Interspeech 2018
2-6 September 2018, Hyderabad, India

Conditional-Computation-Based Recurrent Neural Networks for Computationally Efficient Acoustic Modelling

Raffaele Tavarone[1], *Leonardo Badino*[1]

[1]Center for Translational Neurophysiology of Speeech and Communication

raffaele.tavarone@iit.it, leonardo.badino@iit.it

Abstract

The first step in Automatic Speech Recognition (ASR) is a fixed-rate segmentation of the acoustic signal into overlapping windows of fixed length. Although this procedure allows to achieve excellent recognition accuracy, it is far from being computationally efficient, in that it may produce a highly redundant signal (i.e, almost identical spectral vectors may span many observation windows) that converts into computational overload. The reduction of such overload can be very beneficial for application such as offline ASR on mobile devices.

In this paper we present a principled way for saving numerical operations during ASR by using conditional-computation methods in deep bidirectional Recurrent Neural Networks (RNNs) for acoustic modelling. The methods rely on learned binary neurons that allow hidden layers to be updated only when necessary or to keep their previous value.

We (i) evaluate, for the first time, conditional computation-based recurrent architectures on a speech recognition task, and (ii) propose a novel model specifically designed for speech data that inherently builds a multi-scale temporal structure in the hidden layers. Results on the TIMIT dataset show that conditional mechanisms in recurrent architectures can reduce hidden layer updates up to 40% at the cost of about 20% relative phone error rate increase.

Index Terms: speech recognition, computational efficiency, conditional computation, recurrent neural network.

1. Introduction

In recent years deep bidirectional RNN's have set the state-of-the-art performance in several ASR tasks [1, 2, 3, 4, 5, 6]. RNNs are indeed intrinsically well suited to deal with the dynamical structure of speech due to their internal memory, which allows to capture long-term dependencies in the input signal.

Whether an RNN is integrated into a hybrid RNN-HMM ASR system or is itself an end-to-end [7] ASR system, its input is the result of a fixed-rate (typically 100Hz), fixed window length (typically 25ms) windowing of the acoustic signal.

Although the fixed-rate segmentation scheme allows excellent performances in many ASR task, it has some drawbacks. For example, relevant but very short-timed event within a speech signal (*e.g.* peak or stop consonant) may be averaged out in a too-long observation window [8], a problem that we will not address here. In the opposite case, long duration speech sounds "sampled" every 10ms generate redundant vectors of spectral features (*e.g.*, redundant vectors of MFCCs). During recognition, this redundancy in the input features directly translates into a remarkable amount of superfluous numerical operations. In the past, the limitations imposed by the fixed-rate segmentation scheme inspired the so called segment-based ASR methods [9, 10, 11] that, in contrast to our approach, do not focus primarily on computational efficiency as they typically require many more decoding operations than frame-based systems.

Additionally, the multi-timescale structure of speech [12, 13] along with the multiple levels of abstraction learned by deep neural networks [14] suggests that the deeper layers may encode features with a characteristic timescale longer than that of the lower layers. Nonetheless, in standard deep RNN all the hidden layers are updated at the same rate, a possible supplementary source of unnecessary computational cost. All the previous considerations motivated us to apply conditional computation methods in deep bidirectional RNNs to perform computationally efficient ASR.

The idea of introducing conditional computation in deep neural networks is relatively recent [15, 16]. It consists in the dropping of a portion of the network internal computations in a dynamic (as it depends on time-varying conditions) and learned (as opposed to dropout) fashion. Conditional computation has been used to increase computationally efficiency in feed-forward neural networks [17], to train deep mixture-of-experts [18] and to efficiently control the updating schedule of recurrent neural network [19, 20, 21]. Previous work on computationally efficient feed-forward deep neural networks for ASR relies on low-rank matrix factorization to reduce the model size [22], teacher-student paradigm [23], and multiframe methods where the network either simply skips labels estimations one over two frames or predicts labels of both next and current frames [24]. Another relevant approach is time delay neural networks, whose hidden activations can be sub-sampled following a predefined, not learned, scheme [25].

In this work, for the first time, we apply conditional computation on deep bidirectional RNNs for acoustic modelling. We considered bidirectional implementations of two conditional architectures, Hierarchical-Multiscale LSTM (HM-LSTM) [19] and Skip-GRU [21] and propose a variant of the HM-LSTM, named constrained Hierarchical-Multiscale Hard-Gated Recurrent Unit (cHM-HGRU).

While in a Skip-GRU the decision on whether to update or skip updating (thus keeping previous values) applies to all its hidden layers at once, in a HM-LSTM the decision is applied to each hidden layer separately. Despite HM-LSTM outperformed Skip-GRU in our experiments, it often exhibited an unstable training and actually performed a hidden layer pruning rather than extracting a hierarchical multi-scale temporal representation.

We hypothesize that the much noisier nature of the speech signal and the lack of clear boundaries as opposed to, *e.g.*, text (where word, phrase and sentence boundaries are usually marked by special characters) hamper HM-LSTM training and some prior knowledge of the structure of speech has to be integrated (especially when training data is not huge as in our experiments). To this end we propose a new model, cHM-HGRU, that applies changes to the original HM-LSTM, including simple priors on inter-dependencies between gates of different lay-

10.21437/Interspeech.2018-2195

ers.

We integrated all three architectures in a hybrid RNN-HMM system and tested their accuracy and capability of saving computation in a phone recognition task on TIMIT.

2. Models

2.1. Proposed model

Our proposed model is based on Gated Recurrent Units (GRUs). A GRU is a recurrent unit that handles the vanishing/exploding gradient problem [26] by employing a gating mechanism that updates state $\mathbf{h}_t$ by modulating the memory carried by the recurrent connections. The GRU architecture is defined as follows:

$$\mathbf{h}_t = (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{1a}$$

$$\mathbf{z}_t = \sigma(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{1b}$$

$$\mathbf{r}_t = \sigma(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1} + \mathbf{b}_r) \tag{1c}$$

$$\tilde{\mathbf{h}}_t = \tanh(W_h \mathbf{x}_t + U_h \mathbf{r}_t \odot \mathbf{h}_{t-1} + \mathbf{b}_h)\,, \tag{1d}$$

where $\mathbf{x}_t \in \mathbb{R}^D$, $\{\mathbf{h}_t, \mathbf{z}_t, \mathbf{r}_t, \tilde{\mathbf{h}}_t, \mathbf{h}_t\} \in \mathbb{R}^d$, $W \in \mathbb{R}^{d \times D}$, $U \in \mathbb{R}^{d \times d}$, and $\mathbf{1} \in \mathbb{R}^d$ is a vector of all ones. $\odot$ denotes element-wise multiplication and $\sigma(x)$ is the sigmoid function (applied element-wise). In a GRU, the update gate $\mathbf{z}_t$ and the reset gate $\mathbf{r}_t$ are learned to filter the information contained in the previous state $\mathbf{h}_{t-1}$. These gates are soft, thus a previous state is never entirely conveyed and Eqs. (1) are evaluated anew even when the speech signal is redundant. Here we formalize a model that implements computational efficiency by using hard-gated recurrent units.

Our model, cHM-HGRU, is a bidirectional variant of the HM-LSTM [19] that (i) is based on GRUs rather than LSTM cells in order to reduce computation, number of parameters and simplify the hard-gating mechanisms and (ii) imposes a multi-scale structure of the internal states of the network by constraining deeper layers to be updated less frequently than lower ones.

For sake of space we describe cHM-HGRU starting from the GRU definition rather than from HM-LSTM. Given a vertical stack of L recurrent layers, the update rule of the l-th hidden layer, $\mathbf{h}_t^l \in \mathbb{R}^d$, at time t is defined, for the forward network, as:

$$\mathbf{h}_t^l = (1 - z_t^l)[(1 - z_t^{l-1})\mathbf{h}_{t-1}^l + z_t^{l-1}\mathbf{u}_t^l] + z_t^l \mathbf{f}_t^l\,, \tag{2}$$

with

$$\mathbf{u}_t^l = \tanh\left(\mathcal{LN}(U_{l-1}^l \mathbf{h}_t^{l-1} + U_l^l(\mathbf{r}_t^l \odot \mathbf{h}_{t-1}^l))\right) \tag{3a}$$

$$\mathbf{r}_t^l = \sigma\big(\mathcal{LN}(R_{l-1}^l \mathbf{h}_t^{l-1} + R_l^l \mathbf{h}_{t-1}^l)\big) \tag{3b}$$

$$\mathbf{f}_t^l = \tanh\left(\mathcal{LN}(W_{l+1}^l \mathbf{h}_{t-1}^{l+1} + W_{l-1}^l \mathbf{h}_t^{l-1})\right). \tag{3c}$$

Assuming that all layers have the same dimensionality, $U_h^k \in \mathbb{R}^{d \times d}$, $W_h^k \in \mathbb{R}^{d \times d}$, $R_h^k \in \mathbb{R}^{d \times d}$ for the weights matrices connecting layer h with layer k, and $\mathbf{u}_t^l \in \mathbb{R}^d$, $\mathbf{r}_t^l \in \mathbb{R}^d$, and $\mathbf{f}_t^l \in \mathbb{R}^d$. The function $\mathcal{LN}(\,)$ performs layer normalization as in [27] by scaling and recentering the pre-activations. The backward recurrent network is defined analogously over the reversed input sequence.

The term in square brackets on the right-hand side of (2) is analogous to (1a) of the standard GRU, with the sigmoid vector controlling the gate replaced by a binary scalar. The z's in (2) are binary units that can be broadly interpreted as boundary detectors and are defined as:

$$z_t^l = z_t^{l-1} f_{\text{round}}(\tilde{z}_t^l) \tag{4}$$

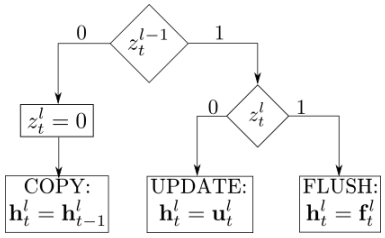

Figure 1: *Gating mechanism of the proposed model.*

with

$$\tilde{z}_t^l = \texttt{hardsigm}(V_l^l \mathbf{h}_{t-1}^l + V_{l-1}^l \mathbf{h}_t^{l-1} + b_z^l)\,. \tag{5}$$

We use $V_h^k \in \mathbb{R}^{1 \times d}$, $b_z \in \mathbb{R}$ and $f_{\text{round}}(x)$ is given by

$$f_{\text{round}}(x) = \begin{cases} 1 & \text{if } x \geq 0.5 \\ 0 & \text{if } x < 0.5\,. \end{cases} \tag{6}$$

The hard-sigmoid function, $\texttt{hardsigm}(x) = \max(0, \min(1, \frac{(ax+1)}{2}))$, is a piecewise-linearized version of the standard sigmoid with slope a, previously used for binary neurons [17]. The layer normalization in Eqs. (3a-3c) is essential to avoid quick saturation of the binary units during training. Hard gates can be interpreted as boundary detectors, where $z_t^l = 1$ corresponds to the detection of a boundary at time t and at layer l.

The presence of z_t^{l-1} in (4) is the second main difference w.r.t. HM-LSTM. It imposes a constraint on the boundary detection in that the model is allowed to find a boundary at layer l only if a boundary is present at the layer below. In our experiments we found that, despite being formally modest, this constraint is essential to obtain a proper hierarchical-multiscale structure of the network when dealing with speech data.

Like HM-LSTM, cHM-GRU has three operational modalities : COPY, UPDATE and FLUSH (see Fig. 1). The COPY operation is performed when no boundary is detected at the lower layer, which also implies no boundary at the current level because of (4). COPY is thus activated when no relevant changes are detected and it copies $\mathbf{h}_{t-1}^l$ into $\mathbf{h}_t^l$.

When a boundary is detected in the layer below but there is no boundary in the current layer, an UPDATE is performed, which resembles a standard recurrent operation.

Finally, the FLUSH operation is performed when a boundary is detected at both levels. The additional boundary at layer l cuts the recurrent connection and sends to the current layer a summary of the previous states only through the top-down connection (when $l < L$). To understand the possible utility of a top-down connection in the FLUSH operation, imagine $\mathbf{h}_{t-1}^l$ being the encoding of some kind of speech unit (*e.g.*, something resembling a phonetic state) and $\mathbf{h}_{t-1}^{l+1}$ the encoding of it plus its left context. When a boundary is detected, a new encoding $\mathbf{h}_t^l$ for the new unit has to be computed, and the computation has to take into account the co-articulation effects due to the left context.

By disregarding the recurrent links, the FLUSH operation introduces a trade-off mechanism between the number of recognized boundaries and the contextual information carried along by the network. As every boundary can trigger either an UPDATE or a FLUSH operation, the model has to find a balance between the number of state updates and the discard of past

knowledge. Indeed, when experimenting with an hard-gated recurrent unit without the flush module we found that the model either performs very poorly or does not save any computation.

Assuming that the classification problem has n classes, the output $\mathbf{y}_t \in \mathbb{R}^n$ of the model is a non-linear classifier that takes as input all the layers in the hierarchy from both the forward and the backward nets. It is defined as:

$$\mathbf{y}_t = \texttt{softmax}\big(\texttt{ReLu}\big(\sum_{l=1}^{L}(O^l_{\text{fw}}\overrightarrow{\mathbf{h}}^l_t + O^l_{\text{bw}}\overleftarrow{\mathbf{h}}^l_t)\big)\big) \quad (7)$$

with $O^l_{\text{fw}}, O^l_{\text{bw}} \in \mathbb{R}^{n\times d}$, $\texttt{softmax}(y_j) = e^{y_j}/\sum_{k=1}^{N} y_k$ and where $\texttt{ReLu}(x) = \max(0, x)$. The $\overrightarrow{\mathbf{h}}^l_t$ and $\overleftarrow{\mathbf{h}}^l_t$ refer to the hidden layers of the forward and backward network, respectively. Note that in (7) we do not weight each layer with a different scalar function as in [19] because in our experiments this appears to induce a tremendous pruning of the deeper layers, thus compromising the hierarchical structure.

2.2. Previous work models

Considered previously proposed models are HM-LSTM and Skip-GRU. Because of space constraints HM-LSTM cannot be described here, but an idea of their structure can be inferred from previous section and a comprehensive account can be found in [19].

Skip-GRU , instead of detecting boundaries at each hidden layer, directly learns to skip input frames (which is equivalent to detecting a boundary shared by all the hidden layers). We give here a short summary of the method and refer the interested reader to [21] for details. The state $\mathbf{s}_t$ of the Skip-GRU is defined as

$$\mathbf{s}_t = u_t\mathbf{h}_t + (1-u_t)\mathbf{s}_{t-1} \quad (8)$$

where $\mathbf{h}_t$ is a standard GRU state as in (1a) and

$$u_t = f_{\text{round}}(\tilde{u}_t)\,, \quad (9a)$$

$$\Delta\tilde{u}_t = \sigma(W_p\mathbf{s}_t + b_p)\,, \quad (9b)$$

$$\tilde{u}_{t+1} = u_t\Delta\tilde{u}_t + (1-u_t)(\tilde{u}_t + \min(\Delta\tilde{u}_t, 1-\tilde{u}_t))\,. \quad (9c)$$

Thus, the Skip-GRU either (i) evaluates a standard GRU or (ii) copies the previous state with a probability that decreases with the number of consecutively skipped timesteps. The overall number of skipped samples is controlled during training by an hyper-parameter λ that weights a secondary term of the objective function, $L_{\text{skip}} = \lambda\sum_t u_t$. We implement a bidirectional version of the model by linearly combining the outputs of the independent forward and backward RNNs.

3. Experimental setup

3.1. Phone recognition systems

The phone recognition systems are hybrid RNN-HMM systems trained and evaluated on the TIMIT dataset. Phone state alignments for RNN training where obtained using the Kaldi [28] s5 recipe. Triphone state alignments computed using the tri3 models were mapped into monophone state alignments. Input features are 40 mel-filter banks plus deltas and delta-deltas.

3.2. RNN architectures

We evaluated three bidirectional RNN architectures with conditional computation: HM-LSTM [19], our cHM-HGRU, and Skip-GRU. These models are compared to two standard recurrent architectures, bidirectional LSTM with peephole connections [2] and bidirectional GRU. In both standard RNNs, the input of a hidden layer is the combination of the forward and backward outputs of the layer below. In the Hierarchical-Multiscale architectures, the top-down connection forces the forward and the backward networks to be independent.

3.3. Training strategy

All networks were trained using the cross-entropy loss and cross-entropy was minimized using standard back-propagation through time.

However, for conditional computation networks, the derivative of (6) is almost always zero except at the threshold, where it is infinite. Hence, the gradient can newer flow through it. To handle this problem several methods have been employed in previous work [16, 17, 29, 19]. Here we use a strategy called *straight-through estimator* that consists in simply replacing the derivative of the non-smooth function with

$$\frac{\mathrm{d}f_{\text{round}}(x)}{\mathrm{d}x} = 1\,. \quad (10)$$

Thus, in the backward pass we just propagate the gradient through the $f_{\text{round}}(x)$ as if it were the identity function. This method is known to introduce a bias in the gradient estimation, but it has nonetheless been proven to ensure effective training [16, 17, 29, 19]. Additionally, we gradually increase the slope a of the hard-sigmoid function following a predefined schedule so that it gradually becomes closer to the step function, a method known as *slope-annealing trick*.

The model's parameters were updated using *Adam* optimizer [30] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1.0\text{e}-08$, and learning rate 10^{-4} over batches of one utterance. All weights were initialized by random sampling from an uniform distribution between -0.1 and 0.1. All biases were initialized to 0.0 except for the biases of the reset gates that were initialized to 1.0. To implement the slope-annealing trick we started with $a = 1.0$ and add $3.0\text{e}-05$ at every optimization step. We used early stopping on the development set.

4. Results

Table 1 summarizes our results. The reported frame-level error rate (FER) and phone error rate (PER) refer to an average over 3 runs with different random initialization. We designed our model to be used in a generic mini-batch setting, where an efficient implementation of conditional operations could not be implemented. Thus, for each layer we report the test set average of the percentage of copies-per-layer (defined as the percentage of copies over the total number of frames in an utterance, averaged on the forward and backward networks) as a proxy for computational efficiency.

The best-performing model is the bidirectional LSTM, which is also the model with the largest number of parameters. The bidirectional GRU reduces the number of parameters by $\approx 38\%$ while giving a 4.3% relative increase in PER. Both models however evaluate the full set of parameters at every timestep (percentage of copies-per-layer is always 0).

The bidirectional HM-LSTM achieved a 22.8% PER on the test set, but a closer investigation reveals that the expected internal hierarchical structure is wiped out in this task. In a proper multiscale scenario [19] the number of copies-per-layer must monotonically increase when moving from lower to deeper lay-

Table 1: *Average FER, PER and rate of per-layer copies on TIMIT development and test set.*

Model	L	d	Num. Params.	% Copies per layer (Test-set average) $l=1$	$l=2$	$l=3$	$l=4$	$l=5$	Dev FER (%)	Test FER (%)	Dev PER (%)	Test PER (%)
Bi-LSTM [2]	5	250	8M	0.0	0.0	0.0	0.0	0.0	30.7 ± 0.2	31.8 ± 0.2	17.8 ± 0.1	18.7 ± 0.2
Bi-GRU	5	250	5M	0.0	0.0	0.0	0.0	0.0	31.8 ± 0.2	32.9 ± 0.3	17.8 ± 0.1	19.5 ± 0.2
Bi-HM-LSTM	5	250	8M	0.0	2.8	17.5	99.4	83.0	35.6 ± 1.5	36.2 ± 1.3	21.6 ± 2.0	22.8 ± 1.7
Bi-cHM-HGRU	5	250	4M	0.0	16.1	43.7	62.9	73.2	35.7 ± 0.1	36.3 ± 0.0	21.3 ± 0.3	22.5 ± 0.4
Bi-Skip-GRU $\lambda = 1.0\text{e}{-04}$	5	250	4M	10.1	10.1	10.1	10.1	10.1	35.5 ± 0.3	36.1 ± 0.1	20.9 ± 0.1	22.5 ± 0.0
Bi-Skip-GRU $\lambda = 5.0\text{e}{-04}$	5	250	4M	38.2	38.2	38.2	38.2	38.2	37.9 ± 0.0	38.8 ± 0.0	22.1 ± 0.4	23.4 ± 0.1
Bi-Skip-GRU $\lambda = 1.0\text{e}{-03}$	5	250	4M	59.3	59.3	59.3	59.3	59.3	41.1 ± 0.0	41.7 ± 0.0	23.1 ± 0.5	24.7 ± 0.5

ers because the corresponding hierarchical levels span increasingly long time intervals. As can be seen in Table 1, the number of copies-per-layer in the HM-LSTM is unstructured. Moreover, it turned out that the model tends to perform a pruning of the deeper layers instead of building an internal multiscale structure, which sometimes results in numerical instability during training.

Our interpretation of this result is based on the observation that in all previous tasks where the HM-LSTM has been tested [19], the learned boundaries are aligned with a well defined break-point in the sequential data (*e.g.* a blank character in the character-level language model). The speech signal does not have such clear markers of its structure, which we speculate would act as anchor points for the boundary detectors. We cannot exclude however that the small size of TIMIT also plays a role in the vanishing of the multiscale organization.

The bidirectional cHM-HGRU, reaches a 22.5% PER while simultaneously halving the number of parameters to ≈ 4 millions and recovering a proper hierarchical multiscale structure. The number of copies-per-layer gradually increases with the layer depth until it reaches the $\approx 73\%$ of copies on the deeper layer. To better demonstrate the hierarchical structure of the internal states we plot them in Fig. 2 for a portion of a TIMIT test utterance.

Finally, we report the performances of the bidirectional Skip-GRU with three different values of the hyper-parameter λ that controls the computational budget. Since Skip-GRU learns to fully skip input frames, the percentage of copies is the same for all layers. As expected, PER increases with increasing computational saving rates. At an approximately equal overall computational load, Skip-GRU performs worse than cHM-HGRU and HM-LSTM.

In the future we plan to evaluate the performances of the presented architectures over larger dataset and to integrate conditional computation into end-to-end training strategies as connectionist temporal classification [3] or sequence-to-sequence models [31], where we expect the stable hierarchical structure of the cHM-HGRU to be beneficial.

5. Conclusion

In this work we evaluated, for the first time, conditional computation deep RNN for acoustic modelling. Our aim was to exploit conditional computation to increase computational efficiency during inference. We experimented with 2 previous work models, Hierarchical-Multiscale LSTM and Skip-GRU, and a proposed variant of HM-LSTM, namely constrained Hierarchical-Multiscale Hard-GRU, on a phone recognition task on the TIMIT dataset. Their accuracy and computational efficiency was compared to those of standard LSTM and GRU.

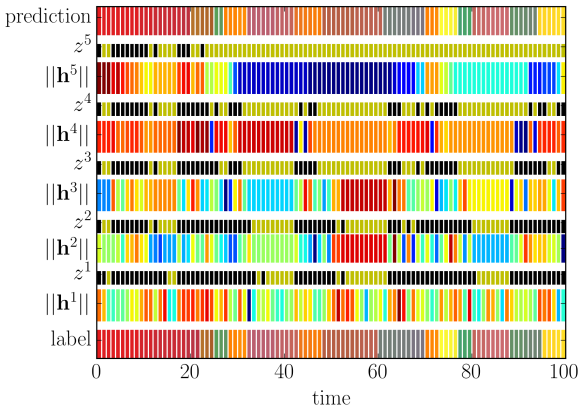

Figure 2: *Internal states of our proposed model (forward network) versus time for a portion of a TIMIT test utterance. Labels and predictions are shown at the bottom and at the top of the plot respectively, with a different color for each classe. The color-coded L_1 norms of the hidden layers are used to succinctly visualize their internal activation in a single scalar. The z's are black when $z = 1$ and yellow when $z = 0$. The COPY operation is performed only when $z^{l-1} = 0$ and $z^l = 0$, resulting in no change of color of $||\mathbf{h}^l||$.*

Both HM-LSTM and cHM-HGRU outperform Skip-GRU while matching its computational load. Results show that while our proposed model cHM-HGRU matches HM-LSTM accuracy it has the following advantages: (i) it retains a proper hierarchical multiscale structure, (ii) it exhibit stable training, and (iii) it has half of the parameters. All of the discussed methods fit especially well for applications where a small portion of accuracy can be dropped in exchange for numerical efficiency, as in offline ASR for mobile devices.

6. Acknowledgements

The authors acknowledge the support of the European Unions Horizon2020 project ECOMODE (grant agreement No 644096) and thank Chiara Bartolozzi for inspiring discussions.

7. References

[1] D. Yu and L. Deng, *Automatic Speech Recognition - A Deep Learning Approach*. Springer, 2016.

[2] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 273–278.

[3] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[4] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[5] H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao, "Sequence discriminative distributed training of long short-term memory recurrent neural networks," in *Fifteenth annual conference of the international speech communication association*, 2014.

[6] F. Weninger, J. Geiger, M. Wöllmer, B. Schuller, and G. Rigoll, "Feature enhancement by deep lstm networks for asr in reverberant multisource environments," *Computer Speech & Language*, vol. 28, no. 4, pp. 888–902, 2014.

[7] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 167–174.

[8] B. Lee and K.-H. Cho, "Brain-inspired speech segmentation for automatic speech recognition using the speech envelope as a temporal reference," *Scientific reports*, vol. 6, p. 37647, 2016.

[9] O. Abdel-Hamid, L. Deng, D. Yu, and H. Jiang, "Deep segmental neural networks for speech recognition." in *Interspeech*, vol. 36, 2013, p. 70.

[10] L. Lu, L. Kong, C. Dyer, N. A. Smith, and S. Renals, "Segmental recurrent neural networks for end-to-end speech recognition," in *Interspeech 2016*, 2016, pp. 385–389. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2016-40

[11] J. R. Glass, "A probabilistic framework for segment-based speech recognition," *Computer Speech & Language*, vol. 17, no. 2-3, pp. 137–152, 2003.

[12] A.-L. Giraud and D. Poeppel, "Cortical oscillations and speech processing: emerging computational principles and operations," *Nature neuroscience*, vol. 15, no. 4, p. 511, 2012.

[13] J. E. Peelle and M. H. Davis, "Neural oscillations carry speech rhythm through to comprehension," *Frontiers in psychology*, vol. 3, p. 320, 2012.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[15] Y. Bengio, "Deep learning of representations: Looking forward," in *International Conference on Statistical Language and Speech Processing*. Springer, 2013, pp. 1–37.

[16] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[17] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.

[18] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[19] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," *arXiv preprint arXiv:1609.01704*, 2016.

[20] A. S. Davis and I. Arel, "Faster gated recurrent units via conditional computation," in *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on*. IEEE, 2016, pp. 920–924.

[21] V. Campos, B. Jou, X. Giró-i Nieto, J. Torres, and S.-F. Chang, "Skip rnn: Learning to skip state updates in recurrent neural networks," *arXiv preprint arXiv:1708.06834*, 2017.

[22] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 6655–6659.

[23] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size dnn with output-distribution-based criteria," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[24] V. Vanhoucke, M. Devin, and G. Heigold, "Multiframe deep neural networks for acoustic modeling," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 7582–7585.

[25] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Interspeech*, 2015.

[26] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[27] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[28] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[29] A. Vezhnevets, V. Mnih, S. Osindero, A. Graves, O. Vinyals, J. Agapiou *et al.*, "Strategic attentive writer for learning macro-actions," in *Advances in neural information processing systems*, 2016, pp. 3486–3494.

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

[31] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4960–4964.