Memorization in Binarized Neural Networks

Nikhil Mehta [*] [1] [2] **Vinay Uday Prabhu** [*] [1] **John Whaley** [1]

Abstract

In this short paper, we examine memorization in Binarized neural networks. Specifically, we investigate if this explicit restriction over the alphabet of weights (and activations) of the deep network has any effect on its capacity to shatter the training dataset when the labels are randomized. We qualitatively reveal that binarized neural networks do indeed exhibit the ability to memorize the training data even in the presence of explicit weight-decay regularization. We also showcase that two regularization techniques, namely dropout and stochastic binarization can be used to effectively decelerate memorization in the noisy labels scenario.Through this work, we hope to add to the burgeoning body of literature concerning memorization and generalization in deep neural networks.

1. Introduction

The need to run deep learning models on memory, computation and power constrained devices such as mobile phones and embedded systems has recently inspired a large body of work in the area of deep-net compression. One specific methodology that has been shown to be extremely promising in terms of both inference time as well as memory footprint, while still achieving state-of-the-art accuracy involves constraining the weights and activations of the deep-net to be $+1$ or -1, termed as Binarized Neural Networks (BNNs or BinaryNet) (Courbariaux et al., 2016). As shown in (Courbariaux et al., 2016), employing BNNs results in up to $32\times$ network compression and a dramatic reduction in inference time given that the 32-bit floating point multiply accumulations can now be replaced by 1-bit `xnor-popcnt` operations.

This work was followed by efforts such as Ternary Weight Networks (TWNs) (Li et al., 2016) and Two-Bit Networks (Meng et al., 2017) that sought to strike a balance between model compression rate and model capacity by allocating more bits to specify the weights of the neural network. The common theme explored in these works is that allowing for greater precision in specifying the weights endows the deepnets *stronger expressive abilities* than their binary precision counterparts which lie at one extreme of the model capacity versus compression rate trade-off.

With this narrative in tow, we now draw our attention towards a recent work (Zhang et al., 2016), where the authors raised the questioned if the heavily parameterized state-of-the-art deep learning behemoths did require a *rethinking* of generalization analysis tools used in traditional statistical learning theory. In doing so, they exposed this intriguing behavior that these deep-nets were able to achieve near 0% training set error rate even when the labels were completely randomized. They also challenged the conventional wisdom that attributed the small generalization error enjoyed by deep-nets either to properties of the model family, or to the explicit and implicit regularization techniques used during training. This is especially relevant to the work represented here as we would be well served to remind ourselves that (stochastic) weight binarization was also introduced as a novel form of *explicit regularization* in the original papers (See (Courbariaux et al., 2016; 2015)).

All of this leads us to explore the answer to the following question, which forms the premise of this paper:

Are compressed neural networks, even with explicitly constrained binary weights, able to fit random labels and hence shatter the training dataset?

The rest of the paper is organized as follows.

In Section-2, we present the details of the experiment procedure used and the datasets covered in this paper. In section-3, we cover the scenarios where we observed that the BNNs did in fact memorize the dataset (in terms of achieving near zero training error rate). In section-4, we cover the two specific regularization techniques that curtailed the memorization capacity of the BNNs. In section-5, we present the conclusions and elucidate on some of the future directions we are taking in extending this research.

[*]Equal contribution [1]UnifyID Inc,San Francisco, USA [2]Purdue University, West Lafayette, IN , USA. Correspondence to: Nikhil Mehta <nikhil@unify.id >, Vinay Uday Prabhu <vinay@unify.id>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

2. Experimentation details

Before we dive into the experimentation details, we first cover the definitions of the binarization functions we used. In this paper, we experimented with both the deterministic and stochastic binarization functions that were introduced in (Courbariaux et al., 2016).

2.1. Deterministic and Stochastic binarization

The deterministic binarization function is just the classical *sign* function, which is defined as,

$$x^{(db)} = Sign\left(x\right) = \begin{cases} +1 & if \quad x \geq 0 \\ -1 & otherwise, \end{cases} \tag{1}$$

where, $x^{(db)}$ represents the (deterministically) binarized variable (either weight or activation) and x is the input real-valued variable.
The stochastic binarization function is defined as,

$$\begin{aligned} x^{(sb)} &= SoftSign\left(x\right) \\ &= \begin{cases} +1 & with\,probability \quad p = \sigma(x) \\ -1 & with\,probability \quad (1-p) \end{cases} \end{aligned} \tag{2}$$

where,

$$\sigma(x) = clip\left(\frac{x+1}{2}, [0,1]\right) = \max\left(0, \min\left(1, \frac{x+1}{2}\right)\right). \tag{3}$$

2.2. Datasets and procedure

In this paper, we considered the CIFAR-10 (Krizhevsky et al.) and SVHN (Netzer et al.) datasets.
With regards to SVHN, we would like to clarify that unlike (Courbariaux et al., 2016), we omitted the $531k$ additional *somewhat less difficult samples* from the training set (as termed in `http://ufldl.stanford.edu/housenumbers/`), leaving us with a training set size of 73257 and a test-set size of 26032 images.
All experiments were run on Theano (Bergstra et al., 2011) and the Lasagne (Dieleman, 2015) library. The architecture of our deep nets is the same as the ones specified in the original Binary-Net paper(Courbariaux et al., 2016), and we train for 500 epochs.
In order to test the memorization capacity of the BNN, we took a candidate network, and trained it on a copy of the data in which all of the true labels were replaced by random labels. This is what the authors in (Arpit et al., 2017) termed as the `randY` scenario.
We have duly open sourced the code in order to facilitate easy reproducibility of the results at the following url: `https://github.com/hockeybro12/BinaryNetGeneralizationExperiments`

3. Results-1: Instances of successful memorization with randomized labels

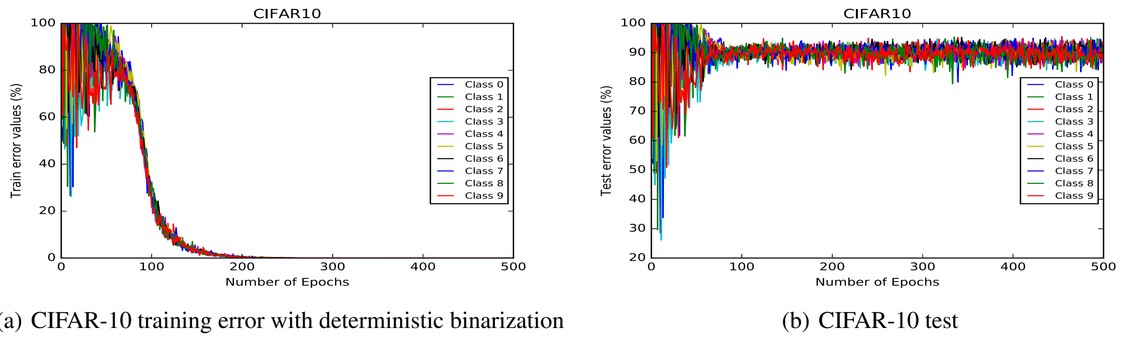
Upon using deterministic binarization, we were able to obtain $\sim$ 0% training error and 90.58% test error on the CIFAR-10 dataset. Further, we obtained 2.54% training error and 89.89% test error for the SVHN dataset. In figure 1, we present the class-wise evolution of the train/test errors with increasing number of epochs. This positive results adds to the discovery made in (Zhang et al., 2016), where the authors showed that different network architectures (InceptionNet, AlexNet, MLP) were able to effectively memorize the CIFAR-10 dataset. Now, we focus our attention to Fig. 2, where we see that the memorization capacity of the BNN did not wither away in spite of explicit l_2-regularization being employed[1]. This result is in keeping with the results in (Zhang et al., 2016), where a similar phenomenon was noticed.
This persistence of the memorization ability in spite of binarization of the weights and biases of the network and explicit weight-decay (l_2-regularization), that too with a mere few hundred epochs of training was rather striking.

Now, we move on to the next section, where we were able to successfully hinder this memorization in BNNs.

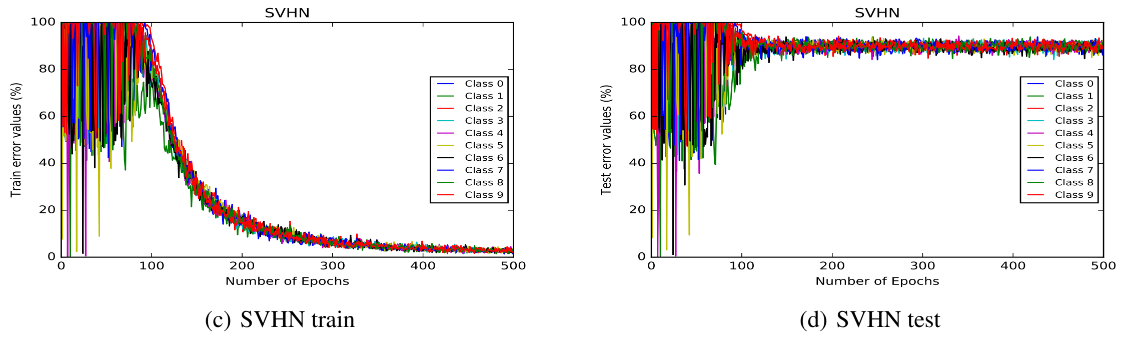
4. Results-2: Instances of incomplete memorization with randomized labels

We were able to qualitatively degrade the memorization ability of BNNs with two strategies: Using dropout as a regularization strategy and using Stochastic binarization (2) in lieu of deterministic binarization (1).
With reference to Fig 3, we see that the use of dropout regularization did result in the marked slowdown in the training error rate decreasing with the number of epochs. Interestingly, we also observed that there was a non-trivial interclass variance in the training error rates observed. After 500 epochs, some of the classes had attained an error rate of $\sim$ 18% whereas the error rate was as high as $\sim$ 46% for some other classes.
Here, we would like to remark that this ineffectiveness of the weight-decay regularization strategy seen in the previous section and the effectiveness of the dropout strategy in terms of hindering memorization in deep-nets was also recently observed in (Arpit et al., 2017) for general Deep-Nets.
Continuing with our efforts to hinder memorization in BNNs, we tried the stochastic binarization strategy (in conjunction with Hinton's `straight-through estimator`) during training and the results are as shown in Fig 4. As seen, the train error rates for all the classes

[1]While only the SVHN dataset is covered here, we observed similar results for the CIFAR-10 dataset as well

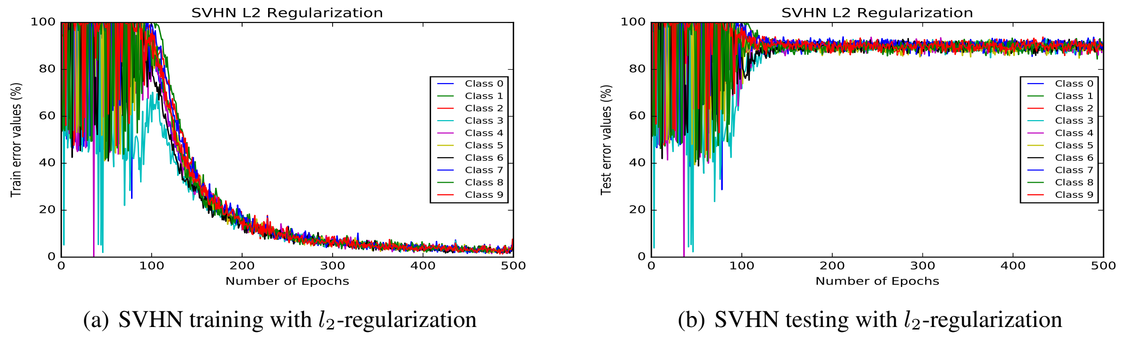


(a) CIFAR-10 training error with deterministic binarization (b) CIFAR-10 test



(c) SVHN train (d) SVHN test

Figure 1. The evolution of class-wise train and test error rates by epochs for the CIFAR-10 and SVHN datasets with deterministic binarization



(a) SVHN training with l_2-regularization (b) SVHN testing with l_2-regularization

Figure 2. Effect of explicit l_2 regularization on the evolution of class-wise train and test error rates by epochs for the SVHN dataset with deterministic binarization

hit an error floor at around $\sim 30\%$. We feel that this observation of stochastic binarization not impeding training with true labeled data (as seen in (Courbariaux et al., 2016; 2015)) but only impeding the noisy labeled case merits further attention.

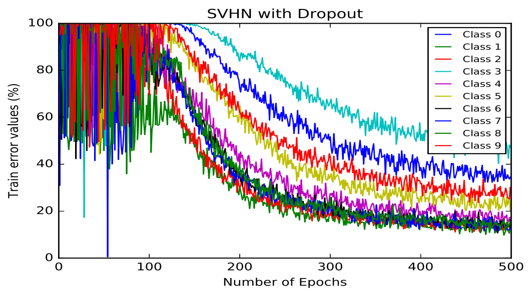
5. Conclusion and future work

In this paper, we embarked on a qualitative analysis of memorization in Binary Neural Networks. We were able to confirm that the strict constraint of binarization on the weights and activations of the network fails extinguish the ability of BNNs to shatter the training set. We also confirmed the continued memorization ability even in the wake of weight-decay regularization. Finally, we found that two regularization techniques, namely dropout and stochastic binarization, did indeed impede the memorization process which calls for further attention.

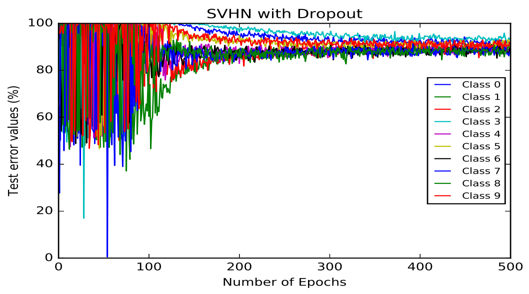
The research represented here is a work-in-progress and we are currently conducting qualitative analysis on the following fronts:

1: Repeating the experiments with randomly generated data (the `randX` case as termed in (Arpit et al., 2017))

2: Single epoch training error analysis (as in (Arpit et al.,

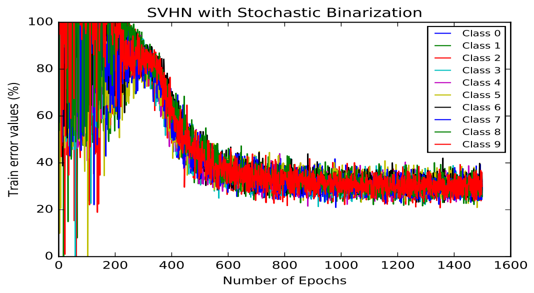


(a) SVHN training with dropout regularization

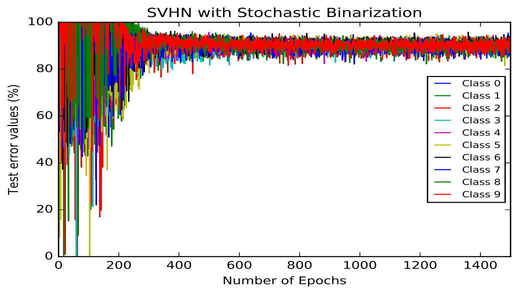


(b) SVHN testing with dropout regularization

Figure 3. Effect of explicit dropout regularization on the evolution of class-wise train and test error rates by epochs for the SVHN dataset with deterministic binarization



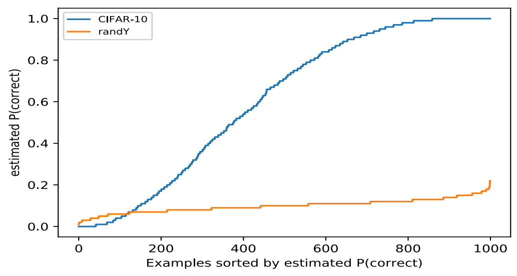
(a) SVHN training with stochastic binarization



(b) SVHN testing with with stochastic binarization

Figure 4. The evolution of class-wise train and test error rates by epochs for the CIFAR-10 and SVHN datasets with stochastic binarization

2017)) to understand the BNNs' ability to first learn the *easy* examples in the datatsets used. In this regard, we do have some preliminary results that we would like to showcase via Fig 5(a). With deterministic binarization, just after a single epoch of training, around 600 of the randomly chosen 1000 image samples were correctly classified at least 50% of the time (with each repetition being a random initialization). This alludes towards the presence of *easy* examples in the training that the BNN first learn to classify correctly, a phenomenon that would not replicate itself for random noisy data.



(a) CIFAR-10 with single epoch training

Figure 5. Single epoch training results

References

Arpit, Devansh, Jastrzebski, Stanisław, Ballas, Nicolas, Krueger, David, Bengio, Emmanuel, Kanwal, Maxinder S, Maharaj, Tegan, Fischer, Asja, Courville, Aaron, Bengio, Yoshua, et al. A closer look at memorization in deep networks. *arXiv preprint arXiv:1706.05394*, 2017.

Bergstra, James, Breuleux, Olivier, Lamblin, Pascal, Pascanu, Razvan, Delalleau, Olivier, Desjardins, Guillaume, Goodfellow, Ian, Bergeron, Arnaud, Bengio, Yoshua, and Kaelbling, Pack. Theano: Deep learning on gpus with python. 2011.

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

Courbariaux, Matthieu, Hubara, Itay, Soudry, Daniel, El-Yaniv, Ran, and Bengio, Yoshua. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Dieleman, Sander. Lasagne: First release., August 2015. URL http://dx.doi.org/10.5281/zenodo.27878.

Krizhevsky, Alex, Nair, Vinod, and Hinton, Geoffrey. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Li, Fengfu, Zhang, Bo, and Liu, Bin. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Meng, Wenjia, Gu, Zonghua, Zhang, Ming, and Wu, Zhaohui. Two-bit networks for deep learning on resource-constrained embedded devices. *arXiv preprint arXiv:1701.00485*, 2017.

Netzer, Yuval, Wang, Tao, Coates, Adam, Bissacco, Alessandro, Wu, Bo, and Ng, Andrew Y. Reading digits in natural images with unsupervised feature learning.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.