9th International IEEE EMBS Conference on Neural Engineering
San Francisco, CA, USA, March 20 - 23 , 2019

Hardware Complexity Analysis of Deep Neural Networks and Decision Tree Ensembles for Real-time Neural Data Classification

Milad Taghavi and Mahsa Shoaran

***Abstract*— A fast and low-power embedded classifier with small footprint is essential for real-time applications such as brain-machine interfaces (BMIs) and closed-loop neuromodulation for neurological disorders. In most applications with large datasets of unstructured data, such as images, deep neural networks (DNNs) achieve a remarkable classification accuracy. However, DNN models impose a high computational cost during inference, and are not necessarily ideal for problems with limited training sets. The computationally intensive nature of deep models may also degrade the classification latency, that is critical for real-time closed-loop applications. Among other methods, ensembles of decision trees (DTs) have recently been very successful in neural data classification tasks. DTs can be designed to successively process a limited number of features during inference, and thus impose much lower computational and memory overhead. Here, we compare the hardware complexity of DNNs and gradient boosted DTs for classification of real-time electrophysiological data in epilepsy. Our analysis shows that the strict energy-area-latency trade-off can be relaxed using an ensemble of DTs, and they can be significantly more efficient than alternative DNN models, while achieving better classification accuracy in real-time neural data classification tasks.**

I. Introduction

Today, machine learning (ML) techniques can be used to interpret complex and noisy sensor data in a variety of applications such as medical devices, wearables, and internet of things (IoT). To enable fast and energy-efficient classification of neural data in real-time applications such as motor decoding for BMI [1] or seizure detection for epilepsy [2], the application-specific integrated circuit (ASIC) implementation of ML algorithms is required. Furthermore, embedded learning at the edge and near the sensors is preferred over the cloud, due to latency or privacy concerns as well as limited communication bandwidth.

Different architectures of DNNs such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have recently been used for neural data classification tasks such as epileptic seizure detection and movement intention decoding [1], [3]. For example, a 5-layer CNN followed by a logistic regressor was used to detect interictal epileptiform discharges from intracranial EEG recordings in [4]. An 8-layer CNN classifier was used to detect the ictal, pre-ictal, and interical periods from scalp EEG following wavelet transform in [5]. To partially relax the large storage requirements of CNNs, an integer CNN architecture for detection of epileptic seizures from scalp or intracranial EEG was proposed in [6]. However, DNN models can be computationally demanding during inference, and require extensive hardware resources and large amounts of memory to store many parameters on chip. Moreover, the computationally intensive nature of DNNs may also degrade the detection latency, that is critical for real-time and closed-loop applications such as responsive stimulation and prosthetic arm control. Support vector machine (SVM) is an alternative classification model that has been used in biomedical system-on-chips (SoCs). However, the size of feature vector in SVM, and consequently, the number of multiplications and additions required for classification linearly increase with number of input channels. Moreover, for cases with a highly nonlinear separation boundary, the use of nonlinear kernels can further increase the hardware complexity of SVM.

Recently, prediction models based on gradient boosting technique [7],[8] have achieved an unprecedented accuracy in ML competitions on Kaggle, such as classification of intracranial EEG data for epilepsy [9]. A combination of gradient boosting (XGBoost) and neural networks was the winning solution to the grasp-and-lift EEG detection contest on Kaggle. This technique employs gradient-based optimization and boosting to form an accurate classification model, by adaptively combining simple weak predictors, typically decision trees. Tree-based classifiers with simple comparators as their processing units are inherently more hardware-friendly compared to DNNs and SVMs [10]. For instance, a 32-channel embedded gradient-boosting classifier for epileptic seizure detection achieved 27× improvement in energy-area-latency product compared to state-of-the-art SVM models [2], [11]. In contrast to DTs, other classification models extract all required features from every input channel, or directly process raw data with intensive computations, which may increase the hardware and memory overhead.

In our previous work [2], [11], we showed the superiority of gradient boosted trees over linear/non-linear SVM and LLS classifiers with a low-power microchip implementation. In this work, we specifically compare the computational complexity and energy-area requirements of neural networks and DT ensembles for neural data classification in real-time applications such as seizure detection. The complexity analysis and results are however applicable to other domains and similar classification tasks.

II. Computational Complexity Analysis

The complexity of a classifier during inference is defined by the number of computational resources and computations required to classify the input data. The memory and hardware requirements can be mathematically formulated by

The authors are with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY. Email: (mt795, shoaran)@cornell.edu.

978-1-5386-7921-0/19/$31.00 ©2019 IEEE

the number of parameters and multiply-accumulate (MAC) operations required to compute the classification model. In the following, we discuss the computational cost of DNN and DT ensemble models.

A. Deep Neural Networks

With multiple stacked layers for feature extraction and transformation, DNNs can model complex and non-linear relationships in data. Convolutional neural network (CNN) is a popular class of deep learning models with translation invariance characteristics, that can extract spatiotemporal features from raw input [12]. The hidden layers in a CNN typically consist of convolutional (CONV), fully connected (FC), and pooling layers [12]. The pooling layers are used to downsample the spatial dimensions of the input, and do not require any parameters to be learned. The computational complexity of pooling layer depends on the type of pooling function (e.g., average or max pooling). A pooling layer of size $u \times v$ downsamples the input feature maps by a factor of uv. In an FC layer, all neurons at the current layer are connected to all neurons of the previous layer. For a neuron with label i at the FC layer l which receives outputs o from neurons in the preceding layer, the propagation function is defined by $n_i = f(\sum_{j=1}^{dim(l-1)} w_{ij} o_j)$, in which f is the activation function (e.g., Sigmoid or ReLU), o_j denotes the output from neuron j at layer $l-1$, and the weight parameters are shown by w_{ij}. Therefore, a total of $dim(l-1)$ MAC operations and parameters are required for each neuron at layer l. Thus, the total number of parameters and MAC operations for the FC layer l would be equal to $dim(l-1) \times dim(l)$.

The CONV layers can extract spatial features from the input, as well as temporal features from time series data. In CONV layers, a group of kernels (filters) are applied to the input, while passing the result to the next layer. Such layers may also provide downsampling (depending on the stride size of the layer) [12]. Let's assume a group of g kernels of size $u \times v$ are applied to f feature maps of dimension $m \times n$, as shown in Fig. 1(a). Here, p_m and p_n are the amount of zero-padding on the borders of input feature maps, while filters are applied with a stride of s. The dimensions of the output feature map in the m and n directions can be written as $o_m = (m-u+2p_m)/s+1$ and $o_n = (n-v+2p_n)/s+1$, respectively, as shown in Fig. 1(b). To reduce the number of parameters required for a CONV layer, a parameter sharing

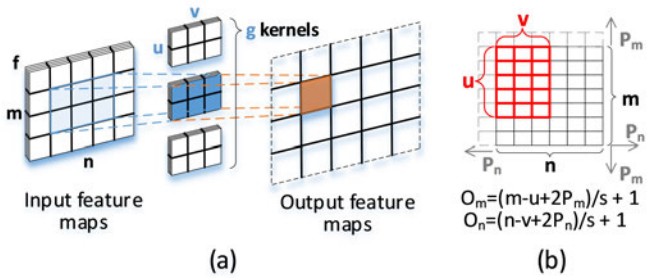

Fig. 1: (a) A group of g kernels with a size of u×v and depth of f are applied to f input feature maps of size m×n; (b) sliding of filters on input data.

TABLE I: Computational complexity of CNN layers and DT ensemble.

	Convolutional Neural network		DT ensemble
	FC layer	CONV layer	
MAC count	$dim(l-1) \times dim(l)$	$f \times u \times v \times g \times o_m \times o_n$	$k \times l \times (t+2) \times N_s$*
Parameters	$dim(l-1) \times dim(l)$	without sharing: $(f \times u \times v+1) \times g \times o_m \times o_n$ with sharing: $(f \times u \times v+1) \times g$	$k \times (2^l-1) \times 3 + T$

* Two extra additions are required for classification and thresholding.

scheme has been proposed, in which the neurons of each output feature map share the same weights and bias for filtering the inputs. Thus, the total number of weights and bias values with and without sharing can be written as:

$$\begin{aligned} \text{with sharing}: &\quad (f \times u \times v+1) \times g \\ \text{without sharing}: &\quad (f \times u \times v+1) \times g \times o_m \times o_n \end{aligned} \tag{1}$$

To calculate the input to the activation function for each element in the output feature maps, $f \times u \times v$ MAC operations are required. Thus, the total number of MACs required to calculate the output of a CONV layer with a group of g filters is given by:

$$f \times u \times v \times g \times o_m \times o_n \tag{2}$$

B. Decision Tree Ensembles

Decision trees are obtained by recursively partitioning the data space based on a sequence of queries, in the form of comparison to a threshold. For a trained model, successive comparisons are then performed on input features during inference, that start at the root node and terminate in a leaf node. To improve the classification performance, various ensemble methods such as gradient boosting and random forest have been widely used. In gradient boosting, multiple trees are built in a greedy fashion to minimize a regularized objective on the training loss, while the output of classifier is defined by a weighted additive combination of individual trees [2]. Given the sequential process of decision making in a tree, only a subset of nodes (along the path from root to the leaf) will be visited during the top-down flow. As a result, a DT would only require the extraction of a limited number of features for classification of input data [2].

In the case of epileptic seizure detection, the input feature vector consists of spectral power features from input channels. Therefore, for comparison at each node of a tree, the channel and feature number, and corresponding threshold value need to be determined. As a result, in a k-size ensemble of trees with a depth of l (i.e., 2^l-1 nodes), the total number of parameters required for inference is equal to:

$$k \times (2^l-1) \times 3 \tag{3}$$

In addition, a total of T coefficient parameters for FIR filters in non-overlapping bands is required, that is stored and shared among trees, as shown in the hardware architecture of Fig. 2(b).

The maximum number of MAC operations per classification in a decision tree is for the case when all the active queries are on spectral power features, as the most computationally intensive attribute. Therefore, assuming that feature extraction is done with t-tap FIR filters followed by an energy extractor, the maximum number of MAC operations required

for feature extraction in a k-size ensemble of trees with a depth of l can be written as $k \times l \times (t+2) \times N_s$, where N_s is the number of samples used for feature extraction at each node. This can be physically implemented with a serial filter architecture employing a single MAC unit per tree. Finally, a summation of the leaf values is required to calculate the output decision of the ensemble, followed by thresholding. The summary of the computational cost for CNN and DT ensembles is shown in Table I.

III. Hardware and Performance Comparison

A high classification accuracy, low latency, small footprint, and power efficiency are the key requirements for an on-chip neural data classifier. For seizure detection, the gradient boosted DT ensemble in [2] achieved an AUC of 92%, outperforming the CNN classifiers studied here [4]-[6], with an average detection latency of 1.1s. The classifier in [5] captures the interictal to preictal transition approximately 9-10 minutes prior to seizure onset. The latency was not reported for the remainder of classifiers studied here. However, given that each method is verified on a different dataset, the comparison of classification performance may not be fair.

In the following, we analyze the energy efficiency and required hardware resources to implement the CNN and DT classifiers in an integrated neural interface. In this analysis, the number of parameters and MAC operations (normalized to sampling frequency) are calculated based on the discussions and equations derived in Section II. A summary of hardware and performance metrics for these classifiers is provided in Table II. A detailed quantitative comparison in terms of energy efficiency and footprint would require the details of hardware implementation. However, except [2], the other methods have not been implemented in hardware.

A. Energy Efficiency

The energy consumption of classifiers can be categorized into computation and data movement energy. The computational energy accounts for the required energy to perform multiply-accumulation, comparison, as well and other mathematical operations during inference. Data movement energy is associated with various types of memory access during inference, such as storing partial sums in CONV layers or reading the weight and bias values from memory. In CNNs, data movement is commonly more energy consuming than computation [13].

The massively interconnected structure of neural networks would require a large number of MAC operations during inference. A number of techniques such as pruning [14] and data reuse [13] have been recently proposed to relax the computational requirements of CNNs. Although the energy estimation based on the analysis in Section II would not reflect the potential gain that can be achieved by pruning and data reuse, the computational requirements of CNNs is generally significantly higher than DT ensembles. For example, the authors in [6] estimate the computational energy of a 4-bit integer CNN model for seizure detection based on the energy consumption of multiplications and additions as

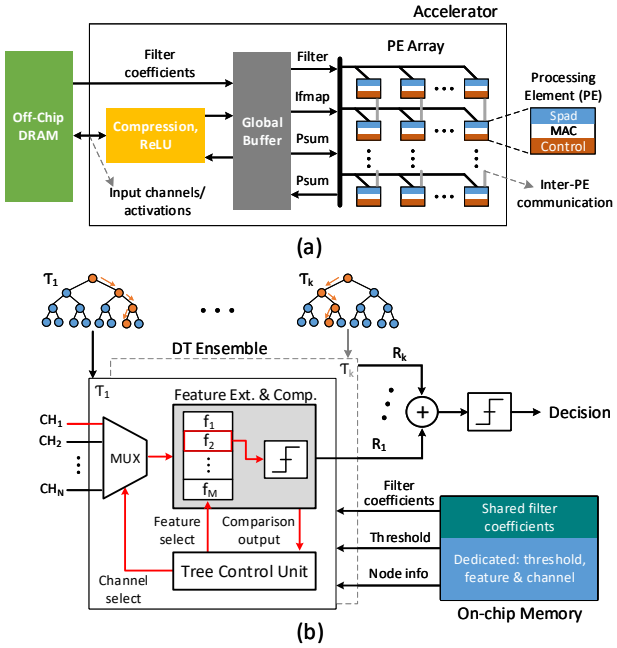

Fig. 2: (a) A spatial architecture for CNN accelerator proposed in [13]. Spad: Scratch pad (local memory space in PEs), Ifmap: Input feature map, Psum: Partial sum. (b) Hardware architecture for the DT ensemble in [2]. Filter coefficients are shared among trees, while each tree has a dedicated memory to store its parameters. R_1: leaf value at the end of Tree 1 execution.

34–90 µJ/class (in a 45nm process, 0.9V). Although their proposed 4-bit Integer-Net enables 10× reduction in energy consumption compared to a conventional 32-bit floating point implementation, it is still significantly more energy consuming compared to the DT ensemble model proposed in [2]. It is worth mentioning that the energy requirements for FFT transform applied to the time series neural data that generates the input to the CNN is not included in the reported computational efficiency.

On the other hand, the large number of MAC operations in CNNs would require significant data movement between computational units and the on-chip or off-chip memories. As a result, there have been extensive efforts to design DNN accelerators [13] and specialized computational platforms, like Google's TPU. To minimize the energy consumption for data movement in CNNs, a spatial architecture with four levels of memory hierarchy was proposed in [13], Fig. 2(a). This proposed scheme for data flow reduces the energy consumption of CONV layers by 1.4–2.5× compared to conventional data flows. For the gradient boosted DT

TABLE II: Comparison of Performance and Hardware Metrics

Parameter	TNSRE'18 [4]	TBME'18 [5]	JETCAS'18 [6]*	JETCAS'18 [2]
Classifier	CNN	CNN	CNN	XGB
Design parameters	5-layer	8-layer	5-layer	8-trees, depth of 4
AUC (# of patients)	90% (18)	86.6% (22)‡	85.1% (12)	92% (26)
Singal Modality	iEEG	scalp EEG	scalp EEG	iEEG
Task	IED detection§	ictal/pre-ictal/interictal	seizure/non-seizure	seizure/non-seizure
# of Parameters†	1980 / 420k	1590 / 125k	8960 / 41.2k	512
# of MACs†**	28.4k / 2.1k	181k / 489	N.A.	704††
Memory	N.A.	N.A.	26.2 kB	<1 kB
Chip Area	N.A.	N.A.	N.A.	1 mm^2
Energy Efficiency	N.A.	N.A.	34–90 µJ	41.2 nJ

* 4-bit integer tested on iEEG from 4 dogs and 8 patients
** normalized to sampling frequency. §Interictal epileptiform discharges
‡ performance reported on CHB-MIT dataset
† reported numbers for CNN models follow the (CONV/FC) format
†† number of MACs required for feature extraction

ensemble in Fig. 2(b) [2], however, the data movement is significantly lower. In this DT architecture, the only parameters to be retrieved from memory include the input channel number to be processed, threshold value for comparison, feature number, and filter coefficients, with no need for intermediate memory access from on-chip memory during feature extraction or classification. As a result, a state-of-the-art energy efficiency of 41.2 nJ/class was achieved in SoC measurements [2].

B. Hardware Utilization and Storage Requirements

As discussed earlier, deep neural networks require specialized hardware platforms for energy efficient computation of MAC operations. Such a platform would process CNNs in a layer by layer fashion. Based on the size and shape of the layer (i.e., CONV or FC), a portion of the on-chip resources would be reused to compute the MAC operations in a sequence. In [13], the computation mapping of the processing element (PE) array for each layer is found by maximizing the data reuse in the form of convolutional, filter, and input feature map reuse. The total core area of this accelerator with 168 PEs and 108kB of global buffer is 12.25 mm^2. Moreover, a large off-chip DRAM and a bulky on-chip buffer is required for energy efficient computations and storage of classifier parameters in DNNs, as opposed to minimal storage requirements of DT ensembles. While the network quantization technique proposed in [6] saves 7.2–7.6× in storage requirements compared to the 32-bit floating point, it still needs 26.2× more storage compared to the DT ensemble in [2]. This 8-tree ensemble requires less than 1kB of register-type memory (690b dedicated per tree and 228B shared), with no need for off-chip memory. Figure 3 further confirms the large number of parameters required for DNN classifiers compared to DT ensembles, based on the equations derived in Section II. The accurate estimation of hardware resources and chip area would depend on the customized hardware architecture for each model, which is not available for the cited references that use DNN models.

Furthermore, the hardware architecture in Fig. 2(b) processes the top-down flow of a tree in a sequential way, through reusing a universal feature extraction engine. This architecture allows significant reduction in the overall chip area and power consumption. Therefore, the energy and area trade-off is relaxed. As discussed in [2], the hardware complexity of this architecture would not scale with number of input channels, and can therefore serve as a compact and low-power solution for multichannel neural data classification. During feature extraction at each node of a tree, the small number of MAC operations in FIR filters would allow the implementation of a filter with a fully serial architecture, without penalizing the detection latency. As a result, the footprint can be further reduced by implementing one MAC per feature extraction unit.

To further improve the area and energy efficiency of the DT ensemble, a reduced bit-precision for parameters and efficient MAC implementation in filters, such as distributed arithmetic or memristive-based multiplication can be employed, that remains as future work.

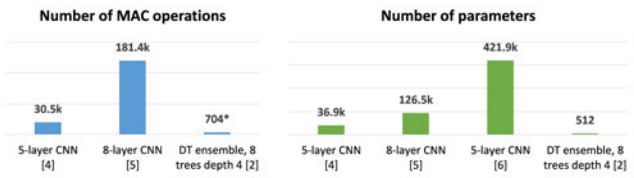

Fig. 3: Comparison of number of parameters and MAC operations required for CNN and DT classifiers, *assuming that all visited nodes in a tree extract features from 1s windows of input.

IV. Conclusions

In this work, we compared the hardware complexity of DNN and DT ensembles for neural data classification. Our study shows that DNNs are computationally demanding, and require large number of parameters and MAC operations for inference. Therefore, such classifiers do not meet the stringent power and area requirements for applications such as implantables or wearables. Prior work [2], [11] have proposed hardware friendly architectures for DTs that substantially relaxes the energy-area trade-off, while outperforming the deep learning methods in accuracy. Our analysis further confirms that ensembles of DTs can serve as an attractive solution for embedded neural data classification.

References

[1] D. Pei, M. Burns, R. Chandramouli, R. Vinjamuri, "Decoding asynchronous reaching in electroencephalography using stacked autoencoders," *IEEE Access*, vol. 6, pp. 52889-52898, 2018.

[2] M. Shoaran, B. A. Haghi, M. Taghavi, M. Farivar, A. Emami, "Energy-Efficient Classification for Resource-Constrained Biomedical Applications," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, vol. 8, no. 4, pp. 693-707, 2018.

[3] P. Thodoroff, J. Pineau, A. Lim, "Learning robust features using deep learning for automatic seizure detection," *Machine learning for healthcare conference*, pp. 178-190, 2016.

[4] A. Antoniades, L. Spyrou, D. Martin-Lopez, A. Valentin, G. Alarcon, S. Sanaei, C.C. Took, "Detection of interictal discharges with convolutional neural networks using discrete ordered multichannel intracranial EEG," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 12, pp. 2285-2294, 2017.

[5] H. Khan, L. Marcuse, M. Fields, K. Swann, B. Yener, "Focal onset seizure prediction using convolutional networks," *IEEE Transactions on Biomedical Engineering*, vol. 65, no. 9, pp. 2109-2118, 2018.

[6] N. D. Truong, A. D. Nguyen, L. Kuhlmann, M. R. Bonyadi, J. Yang, S. Ippolito, O. Kavehei, "Integer Convolutional Neural Network for Seizure Detection," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, 2018.

[7] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, pp.1189-1232, 2001.

[8] T. Chen and T. He, "xgboost: eXtreme Gradient Boosting," *R package version 0.4-2*, 2015.

[9] Available online at www.kaggle.com/c/seizure-detection

[10] M. Shoaran, M. Farivar, A. Emami, "Hardware-Friendly Seizure Detection with a Boosted Ensemble of Shallow Decision Trees," *Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2016.

[11] M. Taghavi, B. A. Haghi, M. Farivar, M. Shoaran, A. Emami, "A 41.2 nJ/class, 32-channel on-chip classifier for epileptic seizure detection," *Int. Conf. IEEE Eng. Medicine and Biology Society (EMBC)*, 2018.

[12] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[13] Y. H. Chen, T. Krishna, J. S. Emer, V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127-138, 2017.

[14] B. Hassibi, D. G. Stork, G. J. Wolff, "Optimal Brain Surgeon and general network pruning," *IEEE International Conference on Neural Networks*, 1993.