Real-time keyword spotting on RaspberryPi with pruned LSTM

Martin Bulín[1]

1 Introduction

The recent trend of integrating smart electronic devices into human every-day life calls for new methods of making the software both capable of performing high accuracies and meeting hardware limitations. This so called ”smartness” is often supported by sophisticated machine learning models, being developed on powerful computing machines and usually using a huge amount of data, which makes them robust and recently even surpassing human skills in a variety of cognitive tasks. The next step for a practical use, however, is to take the trained models and run them on low-cost devices, where the resources are constrained in terms of computing power and memory size.

This work presents a general method for a massive reduction of parameters (80-90%) of trained (DNN or LSTM) models by removing redundant synapses. The pruning process is performed with no influence on the classification accuracy and the resulting model is minimized in terms of the number of needed parameters, which makes it suitable for on-board applications.

Motivated by the recent call for keyword spotting (KWS) systems and inspired by (Chen at al. (2015)), the pruning method is shown on an LSTM-based KWS model deployed on RaspberryPi. The goal is to make the constrained low-cost device capable of spotting a keyword in real-time, with no delay, by taking advantage of the invented minimization method.

2 Methods and Results

The pruning algorithm, firstly introduced in (Bulín (2017)), has been further elaborated and enabled (besides dense) for LSTM layers as well. Figure 1 shows an example, where even only 20% of all synapses managed to keep a sufficiently high accuracy. The remaining parameters (in this case roughly 400K float numbers) could have simply been removed.

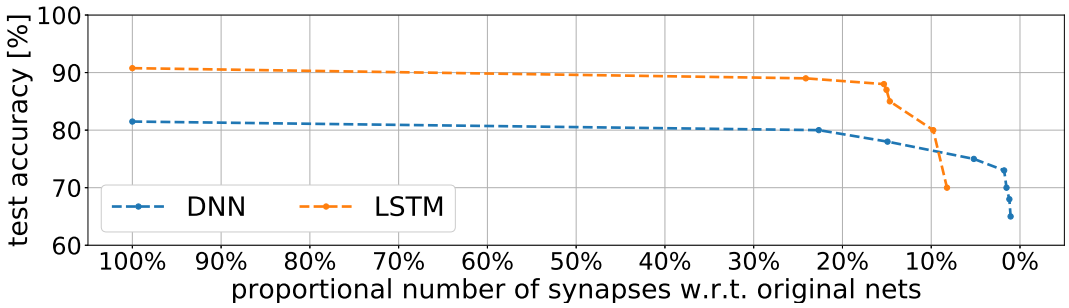

Figure 1: Demonstration of the pruning process and its (non) influence on models accuracy. Shown for two models (DNN and LSTM) on the Speech commands dataset (Warden (2017)).

[1] PhD student of Applied Sciences and Computer Engineering, field of study Cybernetics and Control Engineering, focused on Neural Networks, e-mail: bulinm@kky.zcu.cz

The implemented KWS system is based on (Chen at al. (2015)), where two LSTM layers (each of 128 neurons) are used to deal with the speech signal. In our case, we are limited by the RPi computing power and as the goal is to keep the application in real-time, we need to design a faster predictor. Figure 2 compares considered layers in terms of their size and prediction time.

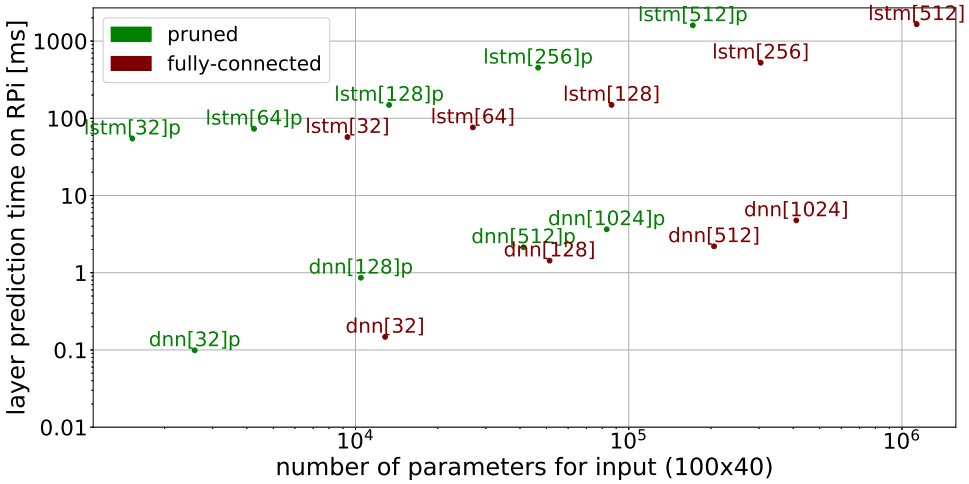

Figure 2: Collection of layers available for the model design (number of cells in brackets).

Based on the experience, the prediction is needed to be done 4-5 times per second in order to catch the keyword reliably, which means that the processing time of the feature extractor must be less than `200ms`. This led us to design a model illustrated in Figure 3. Similarly as in the baseline paper, the model was trained on a huge dataset (2000 words - classes), but apart from the baseline, we keep the model small enough to meet the RPi (model 2B) limitations. Hence, the mission has been accomplished - the system works well, however, to make it more robust, the task is to take the advantage of the massive parameter deletion and find a way to reduce the prediction time of pruned layers (see Figure 2) - left for the future work.

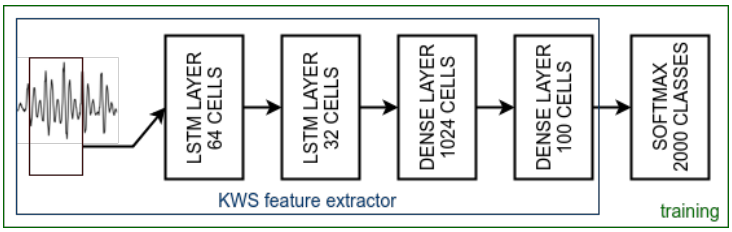

Figure 3: Designed KWS model. The methodology is adopted from (Chen at al. (2015)).

References

Chen, G., Parada, C., Sainath, T. N. (2015) Query-by-example keyword spotting using long short-term memory networks. *In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brisbane, Australia.

Bulín, M. (2017) *Optimization of neural network*. Master thesis. University of West Bohemia, Univerzitní 8, 30100 Pilsen, Czech Republic.

Warden, P. (2017) *Speech commands*. A public dataset for single-word speech recognition. Dataset available from `http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz`.