Attention based recurrent neural networks for keyword spotting

Nicolo' Dal Fabbro†, Davide Masiero†

***Abstract*—Keyword spotting (KWS) is still an open research topic. Many models have been proposed in literature, and deep learning offers a lot of possibilities. Attention based models have shown outstanding results combined with lightweight architectures, and they are getting more and more important in the developing of Sequence to Sequence models. In this paper we present an attention mechanism based on recent developments, for the KWS task. We try to get the best from the combination of the CNN and RNN models, and we develop and analyze a new framework for the attention layers. The main contribution of this paper is to provide a way and show the good impact of computing more than one Attention Vector, and also a performance comparison of the different possible choices. The new model proposed in this paper reaches state of the art performances in the classification task with 35 classes.**

***Index Terms*—Human Voice, Keyword Spotting, Recurrent Neural Networks, Attention Based Models, Deep Learning.**

I. Introduction

Given the increasing interest for speech-related technologies and the wide variety of applications, speech recognition is an active area of research. In particular, the development of lightweight speech command models would be of great importance for a multitude of novel engineering applications, such as voice controlled robots.

Attention-based recurrent network architectures have been recently studied in the literature and show outstanding learning capabilities, reaching state of the art performances in the speech command recognition task, having at the same time a relatively small number of parameters. Starting from the recent work of [2], we propose a model that produces good performances, comparing results and explaining our design choices.

We propose a different convolutional feature extraction architecture, combined with alternative ways of computing the attention weights from the feature vectors. The novelty of this work with respect to [2] is the fact that we explore the possibility of getting multiple vectors of attention weights from a sequence, and we see also the effect of extracting the attention weights from the final states of the deep recurrent network. The purpose of this approach is to better understand the role that the attention mechanism has and to be able of getting more explanations of the model in the same spirit of [2]. The results we get can be of interest for who wants to find new ways of making the network learning the data through attention mechanisms, since our encoder structure gives state of the art performances, and the visualization of the attention weights with our new framework allows to see new possible developments in the keyword spotting task. The main contributions of this work are:

1) A new effective feature extraction architecture that is compared with the one in [2] and with adapted convolutional architectures based on the ones in [1].
2) An attention-based model designed to force the recurrent network to learn from multiple attention weights computed from the same sequence.
3) Visualization of the attention weights with the proposed framework. Interpretation of the different weights obtained from the same sequence showing a different behaviour.

II. Related Work

In [1], several versions of a deep convolutional neural network architecture are proposed for the word recognition based keyword spotting (KWS) task. The aim of the authors is to decrease parameters and multiplies, for the purpose of a lightweight model. Their contribution is useful for the convolutional structure and characteristics, and we used it as a reference. In our work, we compare results obtained by our model with results obtained by an adaptation of their network as a feature extraction architecture preceding an attention-based recurrent network. For the attention-based models literature, we mainly refer to [3], where a detailed description of the attention mechanism in its possible implementations is shown. In the paper they apply the attention mechanism to the speech recognition task, in particular they also focus on the capability of a network to work with different length utterances (Forced Alignment of Long Utterances). In [2], a very well performing implementation of the attention mechanism for KWS is proposed. With respect to [3], the implementation of [2] is a special case: in [3] the approach is general and also long utterances are considered, and in particular the attention weights are computed also on limited parts of the sequence. Instead, in [2], for the specific command recognition task, and disposing of a dataset of 1 second long recordings, the attention weights are always computed considering the whole sequence. The development of neural attention models from [4] increased performance on multiple tasks, especially those related to long Sequence to Sequence models. These models are extremely powerful ways to understand what parts of the input are being used by the neural network to predict outputs. Given our interest in KWS, we base our work on the

†Universita' degli studi di Padova: nicolo.dalfabbro@studenti.unipd.it
†Universita' degli studi di Padova: davide.masiero.5@studenti.unipd.it

recent implementation of [2] because it is the first in which the attention mechanism is used for single word recognition, making changes to the original model proposed in [4]. The main difference between the two models is that the original presented in [4] is used in an Encoder-Decoder framework, computing the context vector for each single time frame of the sequence through a weighted average, where the weights are learned by a dense layer positioned between the encoder and the decoder, and then the context vector is given in input to the decoder for each time frame. Instead, in the implementation proposed in [2], they compute a single context vector (Attention Vector / Alignment), to represent the whole sequence, this vector is computed in a similar way with respect to the previous model, but instead of using each vector of the sequence, they use a specific vector (the middle one), from which the parameters used to generate the alignment are learnt.

In our implementation we want to go deeper in the attempt to explain both the convolutional feature extraction and the attention mechanism capabilities in its possible variations. In particular in our work we investigate the possibility of using other feature vectors from which the network would learn the representation of the sequence. In [2], they also propose the visualization of the attention weights to explain the speech recognition model. In particular, they state how the vowels characteristics and transitions are dominant in the network learning and classification, identified by the attention mechanism as the main discriminant to distinguish words. From this consideration, our work aims at experimenting the possibility of making the network learn strongly also other characteristics of a command word, going beyond the vowels description. To do so, we compute two attention weights vectors in parallel from each sequence. This parallel training aims at forcing the network to learn and to characterize more specifically the data.

III. High Level Overview

The architecture that we propose can be described as a concatenation of three main blocks. The first block has shown to have a crucial impact on the learning process and has an important role in our contribution. It is a convolutional neural network(CNN) block, with the aim of extracting features from the input, as a sort of learned pre-processing phase, applied to the sequence of feature vectors that it receives in input. The output of the block is again a two-dimensional feature matrix. A similar block is also present in [2], but the characteristics are different, and will be explained better in the next sections. The second block is a deep bidirectional recurrent neural network(Bi-RNN), from the output of which the attention weights are computed. We implemented the Bi-RNN using GRU units. In [2] LSTM are instead used.

The final block is a sequence of dense layers that takes in input the attention weights. The novelty that we introduce in the attention mechanism is in the fact that we compute two vectors of attention weights in parallel from the same sequence. Doing so, we have two sets of parameters that are trained to generate the so called "queries", that are vectors of coefficients with which the output matrix is dot-multiplied to get the alignment vectors. The proposed structure has the main characteristic that the learning from a certain point of the block is split in two parts that are trained independently to represent the data. What we are doing is forcing the network to learn more than one single strong feature of the sequence, that is like saying that, if the vowels are the dominant component, we would like the network to be able to learn *other* features of a sequence, like consonants and peculiar combinations of phonemes.

In `Fig.~VI` a simple high-level diagram is shown, to give a clear idea of the general structure of the deep network.

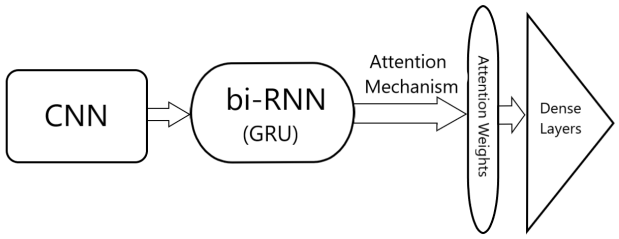

Fig. 1: High level diagram of the deep network model.

IV. Signals and Features

The Google Dataset version 2 collects 105829 recordings from people all over the world, for a total of 35 words. The dataset has been collected with the purpose of allowing keyword spotting studies and experiments.

All the recordings are 1s long, even if some are shorter or damaged, and they all have the same sampling frequency of 16000Hz. For the short recordings, we apply zero padding. We get the feature vectors in a standard and widely used way. We exploit the available python-speech-features [5] library, that we used to compute the mel-scale spectrograms, that is obtained using 80-band mel scale, 1024 discrete Fourier transform points. The framing was done with a standard configuration of parameters: window length of 0.02s, hop size of 0.01s. For the windowing, we didn't apply any window function. We noticed that this produces a visible presence of artifacts in the mel-scale spectrogram. By the way, we have seen that our convolutional feature extraction block, trained, is able to learn the data in a very well performing way, getting rid of the artifacts bad effect. Actually, trying with feature vectors obtained through hamming window applied on the windowing, we see that our network learns less, and reaches lower accuracies, as if the presence of the artifacts brought to a better learning, as a sort of data augmentation by disortion/noise technique.

We split the Google Dataset in train, validation and test set in a standard way, the one proposed and suggested by the

creators of the Dataset, that is also the same way used in [2].

V. Learning Framework

The first block of our architecture is a CNN for feature extraction purposes. A diagram of the one that was proposed in [2] is shown in Fig.~2, while the one that we propose is shown in Fig.~3. The one shown in Fig.~2 has the purpose of capturing time correlation between the elements of the sequence, while on the frequency features no convolution is applied. In [1], they also use convolution in frequency, even if it is always a less involving convolution(e.g., they apply a kernel of size 3 in frequency and 20 in time in the first convolutional layer). Instead, as it is possible to see in the figure, our kernels are of square shape. Furthermore, with respect to the one of Fig.~2, our convolutional block is deeper but smoother in the kernel decreasing to go back to a 2D input for the following deep recurrent network.

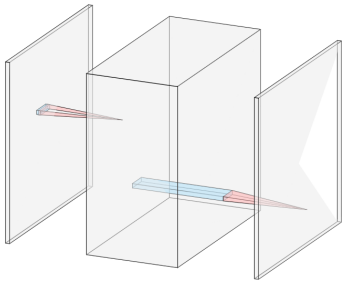

Fig. 2: Convolutional Feature Extraction Architecture of [2]

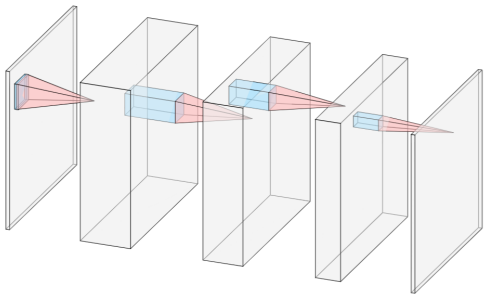

Fig. 3: Our Convolutional Feature Extraction Block

As stated in Sec.~**??** and Sec.~III, we focus on the attention based recurrent neural network structure. A grid-search for the best parameters of the recurrent network has been done. We did train the network with a one layer recurrent network and with a two layers recurrent network. Trying to see if we could decrease the number of parameters, we try with Bi-RNN with 32, 48 and 64 number of units. We see that the best learning configuration is the one with two layers and 64 units.

To illustrate the framework we propose for the attention mechanism, we show in Fig.~4 a detailed diagram of the architecture adopted at the end of the recurrent block. It illustrates how we handle the output of the second bidirectional recurrent layer. What is done is the following: the first element of the output sequence, $\boldsymbol{y_1}$, is taken, projected in a dense layer with linear activation("query") and multiplied matrix-wise with the output sequence. The output of the multiplication is then passed to a softmax layer. The output of this multiplication is multiplied matrix-wise again with the sequence matrix, getting what is a sort of weighted average of the sequence vectors, also called attention vector. For the mid vector of the sequence, $\boldsymbol{y_{mid}}$, an equivalent operation is done. After that, the two attention vectors are put in a dense layer where they are concatenated. We tried several configurations of this multiple attention weights vectors framework. A good working one was a configuration in which one of the two vectors is composed by the final states of the bidirectional recurrent networks, $\boldsymbol{h_{f1}}$ and $\boldsymbol{h_{f2}}$, concatenated together, while the other was $\boldsymbol{y_{mid}}$. By the way the one shown in the figure is the best performing one, the number of parameters is $223k$, so a number of parameters very similar to the one of [2]($202k$). We also give to the final dense layers an additional piece of information, that is the difference between the two attention vectors. The intuitive reason for this choice is the fact that we would like the network to be able to discriminate the characteristics of a sequence exploiting the information collected by two attention weights vectors, but at the same time we think that also the comparison between the two outputs could be a useful distinguishing information to be learnt by the network on the different data. We get a confirmation of this intuition from the results: the best validation accuracy reached on the 35 classes is a 95% accuracy, and the test accuracy is a 94.10% accuracy.

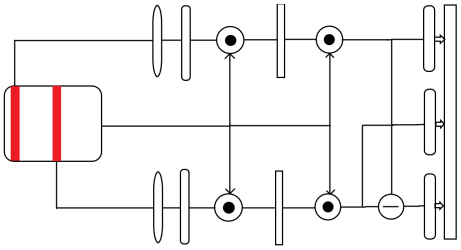

Fig. 4: Our implementation of the attention mechanism.

VI. Results

In this section we present the results obtained by the several networks we compared. In brackets there is the symbolic name we use to recognize the model. We call the results obtained with models similar to [2] *Douglas, because of the name of the first author of the paper. When non specified, the CNN architecture is the one shown in Fig.~3 and proposed by us. The networks we compare are the following:

1) Between the two Bi-GRU the hidden state is not passed, the attention mechanism is the one proposed in [2], so

the mid vector of the recurrent network output is used to compute the Attention Vector. (CNNRNNAttDouglas)

2) The attention mechanism uses the middle vector of the recurrent output to compute the Attention Vector and the hidden state of the recurrent network is passed layer-to-layer(CNNRNNStatefulAttDouglas).
3) With the attention mechanism that uses the mid vector of the RNN output to compute the Attention Vector, we train a different CNN architecture, using as Feature Extraction block a network similar to the one of the convolutional part of [1]. The advantage of this model is that the trainable parameters are less, since the inputs are more reduced in size by the CNN part. The number of parameters in this case is less than $100k$(PCNNRNNAttDouglas)
4) The two Bi-GRU are trained passing the final state layer-to-layer, and the attention mechanism uses the final state of the last Bi-GRU to get the Attention Vector.(CNNRNNAttState).
5) The model is similar to the one proposed in `Sec.~V` and shown in `Fig.~4`, but the difference between the two attention vectors is not given in input to the final dense layers.(MultiAttention)
6) The same CNN proposed in [2] and shown in `Fig.~2` is used, and the attention mechanism works as at the previous point.(MultiAttentionDouglas)
7) The model proposed in `Sec.~V` with the attention mechanism and shown in `Fig.~4`(MultiAttentionDiff)

We present two main metrics: the accuracy obtained on the test set and the confusion matrix. Furthermore, we show the attention weights produced for two sample signals by two networks: the best performing one, (MultiAttentionDiff), and the model in [2] implemented with our signal and feature preprocessing(the attention weights of this are very similar to the ones obtained with any of the presented one-attention vector based model).
In `Tab.~VI` is possible to see the test accuracy results obtained by the networks.

Model	Test Accuracy
CNNRNNAttState	92.55
CNNRNNAttDouglas	92.62
CNNRNNStatefulAttDouglas	92.54
PCNNRNNStatefulAttDouglas	88.26
MultiAttention	93.11
MultiAttentionDouglas	92.50
MultiAttentionDiff	**94.10**
Douglas (paper)	93.90
Douglas (by us)	92.71

The confusion matrix is a meaningful metric, shown in (`Fig.~5`), showing with which percentage words are misclassified by the network. As in [2] we obtain a very good result, as it is possible to see in the main diagonal, we reach true positive classification greater than 90% for the majority of the words, and for the others words still greater than 80%.

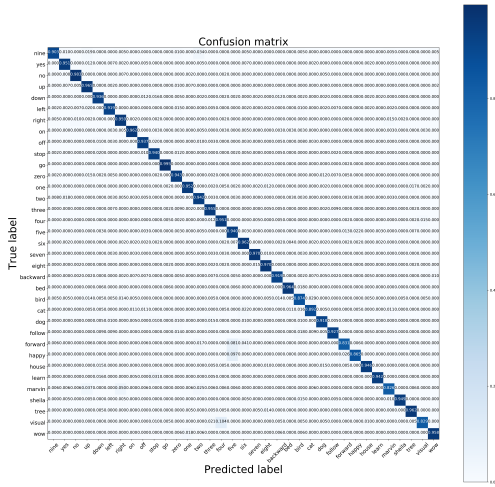

Fig. 5: Confusion Matrix for 35 class classification.

One of the advantages of deep learning models with attention mechanism is the possibility to get an intuition about what the model does, as in the below figures, i.e. (`Fig.~6` and `Fig.~7`). The plots represent the waveform signal, the corresponding mel-scaled spectrogram, two attention weights vectors (produced by MultiAttentionDiff network), and single attention weights (produced by our Douglas network). It's possible to see, in both the cases, as the attention weights are trying to caught the main features of the signal. Moreover it's visible the advantage of using two attention weights, in fact, the MultiAttentionDiff model, can focus on more then one important part, with an additional number of parameters that is not huge, being $20k$ more, and could be reduced in further research.

VII. Concluding Remarks

In this project we have tried to understand the power of an attention mechanism and its implementation. We started from a simple CNN, we inspected the RNN architecture, with more emphasis in GRU units. Dealing with the attention mechanism, we managed to propose our own model based on our research, intuition and trials.
A further work could involve the increasing to three, or even more, of the number of attention vectors extracted from the sequence. Another possibility would be to search for a different way to get the "query" for the computation of the attention vector, or to implement other types of attention

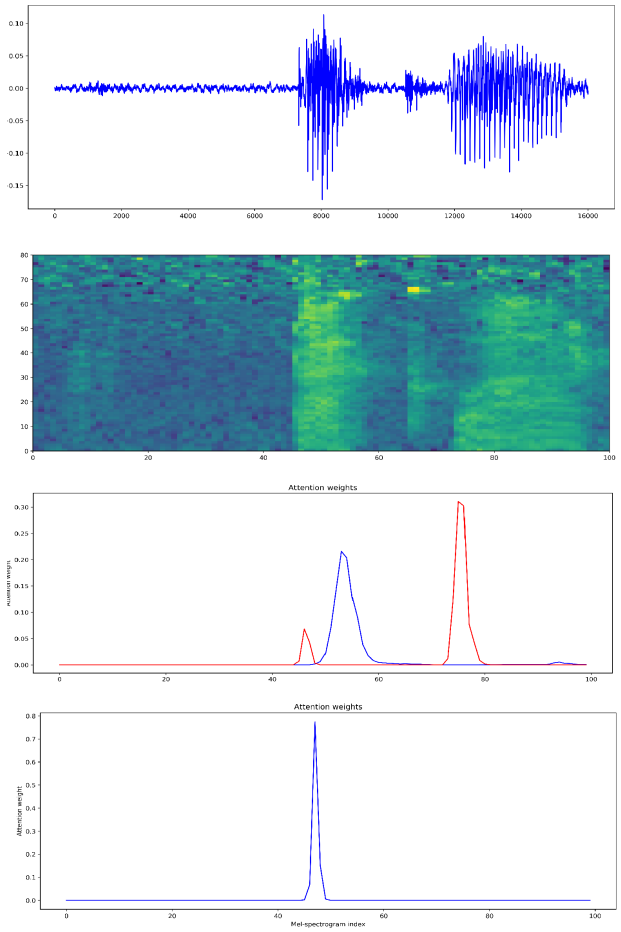

Fig. 6: Waveform, mel-scaled spectrogram, attention weights produced by MultiAttentionDiff and attention weights produced by Douglas (implemented by us), of the word ”Three”.

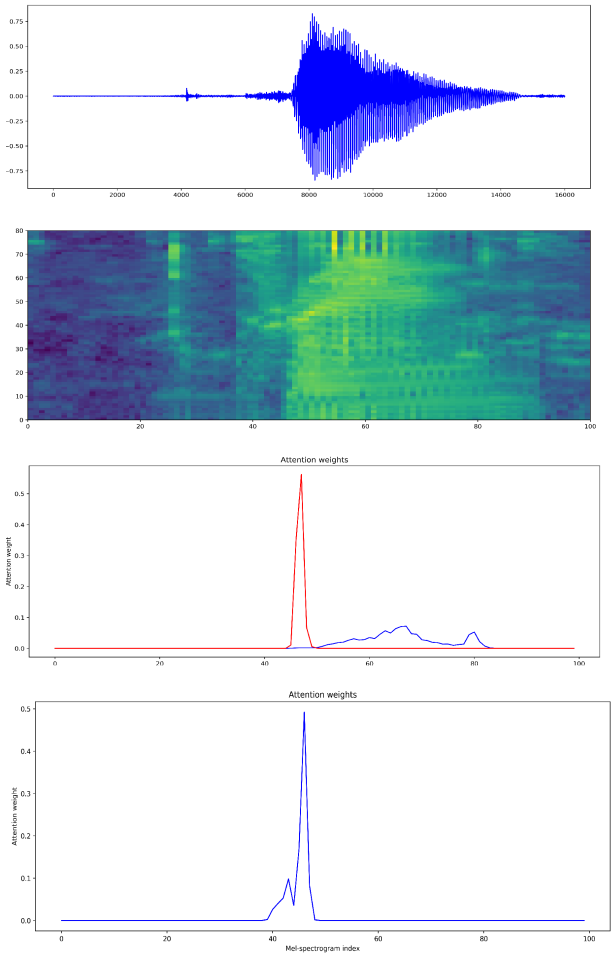

Fig. 7: Waveform, mel-scaled spectrogram, attention weights produced by MultiAttentionDiff and attention weights produced by Douglas (implemented by us), of the word ”Backward”.

mechanisms like those present in [6] and [7], adapting them to the KWS task. Other improvements could be to add random noise at the input signal and to do data augmentation. From a student’s point of view, the project development has been an opportunity to learn a lot of things. We understood how to process an audio file and use it in an application, how to build several networks starting from a very simple CNN model, up to a very complex model with several type of layers, and with some very recent technique in the Machine Learning realm, and as last but not least, we have learnt how to work in a team and the usage of some tools like GitHub to build a working project.

But, obviously we have encountered a lot of difficulties before reaching a good result. We found the some problem with the ”kapre” library used in [2] for the processing of wav files. This lead us to have problem with the loss function that produce Not a Number (NaN) as output, and then the inability to train the network. Then we migrate into another framework to compute the Mel Spectogram coefficients, that resolve the previous problem but lead us to another big problem, the overfitting. In [2] the original network haven’t any types of regularization, and it was our turn to find the best tuning of the parameters to allow the network to learn a good representation of the data.

Anyway after all, we were able to reach a good result and we are happy for this.

REFERENCES