An Energy Efficient Additive Neural Network

Arman Afrasiyabi[1,*], Baris Nasir[1,*], Ozan Yildiz[1,*], Fatos T. Yarman Vural[1], A. Enis Cetin[2]
[1] Department of Computer Engineering, Middle East Technical University, Ankara, Turkey
Email: {afrasiyabi.arman, bnasir, oyildiz, vural}@ceng.metu.edu.tr
[2] Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey
Email: cetin@bilkent.edu.tr

***Abstract*—In this paper, we propose a new energy efficient neural network with the universal approximation property over space of Lebesgue integrable functions. This network, called additive neural network, is very suitable for mobile computing. The neural structure is based on a novel vector product definition, called ef-operator, that permits a multiplier-free implementation. In ef-operation, the "product" of two real numbers is defined as the sum of their absolute values, with the sign determined by the sign of the product of the numbers. This "product" is used to construct a vector product in n-dimensional Euclidean space. The vector product induces the lasso norm. The proposed additive neural network successfully solves the XOR problem. The experiments on MNIST dataset show that the classification performances of the proposed additive neural networks are very similar to the corresponding multi-layer perceptron.**

***Keywords*—energy efficient, efficient ANN, neural network, machine learning, multiplierless ann, mnist, xor, machine learning.**

I. Introduction

Recent developments in VLSI industry create powerful mobile devices which can be used in many practical recognition applications. However, the current structure of the ANNs, especially, deep networks, prohibits us to implement these algorithms effectively on mobile devices due to high energy requirements. A typical neuron needs to perform three main tasks to produce an output: (i) an inner product operation involving multiplication of inputs by weights, (ii) addition, and (iii) pass the result of the inner product through an activation function. According to the [1], the multiplication is the most energy consuming operation.

We introduced the $l1$ norm based vector product for some image processing applications in 2009 [2]–[4]. We also proposed the multiplication free neural network structure in 2015 [5]. However, the recognition rate was below 10% of a regular neural network. Han et al. proposed a model that reduces both computational cost and storage by feature learning [1]. Sarwar et al. used the error resiliency property of neural networks and proposed an approximation to multiplication operation on artificial neurons for energy-efficient neural computing [6]. Abdelsalam et al. approximate the tangent activation function using the Discrete Cosine Transform Interpolation Filter (DCTIF) to run the neural networks on FPGA boards efficiently [7]. Another similar approach is proposed by Rastegari et al. [8]. Binary-Weight-Networks approximates all the weight values to binary values. Since the weight values are binary, convolutions can be estimated by only addition and subtraction, which eliminates the main power draining multiplication operation.

In this paper, we propose an $l1$ norm based energy efficient neural network, called *additive neural network*, that replaces the multiplication operation with a new energy efficient operator, called *ef-operator*. Instead of multiplications, we use sign multiplications and addition operations in a typical neuron. The sign multiplication of two real numbers is a simple bit operation. An addition, it consumes relatively lower energy compared to a regular multiplication as shown in [1] in most processors. Our object recognition experiments on MNIST and CIFAR datasets show that we are able to match the performance of the state of the art neural networks by at most 0.039% error difference, without performing any other changes on the ANN structure.

II. A New Energy Efficient Operator

Let $\mathbf{x}$ and $\mathbf{y}$ be two vectors in $\mathbb{R}^d$. We define an new operator, called ef-operator, as the vector product of $\mathbf{x}$ and $\mathbf{y}$ as follows;

$$\mathbf{x} \diamond \mathbf{y} := \sum_{i=1}^{d} \text{sign}(x_i \times y_i)(|x_i| + |y_i|), \tag{1}$$

which can also be represented as follows;

$$\mathbf{x} \diamond \mathbf{y} := \sum_{i=1}^{d} \text{sign}(x_i) y_i + \text{sign}(y_i) x_i, \tag{2}$$

where $\mathbf{x} = [x_1, \ldots, x_d]^T, \mathbf{y} = [y_1, \ldots, y_d]^T \in \mathbb{R}^d$. The new vector product operation does not require any multiplications. The operation $(x_i \times y_i)(|x_i| + |y_i|)$ uses the sign of the ordinary multiplication but it computes the sum of absolute values of x_i and y_i. ef-operator, $\diamond$, can be implemented without any multiplications. It requires summation, unary minus operation and if statements which are all energy efficient operations.

Ordinary inner product of two vectors induces the ℓ_2 norm. Similarly, the new vector product induces a scaled version of the ℓ_1 norm:

$$\mathbf{x} \diamond \mathbf{x} = \sum_{i=1}^{d} \text{sign}(x_i \times x_i)(|x_i| + |x_i|) = 2||x||_1 \tag{3}$$

Therefore, the ef-operator performs a new vector product, called ℓ_1 product of two vectors, defined in Eq. 1.

Let $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{W} \in \mathbb{R}^{d \times M}$ be two matrices, then the ef-operation between $\mathbf{W}$ and x is defined as follows;

* Equally contributing authors.

978-1-5090-6494-6/17/$31.00 2017 IEEE

$$\mathbf{x} \diamond \mathbf{W} := [\mathbf{x} \diamond \mathbf{w}_1 \quad \dots \quad \mathbf{x} \diamond \mathbf{w}_M]^T \in \mathbb{R}^M, \tag{4}$$

where $\mathbf{w}_j$ is jth column of $\mathbf{W}$ for $j = 1, 2, \dots, M$.

III. Additive Neural Network with ef-operator

We propose a modification to the representation of a neuron in a classical neural network, by replacing the vector product of the input and weight with the $l1$ product defined in ef-operation. This modification can be applied to a wide range of artificial neural networks, including multi-layer perceptrons (MLP), recurrent neural networks (RNN) and convolutional neural networks (CNN).

A neuron in a classical neural network is represented by the following activation function;

$$f(\mathbf{x}\mathbf{W} + \mathbf{b}), \tag{5}$$

where $\mathbf{W} \in \mathbb{R}^{d\times M}$, $b_{\in}\mathbb{R}^M$ are weights and biases, respectively, and $\mathbf{x} \in \mathbb{R}^d$ is the input vector.

A neuron in the proposed additive neural network is represented by the activation function, where we modify the affine transform by using the ef-operator, as follows;

$$f(\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b}), \tag{6}$$

where $\odot$ is element-wise multiplication operator, $\mathbf{W} \in \mathbb{R}^{d\times M}$, $a, b \in \mathbb{R}^M$ are weights, scaling coefficients and biases, respectively, and $\mathbf{x} \in \mathbb{R}^d$ is the input vector. The neural network, where each neuron is represented by the activation function defined in Eq. 6, is called additive neural network.

Comparison of Eq. 5 and Eq. 6 shows that the proposed additive neural networks are obtained by simply replacing the affine scoring function (**xW+b**) of a classical neural network by the scoring function function defined over the ef-operator, $(\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b})$. Therefore, most of the neural networks can easily be converted into the additive network by just representing the neurons with the activation functions defined over ef-operator, without modification of the topology and the general structure of the optimization algorithms of the network.

A. Training the Additive Neural Network

Standard back-propagation algorithm is applicable to the proposed additive neural network with small approximations. Back-propagation algorithm computes derivatives with respect to current values of parameters of a differentiable function to update its parameters. Derivatives are computed iteratively using previously computed derivatives from upper layers due to chain rule. Activation function, f, can be excluded during these computations for simplicity as its derivation depends on the specific activation function and choice of activation function does not affect the remaining computations. Hence, the only difference in the additive neural network training is the computation of the derivatives of the argument, $(\mathbf{a}\odot(\mathbf{x}\diamond\mathbf{W})+\mathbf{b})$, of the activation function with respect to the parameters, $\mathbf{W}, \mathbf{a}, \mathbf{b}$, and input, $\mathbf{x}$, as given below:

$$\frac{\partial(\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b})}{\partial \mathbf{a}} = Diag(\mathbf{x} \diamond \mathbf{W}), \tag{7}$$

$$\frac{(\partial\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b})}{\partial \mathbf{b}} = I_M, \tag{8}$$

$$\frac{\partial(\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b})}{\partial \mathbf{x}_i} \approx \mathbf{a} \odot \mathbf{sign}(\mathbf{w}_i), \tag{9}$$

$$\frac{\partial(\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b})}{\partial \mathbf{W}_{i,j}} \approx a_j \mathbf{x}_i \mathbf{e}_j, \tag{10}$$

where $\mathbf{a}, \mathbf{b} \in \mathbb{R}^M$, and $\mathbf{W} \in \mathbb{R}^{d\times M}$ are the parameters of the hidden layer, $\mathbf{x} \in \mathbb{R}^d$ is the input of the hidden layer, $\mathbf{e}_i \in \mathbb{R}^M$ is the ith element of standard basis of $\mathbb{R}^M$, $\mathbf{w}_i$ is the ith column of $\mathbf{W}$, $\mathbf{sign}(\mathbf{w}_i) = \sum_{j=1}^{M} sign(\mathbf{W}_{i,j})\mathbf{e}_j$ for $i = 1, \dots, M$, δ is the dirac delta function.

The above derivatives can be easily calculated using the following equation suggested by [9]:

$$\frac{d}{dx} sign(x) = 2\delta(x). \tag{11}$$

Approximations to derive the above equation are based on the fact that $\delta(x) = 0$, almost surely.

B. Existence and Convergence of the Solution in Additive Neural Network

In this section, first, we show that the proposed additive neural network satisfies the universal approximation property of [10], over the space of Lebesgue integrable functions. In other words. there exists solutions computed by the proposed additive network, which is equivalent to the solutions obtained by activation function with classical vector product. Then, we make a brief analysis for the convergence properties of the back propagation algorithm when the vector product is replaced by the ef-operators.

1) Universal Approximation Property: The universal approximation property of the suggested additive neural network is to be proved for each specific form of the activation function. In the following proposition, we suffice to provide the proofs of universal approximation theorem for ReLU activation functions, only. The proof (if it exits) for a general activation function requires a substantial amount of effort, thus it is left to a future work.

Proposition III.1. *The additive neural network, defined by the neural activation function with the activation function with Rectified Linear Unit,*

$$f(\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b}) = ReLU(\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b}), \tag{12}$$

is dense in $L^1(I_n)$.

In order to prove the above proposition, the following Lemma is proved first:

Lemma III.2. *If the function $g(x)$ can be computable with activation function*

$$f(\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b}) = \mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b}, \quad (13)$$

then there exist an additive neural network architectures with a Rectified Linear Unit activation function,

$$f(\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b}) = ReLU(\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b}), \quad (14)$$

which can also compute $g(x)$.

Proof: This lemma can be proven using the following simple observations:

- **Observation 1:** If $g(\mathbf{x}) = \mathbf{a} \odot (\mathbf{x} \diamond \mathbf{w}) + \mathbf{b}$, then, $-g(\mathbf{x}) = \mathbf{a}' \odot (\mathbf{x} \diamond \mathbf{w}') + \mathbf{b}'$, where $\mathbf{a}' = \mathbf{a}$, $\mathbf{w}' = -\mathbf{w}$, and $\mathbf{b}' = -\mathbf{b}$.
- **Observation 2:** If $g(\mathbf{x}) = \mathbf{a} \odot (\mathbf{x} \diamond \mathbf{w}) + \mathbf{b}$, then, $g(\mathbf{x}) = \mathbf{a}'' \odot ((-\mathbf{x}) \diamond \mathbf{w}'') + \mathbf{b}''$, where $\mathbf{a}' = \mathbf{a}$, $\mathbf{w}' = -\mathbf{w}$, and $\mathbf{b}' = \mathbf{b}$.
- **Observation 3:** If $g(\mathbf{x}) = \mathbf{a} \odot (\mathbf{x} \diamond \mathbf{w}) + \mathbf{b}$,then, $g(\mathbf{x}) = \mathbf{a}''' \odot (ReLU(\mathbf{x}) \diamond \mathbf{w} + ReLU(-\mathbf{x}) \diamond \mathbf{w}''') + \mathbf{b}'''$, where $\mathbf{a}''' = \mathbf{a}$, $\mathbf{w}''' = -\mathbf{w}$, and $\mathbf{b}''' = \mathbf{b}$.

Lets assume that there exists an additive neural network, using identity as activation function which can compute the function $g(x)$. We can extend each layer using Observation 1, to compute both $g(x)$ and $-g(x)$. Afterwards, we can replace zeros on the weights introduced during previous extension on each layer using Observation 3, to replace the activation function with ReLU. This works, because either $ReLU(x)$ or $ReLU(-x)$ is 0. The modified network is an additive neural network with ReLU activation function, which can compute the function $g(x)$. ■

Proof of Proposition III.1: This can be shown by the universal approximation theorem for bounded measurable sigmoidal functions [10]. This theorem states that finite sums of the form

$$G(\mathbf{x}; \{\alpha_i\}_{i=1}^N, \{\mathbf{y}_i\}_{i=1}^N, \{\theta_i\}_{i=1}^N) = \sum_{i=1}^{N} \alpha_i \sigma(\mathbf{y}_i^T \mathbf{x} + \theta_i), \quad (15)$$

are dense in $L^1(I_n)$, where $\alpha_i, \theta_i \in \mathbb{R}$ and $\mathbf{x}, \mathbf{y}_i \in \mathbb{R}^d$ for $i = 1, 2, \ldots, N$. It can be easily shown that $sign$ function is a bounded sigmoidal function. When the activation function is taken as identity, then one can easily construct an additive network which can compute $sign(\mathbf{y}_i^T \mathbf{x} + \theta_i)$ for $i = 1, 2 \ldots, N$. **Lemma** III.2 shows that there are equivalent networks using ReLU as the activation function which compute the same functions. These networks can be combined with concatenation of layers of the additive neural networks to a single network. Also, proposed architecture contains fully connected linear layer at the output, and this layer can compute superposition of the computed $sign$ functions yielding $G(x)$. Since $G(\mathbf{x})$ can be computable by the additive neural networks, and $G(\mathbf{x})$ functions are dense in $L^1(I_n)$, then functions computed by the additive neural networks are also dense in $L^1(I_n)$. ■

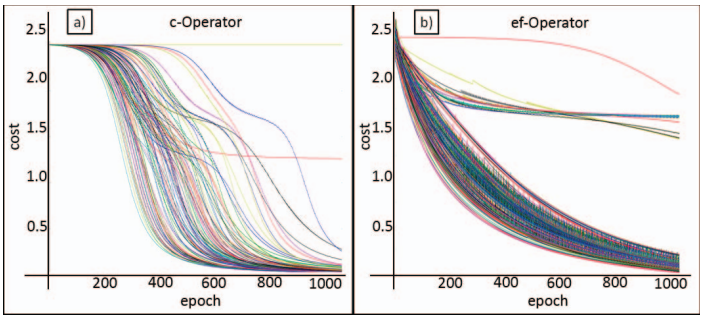

Figure 1: Loss changes in stochastic gradient descent (SGD) algorithm in the training phase of XOR problem with single hidden layer MLP with classical score function **(c-operator)** (left) and with **(ef-operator)** (right). The results are obtained by training the network 200 times in 1000 epochs which are shown by different colors.

2) Computational efficiency: The proposed additive neural network contains more parameters then the classical nuoron representation in MLP architectures. However, each hidden layer can be computed using considerably less number of multiplication operator. A classical neural network, represented by the activation function $f(\mathbf{x}\mathbf{W} + \mathbf{b})$, containing M neurons with d dimensional input, requires $d \times M$ many multiplication operator to compute $\mathbf{x}\mathbf{W} + \mathbf{b}$. On the other hand, the additive neural network, represented by the activation function, $f(\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b})$ with the same number of neurons and input space requires M many multiplication operator to compute $\mathbf{a} \odot (\mathbf{x} \diamond \mathbf{W}) + \mathbf{b}$. This reduction on number of multiplications is especially important when input size is large or hidden layer contains large number of neurons.

IV. Experimental Results

The additive neural network with ef-operator could successfully solve the XOR problem and reached to 100% accuracy. We also investigate the rate of in-loss changes at each epoch. Also, as shown in the Figure 1, some of the runs do not reach to minimum values in 1000 epochs. In other words, more epochs are needed in some runs. Generally, the number of epochs depends on learning rate and initialization condition, and the final epoch can be determined by some stopping criteria. However, in this study, we are only interested to see the variations in the cost; therefore, we fixed the number of epochs to 1000.

Figure 1 show the change of loss in the MLP using classical affine operator (c-operator) and ef-operator, with ReLU as the activation function. We rerun the network for 200 times in 1000 epochs and used k-fold cross validation to specify the learning-rate parameter of SGD. Each color in the plots shows the variations inloss or cost value (x-axis) across the epochs (y-axis) in one specific run of the network. As the figure shows, the cost value of the network with our proposed ef-operator decreases along the epochs and acts similar to c-operator.

In the second experiment, we classified the digits of MNIST dataset of [11] which consists of handwritten examples to examine our proposed additive neural network in multi-class classification problem. We used cross-entropy based cost function and SGD to train the network. We used 150 number

TABLE I: Optimal classification results of classic function c-operator and our proposed ef-operator score functions in different MLP architectures and learning rates.

MLP Architecture	-	ReLU		Tanh		Sigmoid	
	learning rate	c-operator	ef-operator	c-operator	ef-operator	c-operator	ef-operator
2 Hidden Layers	0.01	**98.43**	98.01	96.39	95.57	97.81	96.80
	0.005	98.36	**98.09**	97.23	96.05	98.07	97.10
	0.001	98.03	97.76	97.63	96.77	95.83	96.47
	0.0005	97.61	97.21	96.27	96.10	95.83	95.53
3 Hidden Layers	0.01	96.85	97.80	90.42	92.64	96.31	96.23
	0.005	98.15	**97.95**	95.08	93.33	96.48	96.50
	0.001	**98.22**	97.63	97.49	93.63	95.74	95.85
	0.0005	97.65	96.97	96.78	93.93	94.34	94.83

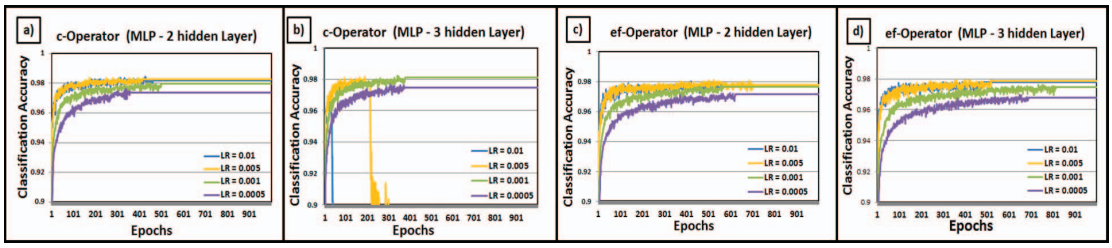

Figure 2: Classification accuracies in different architectures with different score functions. Subplots (a) and (b) shows the results of MLP with 2 and 3 hidden layers using classic c-operator. Subplots (c) and (d) shows the results of MLP with 2 and 3 hidden layers using our proposed ef-operator.

of examples in each iteration of SGD. In other words, the batch size is equal to 150.

Table I contains the classification accuracies of the MLP architecture using three activation functions: ReLU, Tanh and Sigmoid with four different learning rates. As the table shows, our additive neural network over ef-operator reaches to the performance of classic MLP with c-operator. In other words, with a slightly sacrificing the classification performance we can use the proposed ef-operator which is more energy-efficient. Note that, we have not used any regularization methods to improve the accuracy further. Also Table. I shows that ReLU gives the maximum performances across the activation functions.

Figure 2 shows the results of the classification accuracies obtained from the proposed ef-operator and the conventional c-operator. In each epoch, the network is trained with all of the training examples. The plots of the sub-figures are obtained using four different learning rates: 0.1, 0.005, 0.001 and 0.0005. As illustrated in the figure, the proposed additive network catches the performance of the standard backpropagation method.

V. Conclusion

In this study, we propose an energy efficient additive neural network architecture. The core of this architecture is the lasso norm based ef-operator that eliminates the energy-consumption multiplications in the conventional architecture. We have examined the universal approximation property of the proposed architecture over the space of Lebesgue integrable functions and test it in real world problems. We showed that ef-operator can successfully solve the nonlinear XOR problem. Moreover, we have observed that with sacrificing 0.39% accuracy, our proposed network can be used in the multilayer perceptron (MLP) to classify MNIST dataset.

Acknowledgment

Ozan Yildiz is supported by TUBITAK 2210-E fellowship.

References

[1] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[2] H. Tuna, I. Onaran, and A. E. Cetin, "Image description using a multiplier-less operator," *IEEE Signal Processing Letters*, vol. 16, no. 9, pp. 751–753, 2009.

[3] C. E. Akbaş, O. Günay, K. Taşdemir, and A. E. Çetin, "Energy efficient cosine similarity measures according to a convex cost function," *Signal, Image and Video Processing*, pp. 1–8.

[4] H. S. Demir and A. E. Cetin, "Co-difference based object tracking algorithm for infrared videos," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 434–438.

[5] C. E. Akbaş, A. Bozkurt, A. E. Çetin, R. Çetin-Atalay, and A. Üner, "Multiplication-free neural networks," in *2015 23th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2015, pp. 2416–2418.

[6] S. S. Sarwar, S. Venkataramani, A. Raghunathan, and K. Roy, "Multiplier-less artificial neurons exploiting error resiliency for energy-efficient neural computing," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 145–150.

[7] A. M. Abdelsalam, J. Langlois, and F. Cheriet, "Accurate and efficient hyperbolic tangent activation function on fpga using the dct interpolation filter," *arXiv preprint arXiv:1609.07750*, 2016.

[8] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," *arXiv preprint arXiv:1603.05279*, 2016.

[9] R. N. Bracewell, *The Fourier transform and its applications*, 3rd ed., ser. McGraw-Hill series in electrical and computer engineering; Circuits and systems. McGraw Hill, 2000, p. 97.

[10] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[11] Y. B. Y. LeCun, L. Bottou and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, pp. 2278–2324, November 1998.