Tiny-SLU: An End-to-End Spoken Language Understanding for Embedded Devices

Ahmad Bijar[1], Francesco Paci[1], Marco Fariselli[1], Manuele Rusci[1,2], Martin Croome[1], Joel Cambonie[1]
[1]GreenWaves Technologies, France, [2]University of Bologna, Italy
ahmad.bidjar@greenwaves-technologies.com

Introduction

Spoken Language Understanding (SLU) systems automate the process of extracting the meanings or semantics of human speech. Conventional SLUs employ two principal blocks to convert spoken utterances into a set of slots or intents; namely, an Automatic Speech Recognition (ASR) which transcribes spoken words into text and Natural Language Understanding (NLU) which analyzes the intents (Figure 1, left).

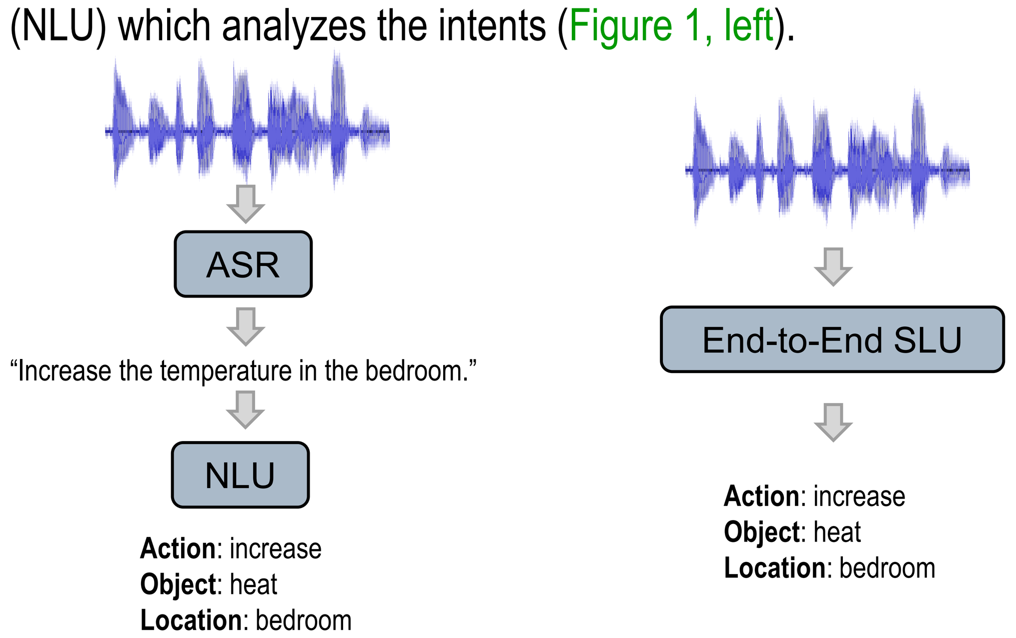

Figure 1. Conventional (left) and End-to-End SLUs.

However, such frameworks have been found very challenging as a two-step training process is required of the ASR and NLU units. Moreover, they need large amounts of memory and computational resources which is not supported by internet-of-things devices (IoTs). Consequently, end-to-end solutions that address the problem of ASR and NLU unification into a single neural network architecture have gained more and more attention during the last few years (Figure 1, right)

It is worth mentioning that these unified solutions also simplify the training-aware or post-training preparation processes necessary for quantized inferences in the embedded devices. Subsequently, in this study, a novel end-to-end solution constructed using recurrent neural networks is proposed. The proposed unified network structure is trained and evaluated on a publicly available dataset, and execution reports on the RISC-V based GAP8 and GAP9 platforms are detailed.

Methods: modeling assumptions

The principal idea is to predict a tuple of intents/slots from a sequence of acoustic features calculated from the spoken utterance. This can be achieved by solving a classification problem using conventional techniques; however, it is assumed that a dataflow streams from the input sequence (i.e., acoustic features) to the first slot and so on till reaching the last slot. Subsequently, an output sequence can simply be formed by a typical order of slots. Also, in order to indicate the starting and ending points of the output sequence, two additional slots (S) with certain values are added accordingly:

Output sequence = { **START**, $S_0, S_1, \dots, S_{n-1}, S_n$, **END**} (1)

This helps to differentiate the beginning and ending of the predicted output sequence from the intermediate sections.

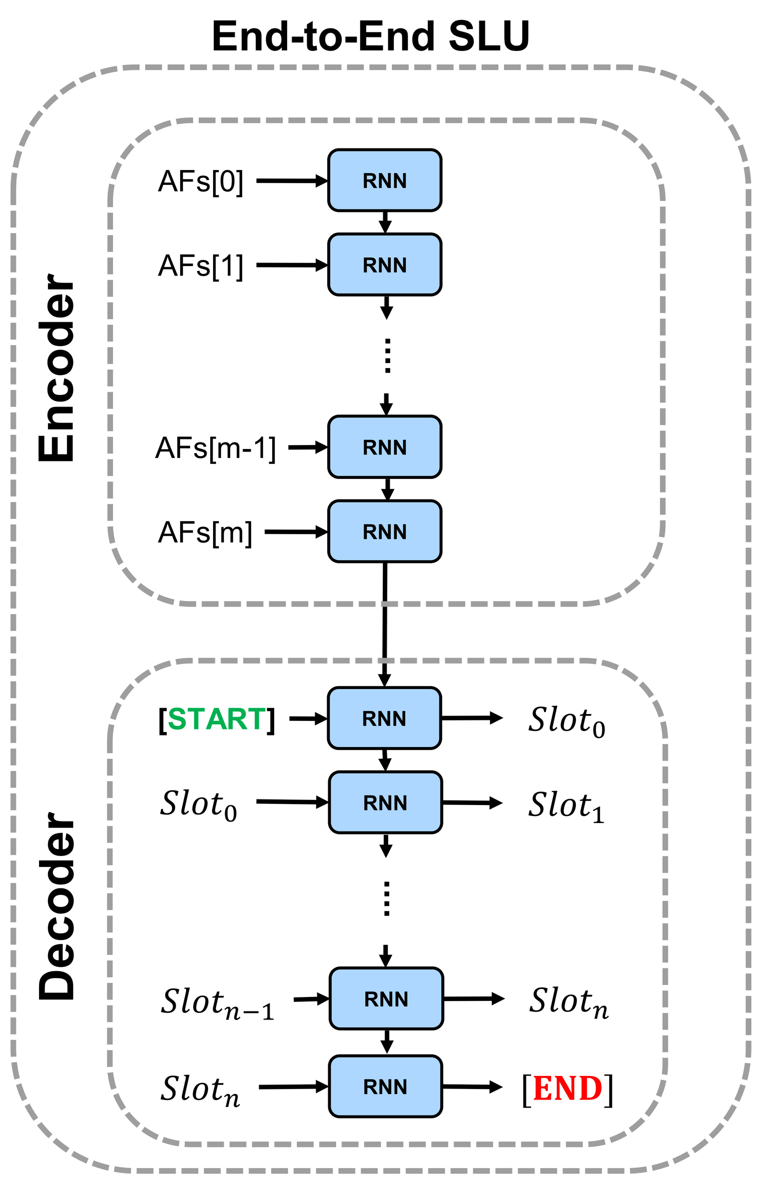

Figure 2. Sequence-to-Sequence SLU architecture

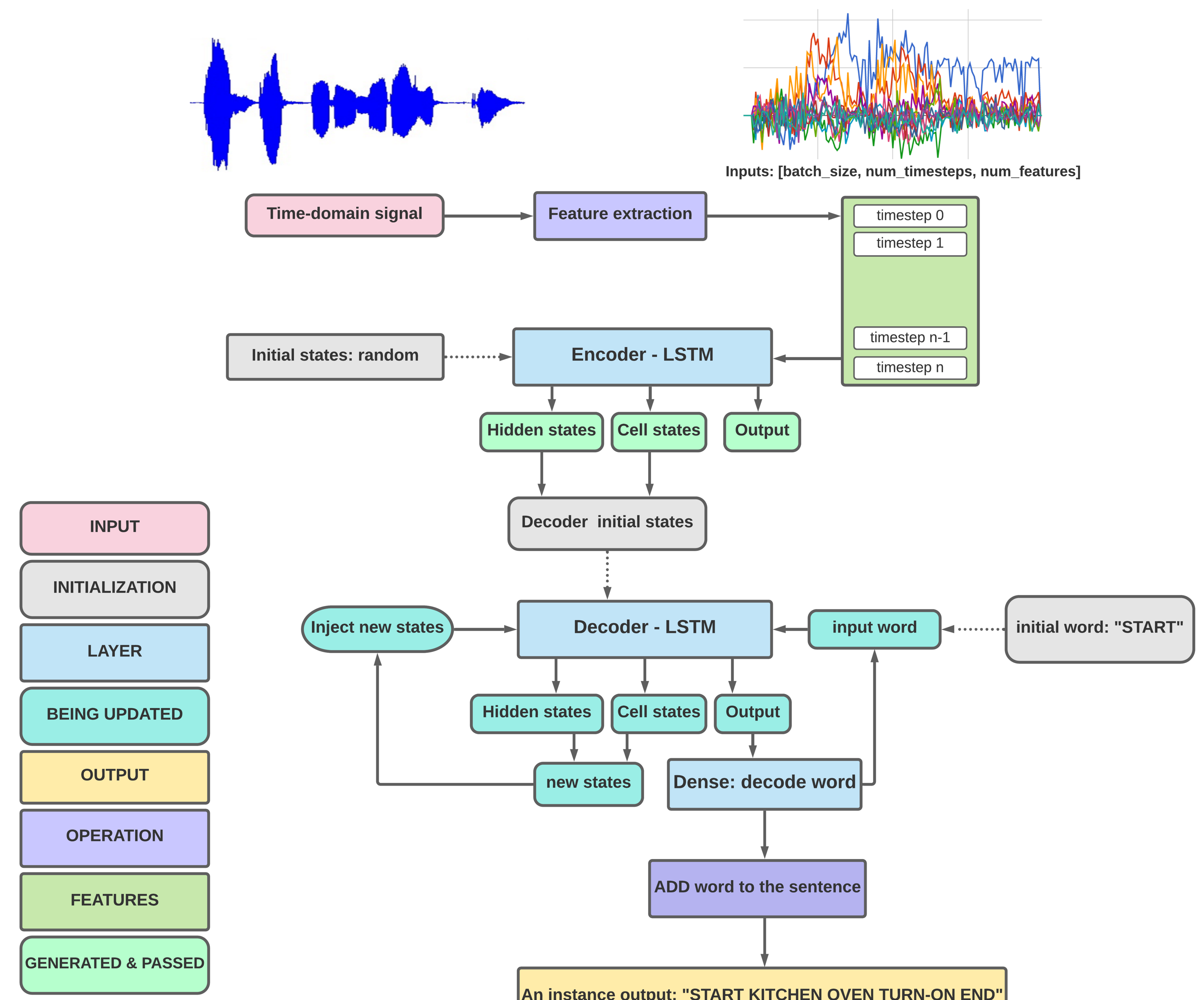

Figure 3. Fully detailed SLU system.

Having inputs and outputs in the form of a sequence enables us to consider a sequence-to-sequence architecture which is composed of two components: an ***Encoder*** module which receives acoustic features (AFs) and provides an intermediate representation of input stream, and a ***Decoder*** that reads in the intermediate representation and turns it into a new form of the stream; i.e., a sequence of tokens forming a set of intents being assigned to the spoken utterance (Figure 2).

Considering the conditional dependence between the slots (s) and in correspondence with probability theory, the intent probability can be estimated as

$$p(S_0, S_1, \dots, S_{n-1}, S_n | \, AFs) =$$

$$p(S_0 | \, AFs).p(S_1 | \, AFs, S_0). \dots .p(S_n | \, AFs, S_0, S_1, \dots, S_{n-1}) \quad (2)$$

Methods: model design

A very detailed architecture of the proposed seq-to-seq modeling is illustrated in Figure 3. Both Encoder and Decoder modules are implemented using recurrent layers. The encoder layer is fed with acoustic features to generate a high level of abstraction from the spoken utterance. Once the encoder processes all input sequence, the extracted features are transferred to the Decoder module by initialization of its recurrent layer. As can be seen, the Decoder is followed by a fully connected layer to map the extracted embedding into the target vocabulary of utterances. As mentioned earlier, the output sequence is supposed to start and finish with additional slots of 'START' and 'END', respectively. This means that the Decoder is first fed by 'START' in order to predict the rest of the slots until it reaches the "END" slot.

Methods: Quantization

For the quantization and deployment of the SLU algorithm, we used NNTool, our internally developed neural network compressor for GAP processors. NNTool is a python library that aims at simplifying a high level DSP/NN graph description (provided as an .onnx or .tflite model) by converting the computational nodes into GAP operators, i.e. SW kernels implemented in our backend library, the Autotiler[2].
Besides standard topology optimizations, such as operation fusions and offline tensors reorganization for more efficient computation, NNTool enables the automatic quantization of different neural networks and DSP layers. Different types of quantization schemes are available in the backend of NNTool, including 16 and 8bits Scaled and POW2 fixed point and IEEE16 or BFloat16 floating point quantization. In this poster, we will focus on the Scaled 8 bits quantization employed in the SLU application. Following [3], our Scaled quantization maps every quantized tensor of the NN inference to the real numerical space with the affine transformation:

$$r = S(q - Z), (3)$$

where q is the integer value quantized to 8bits, r is the representation of that value in the real space, $S = M.2^n$ the scaling factor, represented in our tools and SW library with 8bits for the mantissa M and 8bits for the exponent N, and Z the Zero point, the integer representation of the 0, also in 8bits as the value q. The LSTM is the key operator of SLU, it uses its previous output (state) along with the new data (input frame) to recursively extract the command embedding without losing the long term memory of the sentence.

Applying a scheme similar to the one presented in (3), we quantize the LSTM operations as follows:

$$i_t = \sigma(x_t U_i + h_{t-1} w_i) \rightarrow S_i I_t = \sigma_q \, (S_x X_t S_u U_i + S_h H_{t-1} S_w W_i)$$

$$f_t = \sigma(x_t U_f + h_{t-1} W_f) \rightarrow S_t F_t = \sigma_q (S_x X_t S_u U^f + S_h H_{t-1} S_w W^f)$$

$$o_t = \sigma(x_t U^o + h_{t-1} W^o) \rightarrow S_o O_t = \sigma_q (S_x X_t S_u U^o + S_h H_{t-1} S_w W^o)$$

$$c_t' = tanh\, (x_t U^g + h_{t-1} W^g) \rightarrow S_c C_t' = tanh_q (S_x X_t S_u U^g + S_h H_{t-1} S_w W^g)$$

$$c_t = \sigma(f_t * c_{t-1} + i_t c_t') \rightarrow \; S_c C_t = \sigma_q (S_f F_t * C_{t-1} + S_i I_t * S_c C_t')$$

$$h_t = tanh(c_t) * o_t \rightarrow S_h H_t = \; tanh_q (S_c C_t) * O_t \quad (4)$$

Thanks to NNTool, we can collect the statistics of these tensors, by running inference in full precision over several training samples, in this way we are able to gather the min/max values that they can reach, and extract the S accordingly. As you can notice, the weights are divided into 2 sets: U (to accumulate with the input) and W (to accumulate with the recursive state). They are quantized separately with a single value (per tensor) Scaling factor S_u and S_w, we noted that this approach offers the best tradeoff between accuracy and performance in the quantized inference. The Non-Linear operations ($sigmoid$ and $\tanh$) are fundamental in the numerical accuracy of the LSTM, they have been implemented with a fast 256 point look up table which offers <1e-3 error wrt the floating point numpy version.

Experimental results

The proposed end-to-end SLU is trained from scratch and evaluated on the Fluent Speech Commands (FSC) dataset [4], which contains 30,043 utterances from 97 speakers (train:77, valid:10, test:10). This dataset is designed for controlling smart home appliances and visual assistances with three intents or slots; like: "turn on the light in the kitchen": {'**Action**': activate, '**Object**': lights, and '**Location**': kitchen}. For acoustic features, 40-dimensional Mel-frequency cepstral coefficients (MFCCs) are extracted with window size and hop length of 25 and 5 *ms*, respectively. For 32-bit floating point arithmetic, the proposed solution achieves an accuracy of 95% and 94% for the train and test datasets, respectively. After the quantization process, the accuracy drops to 93.88% (less than 1% reduction).

Conclusions

The proposed end-to-end SLU system achieves state-of-the-art performance with an accuracy of ~94%, while respecting the real-time and edge computing power consumption constraints with an average total power of 7.1 and 0.5 mw in ultra-low power mode of GAP8 and GAP9, respectively. The quantized model efficiently runs on GAP9, being capable of processing over 4 millions of spoken commands (average of 2 seconds each) with a single AA battery (1.5V&1700mAh).

References

Video Demo

[1]. Lugosch, Loren, et al. "Speech model pre-training for end-to-end spoken language understanding." *arXiv preprint arXiv:1904.03670* (2019).
[2]. https://greenwaves-technologies.com/manuals/BUILD/AUTOTILER/html/index.html
[3]. Jacob, Benoit, et al. "Quantization and training of neural networks for efficient integer-arithmetic-only inference." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
[4]. fluent.ai/research/fluent-speech-commands/