INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Ultra-Compact NLU: Neuronal Network Binarization as Regularization

Munir Georges[1], *Krzysztof Czarnowski*[1], *Tobias Bocklet*[1,2]
[1]Intel Corporation
[2]Technische Hochschule Nürnberg
forename.surname@intel.com

Abstract

This paper describes an approach for intent classification and tagging on embedded devices, such as smart watches. We describe a technique to train neuronal networks where the final neuronal network weights are binary. This enables memory bandwidth optimized inference and efficient computation even on constrained/embedded platforms.

The flow of the approach is as follows: tf-idf word selection method reduces the number of overall weights. Bag-of-Words features are used with a feedforward and recurrent neuronal network for intent classification and tagging, respectively. A novel double Gaussian based regularization term is used to train the network. Finally, the weights are almost clipped lossless to -1 or 1 which results in a tiny binary neuronal network for intent classification and tagging.

Our technique is evaluated using a text corpus of transcribed and annotated voice queries. The test domain is "lights control". We compare the intent and tagging accuracy of the ultra-compact binary neuronal network with our baseline system. The novel approach yields comparable accuracy but reduces the model size by a factor of 16: from 160kB to 10kB.
Index Terms: ASR, NLU, SLU, binary DNN, regularization

1. Introduction

Neuronal network inference on edge devices is becoming increasingly important. This may be because of latency reasons but also because of privacy/legal reasons for certain applications in business, military or healthcare. This paper addresses Natural Language Understanding (NLU) on edge devices where the edge device needs to fulfill low power constrains. Example devices are smart watches, headphones or any handhelds which are usually battery-powered. Implications to the available compute and memory requires trade-off decisions between available memory, memory bandwidth and compute. The proposed NLU comprises an intent classifier and tagger which uses a feedforward and recurrent neuronal network, respectively. A preceding automatic speech recognition engine is used to enable spoken language understanding.

Usually, a neuronal network is trained using stochastic gradient descent with error gradient backpropagation according to some loss function chosen to minimize recognition errors. An already trained network is usually quantized for optimal on-device execution. This reduces memory requirements, e.g., an acoustic model with 32-bit floating point weights can be represented by 8-bit integers. Computationally efficient quantization schemes were proposed by, e.g., Raziel Alvarez et al. [1]. The authors propose to apply the quantization already in training which results in an almost zero accuracy degradation when going from 32-bit to 8-bit representation. It was also proposed to apply weight pruning, e.g., by Song Han et al. [2]. A similar approach was proposed for vision, e.g., by Asit Mishra et al. [3]. A general introduction is provided by, e.g., Hao Li et al. [4] and Yunhui Guo [5].

This paper describes a technique to train neuronal networks where the final neuronal network weights are binary. A binary network can be stored and computed significantly more efficiently on any (embedded) platform such as DSP, CPU or FPGA. Although, we only target edge devices in this paper, there is also strong demand for efficient inference on server platforms as outlined by, e.g., Jongsoo Park et al. [6]. Our proposal to train a binary neuronal network is fully data-driven and follows ideas froms Wei Tang et al. [7]. We define a novel double-Gaussian based regularization term that jointly optimizes the target function as well as emphasizing weight values of -1 and 1. The proposed technique is agnostic to the neuronal network topology which enables its use for any kind of model. No specific support is required from the deep learning library employed, hence it is possible to use it with any popular toolkit such as MXNet, TensorFlow, PyTorch, Caffe, Kaldi, etc.

This paper is organized as follows: Section 2 describes the training process where we introduce a double-Gaussian based regularization term. It jointly learns binary weights representation and the target function. Inference of binary neuronal networks is described in Section 3 and followed by Section 4 of experiments. This paper evaluates ultra-compact NLU on two tasks, intent classification and tagging as presented in subsections 4.1 and 4.2, respectively. The paper finalizes with conclusions.

2. Binarization as Regularization

The weights of a neuronal network are estimated by minimizing a loss function using, e.g., stochastic gradient descent with loss gradient backpropagation. Yunchao Gong et al. [8] proposed k-means clustering to the weights for a convolutional neuronal network (CNN) with $k \geq 2$. A fixed-point network with ternary weights was proposed by Kyuyeon Hwang et al. [9] or Diwen Wan et al. [10]. All weights in a ternary neuronal network are -1, 0 or 1. The network was trained with quantized forward pass, but high-precision updates in the backward pass. Fengfu Li et al. [11] provided an overview on ternary weight for neuronal networks. Reducing the precision of gradients was also explored by, e.g., Shuchang Zhou et al. [12]. The papers often use regularization methods to support quantization, e.g., l_1. A theoretical introduction on group sparse regularization is provided by, e.g, Simone Scardapane et al.[13].

In this paper, we envision binary weight values of -1 and 1. This follows the proposals from Matthieu Courbariaux et al. [14]. We achieve binarization at training time by adding a dedicated regularization term, which makes weight values converge to either -1 or 1. Similar approaches were employed by, e.g., Wei Tang et al. [7] or Sajad Darabi et al. [15]. The last-named paper employed the following loss term:

$$R(w) = (\alpha - |w|)^2 \tag{1}$$

which can be considered as a generalization of standard l_2 regularization to two component case. Parameter α is a scaling factor, for $\alpha = 1$ weights are enforced to be close to -1 or 1. Note that for $\alpha = 0$ we get plain l_2 regularization. This R term was successfully applied, e.g., for vision tasks [15].

In this paper, we derive a generalization of l_2 regularization which follows a probabilistic view. Let us consider multivariate-Gaussian density

$$f_{l2}(w) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{-|w|^2/2\sigma^2} \tag{2}$$

with

$$|w|^2 = \sum_{i=1}^{n} w_i^2 \tag{3}$$

where σ^2 is the variance and n the number of weights in w. Within this notation w represents all combined parameters in the model subject to regularization and each w_i is an individual parameter, i.e. a number. The resulting negative log-likelihood loss term is given by:

$$L_{l2}(w) = \frac{1}{2\sigma^2}|w|^2 + \frac{n}{2}\log(2\pi\sigma^2) \tag{4}$$

and is equivalent to l_2 norm of weights. Formula 2 can be rewritten as a product of individual weight densities:

$$f_{l2}(w) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-w_i^2/2\sigma^2} \tag{5}$$

In our case, binarization enforcing prior is derived from this formula by making each individual weight density a mixture of two Gaussian distributions:

$$f_{\text{bin},\sigma}(w) = \prod_{i=1}^{n} \Big(\frac{0.5}{\sqrt{2\pi\sigma^2}} e^{-\frac{(w_i+1)^2}{2\sigma^2}} + \frac{0.5}{\sqrt{2\pi\sigma^2}} e^{-\frac{(w_i-1)^2}{2\sigma^2}} \Big) \tag{6}$$

This forces weights to take values of -1 and 1 and is a direct generalization of plain l_2, i.e., Gaussian prior. Note, ± 1 can be generalized to $\pm\mu$ for symmetrical weights. Moreover, ± 1 can be generalized to μ_1 and μ_2 which allows asymmetric weights, e.g., -5 and 1. Both generalizations to the mean component are beneficial for some problems which are subject to future research. However, in this paper we use $|\mu| = 1$, so that the resulting loss regularization term is given by

$$\begin{aligned} L_{\text{bin},\sigma}(w) &= -\log f_{\text{bin},\sigma}(w) \\ &= -\sum_{i=1}^{n} \log \Big(\frac{0.5}{\sqrt{2\pi\sigma^2}} e^{-\frac{(w_i+1)^2}{2\sigma^2}} \\ &\qquad + \frac{0.5}{\sqrt{2\pi\sigma^2}} e^{-\frac{(w_i-1)^2}{2\sigma^2}} \Big) \\ &= \sum_{i=1}^{n} L_{\text{bin},\sigma}(w_i) \end{aligned} \tag{7}$$

See Figure 1 for plots with different σ values. The closer σ goes to 0, the sharper is the distributed at the minima. In contrast, a $\sigma \gg 0$ balances all weights mostly identical, i.e., the network will not become binary.

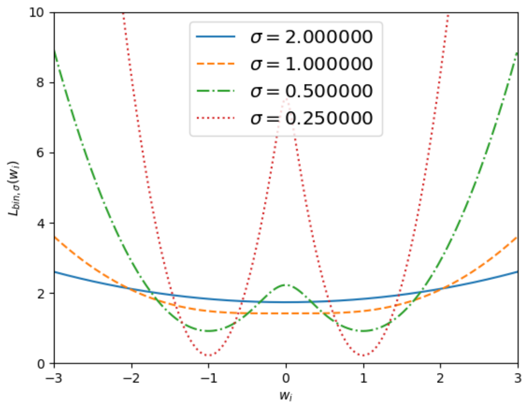

Figure 1: *Binary loss plots for various σ parameter values.*

The final loss used for simultaneous training the model to the target L_{target} while also shaping weight distribution with $L_{\text{bin},\sigma}$, is a convex combination of both. Let ρ be the interpolation weight, $0 < \rho < 1$. Then the overall loss L is given by:

$$L(w) = \rho L_{\text{target}}(w) + (1-\rho) L_{\text{bin},\sigma}(w) \tag{8}$$

We set $\rho = 0.5$ in this paper. Both parameters σ and ρ need to be optimized for the particular task or even controlled dynamically during training for optimal results. In this paper, the target loss L_{target} computes the cross entropy to address the NLU classification tasks. However, it can be any loss function such as ”logistic loss” or ”mean square error”.

It is also common to use different precision for different neuronal network layers, i.e., the input layer uses full-precision, the second layer weights are represented in low-precision, e.g., 8-bit quantized or binary weights. Using layer dependent precision was also studied, e.g., by Patrick Judd et al. [16]. Moreover, the authors propose to alternate quantization, forward computation, backward and weight update during the training process. In summary, it is up to the target task which layers are best binarized. A good rule of thumb is to start with binary representation of hidden layers of an, e.g., feedforward network.

Finally, our approach makes it possible to apply a per layer trainable component standard deviation σ of the double Gaussian distribution. This provides a measure of how well different parts of the network can be binarized. Note, that some regularization to prefer solutions with small σ is required. Additionally, as a generalization, the component means can be made trainable as they are also part of the compute graph, say μ_1 and μ_2 (μ for the symmetric case). Some regularization making 1 a preferred prior μ_1 and μ_2 may be required. We do not explore this in this paper, but this approach gives one more degree of freedom as it will change the position of the local minima. This might support the training process for certain activation functions and certain tasks.

3. Packaging and Inference

A neuronal network comprises matrices, vectors and a topology description. The description defines what operators are applied to the matrices and vectors, respectively. It is most often almost hard coded on resource constrained platforms such as DSPs, i.e., the neuronal network layers that are available. This paper uses binary neuronal networks where all weights of the matrices and vectors are -1 or 1. Zhiyong Cheng et al. [17] investigated the capability of binary networks for image classification.

This paper uses a double Gaussian regularization term $L_{\text{bin}}(\sigma, \mu, w)$ to train the binary neuronal network. Details are described in previous section 2. The result of the training is a network where all weights are very close to either $-\mu$ or μ. However, the weights are still represented in full floating-point precision. We use the following simple clipping rule to move from full precision floating-point weight representations to a binary weight representation:

$$w_i^{\text{bin}} = \begin{cases} -\mu & w_i < 0 \\ \mu & w_i \geq 0 \end{cases} \tag{9}$$

(Note that μ is part of the loss function which can be a trainable parameter using stochastic gradient decent and error back propagation as discussed in the previous section, but often just set to 1. Also c.f. Matthieu Courbariaux et al. [14] for more discussion on the final binarization.)

The matrices and vectors of the resulting neuronal network are stored in a format where one weight takes one bit in memory. This model can be loaded even on platforms with very strong memory bandwidth limitations.

Finally, not just the memory reduction due to the low precision is beneficial. Most common computation is linear (or affine) transformation $y = Wx$, where $x \in \mathbb{R}^N$, $y \in \mathbb{R}^M$ and $W \in \mathbb{R}^{M,N}$:

$$y_i = \sum_{j=1}^{N} w_{i,j} x_j, \quad i = 1, \ldots, M \tag{10}$$

Hence, the dense layer requires $M \cdot N$ multiplications of two floating point weights in full precision. Low precision multiplication application for training deep neuronal networks was proposed, e.g., by Matthieu Courbariaux et al. [18]. However, a binary neuronal network fulfills:

$$w_{i,j} = -1 \text{ or } w_{i,j} = 1 \text{ for all } i, j \tag{11}$$

so that the weight $w_{i,j}$ in Equation 10 computes the sign of the x_j input. The required multiplication of two numbers becomes a sign toggle operator in a binary network, which is very efficiently on any compute platform.

In summary, the dense layer of a binary neuronal network with weight values of -1 and 1 can be computed by a set of sign-toggle and addition operators. This makes inference of binary neuronal network very efficient. Moreover, a binary neuronal network can be inferred by using xor operators. This was proposed by, e.g., Mohammad Rastegari et al. [19] or Itay Hubara et al. [20]. The technique was successfully applied to various tasks including single channel source separation as described by, e.g., Minje Kim et al. [21]

Just for the sake of completeness, Haojin Yang et al. [22] implemented a binary neuronal network with binary activation functions which was later improved by, e.g., Sajad Darabi et al. [15]. Logical operators such as "xnor" and "popcount" are used to implement special layers such as "QActivation", "QConvolution", "QFullyConnected" etc. The activation functions are non-differentiable and gradient discontinuities can be disruptive to learning. Matthieu Courbariaux et al. [23] proposed a weight and activation constrained training of binary neuronal networks. Approximating full-precision weight with a linear combination of multiple binary weight bases was proposed by Xiaofan Lin et al. [24]. Moreover, Liu Zechun et al.[25] proposes to propagate the real-valued activations generated by each 1-bit convolution via a parameter-free shortcut. A different solution was proposed by Yixing Li et al. [26] by applying bit-level sensitivity analysis and bit-level data pruning. Finally, Jorn W. T. Peters et al. [27] proposes a probabilistic training method for neural network with both binary weights and activations. Probably, the most recent paper by Qian Lou et al. [28] proposes a hierarchical deep reinforcement learning based framework for network quantization and binarization.

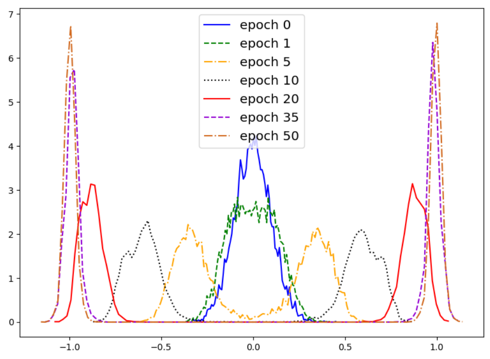

Figure 2: *Weight value histogram as training progresses.*

4. Experiments

We use the MXNet neuronal network training tool which was introduced by Tianqi Chen et al. [29]. All neuronal networks are trained using the Adam optimizer as described by Diederik P. Kingma et al. [30]

Our technique is evaluated on natural language understanding tasks. Section 4.1 shows results on intent classification. It is used to understand the users intention of a voice query, e.g., "it is to dark here" should result in a command that turns on the lights. The results for tagging are described in Section 4.2. Tagging is used to identify certain slots in the voice query, e.g., "reduce brightness to level 5" where "5" is a slot value.

Figure 2 shows the histogram of weight values after some selected training epochs. Epoch 0 is just after initialization. The histogram changes over training time converging to a bimodal distribution as desired. The final epoch 50 shows that most weights are equally distributed at -1 and 1.

4.1. Intent Classification

Spoken language understanding is typically executed in the cloud. However, data privacy regulations as well as increasing demand for cloud offloading motivates execution on low power platforms without any need for cloud connection as proposed by, e.g., Georg Stemmer et al. [31]. It is also in discussion to distribute the computation between embedded devices such as mobile phones and the cloud as proposed, e.g., by Munir Georges et al. [32] or Alice Coucke et al. [33]. Recent development goes in the direction of end-to-end and all neuronal network systems. An example is described by Yanzhang He et al. [34]. The final classification is often done by a feedforward network as proposed, e.g., by Chao Qiao et al. [35] for text classification.

In this paper, intent classification is described by the following conditional probability:

$$\hat{I} = \underset{I \in \text{ Intents}}{\arg\max}\ P(I|\text{words}) \tag{12}$$

where "words" is a vector representation of all recognized words. Here, a bag of word feature vector is used, i.e., the sum of all word vectors which are given by one-hot vectors as described by, e.g., Christopher D. Manning et al. [36].

Let W_1 and W_2 be two weight matrices and let b_1 and b_2 be the corresponding bias vectors. Two layer neuronal network estimates conditional probability of equation 12 as defined by:

$$h = \tanh(W_x x + b_x) \tag{13}$$

$$\hat{I} = \underset{I \in \text{ Intents}}{\arg\max}\ (\text{softmax}(W_h h + b_h)_I) \tag{14}$$

with x being the bag-of-words vector of the voice query, i.e., the result of a speech recognizer.

The intent and tagging task is evaluated on an in-house corpus of transcribed and annotated voice queries. It includes 6 intents and an out-of-intent rejection classes: set brightness, set atmosphere, increase and decrease brightness, turn the lamp on and off. The rejection class helps to suppress out-of-domain input and to properly handle false insertions of the speech recognizer. The in-domain vocabulary has a vocabulary size of 150 words that are selected out of a generic language model corpora with $200k$ vocabulary. We use a tf-idf word selection as described by, e.g., Manning et al. [37]. Table 1 shows results.

Table 1: *Intent classification accuracy*

Type	Memory Size	Accuracy
float weights	160kB	99%
binary weights	10kB	98%
1/2 #weights	5KB	96%
1/2 #weights	1.3KB	89%
1/2 #weights	0.6KB	83%
1/2 #weights	0.3kB	72%

Although the data-set with about 50k unique in-domain sentences is limited and the task is relatively easy, it demonstrates the capabilities of our proposal. The baseline model takes 160kB where all weights are represented by floating point values. The binary neuronal network takes 10kB with no significant accuracy degradation. Further reducing the models by repeatedly shrinking the number of weights by half, results in accuracy degradation. The memory vs. accuracy trade-off depends on the target application. An ultra-compact 0.3kB model achieves still 72% accuracy and can be used for certain always listening applications. The best memory-accuracy trade-off is most likely the 5kB model which achieves 96% accuracy.

4.2. Tagging

Extracting slots from voice queries is important for various command-and-control applications. For example "please play **We Will Rock You** from **Queen**" includes two slots: the music title is '**We Will Rock You**' and the band is '**Queen**', respectively. This task can be represented in various ways, an overview is provided by, e.g., Yoav Goldberg et al. [38], Gkhan Tr et al. [39] or Christian Raymond et al. [40]

We describe the slot extraction as a tagging task which can be formalized as:

$$\hat{s}_t = \underset{s \in \text{ Slot names}}{\arg\max}\ P(s|w_t) \tag{15}$$

with t the word index. This approach was proposed, e.g., by Grégoire Mesnil et al. [41] and [42]. We use also a recurrent neuronal network. Let E be the embedding matrix that maps any one-hot represented word w_i as a continuous vector. The feature vector is a concatenation of embedded word vectors of a sliding window of length n:

$$x = [Ew_{i-n}, ..., Ew_i, ..., Ew_{i+n}] \tag{16}$$

A typical value is $n = 3$ which results in considering the previous and following 3 words of the target word w_i.

$$h_t = \text{sigmoid}(W_x x + W_h h_{t-1} + b_h) \tag{17}$$

$$\hat{s}_t = \underset{s \in \text{ Slot names}}{\arg\max}\ (\text{softmax}(W h_t + b)_s) \tag{18}$$

with $0 < t \leq N$, $h_0 = 0$ where N is the number of words in the sentence. The regularization described in Section 2 is applied to weight matrices W_x, W_h and W, and bias vectors b_h, b.

We evaluate the tagger on the very same in-house natural language understanding corpus as described above. The tagging task is limited to 4 slots, but with a large amount of entities.

Table 2: *Slot classification accuracy*

Type	Memory Size	Accuracy
float weights	1113kB	93%
binary weights	180kB	92%

Results are presented in Table 2. The baseline system with floating point represented weights achieves 93% accuracy. It requires about 1MB of memory to store the neuronal network. The binary recurrent neuronal network weights result in an overall model size of 180KB. It achieves an accuracy of 92% on our test data. The word embedding is for both networks a floating-point valued matrix and takes 150KB; The binarization shrinks the net by a factor of 30. Note that this kind of word embedding can be implemented as a look-up table where no matrix multiplication is required. The index of the word selects the column of the embedding matrix. The network as described by equation 16 and 17 is trained jointly together.

5. Conclusion

We proposed tiny binary neuronal networks for ultra compact NLU on embedded devices. The tininess is achieved by applying tf-idf word selection, shallow network topology and binary weight representation. All weights are quantized to -1 or 1. This was possible by applying a novel regularization term during training the neuronal network weights.

Our approach is agnostic to the network topology and can be applied to any kind of networks. However, further analysis is needed to judge the approach for other tasks. This is subject of our future work where we evaluate binary neuronal networks in the context of speech in general, e.g., speaker identification.

We evaluated the tiny binary neuronal networks for intent classification and slot extraction on lights control task, where we compare it with our baseline system. The text corpus includes transcribed and annotated voice queries. Intent classification achieves an accuracy of 98% (baseline is 99%) with taking 10Kb memory (baseline takes 160Kb). The tagger achieves an accuracy of 92% (baseline is 93%) with taking 180Kb memory (baseline takes 1MB). The proposed binary neuronal models are well suited to speech enabled low-power embedded devices such as a smart watch or headphones.

6. References

[1] R. Alvarez, R. Prabhavalkar, and A. Bakhtin, “On the efficient representation and execution of deep acoustic models,” *Interspeech 2016*, Sep 2016.

[2] S. Han, H. Mao, and W. J. Dally, “Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding,” *CoRR*, vol. abs/1510.00149, 2015.

[3] A. K. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, “WRPN: wide reduced-precision networks,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, Conference Track Proceedings*, 2018.

[4] H. Li, S. De, Z. Xu, C. Studer, H. Samet, and T. Goldstein, “Training quantized nets: A deeper understanding,” in *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 5811–5821.

[5] Y. Guo, “A survey on methods and theories of quantized neural networks,” *CoRR*, vol. abs/1808.04752, 2018.

[6] J. Park, M. Naumov, P. Basu, S. Deng, A. Kalaiah, D. Khudia, J. Law, P. Malani, A. Malevich, S. Nadathur, J. Pino, M. Schatz, A. Sidorov, V. Sivakumar, A. Tulloch, X. Wang, Y. Wu, H. Yuen, U. Diril, D. Dzhulgakov, K. Hazelwood, B. Jia, Y. Jia, L. Qiao, V. Rao, N. Rotem, S. Yoo, and M. Smelyanskiy, “Deep learning inference in facebook data centers: Characterization, performance optimizations and hardware implications,” 2018.

[7] W. Tang, G. Hua, and L. Wang, “How to train a compact binary neural network with high accuracy?” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, California, USA.*, 2017, pp. 2625–2631.

[8] Y. Gong, L. Liu, M. Yang, and L. Bourdev, “Compressing deep convolutional networks using vector quantization,” 2014.

[9] K. Hwang and W. Sung, “Fixed-point feedforward deep neural network design using weights +1, 0, and -1.” in *SiPS*. IEEE, 2014, pp. 174–179.

[10] D. Wan, F. Shen, L. Liu, F. Zhu, J. Qin, L. Shao, and H. Tao Shen, “Tbn: Convolutional neural network with ternary inputs and binary weights,” in *The European Conference on Computer Vision (ECCV)*, September 2018.

[11] F. Li, B. Zhang, and B. Liu, “Ternary weight networks,” 2016.

[12] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, “Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients,” 2016.

[13] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini, “Group sparse regularization for deep neural networks,” *Neurocomput.*, vol. 241, no. C, pp. 81–89, Jun. 2017.

[14] M. Courbariaux, Y. Bengio, and J.-P. David, “Binaryconnect: Training deep neural networks with binary weights during propagations,” 2015.

[15] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia, “BNN+: Improved binary network training,” 2019.

[16] P. Judd, J. Albericio, T. Hetherington, T. Aamodt, N. E. Jerger, R. Urtasun, and A. Moshovos, “Reduced-precision strategies for bounded memory in deep neural nets,” 2015.

[17] Z. Cheng, D. Soudry, Z. Mao, and Z. Lan, “Training binary multilayer neural networks for image classification using expectation backpropagation,” 2015.

[18] M. Courbariaux, Y. Bengio, and J.-P. David, “Training deep neural networks with low precision multiplications,” 2014.

[19] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “Xnor-net: Imagenet classification using binary convolutional neural networks,” *Lecture Notes in Computer Science*, p. 525542, 2016.

[20] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks,” in *Advances in Neural Information Processing Systems 29*, 2016, pp. 4107–4115.

[21] M. Kim and P. Smaragdis, “Bitwise neural networks for efficient single-channel source separation,” *2018 IEEE ICASSP Processing*, pp. 701–705, 2018.

[22] H. Yang, M. Fritzsche, C. Bartz, and C. Meinel, “Bmxnet,” *Proceedings of the 2017 ACM on Multimedia Conference*, 2017.

[23] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1,” 2016.

[24] X. Lin, C. Zhao, and W. Pan, “Towards accurate binary convolutional neural network,” 2017.

[25] Z. Liu, W. Luo, B. Wu, X. Yang, W. Liu, and K.-T. Cheng, “Bi-real net: Binarizing deep network towards real-network performance,” 2018.

[26] Y. Li and F. Ren, “Build a compact binary neural network through bit-level sensitivity and data pruning,” 2018.

[27] J. W. T. Peters and M. Welling, “Probabilistic binary neural networks,” 2018.

[28] Q. Lou, L. Liu, M. Kim, and L. Jiang, “Autoqb: Automl for network quantization and binarization on mobile devices,” *CoRR*, vol. abs/1902.05690, 2019.

[29] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, “Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems,” 2015.

[30] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.

[31] G. Stemmer, M. Georges, J. Hofer, P. Rozen, J. G. Bauer, J. Nowicki, T. Bocklet, H. R. Colett, O. Falik, M. Deisher, and S. J. Downing, “Speech recognition and understanding on hardware-accelerated DSP,” in *Interspeech 2017*, 2017, pp. 2036–2037.

[32] M. Georges, S. Kanthak, and D. Klakow, “Accurate client-server based speech recognition keeping personal data on the client,” in *ICASSP*, 2014.

[33] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet, and J. Dureau, “Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces,” *CoRR*, vol. abs/1805.10190, 2018.

[34] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S. yiin Chang, K. Rao, and A. Gruenstein, “Streaming end-to-end speech recognition for mobile devices,” 2018.

[35] C. Qiao, B. Huang, G. Niu, D. Li, D. dong, W. He, D. Yu, and H. Wu, “A new method of region embedding for text classification,” in *ICLR Proceedings*, 2018.

[36] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999.

[37] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[38] Y. Goldberg and G. Hirst, *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017.

[39] G. Tr, D. Hakkani-Tr, and L. P. Heck, “What is left to be understood in atis?” in *SLT*, D. Hakkani-Tr and M. Ostendorf, Eds. IEEE, 2010, pp. 19–24.

[40] C. Raymond and G. Riccardi, “Generative and discriminative algorithms for spoken language understanding,” in *Interspeech Proceedings*, 2007.

[41] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tr, X. He, L. Heck, G. Tur, D. Yu, and G. Zweig, “Using recurrent neural networks for slot filling in spoken language understanding,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, pp. 530–539, 2015.

[42] G. Mesnil, X. He, L. Deng, and Y. Bengio, “Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding.” in *INTERSPEECH*, 2013, pp. 3771–3775.