try, instead of the proposed Long Short-Term Memory (LSTM) layer as discussed in [5], the lighter Gated Recurrent Unit (GRU) proposed in [6]. While this is of interest also due to the fact that it leads to the reduction of the computational and memory cost, we are mostly interested in the importance of long term information for the final calssification. In fact, LSTM is generally regarded as a better model when dealing with distance information in time with respect to GRU units, as discussed in a comparative study between the two in [7]. Within our implementation, we have tested both approaches while leaving the rest of the architecture unchagned in order to compare the two.

2) Attention Mechanism Information: Another point worthy of consideration is the choice of the authors to feed the attention mechanism with the middle vector of the final output, after the two recurrent layers. The reported motivation in [1] is that, since the benchmark dataset consists of audio samples, they should be centered and therefore all the information should be contained in the middle part of the matrix. While this can be verified by plotting the audio samples, and appears to be generally true for the benchmark dataset, just considering this can be limiting when it comes to generalising this model for other data. As it follows from the brief discussion in the Introduction, many current applications of this technology are required in the context of continuous sound keyword detection, and we would like to expand the capabilities of the system towards a more general solution.

Taking this into consideration, one could try to extract more information from the final matrix of shape $[spec\_len, 128]$ with the aim of feeding it to the attention mechanism. There are a few possible solutions to attempt, including cumulative quantiles in time or other statistics computed over the matrix. In this paper, we report results when averaging over all columns. This empowers the model to run also on samples where information is not centered, which has the potential to lead to improvements on different types of datasets, as well as in real life use cases.

C. Results Improvement Architectures

In order to improve the detection capabilities of the models, we have tested different configurations, modifying the number of convolutions, together with their size, as well as the number of recurrent layers. We found that adding new parameters consistently leads to improvements, and we tuned different models that consistently outperformed the one from [1]. More precisely, we tried to increase the number of convolutions, varying also the number of filters, and the number of LSTM layers.

IV. Results

In this section we report results for all of the models presented. To allow for direct comparison with previous approaches, the benchmark dataset used is the Google Speech Dataset V1. It consists of a set of 35 words, each one contained in a single audio file, with various background noise levels. We felt that it is more relevant to use a benchmark with higher number of words as it fits well within the current need for systems capable of understanding a wider variety of words in order to meet user expectations of usability. All models were trained using Adam [8] with learning rate of 0.1, decreased by a multiplicative factor of 0.7 every 10 epochs.

In the Table 1, containing specifications and results, we use the following notation:

- Models' convolutions denoted as (10,10,1) indicate a model with 3 convolutional layers, each with output of 10, 10 and 1 filters. All convolutions use symmetric padding on either side and (5,1) stride. After every convolutional layer, a batch normalization is inserted.
- Models' recurrent layers are denoted as (LSTM, LSTM) to indicate two consecutive LSTM layers. All recurrent layers are bidirectional and with output dimension $64{\cdot}2 = 128$.
- Models using averaging instead of the middle vector for the attention mechanism are denoted with the addition of *'Avg* in the name of the model.
- We give the number of flops (i.e. multiplications between floats) for every model. Since the architectures change only in the number of convolutions and recurrent layers, we just give the computation of operations impacted by these, to enable comparison.

In regards to the alternative approaches for the recurrent layer, a comparison was done by leaving all other parts of the network unchanged and deploying both variations. As can be seen in Table 1, while performances with GRU are a little worse than with LSTM, suggesting that longer term information is valuable in some cases, this option can still be kept in consideration. In fact, while the benchmark dataset consists of very short audio samples, in reality applications of keyword spotting also arise from continuous sound detection, and being able to shorten the past information window may result in lower computational cost and therefore reduced latency. Other suggestions for achieving lower latency could be to reduce the number of right frames in the mel bank, as well as removing bidirectional recurrent layers, which also require future information not immediately available in real time detection.

For the attention mechanism information, experiments have shown that this approach can sometimes be deleterious to the final accuracy. Nonetheless, we made a choice to keep feeding the attention mechanism with the average vector in all of the models trying to achieve complexity reduction, as we feel this results in a more general model which can be applied in wider variety of problems.

As can be seen from the results in Table 1, the most memory demanding substructure of the network is given by the recursive layer. However, this aspect of the optimization is of lesser significance, since the mel spectrogram is made up of more than 1 million parameters, and therefore the observed variation in memory consumption of the memory itself is almost negligible, at least for the devices that need in situ sound pre-processing. On the other hand, devices that adopt a cloud computing approach can usually exploit availability of powerful machines, making performance optimization a minimal issue.

A more interesting aspect is the number of operations that the model needs to handle. In this case the roles are inverted and the majority of operation costs are due to the convolutional layers. Therefore a reduction in the number of convolutional filters can be translated into quasi-linear reduction of the operations. It is reasonable to assume that cutting the number of operations from $1/2$ to $1/5$, obtaining a score difference in the order of 0.1% would be considered an acceptable tradeoff in majority of applications.

In addition to all the source code for all the models explored, we report confusion matrices and attention plots in our GitHub repository.

A. Some functional considerations

Testing and observing the behaviour of the model with respect to any single specific modification or variation, while leaving all other parts untouched, allows us to hypothesize regarding some functional relations for the layers of the model. This can prove to be useful both in the event of further modifications of the neural attention model as presented here and for combining this approach with others, possibly in alternative use cases:

- Reducing the size of convolutions does not seem to lead to significant accuracy problems. Since all local relations are captured at this stage, we could infer that small number of filters succeed in encoding most of the information required for effective classification. Also, as is to be expected, high level representation seems to be important, since two small concatenated convolutional layers seem to work better that a single bigger one.
- The use of a GRU instead of LSTM perceptibly worsened the performance. This suggests that long term relations are useful for the model and cannot be ignored. Notice that while this can work against a possible reduction of the listening window, the negligible length of the audio samples still allows for low latency if the computational pipeline is optimised well.
- Coherently with the hypothesis in regards to the behaviour of the GRU units model, there is some space left for further capturing of the long term relations. This is confirmed by our best performing model, which is just the original model with an additional LSTM layer appended. This is by far the best performing model of

Reduced Cost Models					
Name	**Convolutions**	**Recurrent**	**Parameters**	**Operations**	**Test Acc**
Original	(10,1)	(LSTM, LSTM)	200K	890K	94.3
Original plus Avg	(10,1)	(LSTM, LSTM)	200K	890K	94.07
GRU model	(10,1)	(GRU, GRU)	160K	820K	94.09
LSTM Avg 2	(5,1)	(LSTM, LSTM)	200K	520K	94.26
LSTM Avg 3	(5)	(LSTM, LSTM)	200K	390K	94.16
LSTM Avg 4	(1)	(LSTM, LSTM)	200K	190K	93.91
LSTM Avg 5	(1)	(LSTM)	100K	120K	92.95
Increased Accuracy Models					
Name	**Convolutions**	**Recurrent**	**Parameters**	**Operations**	**Test Acc**
LSTM Incr 1	(10,10,1)	(LSTM, LSTM)	200K	3390K	94.51
LSTM Incr 2	(10,10,10,1)	(LSTM, LSTM)	200K	5890K	94.53
LSTM Incr 3	(10,1)	(LSTM, LSTM, LSTM)	300K	950K	94.72
LSTM Incr 4	(10,10,10,1)	(LSTM, LSTM,LSTM)	300K	5950K	94.47
LSTM Incr 5 Avg	(10,10,10,1)	(LSTM, LSTM)	200K	5890K	94.46

TABLE 1: tested architectures and results on GoogleSpeech 32 V1

all, and strenghtens our hypotesis about the importance of the information gain. Moreover, we see that using three convolutions with three LSTM layers doesn't lead to better results but instead, we got worse performance. Even if this result can simply be due to the variance in stochastic optimization, we suggest as a future line of work the use of residual learning ([4]) in combination with the attention mechanism with recursive layer. This result is in line with all the conclusions above.

It is important to highlight again that it is not simply the best classification accuracy that we are after. State of art in this area is advanced and recent improvements have been in small steps. All of the models presented here have accuracy above 93%, and depending on the application of the technology, the trade off resulting in accuracy reduction of 2% in return for significant operation and parameter number reduction might be acceptable.

V. Concluding Remarks

In line with the current tendencies in the Keyword spotting community, in this report we presented and analyzed some modifications of a neural network architecture that leverages recurrent layers and attention to improve accuracy and reduce computational cost. We proposed some modifications that aim to result in a model more effective and efficient in real life cases, as well as equivalently efficient on the benchmark dataset.

Using the different architectures that we proposed, we studied the tradeoff between complexity and performances of such a model, and succeded both in reducing computational cost and improving the overall performances.

Future line of works can include: merging this approach with residual learning as suggested by [4], application of this approach to video classification and improvements of the attention layer to exploit all of the information provided after dependencies extraction.

In addition, we feel that testing the Averaging approach we have proposed as an input for the Attention mechanism, possibly with utilisation of non-centered audio samples for training, would be beneficial to understand its impact on the performance.

This was our first complete study of such a complex architecture. It has been very instructive, both on a technical and on a theoretical side, since, apart from learning some tricks and capabilities of the Keras and Tensorflow frameworks which we have not encountered before this class, we also learnt a great deal on the applicability of what we saw during the lessons about neural networks and features extraction from audio samples. Even though it's been a pretty intense and time demanding project, we are very happy of having been able to give our contribution the state of the art (at least when we started out), which is a satisfactory result to report.

References

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

[2] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks.," in *ICASSP*, vol. 14, pp. 4087–4091, Citeseer, 2014.

[3] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[4] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5484–5488, IEEE, 2018.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.