Data Selective Deep Neural Networks For Image Classification

Marcele O. K. Mendonça, Jonathas O. Ferreira and Paulo S. R. Diniz
SMT - Signals, Multimedia, and Telecommunications Lab.
Universidade Federal do Rio de Janeiro, DEL/Poli & PEE/COPPE/UFRJ
P.O. Box 68504, Rio de Janeiro, RJ, 21941-972, Brazil
{marcele.kuhfuss, jonathas.ferreira, diniz}@smt.ufrj.br

***Abstract*—As the volume of data keeps growing, the use of deep neural networks has been widespread in a variety of applications, including image classification. This big available data has led to an increasing interest in designing more efficient systems. Most applications use all training data without taking into account their relevance. A mini-batch gradient descent algorithm is preferable in practice, but the chosen batch size is typically based on empirical tests, and it depends on the dataset characteristics. This work proposes a data-selection strategy applied to classification problems leading to computational savings and, in most cases classification error reduction. A few examples corroborate the effectiveness of the proposed approach.**

***Index Terms*—Neural Network, Data Selection, Gradient Descent, Deep Learning**

I. Introduction

In recent years, machine learning methods have been widely applied to solve many tasks, such as computer vision [1], natural language processing [2], and image-processing [3]. By processing a large amount of available data, deep neural networks (DNNs) have significantly improved the performance in various of areas, including healthcare, climate monitoring, and finance. Nevertheless, enormous success in performance is obtained at the cost of high computational complexity and storage demand. The sheer volume of available data has led to increasing interest in designing more efficient systems. Deep learning is powered by a gradient descent (GD) algorithm or variants [4]. Instead of training using all data examples at once like batch GD or just a single training example, as in stochastic GD, mini-batch GD split the training dataset into small batches used at each epoch. The mini-batch size plays a significant role in the behavior of the GD algorithm. With a large mini-batches, the gradient estimate is more accurate, but the probability of getting trapped in local minima increases [5]. Moreover, using large mini-batch may slow down the convergence rate in practice. In contrast, smaller mini-batches generate a noisy estimate of the gradient, but the model convergence requires fewer epochs [6]. When considering small mini-batches, memory resources can be saved, leading to more efficient systems [7]. The most suitable mini-batch size is usually empirically obtained, as it is highly data-dependent.

In this work, we propose a procedure that exploits this data dependency to manage and save available resources in Neural Network (NN) algorithms [4], [8], [9]. Motivated by the increasing amount of irrelevant and redundant information available for processing, we select the most informative data samples for mini-batch training. In [10], training is improved by emphasizing the uncertain samples. In contrast, our proposed method selects only the informative samples and discards the remaining to reduce the computational complexity. Like our method, in [11]–[14] the batch-selection or hard example, mining is based on the current loss. In our case, the loss is the classification error. Our work is also related to Teaching methods in shallow classifiers [15], [16], but we propose a method to be applied in deep network structures. NN's proposed data-selection method is similar to the one employed in the linear adaptive filtering and kernel adaptive filtering areas [17]–[19]. The approach considers the data's relevance during the learning process at each iteration of the parameter update.

The structure of the paper is as follows. In section II, we describe the basic concepts of a feed-forward neural network [8], [9], [20]. NN's data selection method is proposed in section III, aiming to reduce the computational cost and improve the performance measures related to the standard NNs. Section IV presents some simulation results. Section V includes some concluding remarks.

II. System Model

We consider the feed-forward neural network [21] composed by layers $l = 0, 1, 2, \cdots, L$, where $l = 0$ is the input layer and $l = L$ is the output layer, as illustrated in Figure 1. The full dataset consists of input-output pairs

$$\mathcal{D} = \{(\mathbf{x}(1), \mathbf{y}(1)), (\mathbf{x}(2), \mathbf{y}(2)), \cdots, (\mathbf{x}(M), \mathbf{y}(M))\}, \quad (1)$$

where $\mathbf{x}(m) \in \mathbb{R}^{N\times 1}$ for $m = 1, \cdots M$. In a classification problem with B classes, the desired signal $\mathbf{y}(m) \in \mathbb{R}^{B\times 1}$ is one-hot-encoded, meaning that if c is the correct class, $y_c(m) = 1$ and $y_i(m) = 0$ for $i \neq c$. By utilizing the softmax activation function at the output layer, the output signal $\hat{\mathbf{y}}$ will return a probability distribution on the classes, that is,

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This work was also supported by the research councils: CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), and FAPERJ (Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro).

$\hat{y}_i(m) = P(c = i)$ and the class with greater probability is chosen.

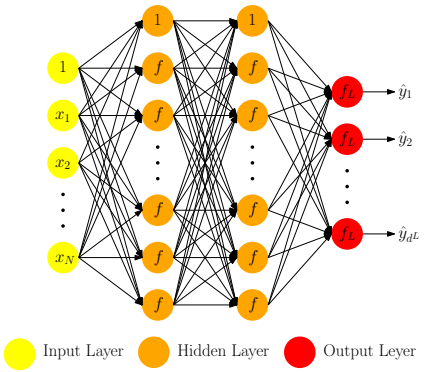

Fig. 1: A feed-forward neural network with $L = 3$ layers.

In mini-batch training, b examples are randomly selected from $\mathcal{D}$ to form $\mathbf{X}_{(t,i)} = [\mathbf{x}(1), \mathbf{x}(2), \cdots, \mathbf{x}(b)]$ and $\mathbf{Y}_{(t,i)} = [\mathbf{y}(1), \mathbf{y}(2), \cdots, \mathbf{y}(b)]$ at each iteration $i = 1, 2, \cdots I$ until complete an epoch t. In the forward pass, the input signals in $\mathbf{X}_{(t,i)}$ flow forward through the network and produce the estimates in $\hat{\mathbf{Y}}_{(t,i)}$ to be compared with the truth labels $\mathbf{Y}_{(t,i)}$. In the backward pass, the objective function is minimized with respect to the weights $\mathbf{W} = [\mathbf{W}^{(1)} \cdots \mathbf{W}^{(L)}]$, where the weight matrix $\mathbf{W}^{(l)}$ connects layers $l-1$ and l. By using the gradient descent algorithm, the weights are iteratively updated following the negative gradient direction,

$$\mathbf{W}^{(l)}(k+1) = \mathbf{W}^{(l)}(k) - \frac{\mu}{b} \frac{\partial \mathbf{J}(\mathbf{W})}{\partial \mathbf{W}^{(l)}}\Big|_{\mathbf{W}(k)}, \tag{2}$$

where μ is the step size and b is the mini-batch size.

III. Data Selection

In this section, we propose a data selection method for classification problems using NN. The selection is performed in each iteration of each epoch. The selection criterion is directly related to the objective function. Among the b examples belonging to a mini-batch set $\mathbf{X}_{(t,i)}$, consider a data pair $(\mathbf{x}, \mathbf{y})$, presented to the network at iteration i. The output layer produces an estimate $\hat{\mathbf{y}}$, which is compared to the target signal $\mathbf{y}$, resulting in an error signal $E(\hat{\mathbf{y}}, \mathbf{y})$ defined according to the objective function. The closer to zero the error is, the less informative or relevant will be the contribution of the example $(\mathbf{x}, \mathbf{y})$ to the parameter update at iteration i of epoch t.

We can express the error signal as the sum of the errors of each output neuron

$$E(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{n=1}^{d^{(L)}} e(\hat{y}_n, y_n), \tag{3}$$

where $\mathbf{y} = [y_1, y_2, \cdots, y_{d^{(L)}}]$ is the target signal and $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{d^{(L)}}]$ is the signal estimated by the neural network.

Figure 2 illustrates the data selection strategy, where we consider only the most relevant subset of examples to the learning process. At each iteration per epoch, a mini-batch set composed by b data samples illustrated in yellow is used in the forward propagation of the data information. We apply the data selection using equation (3) to eliminate the non-informative data represented by the white color. We then proceed with the remainder data in blue in the backpropagation to update the parameter vector.

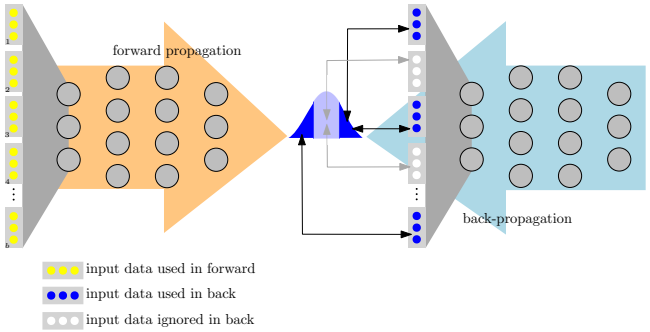

Fig. 2: Data selection neural network diagram.

We consider the cross-entropy,

$$J(\mathbf{W}) = \frac{1}{M} \sum_{m=1}^{M} \sum_{n=1}^{d^{(L)}} [-y_n(m) \ln(\hat{y}_n(m)) - (1 - y_n(m)) \ln(1 - \hat{y}_n(m))], \tag{4}$$

as objective function. Combined with the softmax activation function,

$$\hat{y}_n(m) = h_n^{(L)}(m) = \frac{\exp(a_n^{(L)}(m))}{\sum_{j=1}^{d^L} \exp(a_n^{(L)}(m))}, \tag{5}$$

for $n = 1, \cdots, d^{(L)}$. Each error measure in equation (3) is then defined as

$$e(\hat{y}_n, y_n) = -y_n \ln(\hat{y}_n) - (1 - y_n) \ln(1 - \hat{y}_n), \tag{6}$$

where $\hat{y}_n$ is the estimated output for the n-th class and y_n is the n-th desired value for the output $\mathbf{y}$.

The data selection aims to detect the error signals of equation (3) which are smaller than a given threshold. The related errors are then disregarded in the process of updating the coefficients. This proposal requires selecting a threshold for each iteration of a mini-batch, chosen from a binomial distribution with $n = b$ and $p = P_{\text{up}}$,

$$t_{\text{bin}} \sim \text{Bin}(n, p), \tag{7}$$

where P_{up} is the percentage of selected data.

We define the set of size t_{bin} containing the indexes of data examples that lead to the t_{bin}-th largest error signals E,

$$\mathcal{C} = [k_1, k_2, \cdots, k_{t_{\text{bin}}}]. \tag{8}$$

These t_{bin} data samples are the subset considered in back-propagation. We also define the set of size $b - t_{\text{bin}}$ containing the indexes of data examples that lead to the smallest error signals in E,

$$\mathcal{B} = [k_1, k_2, \cdots, k_{b-t_{\text{bin}}}]. \tag{9}$$

The data examples whose indexes are in set $\mathcal{B}$ are eliminated before the backpropagation process.

Moreover, we consider a fine adjustment factor $0 < \alpha \leq 1$ to randomly select $\alpha \times 100\%$ of the examples in set $\mathcal{B}$ to be definitely eliminated. The remaining $(1-\alpha) \times 100\%$ of the examples in set $\mathcal{B}$ are temporarily included in set $\mathcal{C}$. Then, we randomly select t_{bin} samples to keep in set $\mathcal{C}$. This fine adjustment factor α can be useful when the dataset is too complex, so that eliminating the smallest errors can impair the final results.

The data selection method aims at identifying the non-informative values after the feed-forward propagation process at each iteration, eliminating a portion of the data before the backpropagation process. As we are selecting an estimated portion $\hat{P}_{\text{up}}$ of data in each iteration, the update equation is rewritten as

$$\mathbf{W}^{(l)}(k+1) = \mathbf{W}^{(l)}(k) - \frac{\mu}{b\hat{P}_{\text{up}}} \mathbf{h}_{\mathcal{C}}^{(l-1)} \boldsymbol{\delta}_{\mathcal{C}}^{(l)^T}, \qquad (10)$$

where $\hat{P}_{\text{up}} = \frac{|\mathcal{C}|}{b}$.

As the full dataset is composed of M examples, and each mini-batch has b examples, the number of iterations per epoch is $I = \lfloor M/b \rfloor$. Suppose $M = 60,000$ and we choose $b = 32$, then $I = 1875$. However, if we choose $b = 256$ and $P_{\text{up}} = 0.125$, we are still considering 32 examples per iteration but with a reduced number of iterations $I = 234$ per epoch.

The complete procedure for Data Selection Feed-Forward Multilayer Neural Network is described in Table I. At each epoch, one can compute the objective function for the training dataset J_{train} and also for the validation dataset J_{val}.

TABLE I: Data Selection Feed-Forward Multilayer Neural Network algorithm in classification problem

DS Feed-Forward Multilayer NN algorithm in classification

Initialize
Dataset: $\{(\mathbf{x}(1), \mathbf{y}(1)), (\mathbf{x}(2), \mathbf{y}(2)), \cdots, (\mathbf{x}(M), \mathbf{y}(M))\}$,
Weights: $\mathbf{W} = \{\mathbf{W}^1, \mathbf{W}^2, \cdots, \mathbf{W}^L\}$ (random matrices),
Select: step size $\mu > 0$, number of epochs n_{ep}, mini-batch size b,
number of layers L, number of nodes $(d^1, \cdots, d^L)$,
activation function f, forgetting factor λ_e
Objective function J = Cross-Entropy (CE),
Output function f_L = Softmax;
Define $I = M/b$, prescribe P_{up}
Do for $t = 1 : n_{\text{ep}}$ (for each epoch)
  Do for $i = 1 : I$ (for each iteration)
Randomly select b examples in training dataset
$\mathbf{X}_{(t,i)} = [\bar{\mathbf{x}}(1), \bar{\mathbf{x}}(2), \cdots, \bar{\mathbf{x}}(b)]$, $\mathbf{Y}_{(t,i)} = [\bar{\mathbf{y}}(1), \bar{\mathbf{y}}(2), \cdots, \bar{\mathbf{y}}(b)]$
[Forward Propagation]:
    $\mathbf{H}^{(0)} = [\mathbf{h}^{(0)}(1), \mathbf{h}^{(0)}(2), \cdots, \mathbf{h}^{(0)}(b)] = [ones(1,b); \mathbf{X}_{(t,i)}]$
($ones(1,b)$ is the bias term)
    Do for $l = 1 : L-1$
      $\mathbf{A}^{(l)} = \mathbf{W}^{(l)^T} \mathbf{H}^{(l-1)}$
      $\mathbf{H}^{(l)} = [ones(1,b); f(\mathbf{A}^{(l)})]$
    end
    $\mathbf{A}^{(L)} = (\mathbf{W}^{(L)})^T \mathbf{H}^{(L-1)}$
    $\hat{\mathbf{Y}}_{(t,i)} = [\hat{\mathbf{y}}(1), \hat{\mathbf{y}}(2), \cdots, \hat{\mathbf{y}}(b)] = \mathbf{H}^{(L)} = f_L(\mathbf{A}^{(L)})$
[Data Selection]:
    $e(\hat{y}_n(k), \bar{y}_n(k)) = (1 - \bar{y}_n(k)) \ln(1 - \hat{y}_n(k))$
     for $k = 1, \cdots, b$ and $n = 1, \cdots, d^{(L)}$
    $\mathcal{E}^n = [e(\hat{y}_n(1), \bar{y}_n(1)), \cdots, e(\hat{y}_n(b), \bar{y}_n(b))]$,
     for $n = 1, \cdots, d^{(L)}$
    $\boldsymbol{E} = \left[\sum_{n=1}^{d^{(L)}} e(\hat{y}_n(1), \bar{y}_n(1)), \cdots, \sum_{n=1}^{d^{(L)}} e(\hat{y}_n(b), \bar{y}_n(b))\right]$
    $t_{\text{bin}} \sim \text{Bin}(n, p)$,
    $\mathcal{C} = [k_1, k_2 \cdots, k_{t_{\text{bin}}}]$, $\hat{P}_{\text{up}} = |\mathcal{C}|/b$
     where $\mathcal{C}$ is the index set related to the t_{bin} largest values in the vector $\boldsymbol{E}$
    $\mathbf{Y}_{\mathcal{C}} = [\bar{\mathbf{y}}(k_1), \cdots, \bar{\mathbf{y}}(t_{\text{bin}})]$, $\hat{\mathbf{Y}}_{\mathcal{C}} = [\hat{\mathbf{y}}(k_1), \cdots, \hat{\mathbf{y}}(t_{\text{bin}})]$
[Back-propagation]:
    $\boldsymbol{\Delta}_{\mathcal{C}}^{(L)} = [\boldsymbol{\delta}^{(L)}(k_1), \boldsymbol{\delta}^{(L)}(k_2), \cdots, \boldsymbol{\delta}^{(L)}(t_{\text{bin}})]$
      $= f'_L(\mathbf{A}_{\mathcal{C}}^{(L)}) \otimes (\hat{\mathbf{Y}}_{\mathcal{C}} - \mathbf{Y}_{\mathcal{C}})$
    Do for $l = L-1 : -1 : 1$
      $\boldsymbol{\Delta}_{\mathcal{C}}^{(l)} = f'(\mathbf{A}_{\mathcal{C}}^{(l)}) \otimes [\mathbf{W}^{(l+1)} \boldsymbol{\Delta}_{\mathcal{C}}^{(l+1)}]_1^{d^{(l)}}$
    end
[Updating the weights]:
    Do for $l = 1 : L$
      $\mathbf{W}^{(l)} = \mathbf{W}^{(l)} - \frac{\mu}{b\hat{P}_{\text{up}}} \mathbf{H}_{\mathcal{C}}^{(l-1)} \boldsymbol{\Delta}_{\mathcal{C}}^{(l)^T}$
    end
  end
  $J_{\text{train}}(\mathbf{W}) = \frac{1}{M} \sum_{m=1}^{M} \sum_{n=1}^{d^{(L)}} J_m^n(\mathbf{W})$ (and J_{val} if this set is previously defined)
end

IV. Simulations

In this section, the performance of the proposed data selective neural network is evaluated in three datasets. The algorithms implemented in MATLAB are available on GitHub [22].

The algorithm is tested with cross-entropy error as objective function and softmax as output function, whereas the ReLU function is used as an activation function in the hidden layers. We vary the probability of update so that $P_{\text{up}} \in \{0.125, 0.3, 0.5, 0.7\}$ and compare it to the case where the parameters are always updated, $P_{\text{up}} = 1$. The reduced computational cost is the main benefit as we decrease the probability of update. We verify whether there is an improvement in the algorithm performance, an additional simulation is included in each problem, considering each dataset a correspondent mini-batch size. For example, if the best accuracy is observed by setting $P_{\text{up}} = 0.125$ and $b = 256$, then we compare this result with $P_{\text{up}} = 1$ and $b = 32$.

A. Considered Datasets

1) MNIST Handwritten Digit Dataset: The Modified National Institute of Standards and Technology (MNIST) [23] database is a large data set containing digits handwritten by students and employees of the United States Census Bureau. This dataset consists of $60,000$ and $10,000$ in training and test examples, respectively. The input of this set is a matrix 28×28, where each value represents a pixel of the image. The input signal is normalized in the range 0 to 1. The output dataset presents integer values between 0 and 9.

2) EMNIST Letters Dataset: The EMNIST letter dataset is a set of 26 handwritten characters, and it is provided by the National Institute of Standards and Technology (NIST) [24], [25]. The EMNIST contains $120,000$ and $25,000$ training and test examples, respectively, with samples normalized between 0 and 1. This set's input is a matrix 28×28, where each value represents a pixel of the image.

3) Fashion MNIST Dataset: The Fashion-MNIST dataset consists of a training set of 60,000 examples and a test set of 10,000 examples of images of 10 types of clothing, such as shoes, t-shirts, dresses, and more. It is proposed in [26] as a more challenging classification problem than the MNIST. The

input of this set is also a matrix 28×28, where each value represents a pixel of the image. The input signal is normalized in the range 0 to 1.

B. Experiments without fine adjustment

In the following experiments, we consider $\alpha = 1$, that is, 100% of the elements in set $\mathcal{B}$ are kept for discarding. The considered NN is composed of two hidden layers with 1024 nodes. The step size parameter is $\mu = 0.1$. The mini-batch size is $b = 256$, and the number of epochs is $n_{\text{ep}} = 100$.

Figure 3 depicts the classification error for the MNIST dataset. As shown in Figure 3, the data selective NN achieves a promising performance. As we decrease the amount of update per epoch, an improvement is observed in two-layer NN performance ($b = 256$). Moreover, when compared to simulation $P_{\text{up}} = 1$ and $b = 32$, the data selection with $P_{\text{up}} = 0.125$ achieves the performance with the benefit of requiring a reduced number of iterations.

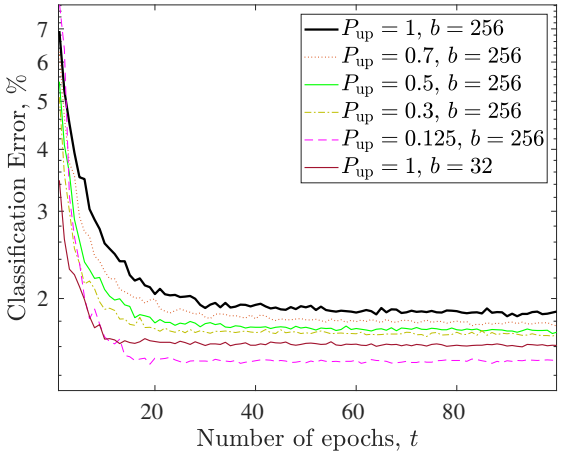

Fig. 3: MNIST Handwritten Digit Simulation.

The classification error for the EMNIST dataset is shown in Figure 4. The proposed method with $P_{\text{up}} = 0.125$ performs quite close to the case where $P_{\text{up}} = 1$ and $b = 32$, but the proposed method requires less iterations per epoch.

Table II includes the data selection method's performance in terms of the averaged MSE over the last ten epochs.

TABLE II: Averaged MSE for different probabilities of update for the EMNIST and MNIST datasets.

	EMNIST dataset	MNIST dataset
$P_{\text{up}} = 1$	9.99±0.069	1.87±0.023
$P_{\text{up}} = 0.7$	9.88±0.056	1.78±0.020
$P_{\text{up}} = 0.5$	9.72±0.059	1.72±0.021
$P_{\text{up}} = 0.3$	9.48±0.042	1.69±0.05
$P_{\text{up}} = 0.1$	9.19±0.034	1.50±0.014
$P_{\text{up}} = 0.005$	96.12±0.047	22.61±4.65
$P_{\text{up}} = 1$, $b = 32$	9.13±0.142	1.60±0.009

We also consider a deep neural network with data selection for the MNIST dataset. In this case, the number of hidden

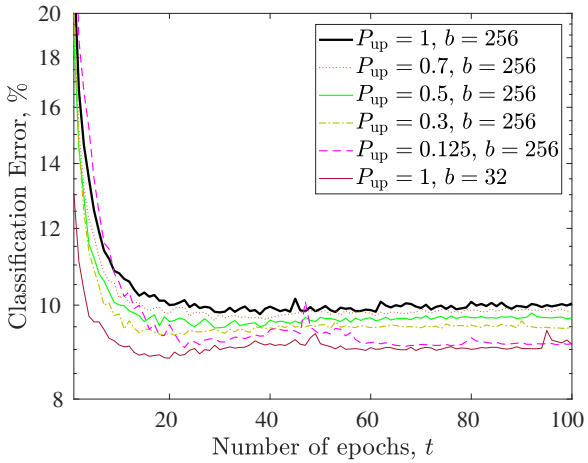

Fig. 4: EMNIST Letters Simulation.

layers is 3, and the mini-batch size is $b = 128$. As can observe in Figure 5, the proposed method performs quite similarly to the case where $P_{\text{up}} = 1$ and $b = 16$, with the advantage of requiring fewer iterations per epoch.

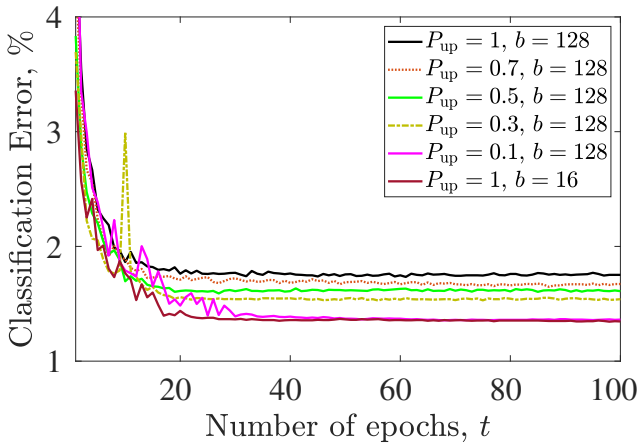

Fig. 5: MNIST Handwritten Digit Simulation.

C. Experiments with fine adjustment

In the following experiments, we consider the case where $\alpha = 0.9$, that is, 90% of the elements in set $\mathcal{B}$ are kept for discarding. The remaining 10% are temporarily included in set $\mathcal{C}$. Then, t_{bin} samples are randomly selected to be in set $\mathcal{C}$. By using an adequate adjustment factor $\alpha < 1$, we can protect the small erros, which are still informative for more complex datasets as the Fashion MNIST. As illustrated in Figure 6, by using $P_{\text{up}} = 0.25$ with $\alpha = 0.9$, we can obtain the same performance of the small mini-batch $b = 32$ case. However, the total number of iterations per epoch is reduced from $I = \lfloor M/32 \rfloor$ to $I = \lfloor M/128 \rfloor$. The fine adjustment factor α does not impair the results as shown in Figure 7 if the dataset is simple as the MNIST dataset.

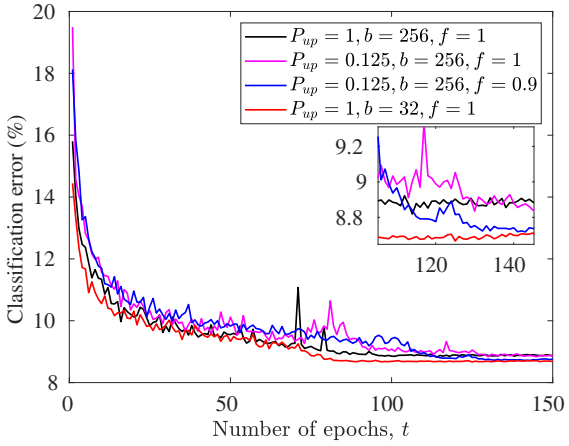

Fig. 6: Fashion MNIST Handwritten Digit Simulation.

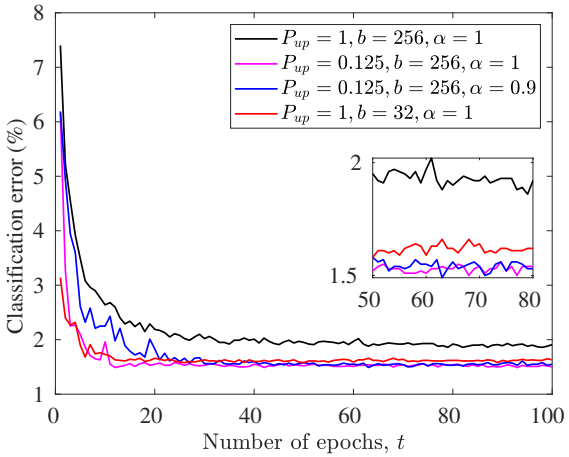

Fig. 7: MNIST Handwritten Digit Simulation.

V. Concluding Remarks

This work introduced a data selection technique to identify non-innovative data examples in a mini-batch training set of neural networks. The decision criteria are based on the error signals at the output layer. In particular, when applied to image classification, the data selection leads to a reduced number of training iterations and a reduction in classification error. The proposed technique can be applied to other architectures. We have also compared our method with pruning techniques such as Dropout. However, Dropout is mainly used for regularization and avoiding overfitting, as our strategy aims at reducing the complexity. We will use the data selection technique in convolutional neural networks (CNN) and more complex datasets in future work.

References

[1] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, Feb 2018.

[2] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, Aug 2018.

[3] D. J. Hemanth and V. V. Estrela, *Deep learning for image processing applications*, vol. 31, IOS Press, Amsterdam, Netherlands, 2017.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, The MIT Press, Cambridge, MA, USA, 2016.

[5] J. Krohn, G. Beyleveld, and A. Bassens, *Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence*, Addison-Wesley Professional, 2019.

[6] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *arXiv preprint arXiv:1804.07612*, Apr 2018.

[7] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, Aug 2014, pp. 661–670.

[8] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*, Academic Press, Inc., Orlando, FL, USA, 1st edition, 2015.

[9] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*, AMLBook, 2012.

[10] H. Chang, E. Learned-Miller, and A. McCallum, "Active bias: Training more accurate neural networks by emphasizing high variance samples," in *Advances in Neural Information Processing Systems*, 2017, pp. 1002–1012.

[11] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 761–769.

[12] I. Loshchilov and F. Hutter, "Online batch selection for faster training of neural networks," *arXiv preprint arXiv:1511.06343*, 2015.

[13] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, and F. Moreno-Noguer, "Fracking deep convolutional image descriptors," *arXiv preprint arXiv:1412.6537*, 2014.

[14] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2794–2802.

[15] D. Gong, C.and Tao, W. Liu, L. Liu, and J. Yang, "Label propagation via teaching-to-learn and learning-to-teach," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 6, pp. 1452–1465, Apr 2016.

[16] W. Liu, B. Dai, A. Humayun, C. Tay, C. Yu, L. B Smith, J. M. Rehg, and L. Song, "Iterative machine teaching," *Proceedings of the 34th International Conference on Machine Learning, ICML*, pp. 2149–2158, May 2017.

[17] P. S. R. Diniz, "On data-selective adaptive filtering," *IEEE Trans. Signal Process.*, vol. 66, no. 16, pp. 4239–4252, Aug. 2018.

[18] M. O. K. Mendonca, J. O. Ferreira, C. G. Tsinos, P. S. R. Diniz, and T. N. Ferreira, "On fast converging data-selective adaptive filtering," *Algorithms*, vol. 12, no. 1, pp. 4, Jan 2019.

[19] C. G. Tsinos and P. S. R. Diniz, "Data-selective lms-newton and lms-quasi-newton algorithms," in *ICASSP 2019 - IEEE International Conference on Acoustics, Speech and Signal Processing*, Brighton, UK, May 2019, pp. 4848–4852.

[20] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 1st edition, 2007.

[21] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[22] J. O. Ferreira, "Code repository for dissertation in github," https://github.com/Jonathasof/Dissertation, Accessed: 2019-12.

[23] Y. LeCun, C. Cortes, and C. J. C. Burges, "Mnist dataset of handwritten digit," http://yann.lecun.com/exdb/mnist/, Accessed: 2019-09.

[24] National Institute of standards and Technology (NIST), "Emnist letters dataset," https://www.nist.gov/node/1298471/emnist-dataset, Accessed: 2019-09.

[25] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.

[26] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.