A Ternary Neural Network with Compressed Quantized Weight Matrix for Low Power Embedded Systems

Son Ngoc Truong
Faculty of Electrical and Electronics Engineering
Ho Chi Minh City University of Technology and Education
HCM City, Vietnam
sontn@hcmute.edu.vn

Received: 14 January 2022 | Revised: 2 February 2022 | Accepted: 11 February 2022

Abstract—In this paper, we propose a method of transforming a real-valued matrix to a ternary matrix with controllable sparsity. The sparsity of quantized weight matrices can be controlled by adjusting the threshold during the training and quantizing process. A 3-layer ternary neural network was trained with the MNIST dataset using the proposed adjustable dynamic threshold. The sparsity of the quantized weight matrices varied from 0.1 to 0.6 and the obtained recognition rate reduced from 91% to 88%. The sparse weight matrices were compressed by the compressed sparse row format to speed up the ternary neural network, which can be deployed on low-power embedded systems, such as the Raspberry Pi 3 board. The ternary neural network with the sparsity of quantized weight matrices of 0.1 is 4.24 times faster than the ternary neural network without compressing weight matrices. The ternary neural network is faster as the sparsity of quantized weight matrices increases. When the sparsity of the quantized weight matrices is as high as 0.6, the recognition rate degrades by 3%, however, the speed is 9.35 times the ternary neural network's without compressing quantized weight matrices. Ternary neural network work with compressed sparse matrices is feasible for low-cost, low-power embedded systems.

Keywords—ternary neural network; deep learning; image recognition; quantized neural network.

I. INTRODUCTION

Deep Neural Networks (DNNs) have achieved impressive success in the field of computer vision [1-4]. Modelling the human brain using DNNs requires a massive number of computation tasks including addition and multiplication. Therefore, it is often challenging to implement DNNs on low-power edge devices such as mobile embedded systems [5]. Edge computing has been attracted much attention recently because it has a lot of advantages in terms of cost and security. To run DNNs on low-power edge devices, many optimized DNN architectures have been proposed. To increase the accuracy, DNNs can be trained on a GPU and then the trained models are loaded to low-cost embedded systems, such as the Raspberry Pi board [6-8]. Another method is to add the external accelerating Neural Computer Stick (NCS) to the Raspberry Pi when deploying the DNNs on it [9]. These deployments of DNNs on the low-cost Raspberry Pi board are based on the full-precision weight, which dramatically consumes power and processing time. The memory usage and inference speed of such models have not been considered. An alternative technique to enhance the performance of DNNs deployed on low-cost computers is to quantize the parameters to speed up the DNNs' run-time and reduce memory consumption [10-17]. Traditionally, 32-bit floating-point is used for numerical formats in DNNs, which has a big impact on speed and memory usage. Reducing the number of bits representing DNN parameters is considered for low-power edge devices. In particular, using numerical formats with lower precision than 32-bit floating point yields numerous benefits. 16-bit floating-point and 8-bit floating-point are commonly used for lightweight DNNs without sacrificing accuracy [5]. Substantial research efforts to use lower precision such as ternary and binary representation of parameters (synaptic weights) have been invested to make possible the implement of DNNs on low-power edge devices [10-17].

A binary neural network constrains synaptic weights to the binary space of {-1,1}. In a binary neural network, the conventional 32-bit floating-point multipliers are replaced by the logical XNOR operations to speed up running time and reduce memory consumption. However, the accuracy of binary neural networks is lower than full-precision neural networks because only one bit is used to represent the synaptic weight and the activation function. To increase the accuracy, ternary neural networks that constraint the synaptic weights to the ternary space {-1, 0, 1} have been proposed [14-16]. When training a ternary neural network, the weights are updated using real-valued variables and are then constrained to -1, 0, or +1 using the ternarization function [14-16]. The ternazization with dynamic threshold yields faster convergence in the training phase and higher accuracy in the inference phase [18]. However, the dynamic threshold, which is based on the mean and standard deviation of real-valued variables, produces an unpredicted number of -1, 0, and +1 bits in the synaptic weight matrices. In this work, we adjust the threshold during the quantization process to obtain sparse weight matrices with

Corresponding author: Son Ngoc Truong

different sparsity (number of zero values), and measure the accuracy of the ternary neural network with different sparsity of quantized weight matrices. It turned out that we can increase the sparsity quantized weight matrices with small loss in accuracy. The sparse weight matrices are then compressed to reduce the size and speed up the matrix multiplication, which consumes most time in the forward pass of DNNs.

II. TERNARY NEURAL NETWORK WITH COMPRESSED WEIGHT MATRICES FOR LOW-POWER EMBEDDED SYSTEMS

Figure 1 shows the concept of a ternary neural network in which the weights are constrained to -1, 0, and 1 [18, 19]. x_1 - x_n are binary inputs and h_1 - h_m are the neuron outputs of the hidden layer, which are also quantized to binary. y_1 - y_k are the neuron outputs for k classes. In Figure 1, Wh is the input-to-hidden layer weight matrix and Wo is the hidden-to-output layer weight matrix. Here, the weight matrices are composed of -1, 0, and 1 representing the inhibitory, contactless, and excitatory synapses.

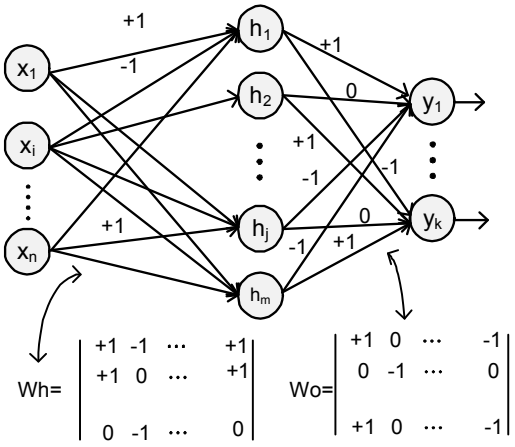

Fig. 1. The conceptual diagram of a ternary neural network, where the synaptic weights are -1, 0, or +1 representing inhibitory, contactless, and excitatory synapses.

Ternary neural network represents weights using fewer bits than a full-precision neural network. The ternary weights can be represented by lower bit signed integer values or complementary binary arrays [19]. The amount of required memory for the model's parameters of ternary neural networks is substantially less than that of the full-precision neural networks. The ternary neural network is trained using the traditional gradient descent method that updates the weights in the direction of the maximum decrease of the loss function. The weights are updated with real values and transformed to the binary values using the following quantization function [18]:

$$w_t = \begin{cases} -1, & \text{if } w_r <= -w_{threshold} \\ 0, & \text{if } -w_{threshold} < w_r < w_{threshold} \\ +1, & \text{if } w_r >= w_{threshold} \end{cases} \quad (1)$$

where $w_{threshold}$ is the threshold weight, w_r is the real-valued weight, and w_t is the ternary weight of -1, 0, or +1. By using (1), the ternary weights are obtained by comparing the real-valued weights with a positive threshold value. It can be observed that for every training iteration, the distributions of synaptic weights are different. Therefore, a dynamic threshold is selected using the Gaussian distribution proposed in our previous work [18]. The proposed method attempts to equalize the number of negative weights, zero weights, and positive weights. The quantization function with dynamic threshold is presented in (2) [18]:

$$w_t = \begin{cases} -1, & \text{if } w_r <= \mu - 0.44\sigma \\ 0, & \text{if } \mu - 0.44\sigma < w_r < \mu + 0.44\sigma \\ +1, & \text{if } w_r >= \mu + 0.44\sigma \end{cases} \quad (2)$$

where μ and σ are respectively the mean and standard deviation of real valued synaptic weights. According to the Gaussian distribution, if the threshold is selected to be μ-0.44σ and μ+0.44σ, we obtain 33%, 34%, and 33% as the number of negative synaptic weights, zero-value synaptic weights, and positive synaptic weights respectively [18]. The percentages of negative, zero, and positive synaptic weights are maintained constantly every epoch of the training process because the threshold is adapted to the distribution of synaptic weights.

Increasing the number of zero values in quantized weight matrices leads to higher sparsity. The sparse matrix can be compressed to reduce the memory consumption and the matrix multiplication time. In this work, we control the percentage of zeros by modifying the quantization function as follows:

$$w_t = \begin{cases} -1, & \text{if } w_r <= \mu - \lambda\sigma \\ 0, & \text{if } \mu - \lambda\sigma < w_r < \mu + \lambda\sigma \\ +1, & \text{if } w_r >= \mu + \lambda\sigma \end{cases} \quad (3)$$

where λ is a variable that controls the threshold. In (3), if we increase λ, the number of zeros will increase. The higher the value of λ, the higher the sparseness of the quantized weight matrices. The sparse weight matrices can be compressed to reduce the memory usage and speed up the forward pass. The sparse weight matrices are compressed using the Compressed Sparse Row (CSR) format, which potentially leads to a substantial decrease in computational time and speeds-up the neural networks [20-23].

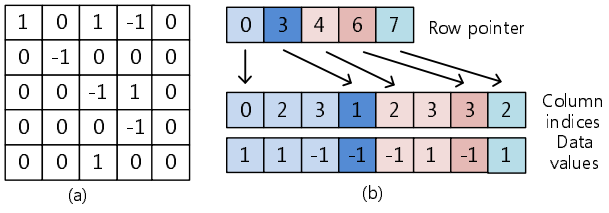

Fig. 2. An example of CSR. (a) Sparse matrix with high sparsity and (b) CSR representation of the sparse matrix

Figure 2 shows an example of the CSR format when representing a sparse matrix. Figure 2(a) shows a sparse matrix and its CSR representation is shown in Figure 2(b). CSR is a popular and general-purpose sparse matrix representation. The matrix is stored using three arrays, which are the row pointer array, the column indices array, and the data values array [23]. The pointer array stores the pointers to the beginning of every row, the column indices stores the corresponding column indices, and the data value array stores the nonzero values, as illustrated in Figure 2. The row pointer array begins with the value of 0, for the first row, the first, the 2nd, and the 3rd column of the sparse matrix have the values of respectively 1, 1, and -1, presented in the column indices array and data values in Figure 2(b). The second row of the sparse matrix is represented by the second element in the row pointer array, which has the value of 3. The value of -1 in the second row is represented by the column indices of 1 and the data value of -1, as shown in Figure 2(b). The sparse matrix in Figure 2(a) can be compressed by using the arrays in Figure 2(b). By doing this, the memory and time consumption for matrix multiplication are significantly reduced.

During the forward-pass propagation, the neuron's output is calculated by using matrix multiplication. Assume that $x = [x_1, x_2, ..., x_n]$ is the $1 \times n$ input vector, Wh is the $m \times n$ input-to-hidden layer weight matrix, $h = [h_1, h_2, ..., h_m]$ is the output vector of the hidden layer, the forward-pass propagation performs the below computational task:

$$h = \sigma(xW_h^T) \quad (4)$$

Equation (4) consumes more power and time as we increase the size of the input vector and the weight matrix. If the weight matrix is sparse and represented by CSR, we can replace the matrix multiplication in (4) with the Sparse matrix-vector multiplication, which is faster than traditional matrix multiplication [20]. In this work, we represent weight matrices using sparse matrices and compress them with CSR. The sparse matrix multiplication is performed by using the Scientific Python (Scipy) library to save computational time [24].

III. EXPERIMENTAL RESULTS

A three-layer ternary neural network was deployed on a low-power Raspberry Pi board for the application of image recognition. The network was trained and tested on the MNIST dataset for recognizing images of handwritten digits [25]. The input layer has 784 units corresponding to 784 image pixels. The inputs are binary. The hidden layer has 512 neurons and the output layer has 10 neurons for recognizing 10 digits. The network is trained using Stochastic Gradient Descent with the Momentum method. The real-valued weights are transformed to binary weights using the proposed adjustable dynamic threshold. By adjusting the variable in (3), we achieved the recognition rate with varied sparsity of quantized weight matrices, as presented in Figure 3. In Figure 3, the sparsity of the quantized weight matrix is varied from 0.1 to 0.6 by adjusting λ, as explained above. The recognition rate slightly degraded as the sparsity of the quantized weight matrix increased. When the sparsity of the quantized weight matrices was as small as 0.1, the ternary neural network produced a recognition rate of 91%. When the sparsity of the quantized weight matrix increased to 0.6, the recognition rate was reduced by 3%. The results indicated that increasing the sparsity of the quantized weight matrix led to a small decrease in accuracy. In this work, the training is performed on edge device, Raspberry Pi board, and the network is simply constituted of an input layer, a hidden layer, and an output layer with the weights quantized to -1, 0, and +1. The ternary weights are obtained by the proposed dynamic threshold quantization with controllable output sparsity. The accuracy of the ternary neural network is slightly lower than the full-precision neural network, however it has the advantages of less memory usage and faster inference time. More importantly, the proposed ternary neural network is promising for low-cost edge devices. Sparse quantized weight matrices were represented using the CSR format, which significantly enhances the inference processing. With a fixed-size sparse quantized weight matrix, the higher sparsity results in the smaller size CSR representation and faster CSR matrix multiplication.

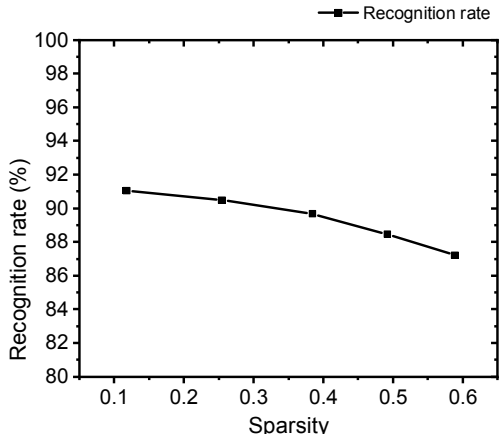

Fig. 3. Recognition rate with varied sparsity of quantized weight matrices.

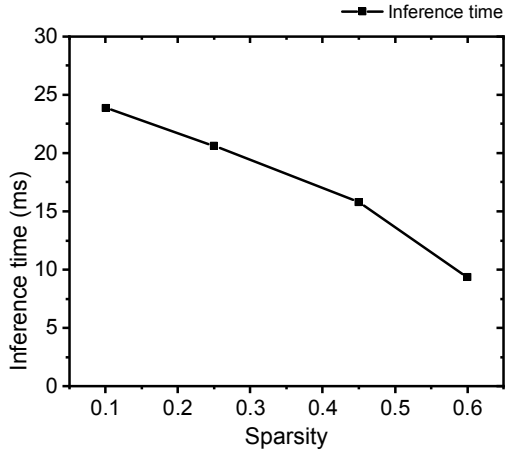

Fig. 4. Inference time with varied sparsity of quantized weight matrices.

Figure 4 shows the inference time of the 3-layer ternary neural network deployed on a low-power embedded system, Raspberry Pi 3, for an input image with varied sparsity of quantized weight matrices. The inference time is the forward-

pass propagation time required to propagate one image from the input layer to the output layer, which is also the time for predicting one image. For the uncompressed quantized weight matrices, the forward pass propagation takes 87.28ms, and such inference time does not depend on the sparsity of quantized weight matrices. Figure 4 shows the inference time with varied sparsity of quantized weight matrices when the quantized weight matrices were compressed with CSR. The matrix multiplication is performed using the Scientific Python library for the input vector and CSR arrays. For a sparsity of 0.1, the forward-pass propagation takes 20.606ms, which 4.24 times faster than the ternary neural network with uncompressed quantized weight matrices. More interestingly, when the sparsity increases, the array size of CSR representation for sparse matrices is more reduced, resulting in faster multiplication. For a sparsity of 0.6, the inference time of the compressed-weight-matrix ternary neural network is 9.335ms, which is 9.35 times faster than the original ternary neural network.

Quantizing the weight matrix is one of the techniques that are suitable for deploying DNNs on low-cost computers. Quantized neural networks can save the required memory for storing the model's parameters and internal parameters, and implementing faster than full-precision neural networks for speech and image recognition, as presented in [19]. In this work, we propose a method to control the sparsity of quantized weight matrices during the training process and compress the weight matrices using CSR representation. The high sparsity of quantized weight matrices sacrifices little accuracy, but speeds up the ternary neural network by a factor that reaches 9.35. The proposed idea is deployed on a simple 3-layer neural network for hand-written character recognition. Utilizing high sparsity quantized weight matrices and CSR makes the ternary neural network possible to implement on low-cost, low-power embedded systems such as general-purpose Raspberry Pi 3 board.

IV. CONCLUSION

In this paper, we proposed a quantization function that can control the sparsity of quantized weight matrices for ternary neural networks. The sparsity of quantized weight matrices of the ternary network varied from 0.1 to 0.6 when the ternary neural network was trained with the MNIST dataset. The obtained recognition rate varied from 91% to 88%. The sparse weight matrices were compressed using the CSR format. The ternary neural network with compressed weight matrices was 4.24 times and 9.35 times faster than the original ternary neural network, when the sparsity of the quantized weight matrices was 0.1 and 0.6 respectively. Ternary neural networks with compressed quantized weight matrices are suitable for implementation on low-power embedded systems for the application of image recognition.

ACKNOWLEDGMENT

This research is supported by the Ho Chi Minh City University of Technology and Education (HCMUTE), Vietnam.

REFERENCES

[1] K. L. Masita, A. N. Hasan, and T. Shongwe, "Deep Learning in Object Detection: a Review," in *International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems*, Durban, South Africa, Aug. 2020, pp. 1–11, https://doi.org/10.1109/icABCD49160.2020.9183866.

[2] A. Alsheikhy, Y. Said, and M. Barr, "Logo Recognition with the Use of Deep Convolutional Neural Networks," *Engineering, Technology & Applied Science Research*, vol. 10, no. 5, pp. 6191–6194, Oct. 2020, https://doi.org/10.48084/etasr.3734.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *26th Annual Conference on Neural Information Processing Systems*, Nevada, USA, Dec. 2012, vol. 25, pp. 1097–1105.

[4] S. Sahel, M. Alsahafi, M. Alghamdi, and T. Alsubait, "Logo Detection Using Deep Learning with Pretrained CNN Models," *Engineering, Technology & Applied Science Research*, vol. 11, no. 1, pp. 6724–6729, Feb. 2021, https://doi.org/10.48084/etasr.3919.

[5] J. Lee, J. Lee, D. Han, J. Lee, G. Park, and H.-J. Yoo, "An Energy-Efficient Sparse Deep-Neural-Network Learning Accelerator With Fine-Grained Mixed Precision of FP8–FP16," *IEEE Solid-State Circuits Letters*, vol. 2, no. 11, pp. 232–235, Aug. 2019, https://doi.org/10.1109/LSSC.2019.2937440.

[6] K. Yokoo, M. Atsumi, K. Tanaka, H. Wang, and L. Meng, "Deep Learning based Emotion Recognition IoT System," in *International Conference on Advanced Mechatronic Systems*, Hanoi, Vietnam, Dec. 2020, pp. 203–207, https://doi.org/10.1109/ICAMechS49982.2020.9310135.

[7] N. Lee, M. H. Azarian, M. Pecht, J. Kim, and J. Im, "A Comparative Study of Deep Learning-Based Diagnostics for Automotive Safety Components Using a Raspberry Pi," in *IEEE International Conference on Prognostics and Health Management*, San Francisco, CA, USA, Jun. 2019, pp. 1–7, https://doi.org/10.1109/ICPHM.2019.8819436.

[8] B. H. Curtin and S. J. Matthews, "Deep Learning for Inexpensive Image Classification of Wildlife on the Raspberry Pi," in *10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference*, New York, NY, USA, Oct. 2019, pp. 0082–0087, https://doi.org/10.1109/UEMCON47517.2019.8993061.

[9] E. Kristiani, C.-T. Yang, and K. L. Phuong Nguyen, "Optimization of Deep Learning Inference on Edge Devices," in *International Conference on Pervasive Artificial Intelligence*, Taipei, Taiwan, Dec. 2020, pp. 264–267, https://doi.org/10.1109/ICPAI51961.2020.00056.

[10] 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," Mar. 2016, Accessed: Feb. 12, 2022. [Online]. Available: http://arxiv.org/abs/1602.02830.

[11] Y. Wang, J. Lin, and Z. Wang, "An Energy-Efficient Architecture for Binary Weight Convolutional Neural Networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 2, pp. 280–293, Oct. 2018, https://doi.org/10.1109/TVLSI.2017.2767624.

[12] T. Simons and D.-J. Lee, "A Review of Binarized Neural Networks," *Electronics*, vol. 8, no. 6, Jun. 2019, Art. no. 661, https://doi.org/10.3390/electronics8060661.

[13] C. Baldassi, A. Braunstein, N. Brunel, and R. Zecchina, "Efficient supervised learning in networks with binary synapses," *Proceedings of the National Academy of Sciences*, vol. 104, no. 26, pp. 11079–11084, Jun. 2007, https://doi.org/10.1073/pnas.0700324104.

[14] K. Hwang and W. Sung, "Fixed-point feedforward deep neural network design using weights +1, 0, and −1," in *IEEE Workshop on Signal Processing Systems*, Belfast, UK, Oct. 2014, pp. 1–6, https://doi.org/10.1109/SiPS.2014.6986082.

[15] H. Yonekawa, S. Sato, and H. Nakahara, "A Ternary Weight Binary Input Convolutional Neural Network: Realization on the Embedded Processor," in *IEEE 48th International Symposium on Multiple-Valued Logic*, Linz, Austria, May 2018, pp. 174–179, https://doi.org/10.1109/ISMVL.2018.00038.

[16] S. Yin *et al.*, "An Energy-Efficient Reconfigurable Processor for Binary- and Ternary-Weight Neural Networks With Flexible Data Bit Width,"

IEEE Journal of Solid-State Circuits, vol. 54, no. 4, pp. 1120–1136, Apr. 2019, https://doi.org/10.1109/JSSC.2018.2881913.

[17] L. Deng, P. Jiao, J. Pei, Z. Wu, and G. Li, "GXNOR-Net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework," *Neural Networks*, vol. 100, pp. 49–58, Dec. 2018, https://doi.org/10.1016/j.neunet.2018.01.010.

[18] S. N. Truong, "A Dynamic Threshold Quantization Method for Ternary Neural Networks for Low-cost Mobile Robots," *International Journal of Computer Science and Network Security*, vol. 20, no. 2, pp. 16–20, 2020.

[19] S. N. Truong, "A Low-cost Artificial Neural Network Model for Raspberry Pi," *Engineering, Technology & Applied Science Research*, vol. 10, no. 2, pp. 5466–5469, Apr. 2020, https://doi.org/10.48084/etasr.3357.

[20] J. L. Greathouse and M. Daga, "Efficient Sparse Matrix-Vector Multiplication on GPUs Using the CSR Storage Format," in *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, New Orleans, LA, USA, Nov. 2014, pp. 769–780, https://doi.org/10.1109/SC.2014.68.

[21] X. Feng, H. Jin, R. Zheng, K. Hu, J. Zeng, and Z. Shao, "Optimization of Sparse Matrix-Vector Multiplication with Variant CSR on GPUs," in *17th International Conference on Parallel and Distributed Systems*, Tainan, Taiwan, Dec. 2011, pp. 165–172, https://doi.org/10.1109/ICPADS.2011.91.

[22] H. Kabir, J. D. Booth, and P. Raghavan, "A multilevel compressed sparse row format for efficient sparse computations on multicore processors," in *21st International Conference on High Performance Computing*, Goa, India, Dec. 2014, pp. 1–10, https://doi.org/10.1109/HiPC.2014.7116882.

[23] J. C. Pichel and B. Pateiro-Lopez, "Sparse Matrix Classification on Imbalanced Datasets Using Convolutional Neural Networks," *IEEE Access*, vol. 7, pp. 82377–82389, 2019, https://doi.org/10.1109/ACCESS.2019.2924060.

[24] J. Ranjani, A. Sheela, and K. P. Meena, "Combination of NumPy, SciPy and Matplotlib/Pylab -a good alternative methodology to MATLAB - A Comparative analysis," in *1st International Conference on Innovations in Information and Communication Technology*, Chennai, India, Apr. 2019, pp. 1–5, https://doi.org/10.1109/ICIICT1.2019.8741475.

[25] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, Aug. 2012, https://doi.org/10.1109/MSP.2012.2211477.