Manipulating SGD with Data Ordering Attacks

Ashlesha Chaudhari, Divya Appapogu, Praneeth Chandra Bogineni, Saurav Vara Prasad Chennuri
{ashu33, divsp, pranchan, saurav07}@bu.edu

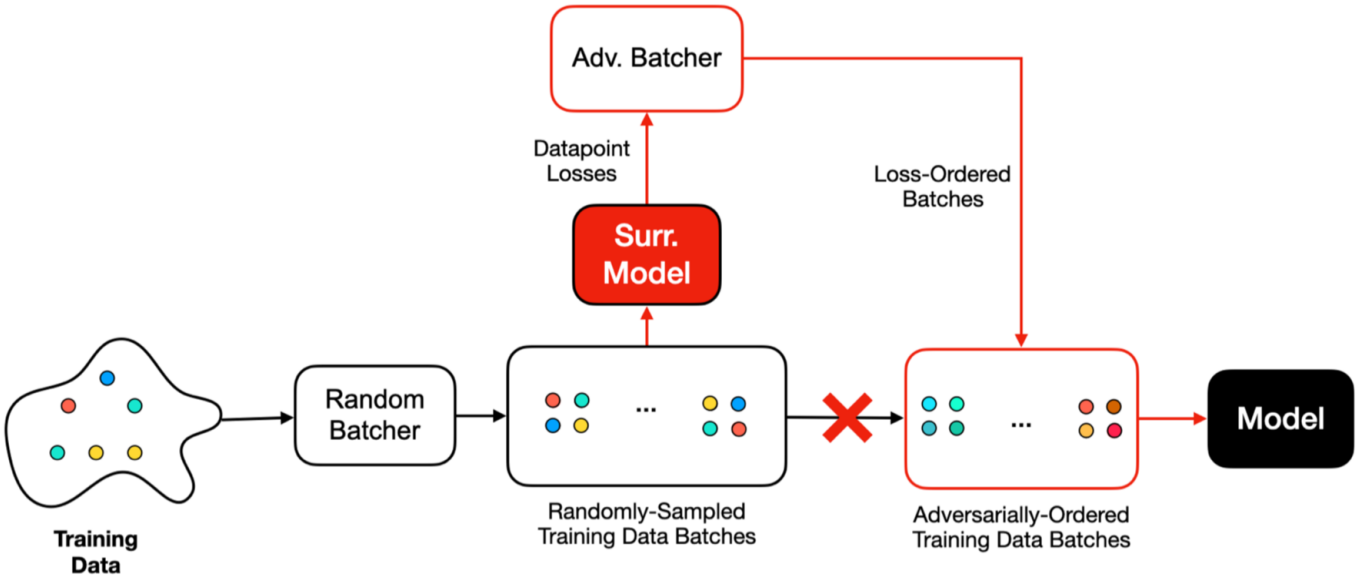

Figure 1. The attacker reorders the benign randomly supplied data based on the surrogate model outputs. Attacker co-trains the surrogate model with the data that is supplied to the source model[1].

1. Task Description

There are a wide array of attacks that deep learning models are vulnerable to. As a result of these vulnerabilities, attackers can poison the models and introduce backdoors to them. Typically these types of attacks require that the attacker has access to the data and manipulate its underlying distribution. The authors in [1] show that by simply reordering the batches and data points during training time, one can influence model behavior. There are two kinds of attacks that can be performed this way without manipulating the data points or the model architecture, *integrity,* and *availability.* For *integrity,* an attacker can reduce the model's accuracy or control its predictions during the presence of certain triggers. For *availability,* the attacker can increase the amount of time the model takes to train or reset its training progress altogether.

Three kinds of approaches can be used to exploit the aforementioned attacks, *Batch Reordering, Reshuffling, and Replacing* (***BRRR***). The attacker can significantly change the performance of the model by

1. Changing the order that the batches are supplied during training.
2. Changing the order in which the individual data points are supplied to the model during training
3. Replacing the data points in batches with other data points from the dataset

Batch Order Poisoning(**BOP**) and Batch Order Backdoor(**BOB**) are the two techniques that are introduced in the paper to perform *integrity* and *availability* attacks respectively. These attacks target the stochastic nature of **SGD** and work by changing the order that the original data set is presented to the model and do not manipulate the data points in any way.The paper [1] doesn't have an official implementation, and our primary goal is to implement the approaches proposed and achieve results similar to the ones presented in the paper(More about this in sections 3 and 4).

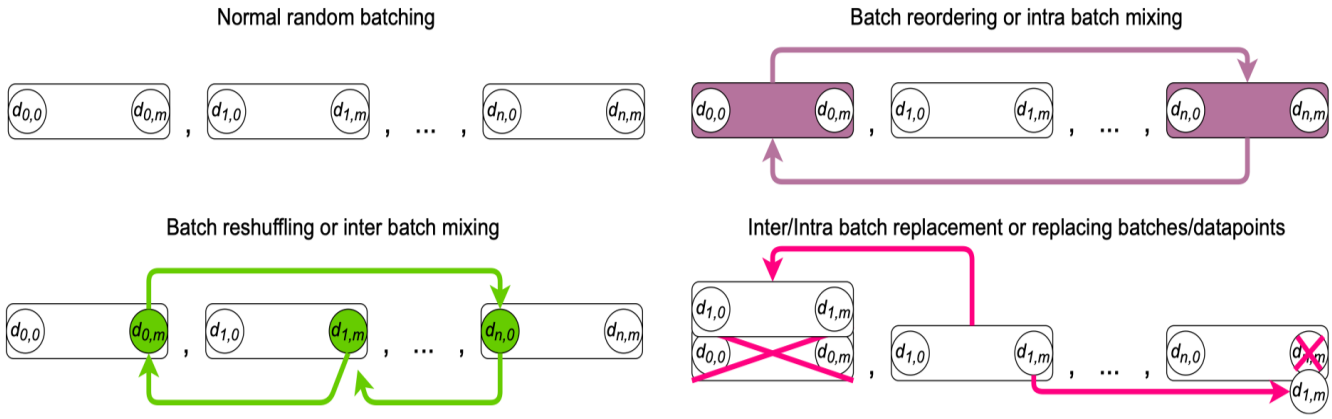

Figure 2: Taxonomy of BRRR attacks. Normal batching assumes randomly distributed data points and batches. Batch reordering assumes the batches appear to the model in a different order, but internal contents stay in the original random order. Batch reshuffling assumes that the individual data points within batches change order, but appear only once and do not repeat across batches. Finally, batch replacement refers to cases where data points or batches can repeat or not appear at all[1].

$$\theta_{N+1} = \theta_1 - \eta\nabla\hat{L}_1(\theta_1) - \eta\nabla\hat{L}_2(\theta_2) - \cdots - \eta\nabla\hat{L}_N(\theta_N)$$

$$= \theta_1 - \eta\sum_{j=1}^{N}\nabla\hat{L}_j(\theta_1) + \eta^2\overbrace{\sum_{j=1}^{N}\sum_{k<j}\nabla\nabla\hat{L}_j(\theta_1)\nabla\hat{L}_k(\theta_1)}^{\text{data order dependent}} + O(N^3\eta^3)\,.$$

Where **N :** Number of batches

L_j : Loss of j^{th} batch

θ_N : Parameters after N^{th} gradient update

η : Learning rate

Equation 1: It can be seen that when updating the parameters over an epoch, the gradients calculated for each batch during the training process are order dependent. This dependence on the data order opens a new layer of attack surface to manipulate the process of SGD and affect the network adversarially.

$$\min_{k=1,\ldots,t}\mathbb{E}\left\|\nabla\hat{L}(\theta_k)\right\|^2 \leq \frac{\hat{L}(\theta_1)-\hat{L}^*}{\eta t} + \frac{M\eta}{2}\mathbb{E}\left\|\nabla\hat{L}_{i_k}(\theta_k)\right\|^2 - \mathbb{E}\left[\hat{L}(\theta_k)^\top\left(\mathbb{E}\left[\nabla\hat{L}_{i_k}(\theta_k)\right]-\nabla\hat{L}(\theta_k)\right)\right].$$

Where **L_{ik} :** Loss at k^{th} gradient step for i^{th} batch

M : Lipschitz Constant

Equation 2: The above equation is the expected value of the gradients of loss function, upper bounded under the assumption of Lipschitz Continuity. This upper bound can be manipulated by the gradients of different batches from the third term on the Right hand side of the equation. A biased bound varies from the unbiased one in last two terms on the RHS. The bound will grow if the second term on the RHS becomes larger, while the third term becomes large and negative.

2. Related Work

Szegedy et al. [2] and Biggio et al. [3] concurrently discovered the existence of adversarial examples, which contain small imperceptible perturbations that affect the *integrity* of the model. These were initially Whitebox models[2, 3, 4, 5], where the attacker has access to the model and the data points. Later BlackBox models were developed[6] where the attacker trained a surrogate model and transferred the adversarial examples to the target model.

All the aforementioned methods manipulate data during test time to influence the model's accuracy. In contrast to them, **BRRR** works during training time without editing the data that is sent to the model.

3. Background

Because of the stochastic nature of SGD, the expectation of the gradients of a single batch in the long term would equal the gradients because of the whole dataset. We also observe that the gradients generated because of different batches of data depends on the order in which those batches have been passed to the model from **Equation 1**. This observation opens an area to skew the stochastic notion of SGD, and manipulate the gradients in an adversarial way.

From **Equation 2**, it can be observed that expectation of gradients can be explicitly manipulated to produce better attack against SGD convergence. The attacker needs to pick the batches such that the difference between the batch gradient and true gradient is opposite to the direction of true gradient.

4. Approach

The taxonomy of the **BRRR** attacks is shown in figure 2. During the first epoch of training, we cache the dataset and the corresponding losses. The attack starts from epoch 2 by following any of the policies mentioned in figure 3. The authors tested the methods on CIFAR10 and CIFAR100[7] using Resnet18 and Resnet50[9] as source models and Mobilenet[10] and lenet[11] as surrogate models, and reported the results. We have implemented these methods from scratch and replicated the results mentioned in the paper.

Network Backdoor

Backdooring is the process of misleading the network to classify the data as wrong classes by adding trigger to a set of classes in the training data. The trigger confuses the model to make wrong predictions. We explore one type of trigger (9 white lines at the top of each image) added to data of 5 classes of the dataset as shown in **Figure 3**. We then replace the data based on the gradients of natural datapoint that are closest to the gradients of the triggered data from the surrogate model. We observe that the classification accuracy decreases significantly when trained in this way concluding that the network is clearly confused on what each of the class is for the datapoints.

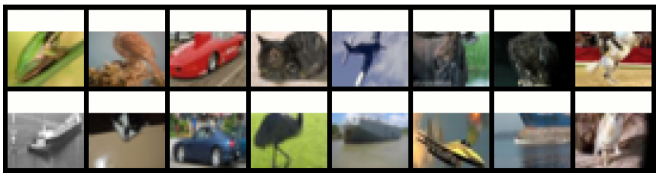

Figure 3: White line trigger examples. 9 lines from the top of the image are colored white as a trigger on the dataset.

Datasets and Metric

As additional experiments we also tested the method on the datasets SVHN[12] and used test Vision Transformers[14] as source model. Oscillation attacks worked well on the transformer model and LeNet Surrogate Model as can be seen in Table 5. The apparent robustness of vision transformers to these attacks maybe misleading. We think this may be due to LeNet being a poor surrogate for transformers. We need to do further testing to verify this assumption. Models Classification accuracy can be used to estimate the effect of the methods on training procedure. We first train a reference model under a setting using the random assumption and later compare the reference model's classification accuracy to the model trained while under several types of **BRRR** attacks.

6. Results

All the extensive results for different models and datasets that we experimented on are mentioned in tables 1-3. Here we can see that our attacks were successful in different surrogate and source models. There was a significant dip in accuracies for both data

reordering and reshuffling attacks for all the experimented source and surrogate models which proves the efficacy of the data ordering attacks.

7. Conclusion

We were able to replicate the results mentioned in the paper, and in turn we were also able to extend it to vision transformers and observe a similar decrease in accuracy.By changing order of the batches and order of the data points within them one can easily manipulate a training model.Careful reordering of model's training data allows it to be poisoned or backdoored without changing the training data at all.The BRRR attacks have multiple attack surfaces and they can be deployed at the the OS level, which makes them harder to detect.

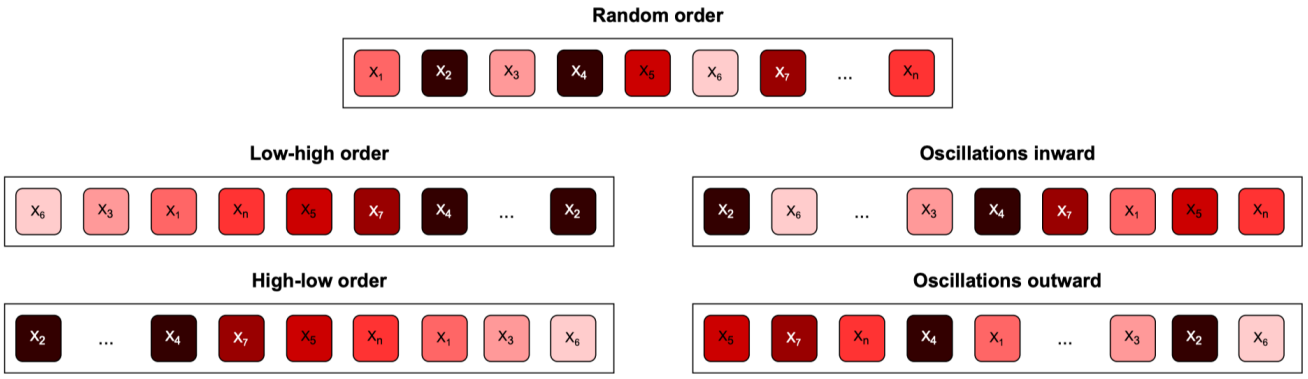

Figure 3: There are four different reorder and reshuffle policies based on the corresponding data point and batch losses. The loss values are color-coded from bright to dark colors, to represent loss values from low to high. Low-high policy orders a sequence by the loss magnitude. High-low policy orders a sequence by the negative loss magnitude. Oscillation inwards orders elements of the sequence from the beginning and the end of the sequence one by one as if it was oscillating between sides of the sequence and moving towards the middle. Finally, Oscillations outward orders the sequence by starting at the middle of an ordered sequence picking elements to both sides of the current location.[1]

Appendix A. Code repository

Github link:
https://github.com/praneethchandraa/Data-Ordering-Adversarial-Attacks

Appendix B.

All the detailed roles are mentioned in Table 4 at the end of this document

References

1) Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A. Erdogdu, Ross Anderson, arXiv:2104.09667v2 **[cs.LG]**
2) C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.
3) B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndic, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks ´ against machine learning at test time. In Joint European conference on machine learning and knowledge discovery in databases, pages 387–402. Springer, 2013.
4) I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples, 2015.
5) A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks, 2019.
6) N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In Proceedings of the 2017 ACM on
7) Asia conference on computer and communications security, pages 506–519, 2017.
8) A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.
9) K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

10) A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.
11) Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.
12) Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.
13) Thomas Mosgaard Giselsson, Rasmus Nyholm Jørgensen, Peter Kryger Jensen, Mads Dyrmann, Henrik Skov Midtiby. arXiv:1711.05458v1
14) Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby. arXiv:2010.11929v2
15) https://github.com/jmjeon94/MobileNet-Pytorch/blob/master/MobileNetV1.py

Table 1: Baseline Accuracies of all the models tested

Dataset	Source	Surrogate	Attack Policy	Attack Type	Test Acc	Delta Acc
CIFAR 10	ResNet-18	-	-	-	90.3%	0
CIFAR 100	ResNet-50	-	-	-	74%	0
SVHN	ViT-b-16	-	-	-	82.76%	0

Table 1. Test Accuracy results of all Attack types of Batch Reordering and Reshuffling Attacks on CIFAR10 dataset using ResNet18 as source model and LeNet5 as surrogate model

Dataset	Source	Surrogate	Attack Policy	Attack Type	Test Acc	Delta Acc
CIFAR 10	ResNet18	LeNet5	Batch Reordering	Low - High	**74%**	**-16.3%**
CIFAR 10	ResNet18	LeNet5	Batch Reordering	High - Low	**78%**	**-12.3%**
CIFAR 10	ResNet18	LeNet5	Batch Reordering	Oscillation Inward	**77%**	**-13.3%**
CIFAR 10	ResNet18	LeNet5	Batch Reordering	Oscillation outward	**82%**	**-8.3%**
CIFAR 10	ResNet18	LeNet5	Batch Reshuffling	Low - High	**37.2%**	**-62.1%**
CIFAR 10	ResNet18	LeNet5	Batch Reshuffling	High - Low	**31.5%**	**-58.8%**
CIFAR 10	ResNet18	LeNet5	Batch Reshuffling	Oscillation Inward	**28.2%**	**-62.1%**

CIFAR10	ResNet18	LeNet5	Batch Reshuffling	Oscillation outward	**30.3%**	**-60.0%**

Table 2. Test Accuracy results of all Attack types of Batch Reordering and Reshuffling Attacks on CIFAR100 dataset using ResNet50 as source model and MobileNet as surrogate model

Dataset	**Source**	**Surrogate**	**Attack Policy**	**Attack Type**	**Test Acc**	**Delta Acc**
CIFAR 100	ResNet50	MobileNet	Batch Reordering	Low - High	**71%**	**-3%**
CIFAR 100	ResNet50	MobileNet	Batch Reordering	High - Low	**65%**	**-9%**
CIFAR 100	ResNet50	MobileNet	Batch Reordering	Oscillation Inward	**70%**	**-4%**
CIFAR 100	ResNet50	MobileNet	Batch Reordering	Oscillation outward	**70%**	**-4%**
CIFAR 100	ResNet50	MobileNet	Batch Reshuffling	Low - High	**5%**	**-69%**
CIFAR 100	ResNet50	MobileNet	Batch Reshuffling	High - Low	**8%**	**-66%**
CIFAR 100	ResNet50	MobileNet	Batch Reshuffling	Oscillation Inward	**32%**	**-43%**
CIFAR100	ResNet50	MobileNet	Batch Reshuffling	Oscillation outward	**9%**	**-65%**

Table 3. Test Accuracy results of all Attack types of Batch Reordering and Reshuffling Attacks on SVHN dataset using Vit-b-16 (Vision in transformers base) as source model and LeNet as surrogate model

Dataset	**Source**	**Surrogate**	**Attack Policy**	**Attack Type**	**Test Acc**	**Delta Acc**
SVHN	ViT-b-16	LeNet5	Batch Reordering	Low - High	**76.1%**	**-6.67%**
SVHN	ViT-b-16	LeNet5	Batch Reordering	High - Low	**78.45%**	**-4.31%**
SVHN	ViT-b-16	LeNet5	Batch Reordering	Oscillation Inward	**54.8%**	**-27.96%**
SVHN	ViT-b-16	LeNet5	Batch Reordering	Oscillation outward	**54.6%**	**-28.16**
SVHN	ViT-b-16	LeNet5	Batch Reshuffling	Low - High	**75.3%**	**-7.16**

SVHN	ViT-b-16	LeNet5	Batch Reshuffling	High - Low	**76.8%**	**-5.96**
SVHN	ViT-b-16	LeNet5	Batch Reshuffling	Oscillation Inward	**51.6%**	**-31.16**
SVHN	ViT-b-16	LeNet5	Batch Reshuffling	Oscillation outward	**52.8%**	**-29.96**

Network Backdoor Attacks

Dataset	**Model**	**Surrogate**	**# Classes triggered**	**Classification Accuracy**	**Delta Accuracy**
CIFAR-10	ResNet-18	LeNet-5	5	17.8%	-72.5

Table 4. Team member contributions

Name	**Task**	**File names**	**No. Lines of Code (Approx)**
Ashlesha	Implemented BRRR attacks on CIFAR-100, Ran experiments with varying oscillation length and Mixed attacks	CIFAR-100_BatchReodering.ipynb,CIFAR-100_BatchReshuffling.ipynb,DataOrderingMixedAttacks.ipynb	160-200
Divya	Implemented Oscillation attack types for batch ordering and shuffling, Implemented and ran experiments on Transformer models	DataOrderingAttacks.ipynb ,DataOrderingAttacks-Transformers.ipynb	150 - 200
Praneeth	Batch Order Poisoning Skeleton Code, Data Ordering Attacks	DataOrderingAttacks.ipynb	287
Saurav	Network Backdooring, Bug Fixes on Batch order poisoning attacks (reordering, reshuffling)	DataOrderingAttacks.ipynb	140 - 190