Common-Source Amplifier Based Analog Artificial Neural Network Classifier

Akshay Jayaraj *, Imon Banerjee† and Arindam Sanyal*
*Electrical Engineering, State University of New York at Buffalo, Buffalo, NY 14260, USA
Email: akshayja@buffalo.edu, arindams@buffalo.edu
†Biomedical Data Science Department, Stanford University, Stanford, CA 94305, USA. Email: imonb@stanford.edu

***Abstract*—An analog artificial neural network (ANN) classifier using a common-source amplifier based nonlinear activation function is presented in this work. A shallow ANN is designed using transistor level circuits and a multinomial (10 classes) classification accuracy of 0.82 is achieved on the MNIST dataset which consists of handwritten images of digits from 0-9. Use of common-source amplifier structure simplifies the ANN and results in 5X lower energy consumption than existing analog classifiers. The classifier performance is validated using Spectre and Matlab simulations.**

I. Introduction

With emergence of internet-of-things (IoT), there is a growing need to impart intelligence to sensor devices for enabling real-time inference and decision making ability. A machine learning classifier embedded inside an IoT sensor can also reduce bandwidth requirements by only transmitting classifier outcomes rather than raw sensor data. However, digital implementation of machine learning classifier on-chip far exceeds the power budget of typical IoT sensors [1]. This has resulted in interest in analog machine learning classifiers [1]–[5]. Classifiers embedded in IoT sensors are usually trained offline and the weights are provided to an on-chip artificial neural network (ANN) for classification at the sensor node. Analog machine-learning classifiers reported so far in the literature has implemented either only multiply-and-add operations or binary classification using linear regression. [2] presented a switched-capacitor based matrix-multiplier which achieved 0.85 classification accuracy on images from Cifer-10 database. [3] embedded a matrix multiplier inside an analog-to-digital converter (ADC) and achieved 0.83 classification accuracy on a gender detection system while consuming 655nJ/classification. [4] embedded a multiply-and-add circuit inside a $\Delta\Sigma$ ADC and achieved 0.87 classification accuracy on MNIST dataset, while [1], [5] presented binary classifiers with modified Adaboost [6] and achieved 0.9 classification accuracy on MNIST dataset. [5] has reported the lowest energy consumption of 534pJ/classification.

In this work, we report a shallow ANN consisting of an input layer, a hidden layer and an output layer that performs multinomial (10 classes) classification on MNIST dataset using nonlinear activation function. MNIST dataset is used because it is commonly available and image is a popular modality for IoT sensors. We use common-source amplifiers for implementing nonlinear activation functions which results in simulated energy consumption of 100pJ/classification. While MNIST dataset has images with 28x28 pixels, it is not feasible to provide 784 external inputs to a chip due to limitations imposed by pad size. We resize the input images to 5x5 pixels to ensure that inputs can be provided to an on-chip classifier. [1] resizes MNIST input to 47 pixels and [5] resizes MNIST input to 82 pixels. While accuracy of our classifier is limited by input resizing which is a lossy process, the classification accuracy can be improved if an image sensor array is integrated with the classifier such that inputs to the ANN do not have to come from outside. In addition, the purpose of this work is to show the feasibility of a complete on-chip analog classifier and we have chosen MNIST dataset as a vehicle for this demonstration to perform fair comparison with existing works. Selecting another dataset with inputs having lower number of pixels will significantly improve classification accuracy of the proposed system. Even with the inputs resized to 25 pixels, if the sensor data is to be transmitted directly, the transmitter will need to send out 250-bits every cycle assuming the analog sensor data is quantized to 10-bit. In contrast, with the proposed on-chip classifier, the transmitter needs to send out only 4-bit digital data every cycle which reduces transmission bandwidth by approximately a factor of 60.

The rest of this paper is organized as follows: the classifier training and circuit implementation is discussed in Section II, simulation results are presented in Section III and the conclusion is brought up in Section IV.

II. Proposed Architecture

The core arithmetic operation that an on-chip classifier needs to perform is given by $f\left(\sum W_i x_j\right)$ where $f(\cdot)$ can be linear or non-linear function (also referred to as activation function), W_i is the weight vector and x_i is the input vector. The arithmetic operation can be performed in either digital or analog domain. However, analog computation using physical properties of MOS devices is more energy efficient than digital computation which only uses MOS devices as switches. We use a simple common-source (CS) amplifier as the building block for the proposed classifier. Fig. 1 shows a CS amplifier with PMOS diode load and the plot of drain-source voltage of the PMOS load versus input voltage. The transfer function has a similar shape as activation functions commonly used in machine learning literature, such as tanh and softmax. In contrast to prior CMOS implementation of activation functions [7], the

978-1-7281-0397-6/19/$31.00 ©2019 IEEE

proposed implementation uses only 2 transistors and has a much lower hardware cost which is an enabling factor for building a complete ANN classifier. Later in this section we will show how the CS amplifier with diode load is used to implement a complete ANN classifier. For our ANN classifier, weight vectors are implemented by varying the widths of banks of input transistors, while addition is performed in current domain by summing the currents through each input transistor using diode-connected load.

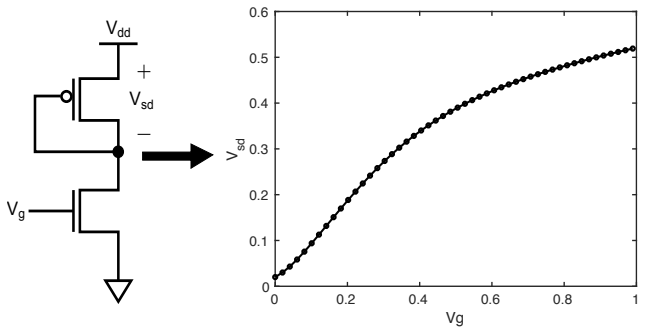
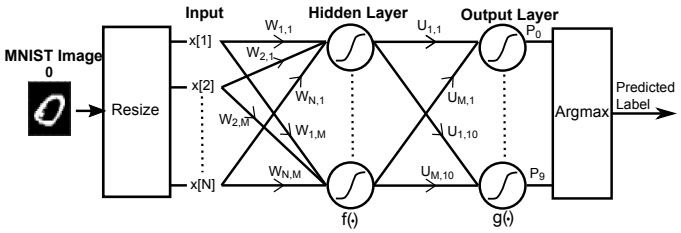
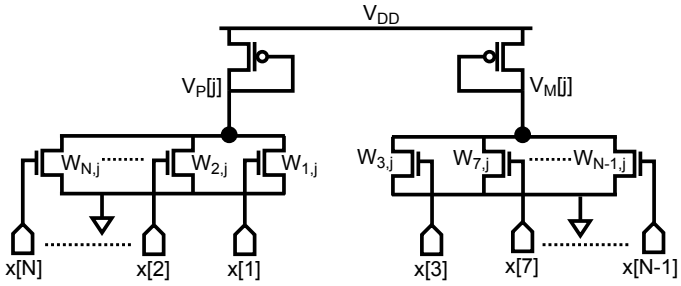


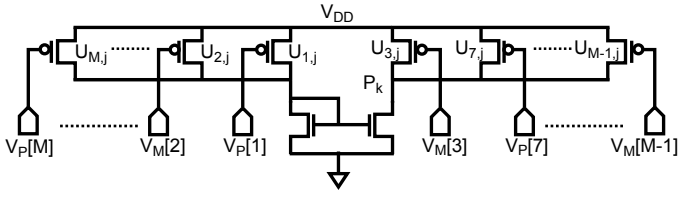
Fig. 1. CS amplifier and its transfer function



(a)



(b)



(c)

Fig. 2. Schematic of (a) ANN for MNIST classification (b) j-th slice of hidden layer activation function (c) k-th slice of output layer softmax function

Fig. 2 shows the transistor level schematic of the proposed classifier with 1 hidden layer. As shown in Fig. 2(a), the input image is resized from 28x28 pixels to 5x5 pixels and flatted into a column vector. Each input is attenuated to limit the input swing from 0 to 0.4V such that the input transistors in the hidden layer never go into linear region. The inputs are sent to a hidden layer which performs a nonlinear activation function (denoted by $f(\cdot)$ in Fig. 2(a)) on the weighted sum of all inputs. The proposed classifier has 28 hidden neurons and the weights for the j-th hidden neuron is denoted by $W_{i,j}$ where the subscript i corresponds to the i-th input. The outputs from the hidden layer are sent to the output layer. Each output neuron performs softmax function (denoted by $g(\cdot)$ in Fig. 2(a)) on the weighted sum of its inputs. The output layer has 10 neurons and the predicted output digit is the label associated with the output neuron with the highest value, i.e, if for a certain image input, the 8-th output neuron has the highest value, the classifier will predict the input image to be digit '8'. Fig. 2(b) shows schematic of the j-th hidden neuron built using the CS amplifier shown in Fig. 1. The inputs to the hidden neuron are $x[i]$ with the corresponding weight denoted by $W_{i,j}$. Since $W_{i,j}$ can be either positive or negative, each hidden neuron is split into 2 halves with the inputs with positive weight in the left half and inputs with negative weight in the right half. The output of the hidden neuron is given by the difference between drain-to-source voltages of the PMOS loads in the two halves. Fig. 2(c) shows the schematic of the k-th output neuron. The output neuron is a pseudo-differential CS amplifier which implements softmax of weighted sum of hidden layer outputs. Depending on the polarity of the weights $U_{j,k}$, $V_P[j]$ and $V_M[j]$ are routed to the positive and negative input of the pseudo-differential amplifier or vice versa. Since the activation functions are based on current mode operations, the proposed classifier is immune to charge injection errors unlike previous switched-capacitor implementations of analog classifiers.

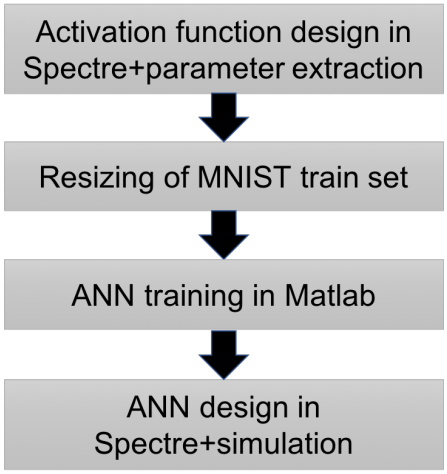


Fig. 3. Flowchart for proposed analog classifier design

Fig. 3 shows the flowchart used for design of the proposed ANN classifier. 1 slice of hidden neuron and output neuron are designed in transistor level and their transfer curves for different widths of input transistors are extracted using Spectre simulations. The extracted transfer curves are imported in Matlab for training the ANN. The MNIST dataset is split randomly into 60,000 training data and 10,000 testing data.

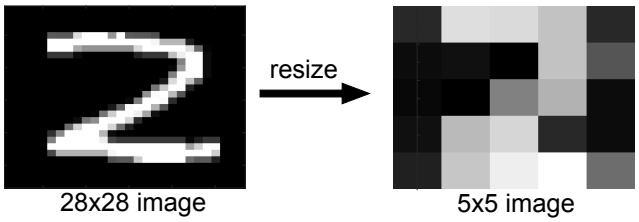


Fig. 4. Example of resizing input image from 28x28 to 5x5

Algorithm 1 ANN Training pseudo-code

W1 ← Weight vector of input layer to hidden layer
W2 ← Weight vector of hidden layer to output layer
$NumberCorrect = 0$
for i<Number of Training Iterations **do**
  for j< Size(Train set) **do**
    $Input \leftarrow Trainset(j)$
    $HiddenOuput \leftarrow \mathrm{f}(Bias; W1, Input)$
    $Ouput \leftarrow \mathrm{g}(Bias; W2, HiddenOutput)$
    $Prediction \leftarrow \mathrm{argmax}(Output)$
    if $Prediction = Train\ label(j)$ **then**
      $NumberCorrect+ = 1$
    end if
    $delta_1 \leftarrow (Output - trainlabel(j)) * (1 - Output^2)$
    $delta_2 \leftarrow (W2 * delta_1) * (1 - HiddenOutput^2)$
    $W1 \leftarrow W1 - alpha * (Input * delta_2')$
    $W2 \leftarrow W2 - alpha * (hiddenoutput * delta_1')$.
  end for
  $accuracy \leftarrow NumberCorrect/N$
end for

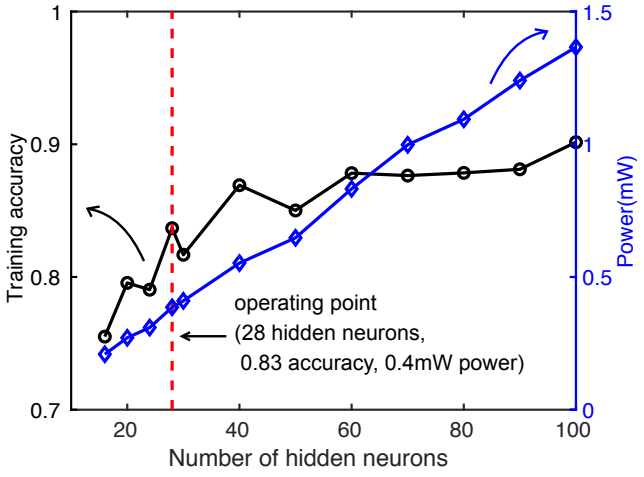


Fig. 5. Classifier accuracy and power versus number of hidden layers

After the training is completed, the weights for hidden layer and output layer are imported into Spectre netlists for transistor level simulation. Fig. 4 shows an example of the resizing operation used to convert 28x28 input image into a 5x5 image. Even though downsizing the input image introduces loss, the input shape is retained which allows the classifier to perform predictions with good accuracy. Algorithm 1 shows the pseudo-code snippet for training the ANN in matlab based on parameters extracted from Spectre simulations. The ANN weights are updated as the network iterates through each input from the training set. The weights are updated using the well known stochastic gradient descent [8] which computes derivative of error (mean squared error for our case) with respect to current weights and calculates new weights based on the learning rate, derivative of error and current weights. The hidden layer weights are quantized to 3-bits (4-bits with sign) and the output layer weights are quantized to 7-bits (8-bits with sign) during ANN training to keep the ANN area small. In order to determine the optimum number of hidden neurons, we performed a sweep of number of hidden neurons versus classifier accuracy and power consumed by the hidden layer. The simulation results are shown in Fig. 5. As the number of hidden neurons increases, classifier training accuracy increases and so does power consumed by hidden layer. Training accuracy increases slowly once the number of hidden layer exceeds 40 but the power keeps increasing steadily. For 100 hidden neurons, the training accuracy is 0.9 but the power consumed by the hidden layer is 1.4mW. For this design, we have selected 28 hidden neurons which correspond to hidden layer power of 0.4mW.

III. Simulation Results

The ANN network trained using Matlab is simulated in Spectre. The test set contains 10,000 images. Fig. 6 shows the confusion matrix for the test set. The average classification accuracy over the 10 classes is 0.82. As shown in the confusion matrix, the accuracy for classifying '2' is the highest at 0.89 while accuracy for classifying '3' is the lowest at 0.77. Digit '3' has a low classification accuracy because it is similar in structure to '8' and '5' and the classifier has incorrectly labeled '3' 7% times each when the input is '5' or '8'. The classification accuracy is limited by the errors introduced due to resizing of the input images and can be improved further if the number of input features can be increased.

While accuracy is a good measure of how effective a classifier is, for multinomial classes with uneven distribution, accuracy does not provide the complete picture. We need other metrics like precision, recall [9] and f1 score to evaluate the performance of the classifier. Table I presents the precision, recall, f1 score and support values for the classifier using 10,000 test data. Considering the class '1', "precision" of class '1' is the ratio of correctly predicted '1's to the total number of images which are predicted as '1' by the classifier. "Recall" for class '1' is the ratio of correctly predicted '1's to the actual number of '1's in the dataset. In other words, "precision" is the ratio of true positive to (true positive + false positive), while "recall" is the ratio of true positive to (true positive + false negative). Thus, a high precision and high recall value indicates that the classifier is performing well. "f1 score" captures the effect of both false positives and false negatives by taking the harmonic mean of precision and recall. The "support" column in Table I indicates the number of times each class appears in the test dataset. The proposed classifier has the lowest f1 score for '8'. This observation is consistent

with the confusion matrix plot in Fig. 6 and the low f1 score is due to the fact that after reducing the input features, the distinction between '3', '5' and '8' are reduced.

It is interesting to assess the effect of mismatches on the classifier accuracy. The transistors are sized up to ensure that random mismatch does not exceed 2%. Fig. 7 shows the histogram of classification accuracy extracted from 10 monte-carlo runs across process and mismatch corners. It can be seen that the classifier has an average accuracy of 0.819 with a standard deviation of 0.02. Thus, the classifier accuracy is not affected by random variations in transistor sizes. Fig. 8 shows histogram of classifier accuracy as the proposed ANN performs classification repeatedly on the same test set in presence of transient noise. The standard deviation of classification accuracy is only 0.17m indicating that the proposed ANN has very high signal-to-noise ratio.

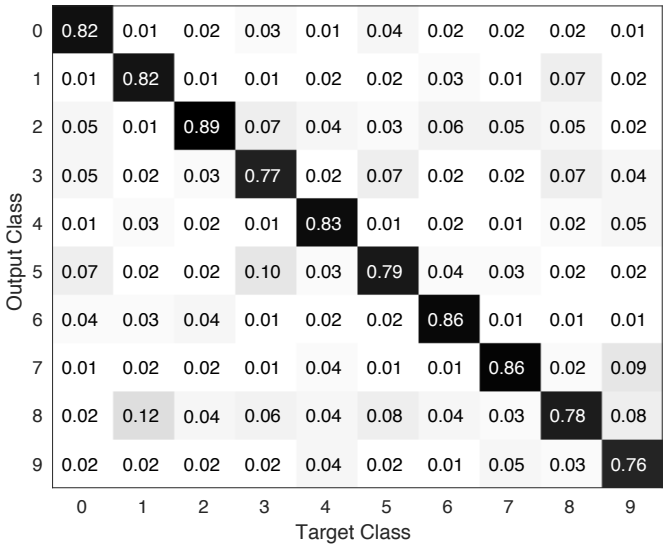


Fig. 6. Confusion matrix plot

TABLE I
PRECISION AND RECALL FOR THE CLASSIFIER

Label	Precision	Recall	f1 score	Support
0	0.9122	0.8135	0.86	980
1	0.91724	0.8076	0.8589	1135
2	0.7355	0.8764	0.7998	1032
3	0.7733	0.7627	0.7679	1010
4	0.8921	0.8241	0.8567	982
5	0.7018	0.7825	0.74	892
6	0.8914	0.8472	0.8688	958
7	0.8424	0.8507	0.8465	1028
8	0.5287	0.771	0.6273	974
9	0.8662	0.7502	0.804	1009

Table II compares our work with state-of-the-art classifiers. All of [1], [3], [5] implements only the weak linear regression classifiers on-chip and the boosting is done off-chip to improve accuracy. As can be seen from Table II, the proposed classifier

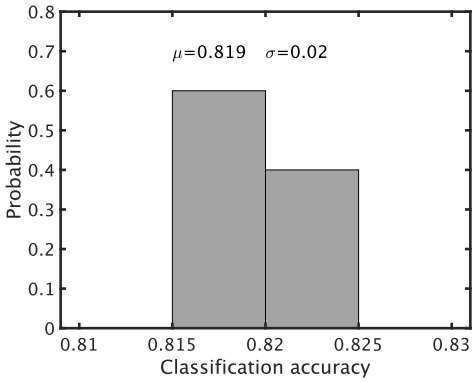


Fig. 7. Classifier accuracy versus monte-carlo iterations

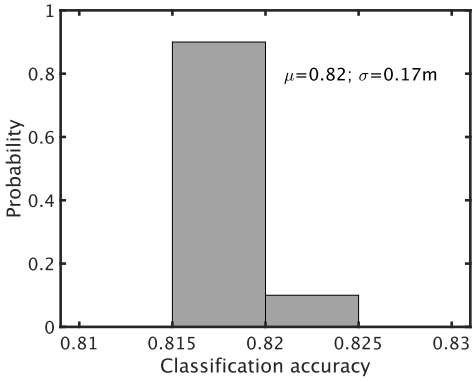


Fig. 8. Classifier accuracy versus noise

consumes 5X lower energy/classification than state-of-the-art while achieving comparable classification accuracy as existing analog/mixed-signal classifiers. Accuracy of our classifier can be improved if we can accommodate a larger input size.

TABLE II
COMPARISON WITH STATE-OF-THE-ART CLASSIFIERS

	[3]	[1]	[5]	This work
Process(nm)	130	130	130	65
Architecture	ADC	Comparator	SRAM	Amplifier
Application	Gender	MNIST	MNIST	MNIST
Accuracy	0.83	0.9	0.9	0.82
Speed (MHz)	0.02	0.2	50	1
Energy(nJ/ classification)	655^a	0.534^a	0.633^a	0.1^b

^{a} measured; ^{b} simulated results

IV. CONCLUSION

In this paper, we have presented an analog machine learning classifier which uses two-transistor common-source amplifier as the basic building block. We have used the popular MNIST dataset with hand written images from 0-9 to demonstrate the effectiveness of the proposed classifier. The proposed classifier has an accuracy of 0.82 and consumes only 100pJ/classification which is 5X better than state-of-the-art.

REFERENCES

[1] Z. Wang and N. Verma, "A low-energy machine-learning classifier based on clocked comparators for direct inference on analog sensors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 11, pp. 2954–2965, 2017.

[2] E. H. Lee and S. S. Wong, " A 2.5 GHz 7.7 TOPS/W switched-capacitor matrix multiplier with co-designed local memory in 40nm," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2016, pp. 418–419.

[3] Z. Wang, J. Zhang, and N. Verma, "Realizing Low-Energy Classification Systems by Implementing Matrix Multiplication Directly Within an ADC." *IEEE Trans. Biomed. Circuits and Systems*, vol. 9, no. 6, pp. 825–837, 2015.

[4] F. N. Buhler, A. E. Mendrela, Y. Lim, J. A. Fredenburg, and M. P. Flynn, "A 16-channel noise-shaping machine learning analog-digital interface," in *IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, 2016, pp. 1–2.

[5] J. Zhang, Z. Wang, and N. Verma, "A machine-learning classifier implemented in a standard 6T SRAM array," in *IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, 2016, pp. 1–2.

[6] D. P. Solomatine and D. L. Shrestha, "AdaBoost. RT: a boosting algorithm for regression problems," *Neural Networks*, vol. 2, pp. 1163–1168, 2004.

[7] M. Carrasco-Robles and L. Serrano, "A novel CMOS current mode fully differential tanh (x) implementation," in *IEEE International Symposium on Circuits and Systems*, 2008, pp. 2158–2161.

[8] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.

[9] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.