TRUE GRADIENT-BASED TRAINING OF DEEP BINARY ACTIVATED NEURAL NETWORKS VIA CONTINUOUS BINARIZATION

Charbel Sakr[⋆]*, Jungwook Choi*[†]*, Zhuo Wang*[†‡]*, Kailash Gopalakrishnan*[†]*, Naresh Shanbhag*[⋆]

[⋆]Dept. of Electrical and Computer Engineering, University of Illinois at Urbana Champaign

[†]IBM T. J. Watson Research Center

ABSTRACT

With the ever growing popularity of deep learning, the tremendous complexity of deep neural networks is becoming problematic when one considers inference on resource constrained platforms. Binary networks have emerged as a potential solution, however, they exhibit a fundamental limitation in realizing gradient-based learning as their activations are non-differentiable. Current work has so far relied on approximating gradients in order to use the back-propagation algorithm via the straight through estimator (STE). Such approximations harm the quality of the training procedure causing a noticeable gap in accuracy between binary neural networks and their full precision baselines. We present a novel method to train binary activated neural networks using true gradient-based learning. Our idea is motivated by the similarities between clipping and binary activation functions. We show that our method has minimal accuracy degradation with respect to the full precision baseline. Finally, we test our method on three benchmarking datasets: MNIST, CIFAR-10, and SVHN. For each benchmark, we show that continuous binarization using true gradient-based learning achieves an accuracy within 1.5% of the floating-point baseline, as compared to accuracy drops as high as 6% when training the same binary activated network using the STE.

***Index Terms*—** deep learning, binary neural networks, activation functions

1. INTRODUCTION

Deep neural networks are becoming the de facto predictive models used in many machine learning tasks. Their popularity is the result of numerous victories deep learning has enjoyed in the past decade. Most notably, with AlexNet [1] winning the 2012 ImageNet Large Scale Visual Recognition challenge, extensive research efforts in the area followed and culminated with machines outperforming humans in recognition tasks [2]. However, this outstanding representational power comes at the price of very high complexity. Some of these networks require around 1 billion multiply-accumulates (MACs) [3]. It is in fact not uncommon to find networks with over 100 million parameters [4] and over 1000 layers [5]. Such high complexity makes these models hard to train, but most importantly, their deployment on resource-constrained platforms, such as ASICs, FPGAs, and microcontrollers, becomes problematic.

1.1. Related work

The importance of reducing the complexity of deep learning systems is well appreciated today. One approach is to optimize the structure of a neural network itself, such as by pruning [6], where weak connections are deleted and the reduced network is retrained. Dominant layers can also be decomposed into a cascade of smaller layers with an overall lesser complexity [7]. Taking advantage of the sparsity also enables efficient implementation via zero-skipping [3].

An orthogonal approach is to consider reduced precision neural networks. This can be done in one of two ways: quantizing pre-trained networks, or directly training in low precision. The first option is justified by the inherent robustness of neural networks suggesting that moderate quantization should not be catastrophic. This has led to interesting analytical investigations such as determining the correspondence between precision and signal-to-quantization-noise ratio [8]. A better understanding of accuracy in the presence of fixed-point quantization was presented in [9]. This study led to the discovery of an interesting trade-off between weight and activation precisions.

Directly training in low precision has also seen many advances. It was shown that training 16-bit fixed-point networks is possible using stochastic rounding [10]. It was later realized that training binary weighted neural networks, such as BinaryConnect [11], is possible provided a high precision representation of the weights is maintained during the training [12]. A natural extension is to consider activation binarization such as BinaryNet [13], XNOR-Net [14], and DoReFa-Net [15]. A close investigation of these works' reported performances reveals failure in achieving state-of-the-art accuracy. For instance, the accuracy gap between BinaryNet and BinaryConnect is slightly over 3% (on the CIFAR-10) dataset, despite using numerous optimization tricks such as stochastic rounding, shift-based ADAMAX, and early model selection. These shortcomings may arguably be attributed to the inability to use gradient-based learning because of the use of the non-differentiable binary activation function. Instead, these works have relied on gradient approximations via the straight through estimator (STE) [16] to enable back-propagation.

This work was supported in part by Systems On Nanoscale Information fabriCs (SONIC), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA.

This work is supported in part by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network.

[‡] Zhuo Wang is now an employee at Google.

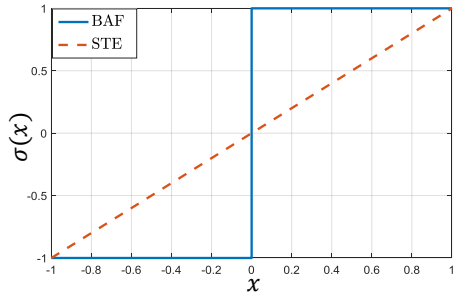


Fig. 1: A feedforward binary activation function (BAF) and its straight through estimator (STE). Conventional training of binary activated neural networks uses the non-differentiable binary activation during the feedforward computations but replaces it with the identity during back-propagation.

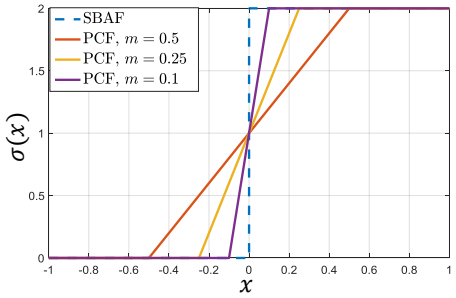


Fig. 2: A scaled binary activation function (SBAF) and some parametrized clipping functions (PCFs). The SBAF considered is $\sigma(x) = 2 \times \mathbb{1}_{x>0}$. The PCF is $\sigma(x) = \text{clip}(\frac{x}{m} + 1, 0, 2)$. With smaller value of m, the PCF approaches the SBAF.

The STE estimates the gradient of the binary activation function by replacing it with the identity function as shown in Fig. 1. Effectively, the back-propagation procedure "sees through" the binary activation function. The STE is justified in the context of stochastic gradient descent (SGD) because the computed gradient is an approximation to the true gradient with respect to the loss function being minimized. However, in the case of SGD, almost half a century of research has led to a well-established theory with performance guarantees [17]. In contrast, the STE method still lacks a complete analytical basis.

1.2. Contributions

We propose a new method to train binary activated networks. Instead of relying on gradient approximation, our method leverages the true SGD algorithm. We analytically demonstrate that our method guarantees minimal accuracy degradation with respect to the full precision baseline. We present numerical experiments that show successful training of binary activated networks on the MNIST [18], CIFAR-10 [19], and SVHN [20] datasets with accuracies within 1.5% with respect to the full precision baseline. We also train the same binary activated networks using the STE and observe accuracy drops as high as 6% justifying the superiority of our proposed method. Thus, the complexity benefits of binary activated networks are presented with minimal loss in accuracy over full precision networks.

The rest of this paper is organized as follows. Section 2 describes our proposed continuous binarization method and includes our analytical justification. Numerical results are reported in Section 3. We conclude our paper in Section 4.

2. PROPOSED METHOD: CONTINUOUS BINARIZATION

This section describes our proposed continuous binarization technique and its analytical justification.

2.1. Principle

Our main motivation is to employ true gradient-based learning for binary activated networks. Consequently, the use of the STE is not suitable. Instead, we replace the binary activation by a continuous, piecewise differentiable function. More specifically, we consider a scaled binary activation function (SBAF) of the form

$$\sigma(x) = \alpha \times \mathbb{1}_{x>0}$$

where α is a scale parameter and $\mathbb{1}$ is the indicator function. We also consider a parametrized clipping function (PCF) of the form

$$\sigma(x) = \text{clip}(\frac{x}{m} + \frac{\alpha}{2}, 0, \alpha) = \min(\max(\frac{x}{m} + \frac{\alpha}{2}, 0), \alpha) \tag{1}$$

where m and α are the slope and scale parameters, respectively. Observe that, for a small value of m, the PCF has a steeper slope and approaches the SBAF as shown in Fig. 2 for $\alpha = 2$. This observation is the key insight our method relies on. We propose to train a neural network using the PCF as the activation function and *learn* its parameter m while constraining it to be small. This is done by regularizing m using the L_2 or L_1 regularization schemes. When m is small enough, we replace the PCF by the SBAF to obtain the final binary activated network. Thus, our procedure uses SGD during training, but generates a binary activated network for inference.

2.2. Procedure

The proposed continuous binarization procedure is provided in Algorithm 1. An important caveat is that training the slope parameters of all layers simultaneously might lead to a *bottleneck effect* in the backward pass. This is because the gradients flowing through the PCF become sparse as the slope becomes steep, causing convergence to slow down or even stop. Hence, we learn the slope parameter m in stages, *one layer at a time*, where at stage l, all layers up to layer $l-1$ are binary activated and have frozen weights and only the slope parameter m at layer l is learned.

As the parameter m needs to be regularized, e.g. via L_1 or L_2 regularizations, the usual caveats apply to the choice

Algorithm 1 Continuous Binarization procedure: L is the number of layers, $I^{(l)}$ is the number of iterations for which the PCF parameters of layer l are trained, μ is the learning rate, and $\lambda^{(l)}$ is the slope regularization coefficient at layer l

for $l = 1$ to L **do**
  for $i = 1$ to $I^{(l)}$ **do**
    (inputs, labels) ← getMiniBatch()
    hidden ← inputs
    ▷ Feedforward pass:
    for $l = 1$ to L **do**
      W ← getLayerWeights(j)
      preActivation ← computePreActivation(hidden, W, l)
      if $j < l$ **then**
        hidden ← SBAF(preActivation, l)
      end if
      if $l \leq j < L$ **then**
        hidden ← PCF(preActivation, j)
      end if
      if $j == L$ **then**
        hidden ← outputActivation(preActivation)
        ▷ Generally a softmax
      end if
    end for
    predicted ← argmax(hidden)
    C ← costFunction(predicted, labels)
    ▷ Generally a cross entropy
    ▷ Backward pass and updates:
    for $j = l$ to L **do**
      W ← getLayerWeights(j)
      G ← computeGradients(C, W)
      $W \leftarrow W - \mu G$
    end for
    m ← getPCFSlope(l)
    α ← getPCFScale(l)
    g_m ← computeGradient(C + regularizer(m, $\lambda^{(l)}$), m)
    g_α ← computeGradient(C, α)
    $m \leftarrow m - \mu g_m$
    $\alpha \leftarrow \alpha - \mu g_\alpha$
  end for
end for

of the regularization coefficient λ. We identify three types of regularization in a network: 1) activations preceding fully connected layers: λ_1, 2) activations preceding convolutional layers: λ_2, and 3) activations preceding pooling layers: λ_3. A good strategy is to choose $\lambda_1 < \lambda_2 < \lambda_3$.

2.3. Analytical Justification

Given a network, let us denote by $N^{(l)}$ the obtained network using the SBAF for all layers until and including the l-th one and the PCF for all final layers past the l-th one. Consequently, $N^{(0)}$ is the original full precision network using the PCF at all layers and $N^{(L-1)}$ is the final binary activated network. Note that at the start of stage l ($l \geq 2$) of the continuous binarization procedure, we replace $N^{(l-1)}$ by $N^{(l)}$ after having regularized the parameter m at layer l.

We first show that the mean squared error (MSE) of approximating the PCF by the SBAF decreases linearly in m. Note that from (1), the intercepts with the lines $y = 0$ and $y = \alpha$ of the clipping function are at $x = -\frac{m\alpha}{2}$ and $x = \frac{m\alpha}{2}$, respectively. We assume the pessimistic scenario where the input x is uniformly distributed in $\left[-\frac{m\alpha}{2}; \frac{m\alpha}{2}\right]$. Then, the MSE between PCF and SBAF activations is computed as follows:

$$MSE = k \int_{-\frac{m\alpha}{2}}^{-\frac{m\alpha}{2}} \left(\frac{x}{m} + \frac{\alpha}{2} - \alpha \cdot \mathbb{1}_{x>0}\right)^2 dx = c \cdot m$$

where k is a normalization constant and $c = k\frac{\alpha^3}{12}$ is a constant. Hence, we establish that as m decreases, the perturbations due to switching activation functions at layer l decrease linearly in the mean squared sense.

Next, for a given input, let $\mathbf{a}_o$ and $\mathbf{a}_p$ be the feature vectors at layer l of $N^{(l-1)}$ and $N^{(l)}$, respectively. We will show that there is no mismatch between the predicted labels $\hat{y}_o$ and $\hat{y}_p$ of $N^{(l-1)}$ and $N^{(l)}$, respectively, provided the perturbation vector at layer l, $\mathbf{q_a} = \mathbf{a}_p - \mathbf{a}_o$, has a bounded magnitude. To do so, we use the output re-ordering argument similar to that in [9]. There is a mismatch when $\hat{y}_o = i$, $\hat{y}_p = j$, and $i \neq j$, where i, j are predicted classes. But this event occurs only if $f_i(\mathbf{a}_o) > f_j(\mathbf{a}_o)$ and $f_i(\mathbf{a}_p) < f_j(\mathbf{a}_p)$ where f_i, f_j are the soft outputs at indexes i, j, respectively. For small perturbations, we can use a first order Taylor approximation as in [9] to show:

$$\begin{aligned} &f_i(\mathbf{a}_p) < f_j(\mathbf{a}_p) \Rightarrow f_i(\mathbf{a}_o + \mathbf{q_a}) < f_j(\mathbf{a}_o + \mathbf{q_a}) \\ &\Rightarrow f_i(\mathbf{a}_o) - f_j(\mathbf{a}_o) < \|\mathbf{q_a}\| \, \|\nabla_{\mathbf{a}_o} f_j(\mathbf{a}_o) - \nabla_{\mathbf{a}_o} f_i(\mathbf{a}_o)\| \end{aligned}$$

where we used the Cauchy-Schwarz inequality. Therefore, by the contrapositive, we have no mismatch if

$$\|\mathbf{q_a}\| < \frac{f_i(\mathbf{a}_o) - f_j(\mathbf{a}_o)}{\|\nabla_{\mathbf{a}_o} f_j(\mathbf{a}_o) - \nabla_{\mathbf{a}_o} f_i(\mathbf{a}_o)\|} \tag{2}$$

For an M-class classification problem, we may extend the condition in (2) as follows:

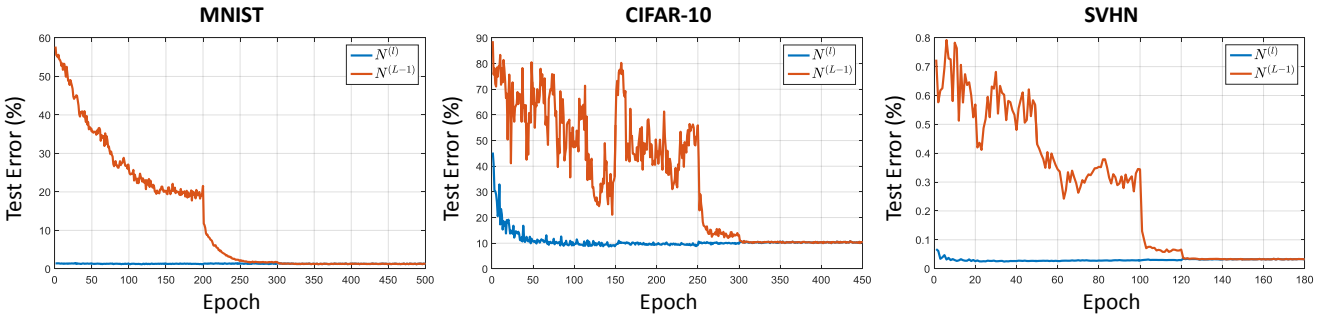
$$\|\mathbf{q_a}\| < \min_{j=1\ldots M,\ j\neq i} \frac{f_i(\mathbf{a}_o) - f_j(\mathbf{a}_o)}{\|\nabla_{\mathbf{a}_o} f_j(\mathbf{a}_o) - \nabla_{\mathbf{a}_o} f_i(\mathbf{a}_o)\|}$$

Thus, we establish an upper bound on the perturbation vector magnitude to avoid mismatch. But this magnitude decreases linearly as a function of m in the mean square sense. Hence, we argue that replacing the PCF by the SBAF at layer l has minimal effect on accuracy for small enough values of m.

3. NUMERICAL RESULTS

In order to evaluate our proposed continuous binarization method, we utilize three datasets: MNIST [18], CIFAR-10 [19], and SVHN [20]. For each, we consider a unique network inspired by BinaryNet [13], defined below:

- MNIST: A multi-layer perceptron with architecture $784 - 2048 - 2048 - 2048 - 10$.
- CIFAR-10: A convolutional neural network with architecture $128C3 - 128C3 - MP2 - 256C3 - 256C3 - MP2 - 512C3 - 512C3 - 1024FC - 1024FC - 10$.



(a) (b) (c)

Fig. 3: Illustration of the continuous binarization procedure for: (a) MNIST, (b) CIFAR-10, and (c) SVHN. Both blue and orange curves are for the same network. The blue curve is obtained by observing the test error for $N^{(l)}$. The orange curve is obtained by observing the test error for $N^{(L-1)}$.

	MNIST	CIFAR-10	SVHN
Full-precision Baseline	1.45%	9.04%	2.53%
Binarization via STE	1.54%	14.80%	4.05%
Continuous Binarization	1.27%	10.41%	3.20%

Table 1: Summary of Test Errors for the three networks trained for each dataset. Note the small gap in accuracy between baseline and binary activated networks for each dataset when using the proposed continuous binarization method. In contrast, binarization using the STE is not as successful.

- SVHN: A convolutional neural network with architecture $64C3-64C3-MP2-128C3-128C3-MP2-256C3-256C3-1024FC-1024FC-10$.

The network corresponding to each dataset is trained in three ways: 1) *full-precision baseline* using the clipping activation function, 2) *binarization via STE* [16], and 3) using the proposed *conitnuous binarization*. All results are summarized in Table 1.

Figure 3 illustrates the outcome of the continuous binarization procedure for each datasets. Plotted is the test error as a function of training epoch for $N^{(l)}$ and $N^{(L-1)}$. The networks are pre-trained with $m = 0.5$ and $\alpha = 2$ for all layers which is why the test error for $N^{(l)}$ has good initial conditions. Recall that $N^{(l)}$ uses the SBAF for all layers up to the l-th one and the PCF for all other layers, whereas $N^{(L-1)}$ uses only the SBAF. As l is progressively incremented in stages, the number of binary activated layers in $N^{(l)}$ is also incremented until all the network is binary activated and $N^{(l)}$ is identical to $N^{(L-1)}$, making both curves meet.

For the MNIST dataset, the full precision baseline is obtained using SGD and achieves 1.45% test error after 500 epochs. The STE implementation is obtained by training with Vanilla Adam [21] (we observed significant improvements over SGD). All network configurations are otherwise identical to the baseline. The test error obtained is 1.54% after 500 epochs. For continuous binarization, only L_2 regularization was used on m with $\lambda = 1$ for all layers. The first layer is trained for 200 epochs, and all other layers are trained for 100 epochs each. The test error at the end of the last training epoch we obtain is **1.27%**.

For the CIFAR-10 dataset, the baseline network is trained using SGD and achieves a test error of 9.04% using clipping after 500 epochs. The STE implementation is trained using Vanilla Adam and with the same configurations as the baseline otherwise. The test error obtained is 14.80% after 500 epochs. During continuous binarization, Vanilla Adam is used instead of SGD. This slightly negated the warm up effect due to the change in optimization technique. both L_2 and L_1 regularizations were used with coefficients $\lambda_1 = 0.001$, $\lambda_2 = 0.01$, $\lambda_3 = 0.1$. Each layer is trained for 50 epochs and the final test error obtained is **10.41%**.

For the SVHN dataset, the baseline network is trained using SGD and achieves a test error of 2.53% using clipping after 200 epochs. The STE implementation is trained using Vanilla Adam and with the same configurations as our baseline otherwise. The test error obtained is 4.05% after 200 epochs. During continuous binarization, the same procedure as that for CIFAR-10 is followed with a few modifications: each layer is trained for 20 epochs instead of 50 because this dataset is much larger; and only L_2 regularization is used for all layers past the fifth one. We obtain a final test error of **3.20%**.

In summary, for each of the three datasets, the test error using continuous binarization is within 1.5% of the full precision baseline. However, for binarization using the STE, the accuracy drop reaches up to 6%. This demonstrates the superiority of our proposed training procedure for binary activated networks.

4. CONCLUSION

We have presented a novel method for binarizing the activations of deep neural networks. We have presented a theoretical justification for our method. To demonstrate the validity of our approach, we have tested it on three deep learning benchmarks.

Future work includes further experimentations on larger models and datasets, combining the proposed activation binarization to weight binarization, and extension to the multi-bit activation.

5. REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[3] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne 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.

[4] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[5] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision*. Springer, 2016, pp. 646–661.

[6] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 1135–1143.

[7] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[8] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 2849–2858.

[9] Charbel Sakr, Yongjune Kim, and Naresh Shanbhag, "Analytical guarantees on numerical precision of deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 3007–3016.

[10] S. Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan, "Deep learning with limited numerical precision," in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 1737–1746.

[11] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.

[12] Hao Li, Soham De, Zheng Xu, Christoph Studer, Hanan Samet, and Tom Goldstein, "Training quantized nets: A deeper understanding," in *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[13] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4107–4115.

[14] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[15] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[16] Yoshua Bengio, Nicholas Léonard, and Aaron Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[17] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

[18] Yann LeCun, Corinna Cortes, and Christopher JC Burges, "The MNIST database of handwritten digits," 1998.

[19] Alex Krizhevsky and Geoffrey Hinton, "Learning multiple layers of features from tiny images," 2009.

[20] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS workshop on deep learning and unsupervised feature learning*, 2011, vol. 2011, p. 5.

[21] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.