Attention Based Models for Keyword Spotting

Riccardo Mazzieri

***Abstract*—In this work we explore a variety of Deep neural newtork architectures, with particular focus on the attention mechanism, for the Keyword Spotting task (KWS). In recent years, attention based models have proven to be successful in a wide variety of domains, including image recognition, neural machine translation and speech recognition. We analyize recent attention based architectures: we first focus on the model proposed by de Andreade et al, and explore the role of the size of filters in the convolutional part of the model.**

***Index Terms*—Keyword Spotting, Convolutional Neural Networks, Recurrent Neural Networks, Attention Mechanism, Transformers.**

I. Introduction

The Keyword Spotting (KWS) task consists in the detection of a certain predetermined set of keywords from a stream of user utterances. In recent years, this problem has become increasingly popular and, with the rapid development of mobile devices, it is now playing an important role in human-computer interaction, as well as encouraging the adoption of hands-free interfaces for a wide variety of use cases, that can range from "smart home" devices like Amazon Alexa, to virtual assistants such as Apple's Siri or Google Assistant.

At present, Deep Learning techniques represent the state of the art approach for the KWS problem and have proved to give good results on tradeoff between model accuracy and memory footprint [1] [2] [3]. Fully connected and CNN based architectures have been widely explored, yielding good results [4] [2] [5] [6]. In recent years, mostly thanks to the Transformer architecture introduced by Vaswani et al. [7], interest towards attention-based architectures has grown dramatically. Indeed, recent works in machine learning have shown that models incorporating attention are able to provide groundbreaking results in a wide variety of domains [8] [9] [10] [11] [12]. Another charming feature of attention-based systems is their inherent interpretability. Indeed, Explainable AI (XAI) is quickly becoming an hot topic in modern machine learning research, and the development of models capable of giving some sort of explanation for their predictions will in time become more and more desirable, if not required [13].

Motivated by those increasing trends, in this work we explore several applications of the attention mechanism for the KWS task. Specifically, we first explore an hybrid architecture, introduced by de Andreade et al. [14], constituted by a convolutional, a recurrent and an attention part. We explore the role of filter size for the convolutional block, to understand if vocal features can be extracted mostly from time or frequency domain. We also propose some small adjustment to the attention layer and compare the performances with the more modern variant presented in [3]. We also examine the Keyword Transformer [15], a recent architecture that exploits the strength of Transformers in a similar fashion to Vision Transformers [8].

II. Related Work

A. Foundations and state of the art

In recent years, machine learning techniques have proven to be the de-facto standard for approaching the KWS problem. Such models typically perform segmentation of the audio sequence on the time domain and extract log-mel scale spectrograms or mel-frequency cepstral coefficients (MFCC) [16] from each frame. Those are then used as input feature vectors for the models. One of the first works exploring deep neural networks for the KWS task is from Chen et al [1]: the authors explore small footprint fully connected architectures and show how they improved performances with respect to the baseline HMM models. This kind of architecture accounted for the sequentiality of audio data by stacking feature vectors of adjacent audio frames: while this gives good results compared to the baseline, it is a very simplistic way to model sequential data. Sainath and Parada [2] approached the problem by using CNNs, a more fitting class of models which are able, by design, to capture several essential features of speech data (like input topology or translational invariance) and at the same time having a much smaller memory footprint due to parameter sharing. Indeed, CNNs have proven to be very successful for KWS: the current state of the art has recently been achieved in [17], where the authors introduce Broadcasted Residual Networks (BC-ResNets), a particular class of ResNets that are able to capture both 1D and 2D features thanks to the introduction of a new kind of residual block.

B. Models based on Attention

In [14], De Andreade et al. propose an attention based convolutional recurrent neural network (Att-RNN), which takes as input a mel-scale spectrogram and extracts features in the frequency dimension with two convolutional layers. Then, such features are given as input to two bidirectional LSTM [18] layers, in order to extract long range dependencies from the inherently sequential audio data. The last[1] output feature from the LSTM layer is then transformed with a linear dense layer and used as a query vector for the attention mechanism. Finally, the sum of the LSTM outputs, weighted

[1]In the paper, the authors report to use the middle output, but looking at their implementation, they use the last one. Either way, as they point out, any output of the LSTM layer should work well as a query vector, since the bidirectional LSTM layer should be able to summarize the whole sequence in any of its outputs.

with respect to the attention scores, are given as input to a final dense MLP, which performs the classification. An immediate drawback of this approach is its impossibility to function in a streaming fashion (at least without introducing delay), since bidirectional recurrent layers require the whole sequence of data to work. Nevertheless, the introduction of the attention mechanisms allows us to inspect the sections of audio which were responsible for the classification, increasing model transparency. In [3] the authors propose a more modern variant of the same architecture (referred as MHAtt-RNN): LSTM is substitued by GRU [19], and the attention mechanism is replaced by the multi-head attention layer introduced in [7] (using 4 heads). The latest contribution in attention based models for KWS comes from [15], where the authors, inspired by the success of the newly introduced Vision Transformer (ViT) [8] for computer vision tasks, propose an adaptation of such architecture for keyword spotting, called the Keyword Transformer (KWT). The results were suprising: despite ViT proved to be competitive only when supported by pre-training on large datasets, KWT outperformed more complex models based on mixes between CNN, RNN and attention, even when trained on relatively small datasets, like the Google Speech Commands dataset [20].

III. Experimental Setup

A. Signals and Features

Each model was trained on the Google Speech Commands dataset V2 [20], which consists on a total of 105829 user utterances of a total of 35 keywords. Each utterance is stored as a one second long[2] `WAVE` file, sampled at a 16KHz rate. From each signal, 80 log-mel spectrogram features are computed, using a window size of $25ms$ (400 samples) and a hop size of $10ms$ (160 samples). Following the approach from past literature, the models are trained for two different tasks, which are the following:

- **12kws** task: the model must discriminate among 12 different keywords: "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go", unknown or silence. The unknown keywords are randomly chosen from the set of remaining keywords and the silence samples consist in one second long crops, randomly extracted from the noise files provided by the dataset[3]. Since the total amount of words belonging to the "unknown" class was much more than number of representatives for each of the other keywords, we decided to randomly extract them to create a balanced dataset. In this way, we have 36921 samples for training, 4443 for validation and 4888 for testing.
- **35kws** task: the model must discriminate among all the 35 keywords present in the dataset. No unknown class or silence class is introduced, therefore, for this task, the entire dataset is used. The training set consists in 84843 samples, the validation in 9981 and the test in 11005.

[2]Some clips are a bit shorter than one second: in those cases, the clips are zero padded towards the end.

[3]Those consist in 6 files containing noisy background sounds, both artificially generated and recorded from real environments.

Following Google's suggestions from [20], the dataset is split in 80% for training set, 10% for validation set and 10% for test set, in such a way that the same speakers are never present in two different splits.

For each task, the training set is augmented, following existing approaches from the literature. Specifically, the augmentation process follows this order:

1) Each signal is randomly shifted left or right (zero padding the remaining portion), by x samples, where x is drawn from a uniform distirbution $U(0, 100)$;
2) Samples belonging to the "silence" class do not remain the same across epochs, but are randomly generated each time with the same process described above;
3) With a probability of 0.8, each signal is mixed with a randomly generated background noise, multiplied by a factor drawn from $U(0, 0.2)$;
4) After the conversion in MFCC/log-mel the features are augmented using SpecAugment [21], a simple data augmentation method which consists in adding randomly sized frequency and time masks to the feature matrix: this is done in order to render the model more robust to partial loss of frequency information or of small sections of speech.

The image resulting from the last step constitutes the actual input for the model. We remark that validation and test sets are not touched by this process.

All the project was carried out using an NVIDIA GeForce GTX 1060 6GB GPU and an Intel i5-64000 CPU, on a Linux machine. All models were built and trained using TensorFlow [22]. All source code is available on Github[4].

B. Input Pipeline

Since data augmentation is applied, storing the entire dataset in memory would not be possible: this would result in the same data being reused across epochs. For this reason, a core part of this work was to build an efficient input pipeline that could handle data augmentation on the fly, during training. In this section we explain how the data generation process described in Section III-A works. The augmentation takes place in two different moments during training:

1) Phase 1: this is the phase when random noise samples are extracted, and each sample is shifted. This is performed by the CPU, while the GPU is performing training.
2) Phase 2: this phase is performed inside the model by the GPU, with a series of preprocessing layers. Specifically, several custom preprocessing layers were built:
   a) `RandomNoiseAugment`
   b) `LogMelSpectrogram`
   c) `MFCC`
   d) `SpecAugment`

Note that in Phase 1, each operation is performed one sample at a time, while in Phase 2 each operation is performed

[4]Link here.

in parallel on an entire batch of samples: this results in a much more optimized implementation. Given the hardware setup with which the project was carried on, this framework was a necessity: just computing the MFCC for each sample on CPU would cause a performance bottleneck, wasting the computational capabilities of the GPU. In Figure 1 we report a detailed diagram, showing all the steps in the input pipeline for the generation of the training set.

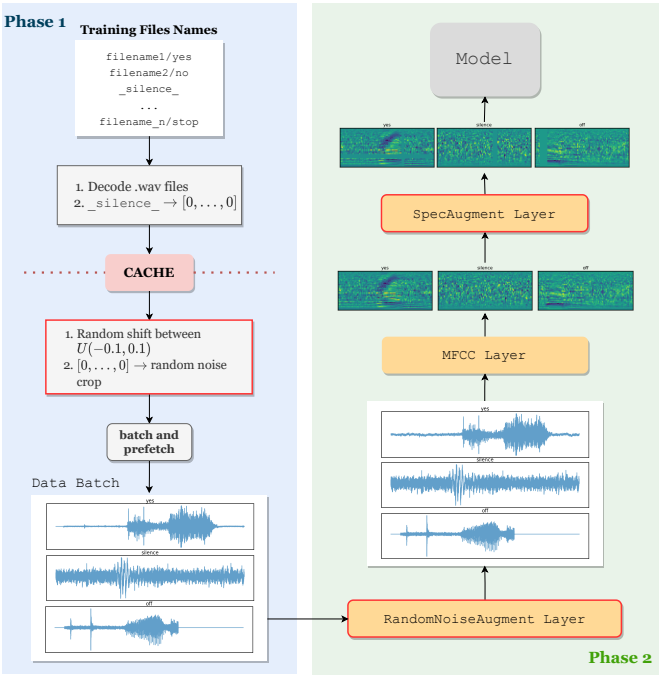

Fig. 1: Detailed description of the system that generates the training data. In the cache block, the data that was produced until that moment is saved to file. Each step which is performed before the caching operation happens only one time, during the first iteration of the dataset; on all the successive iterations, the data is read from the cached file. For the validation and test sets, the augmentation is not applied. Furthermore, to ensure complete separation between training and validation/test sets, the noise samples were generated from different portions of the noise files based on whether they were used for training of for validation or testing.

C. Learning Framework

Descrivere nel dettaglio i modelli [...]

Each model was trained for 50 epochs, using early stopping and reduce lr on plateau

IV. Results

In Table 1 we report the different accuracies...

TABLE 1: Table with results

Model	Param.	Mult.	Acc.
res8-narrow[7]	20 K	5.65M	90.1%
res15-narrow[7]	43 K	160M	94.0%
res8[7]	111 K	30M	94.1%
res15[7]	239 K	894M	95.8%
DS-CNN-S[5]	24 K	5.4M	94.4%
DS-CNN-M[5]	140 K	19.8M	94.9%
DS-CNN-L[5]	420 K	56.9M	95.4%
TC-ResNet8[8]	66 K	1.12M	96.1%
TC-ResNet8-1.5[8]	145 K	2.20M	96.2%
TC-ResNet14[8]	137 K	2.02M	96.2%
TC-ResNet14-1.5[8]	305 K	4.13M	**96.6%**
TENet6-narrow	**17K**	**553K**	96.0%
TENet12-narrow	31 K	895 K	96.3%
TENet6	54 K	1.68M	96.4%
TENet12	100 K	2.90M	**96.6%**

V. Concluding Remarks

References

[1] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4087–4091, 2014.

[2] T. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *INTERSPEECH*, 2015.

[3] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurenzo, "Streaming keyword spotting on mobile devices," *Interspeech 2020*, Oct 2020.

[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. Mittermaier, L. Kürzinger, B. Waschneck, and G. Rigoll, "Small-footprint keyword spotting on raw audio data with sinc-convolutions," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7454–7458, IEEE, 2020.

[6] 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.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[9] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*, pp. 10347–10357, PMLR, 2021.

[10] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[11] M. Kumar, D. Weissenborn, and N. Kalchbrenner, "Colorization transformer," *arXiv preprint arXiv:2102.04432*, 2021.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019.

[13] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a 'right to explanation'," *AI Magazine*, vol. 38, pp. 50–57, Oct 2017.

[14] 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.

[15] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword transformer: A self-attention model for keyword spotting," *arXiv preprint arXiv:2104.00769*, 2021.
[16] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.
[17] B. Kim, S. Chang, J. Lee, and D. Sung, "Broadcasted residual learning for efficient keyword spotting," *arXiv preprint arXiv:2106.04140*, 2021.
[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[19] K. Cho, B. V. Merrienboer, Çaglar Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder?decoder for statistical machine translation," in *EMNLP*, 2014.
[20] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.
[21] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.
[22] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zhang, "Tensorflow: A system for large-scale machine learning," in *OSDI*, 2016.