UNDER REVIEW

RESCIENCE C

Reproduction / Computer Science

[Re] Network Deconvolution

Rochana R. Obadage[1], Kumushini Thennakoon[1], Sarah M. Rajtmajer[2], Jian Wu[1]
[1]Old Dominion University, Norfolk, VA, USA – [2]IST, Pennsylvania State University, University Park, PA, USA

Edited by
(Editor)

Reviewed by
(Reviewer 1)
(Reviewer 2)

Received
–

Published
–

DOI
–

Reproducibility Summary

Scope of Reproducibility – Our work evaluates the claim that network deconvolution [1] improves deep learning model performance compared with batch normalization. We reran the paper's original experiments using the same software, datasets, and evaluation metrics to determine whether the reported results could be reproduced. We examine the consistency of reported values, document discrepancies, and discuss reasons that some results were not able to be consistently reproduced.

Methodology – We used the original study's GitHub codebase [2] with thorough documentation to repeat experiments to generate results in Tables 1 and 2. For each CNN architecture, we conducted three attempts per reported value, resulting in 360 reproduced values as compared with 120 results reported in the original paper. The ImageNet dataset was not available from the URL in the original paper and was obtained from a different site, requiring us to restructure it to match the study's format. Hyperparameters were consistent with the original study and the experiments were conducted on the on-site GPU cluster we have access to. The reproducibile codes are available at our GitHub repository https://github.com/lamps-lab/rep-network-deconvolution.

Results – Our reproduced results confirm that network deconvolution improves model performance compared with batch normalization. Consistent higher accuracy was observed with network deconvolution for CIFAR-10 and CIFAR-100 datasets, particularly with 20 and 100 epochs. Single-epoch discrepancies were minimal. Results for VGG-11, ResNet-18, and DenseNet-121 on the ImageNet dataset also support the original claim, with improved top-1 and top-5 accuracy values. However, PNASNet-18 showed weaker performance overall.

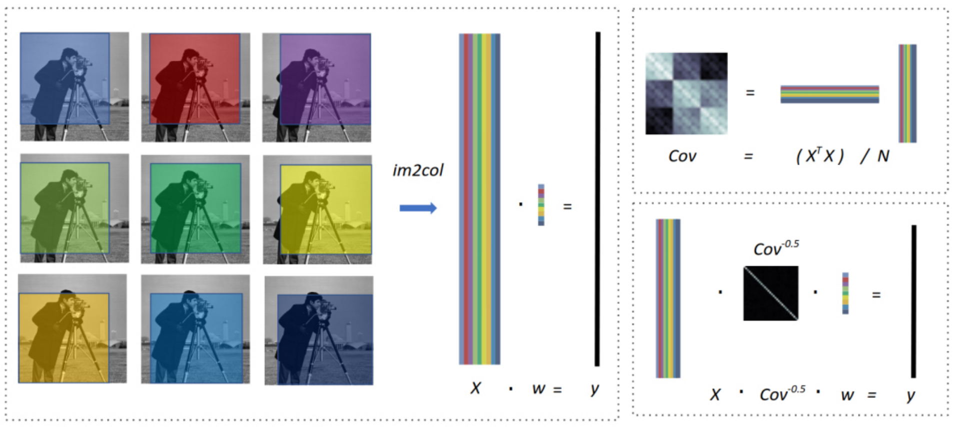
What was easy – The use of benchmark datasets, e.g., CIFAR-10 and CIFAR-100 with PyTorch, simplified reproducing the experimental setup and comparing our results with the original study.

What was difficult – We faced PyTorch compatibility issues, module import errors, and significant computational demands. Handling the large ImageNet dataset required extensive computational resources. We needed to adopt a GPU with 80GB memory to use the data.

Communication with original authors – Although we did not receive a response from the original authors, their well-documented code-base and clear methodology description provided sufficient information to reproduce the results.

1 Introduction

Batch normalization (BN) is a widely adopted technique in modern deep learning architectures, which has been shown to accelerate training and improve prediction performance [3]. Recent studies have explored alternatives to BN to further improve model performance. One such approach is network deconvolution, proposed in 2020 by Ye et al. [1] (hereafter, **original study**). As shown in the Figure 1, the authors of the original study claim that network deconvolution enhances the training of convolutional neural networks (CNNs) by removing pixel-wise and channel-wise correlations in the input data. In CNNs, the blur effect occurs when an image is convolved with a Gaussian kernel or a similar smoothing filter. This process involves overlaying the kernel onto the image and performing a mathematical operation at each pixel. The blur effect which occurs due to the correlation of pixels reduces the clarity of object boundaries and features, making it difficult to accurately identify and localize objects.



Algorithm 1 Computing the Deconvolution Matrix

```
1: Input: C channels of input features $[x_1, x_2, ..., x_C]$
2: for $i \in \{1, ..., C\}$ do
3:     $X_i = im2col(x_i)$
4: end for
5: $X = [X_1, ..., X_C]$ %Horizontally Concatenate
6: $X = Reshape(X)$ %Divide columns into groups
7: $Cov = \frac{1}{N} X^t X$
8: $D \approx (Cov + \epsilon \cdot I)^{-\frac{1}{2}}$
```

Figure 1. Network deconvolution. Adopted from Figure 3 and Algorithm 1 in the original study [1].

Per the original study, the architecture of network deconvolution involves replacing BN layers with deconvolution layers before a convolutional or fully-connected layer. This process entails calculating the covariance matrix of the input data, approximating its inverse square root, and then applying this transformation to decorrelate the data before feeding it into each subsequent layer (Figure 1).

In the results reported in Table 1 of the original study [1], the authors evaluated their approach on 10 CNN architectures, including VGG-16, ResNet, PreAct-18, DenseNet-121, ResNeXt-29, MobileNet v2, DPN-92, PNASNetA, SENet-18, and EfficientNet. They used two benchmark datasets: CIFAR-10 [4] and CIFAR-100 [5]. Their results demonstrated improved performance compared with the baseline models using BN. In Table 2 of the original study, the authors compared VGG-11, ResNet-18, and DenseNet-121 architectures over BN and network deconvolution using the ImageNet [6] dataset to further validate the claims they obtained using CIFAR-10 and CIFAR-100.

Motivated by use cases including preserving spatial information [7], reconstructing high-resolution features [8], improving localization accuracy [9], and image segmentation [10], our study attempts to reproduce the results reported in the original paper with the most recent versions of software libraries. Therefore, our work used the same datasets and methods but different versions of the software packages adopted by the original paper. We define our work as a soft-reproducible study, to be distinct from a strict reproducible study, which adopts exactly the same datasets, and methods (including its software implementations). Soft-reproducibility studies circumvent many dependency compatibility obstacles are thus easier to implement. The results are still meaningful to understand the reliability of the original work.

2 Citations of the Original Study from Scholarly Papers

To analyze the scholarly reception of the original study, we reviewed 59 citation contexts from 31 citing papers extracted using the Semantic Scholar Graph API [11]. The sentiment of each citation context was manually categorized as negative, positive, or neutral where **negative** was taken to indicate irreproducibility (e.g., unavailability of the cited paper's data/code or unsuccessful attempts in reproducing), **positive** context suggests reproducibility (e.g., re-usage of the cited paper's data/code or key concepts), and **neutral** means the context simply mentions (cites) the paper without offering information about its reproducibility. The classification process followed the same methodology as our previous study [12]. Of the 59 citation contexts, 13 were determined to be positive. The remaining 46 contexts were neutral. Importantly, there were no negative citation contexts. Two examples are shown in Table 1. This analysis supports the generally positive reception of the original study in the scholarly community.

Table 1. Examples of citation contexts categorized by class.

Class	Citation Context Example
Positive	"Drawing inspiration from this study, the structure of animal visual systems mentioned above, and recent related work **(Ye et al. 2020)**, we derive two invariance properties that enhance the training of deep neural networks."
Neutral	"Some complementary information can be found in recent works **(Ye et al. 2020)**"

3 Scope of reproducibility

Our work seeks to evaluate and verify the primary claim of the original paper, namely that when network deconvolution is used instead of batch normalization in a deep learning architecture, the performance is generally improved [1]. Verification of the claim required repeating the original experiments and analyses using the same methodologies, datasets, and computational tools as described in the original study. The motivation for this reproducibility study extends beyond simply reproducing the original results. Our work also examines the consistency of the reported values in Table 1 of the original study by re-running the experiment multiple times under the same configurations. This approach is necessary because the performance of deep learning models, despite their deterministic nature, may vary due to random factors such as stochastic gradient descent and data shuffling. By performing repeated experiments, we can assess whether the reported improvements are consistently achievable or if they might be influenced by favorable random initializations. We document discrepancies and challenges encountered during the process, providing a comprehensive account of our findings.

4 Methodology

We use the code provided by the original study [2]. The code base requires Python GPU-based processing capabilities and the PyTorch framework. We discuss these requirements in detail in Computational Requirements, Subsection 4.5. In Table 1 of the original study, authors reported 12 values, specifically 6 for BN and 6 for network deconvolution, for each CNN architecture under three different epoch settings. For a single CNN architecture, we conducted three attempts for each value reported in the original study (using the same hyperparameter setup), resulting in 36 reproduced values, to examine the model's consistency. Therefore we obtained 360 values for 10 architectures (Table 2). Additionally, we reproduced the 14 values from the original study Table 2.

Table 2. Our approach in reproducing the results from the original study [1]: Table 1 (BN: Batch Normalization, ND: Network Deconvolution, rep.: reproduced/reproducing)

Dataset	Technique	# tested CNNs	# epoch settings	# rep. attempts per reported value	Total rep. values
CIFAR-10	BN	10	3	3	90
	ND	10	3	3	90
CIFAR-100	BN	10	3	3	90
	ND	10	3	3	90

4.1 Datasets

The authors of the original study have tested the proposed model and compared it against the state-of-the-art against three benchmark datasets:

1. CIFAR-10: A dataset consisting of 60,000 32x32 color images across 10 classes. It is split into 50,000 training images and 10,000 test images (Size: 163 MB) [4].
2. CIFAR-100: A dataset with 60,000 32x32 color images spanning 100 classes. Similar to CIFAR-10, this dataset is divided into 50,000 training images and 10,000 test images (500 training and 100 testing images for each class) (Size: 161 MB) [5].
3. ImageNet ILSVRC: The dataset released for ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[1] 2012 with 1.2 million training images, 50,000 validation images, and 100,000 test images. This dataset contains 1000 object classes. (Size: 167.62 GB) [6].

The results of CIFAR-10 and CIFAR-100 are shown in Table 1 in the original study. The results of ImageNet are shown in Table 2 in the original study.

4.2 Hyperparameters

We used the same hyperparameter settings as those employed in the original study (see our study Table 3) :

[1]https://www.image-net.org/challenges/LSVRC/2012/index.php

Table 3. Hyperparameter settings used to reproduce results from the original study.

Original Study	Learning rate	Optimizer	Batch size	Stride	Epochs
Table 1	0.1	SGD	128	3	[1,20,100]
Table 2	[0.01, 0.1, 0.1]	SGD	256	3	90

4.3 Preparing the ImageNet dataset

The CIFAR-10 and CIFAR-100 datasets required for reproducing Table 1 of the original study are automatically downloaded through the codebase using torchvision [13], eliminating the need for manual dataset preparation. The ImageNet ILSVRG [6] subset was required to reproduce the results reported in Table 2 of the original study. Downloading the ImageNet ILSVRC dataset is no longer possible from the original source on the ImageNet website[2]. The URL redirected to a new download link hosted on Kaggle.[3]. After inspection, we found that the folder structure of the Kaggle version is different from the original data used by the original paper. Therefore, we followed the below steps to modify the folder structure to make it compatible with the input format required by the code (the detail is described at our GitHub repository https://github.com/lamps-lab/rep-network-deconvolution).

1. Clone the GitHub repository https://github.com/lamps-lab/rep-network-deconvolution to a workspace on a local server (hereafter ***workspace***). This ***workspace*** is an extended fork of the GitHub repository[4] provided by the authors of the original study.
2. Download the "*ILSVRC.zip*" file from Kaggle Imagenet-Object-Localization-Challenge https://www.kaggle.com/c/imagenet-object-localization-challenge/data.
3. Inside the '*imagenet*' folder of the ***workspace***, unzip the "*ILSVRC.zip*".
4. Move the "*valprep.sh*" inside "*imagenet/ILSVRC/Data/CLS-LOC/val*" folder which is initially available in the root directory of the ***workspace*** and execute "*valprep.sh*" (on successful execution, you should see the created 1000 sub-directories inside the same 'val' folder representing the 1000 object categories as described in our GitHub repository[5]).

4.4 Experimental setup

After preparing the datasets, we follow the steps below to set up the experimental environment.

1. Create a python virtual environment following the instructions from https://docs.python.org/3/library/venv.html and activate it.
2. Install the python dependencies using the "*requirements.txt*" file (available in the root directory of the ***workspace***).
3. Perform initial testing using the "*deconv_rep.ipynb*" notebook, which is also available in the root directory of the ***workspace***, with 1 epoch to verify the code's functionality (GPU processing capabilities required).
4. Conduct more extensive experiments using the bash scripts under the ***workspace*** on a local GPU cluster.

[2]https://www.image-net.org/
[3]https://www.kaggle.com/c/imagenet-object-localization-challenge/data
[4]https://github.com/yechengxi/deconvolution
[5]https://github.com/lamps-lab/rep-network-deconvolution

4.5 Computational requirements

Table 1 in the original paper was reproduced on an HPC cluster equipped with an NVIDIA V100 GPU with 16GB memory. Because ImageNet is significantly larger than CIFAR-10 and CIFAR-100, we moved the experiments to a more powerful server with NVIDIA A100, providing 80GB GPU. The original study does not specify the software versions used, and several software packages have been upgraded since the paper was published. Below is the list of resources we utilized in our study.

Hardware:

- Local GPU Cluster:
  - ◇ 16 GB GPU – NVIDIA V100 (For original study Table 1)
  - ◇ 80 GB GPU – NVIDIA A100 (For original study Table 2)
- Disk : 2 TB (Shared for both Table 1 and 2 of the original study)
- CPU : Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz (multi-core)
- RAM: 64 GB (Shared for both Table 1 and 2 of the original study)

Software:

- Operating System: Rocky Linux 9.3 (Blue Onyx)
- Python 3.10.2
- SciPy 1.10.1
- NumPy 1.23.5
- TensorBoard 2.12.0
- Matplotlib 3.7.1
- PyTorch 1.13
- Torchvision 0.14.1
- TensorFlow 2.12.0
- Slurm (for job scheduling on the internal GPU cluster)

4.6 Recommendations for Minimal Computational Requirements

For researchers attempting to reproduce the original study, we recommend the following minimum hardware requirements:

- GPU: 16 GB GPU memory for Table 1, 40 GB GPU memory for Table 2
- CPU: Multi-core processor, such as Intel Xeon or AMD Ryzen
- RAM: 16 GB
- Disk: At least 240 GB for data and model storage

5 Results

5.1 Reproduced Results of Table 1 in the Original paper

The authors of the original paper reported the model performance using "Accuracy" as the metric. We report the reproduced accuracy for each architecture using each of the three datasets, namely, CIFAR10, CIFAR100, and ImageNet. To investigate the consistency, we obtained 3 results per one value reported in the original study Table 1. For example, we ran the experiment under the same hyper-parameter settings for batch normalization for 1 epoch for a particular architecture three times and recorded the results per each attempt. Then we averaged the results of three attempts and compared them with the original study's reported values. Table 4 and Figure 2 compare the averaged reproduced results for CIFAR-10 dataset with the original study's values. A similar comparison was made for CIFAR-100 in Table 5 and Figure 3.

Table 4. Results from the original study Table 1 and the reproduced averaged values from our study for CIFAR-10 dataset with 1, 20, and 100 epochs. Architectures: (1) VGG-16, (2) ResNet-18, (3) Preact-18, (4) DenseNet-121, (5) ResNext-29, (6) MobileNet v2, (7) DPN-92, (8) PNASNet-18, (9) SENet-18, (10) EfficientNet (All values are presented as percentages). Color codes: Red - If the reproduced result is lower than the original value by more than 10%, Green - If the reproduced value is greater than the original value, Black - If the reproduced value is less than the original value, but the difference between two values is no more than 10%.

Arch.	CIFAR-10											
	BN 1		ND 1		BN 20		ND 20		BN 100		ND 100	
	Org. Value	Rep. Avg	Org. Value	Rep. Avg	Org. Value	Rep. Avg	Org. Value	Rep. Avg	Org. Value	Rep. Avg	Org. Value	Rep. Avg
(1)	14.12	14.38	74.18	74.20	90.07	90.36	93.25	92.83	93.58	99.86	94.56	99.88
(2)	56.25	46.56	72.89	58.69	92.64	92.35	94.07	94.20	94.87	96.36	95.40	96.42
(3)	55.15	16.42	72.70	72.33	91.93	90.32	94.10	94.26	94.37	99.92	95.44	99.96
(4)	59.56	55.78	76.63	76.31	93.25	93.11	94.89	94.87	94.71	99.95	95.88	99.97
(5)	52.14	25.70	69.22	69.35	93.12	92.34	94.05	93.57	95.15	99.99	95.80	99.97
(6)	54.29	53.26	65.40	59.85	89.86	90.70	92.52	92.11	90.51	96.34	94.35	99.55
(7)	34.00	23.27	53.02	50.61	92.87	91.79	93.74	93.55	95.14	99.96	95.82	99.96
(8)	21.81	10.55	64.19	65.19	75.85	58.84	81.97	81.61	81.22	84.12	84.45	88.98
(9)	57.63	58.62	67.21	67.80	92.37	92.74	94.11	94.25	94.57	99.95	95.38	99.95
(10)	35.40	40.10	55.67	59.91	84.21	84.68	86.78	87.77	86.07	89.16	88.42	90.10

Table 5. Results from the original study Table 1 and the reproduced averaged values from our study for CIFAR-100 dataset with 1, 20, and 100 epochs. Architectures: (1) VGG-16, (2) ResNet-18, (3) Preact-18, (4) DenseNet-121, (5) ResNext-29, (6) MobileNet v2, (7) DPN-92, (8) PNASNet-18, (9) SENet-18, (10) EfficientNet (all values are presented as percentages). Color codes: Same as Table 4

Arch.	CIFAR-100											
	BN 1		ND 1		BN 20		ND 20		BN 100		ND 100	
	Org. Value	Rep. Avg	Org. Value	Rep. Avg	Org. Value	Rep. Avg	Org. Value	Rep. Avg	Org. Value	Rep. Avg	Org. Value	Rep. Avg
(1)	2.01	1.47	37.94	24.02	63.22	69.04	71.97	88.43	72.75	99.23	75.32	99.30
(2)	16.10	9.15	35.73	15.41	72.67	68.49	76.55	74.32	77.70	97.42	78.63	94.31
(3)	15.17	8.39	36.52	22.60	70.79	87.73	76.04	94.71	76.14	99.90	79.14	99.75
(4)	17.90	11.12	42.91	28.06	74.79	92.95	77.63	98.12	77.99	99.89	80.69	99.93
(5)	17.98	3.89	30.93	22.68	74.26	84.44	77.35	98.32	78.60	99.93	80.34	99.96
(6)	15.88	10.11	29.01	14.32	66.31	75.05	72.33	82.92	67.52	78.53	74.90	98.52
(7)	8.84	1.58	21.89	12.82	74.87	76.59	76.12	96.77	78.87	99.93	80.38	99.97
(8)	10.49	1.45	36.52	20.47	44.60	35.67	55.65	58.27	54.52	42.50	59.44	65.66
(9)	16.60	12.58	32.22	20.50	71.10	87.87	75.79	94.12	76.41	99.92	78.63	99.66
(10)	19.03	5.27	22.40	16.11	57.23	62.14	57.59	66.83	59.09	66.78	62.37	68.79

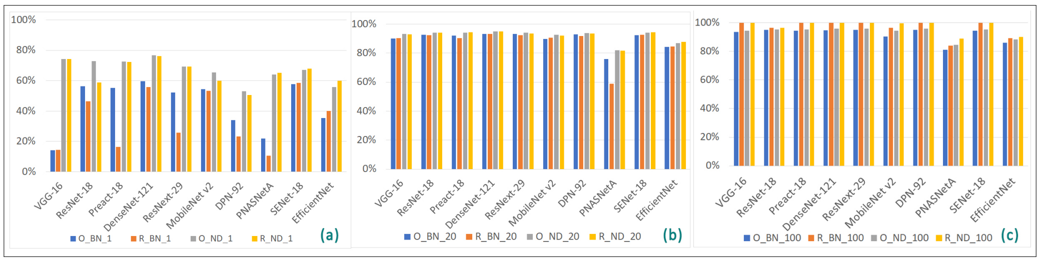


Figure 2. Comparing original and reproduced accuracy for CIFAR-10 dataset: Original Study [1] - Table 1 (a) with 1 epoch, (b) with 20 epochs, (c) with 100 epochs. (Legend: O_BN_20 - original accuracy using batch normalization with 20 epochs, R_ND_100 - reproduced accuracy using network deconvolution with 100 epochs).

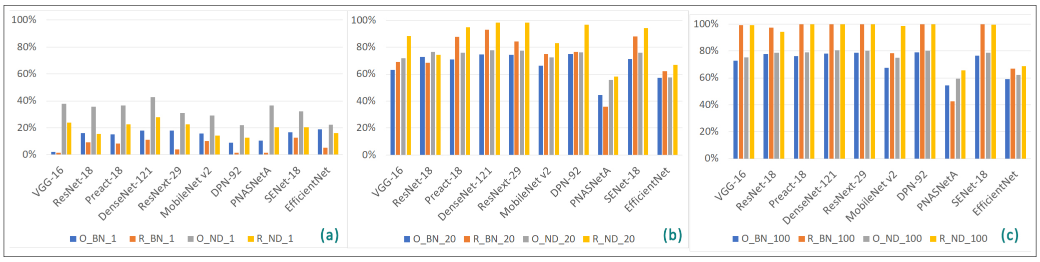


Figure 3. Comparing original and reproduced accuracy for CIFAR-100 dataset: Original Study [1] - Table 1 (a) with 1 epoch, (b) with 20 epochs, (c) with 100 epochs. Labels: ((Legend: O_BN_20 - original accuracy using batch normalization with 20 epochs, R_ND_100 - reproduced accuracy using network deconvolution with 100 epochs).

The results show that the model performance with network deconvolution is always better than the model performance with batch normalization. To determine if the values are reproducible, we consider a 10% reproducibility threshold for the accuracy metric as our criterion. In both CIFAR-10 and CIFAR-100 datasets 36 out of 60 values (60%) were reproduced with better results. In the ImageNet dataset 9 out of 14 values were reproduced with better results. There are few instances in CIFAR10 and CIFAR-100 datasets where the reproduced value is lower than the originally reported value. Most of these cases occur when the models were trained for 1 epoch. However, for the cases where the models were trained for 20 epochs and 100 epochs, the reproduced values show improved results, showing high accuracy values similar to those reported in the original study. The architecture PNASNet-18 shows weak performance in both batch normalization and network deconvolution as its accuracy values are comparatively low.

We calculated the averaged squared deviation from the original value for the three reproduced values obtained at three attempts . We used this calculated value to understand the consistency of the reproducing results. Consistency, in this context, means that the deviation of the reproduced values from the originally reported value should be minimal. Therefore, if the averaged squared deviation of reproduced values fall below 0.5 threshold we consider that the reproduced results are consistent. For CIFAR-10 dataset the averaged squared deviation lies between 0.00 to 0.23 and for CIFAR-100 dataset it lies between 0.00 and 0.11. Therefore the reproduced results are consistent.

5.2 Reproduced Results of Table 2 in the Original Paper

Table 6. Accuracy values reported by the original study Table 2 and the reproduced values for VGG-11 with 90 epochs (Rep.:Reproduced value).Color codes: Red - If the reproduced result is lower than the original value by more than 10%, Green - If the reproduced value is greater than the original value, Black - If the reproduced value is less than the original value, but the difference between two values is no more than 10%.

	VGG-11					
	Original	Original Rep.	BN	BN Rep.	Deconv	Deconv Rep.
ImageNet top 1	69.02	69.61	70.38	71.35	71.95	71.95
ImageNet top 5	88.63	89.08	89.81	90.25	90.49	90.60

Table 7. Accuracy values reported by the original study's Table 2 and the reproduced values for the architectures ResNet-18 and DenseNet-121 with 90 epochs (Rep.:Reproduced value). Color codes: Same as Table 6.

	ResNet-18				DenseNet-121			
	BN	BN Rep.	Deconv	Deconv Rep.	BN	BN Rep.	Deconv	Deconv Rep.
ImageNet top 1	69.76	70.72	71.24	71.58	74.65	75.88	75.73	75.67
ImageNet top 5	89.08	89.73	90.14	90.27	92.17	92.96	92.75	92.69

We reported the top-1 and top-5 accuracy values for BN and network deconvolution. Reproduced results for the VGG-11, ResNet-18, and DenseNet-121 using the ImageNet dataset are consistent since they fall within our reproducibility threshold. Therefore it confirms the main claim in the original study (Table 4, 5, 6, and 7).

5.3 Training Time Comparison

During our reproducibility study, we observed a remarkable difference in the training time between BN and network deconvolution, which was not reported in the original paper. Training time is often a factor to consider when building a deep learning architecture. In this section, we compare the training times of the BN and network convolution observed when testing them on 10 deep learning architectures. The results of CIFAR-10 are shown in Table 8 and Figure 4. The results of CIFAR-100 are shown in Table 9 and Figure 5. (Available at our GitHub repository [6] "*results/rep_values.xlsx*").

Table 8. Training time reported for the batch normalization and network deconvolution for CIFAR-10 dataset with 1, 20, 100 epochs. (Measuring unit: seconds).

Architecture	CIFAR-10					
	BN 1	ND 1	BN 20	ND 20	BN 100	ND 100
VGG-16	37.38	118.69	407.35	1183.07	990.47	3356.51
ResNet-18	35.41	112.63	1235.03	3586.99	1122.20	5142.16
Preact-18	47.88	180.79	639.52	1061.18	1356.69	5119.86
DenseNet-121	101.96	603.26	1156.61	4792.65	5549.08	23524.84
ResNext-29	66.39	324.62	906.48	2595.79	4473.94	12885.14
MobileNet v2	43.88	393.73	492.24	1738.16	2183.71	8106.39
DPN-92	190.40	708.68	2768.02	6500.02	13646.93	31776.99
PNASNet-18	35.22	153.21	369.79	1593.17	1811.29	7948.39
SENet-18	39.52	367.79	344.22	1517.79	1670.13	7020.19
EfficientNet	58.05	133.67	550.25	578.58	2647.03	2712.74

Table 9. Training time reported for the batch normalization and network deconvolution for CIFAR-100 dataset with 1, 20, 100 epochs. (Measuring unit: seconds).

Architecture	CIFAR-100					
	BN 1	ND 1	BN 20	ND 20	BN 100	ND 100
VGG-16	22.54	42.85	204.81	1104.37	1274.84	3424.08
ResNet-18	20.17	56.87	446.33	1076.18	1713	4589.303
Preact-18	22.70	68.50	294.96	1370.68	1856.72	5098.77
DenseNet-121	68.12	252.16	1193.88	5578.77	9109.30	25429.58
ResNext-29	56.71	142.49	909.63	4498.96	4574.03	12895.70
MobileNet v2	32.96	89.18	449.72	2845.72	2880.88	8181.95
DPN-92	158.08	342.03	2385.98	13684.59	11472.35	32227.37
PNASNet-18	27.09	89.98	378.45	1815.56	2956.37	8219.82
SENet-18	26.56	81.89	347.26	1691.54	2433.31	6813.16
EfficientNet	35.5	41.53	387.08	2683.59	1090.063	2784.08

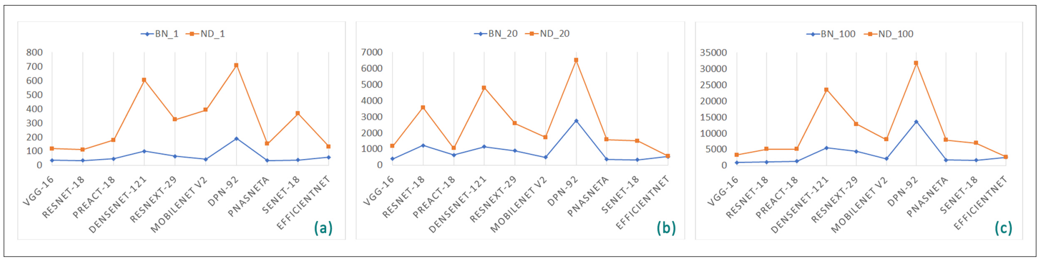


Figure 4. Training times for each CNN architecture with CIFAR-10 dataset: (a) with 1 epoch, (b) with 20 epochs, (c) with 100 epochs

[6] https://github.com/lamps-lab/rep-network-deconvolution

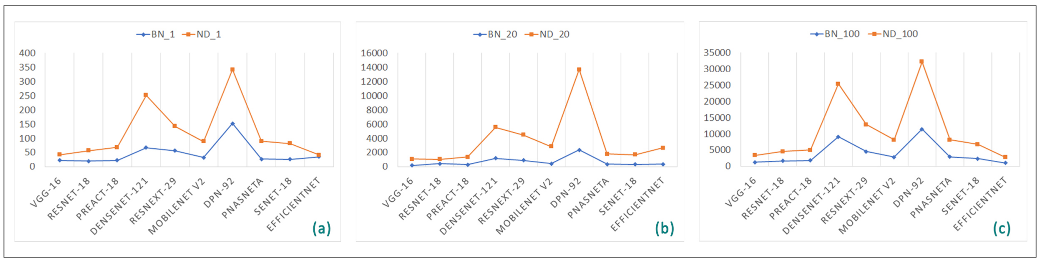


Figure 5. Training times for each CNN architecture with CIFAR-100 dataset: (a) with 1 epoch, (b) with 20 epochs, (c) with 100 epochs

Figures 4 and 5 depict that the training time of architectures with network deconvolution is always higher than the architectures with BN. Remarkable time gaps are visible in DenseNet-121 and DPN-92. DPN-92 always exhibits the longest training time for both BN and network deconvolution and the biggest difference between them. The shortest time is seen in the EfficientNet for the CIFAR-10. The same trend was observed in the CIFAR-100 dataset except ResNet-18 has the shortest time difference in 20 epochs.

Table 10. Model training times for the ImageNet dataset. (Measurements explanation: d-day, h-hour, min-minutes)

VGG-11			ResNet-18		DenseNet-121	
Original Rep.	BN Rep.	ND Rep.	BN Rep.	ND Rep.	BN. rep	ND Rep.
1d,7h, 23min	1d,15h, 38min	2d,21h, 27min	0d,13h, 8 min	1d,11h, 48 min	3d,21h, 18min	5d,14h, 10min

Table 10 shows the training times for the ImageNet dataset using the VGG-11, ResNet-18, and DenseNet-121 architectures. In this table, we report the time taken to reproduce each value stated in our study, as presented in Tables 6 and 7.

6 Discussion

In this work, we rigorously investigated the reproducibility of the reported results of the original paper by conducting experiments under the same hyperparameter settings. Specially, for Table 1 of the original study, we ran each architecture three times under the same hyperparameter configurations. The results of each run slightly vary due to the random initialization but majority of the results fall within the 10% reproducibility threshold. By systematically examining the model's performance under varying conditions, we aimed to provide insights on reprocubility and consistency of the reproduced results.

The reproduced accuracies for Table 1 and Table 2 of the original paper are nearly the same or even higher when compared to the original values. Among 134 reported results , 116 results (green or black) fall within the threshold of 10% (relative error) and 80 results (green) are better than the original values. This improvement is systematic and does not change the conclusion. In our investigation, we identified several factors that could potentially cause the systematic improvement of observed results. Notably, the advancement in numerical stability of the libraries and frameworks, such as the upgrades from NumPy 1.16.1 to NumPy 1.23.5 and from PyTorch 1.0 to PyTorch 1.13, may have contributed to the systematic improvements. Additionally, the improvement can also be caused by the implementation of optimization algorithms and parallelism in TensorFlow.

The calculated averaged squared deviation from the original values for each architecture during the three attempts show that all the results are consistent. The Training time comparison shows that network deconvolution in general requires more training time compared to BN but the performance improvements outweigh the increased training time.

Network deconvolution has been recognized and adopted in many in contemporary architectures, including U-Net[14], SegNet[15], GANs[16], Super-resolution networks[17], Pix2Pix[18], and DeepLab[19]. In most cases, this technique is used to up-sample the convoluted images to produce high-quality images.

6.1 What was easy

The use of a widely adopted deep learning framework: PyTorch, and benchmark datasets, like CIFAR-10, and CIFAR-100 made it easier to set up the experiments and compare our results with the original study.

6.2 What was difficult

Reproducing Table 01 — Despite the availability of the source code, we encountered a few issues during the reproducibility process. Initially, we faced a compatibility issue with different PyTorch versions, which required us to trial and error to resolve the issues across multiple versions. Furthermore, we encountered specific errors related to module imports and undefined models, which was resolved by importing the necessary models into the Python main script. Another challenge was the computational requirements for extensive experimentation across multiple architectures when working with the ImageNet dataset. This necessitated the use of internal GPU resources, as the Google Colab free tier had limitations in terms of GPU capabilities and runtime.

Reproducing Table 02 — We experienced the following difficulties while reproducing the values of Table 02 in the original study:

- The ImageNet dataset was not available in the original source and it redirected to the https://www.kaggle.com/c/imagenet-object-localization-challenge/data website. We downloaded the dataset from the redirected site. The size of the dataset was relatively large ($\approx$ 160GB) and it took around 8 hours to download and unzip the dataset.
- The folder structure of the downloaded dataset was different from the dataset used in the original study. *imagenet/ILSVRC/Data/CLS-LOC/val* folder did not contain the sub-folders for each class, so we had to prepare the dataset folder structure separately.
- We encountered a "*torch.cuda.OutOfMemory: RuntimeError*" due to the initial GPU memory we have been using. We had to transfer from 16GB GPU (NVIDIA A100) to 80 GB GPU (NVIDIA A100) memory in order to train ImageNet related 3 models.

6.3 Communication with original authors

We tried contacting the original authors during our reproducibility study but we didn't receive any response. However, their well-documented code base and clear methodology description in the paper provided sufficient information for us to reproduce the results successfully.

7 Conclusions

Our study finds that the accuracy results reported in the original paper are reproducible within a threshold of 10% with respect to the accuracy values reported in the original paper, and thus the authors' primary claim (network deconvolution improves the performance of deep learning models compared with batch normalization) is successfully verified. Network deconvolution emerges as a promising technique for improving model accuracy; although it may increase the training time by 2% to 358% depending on the architecture and training epoches.

References

1. C. Ye, M. Evanusa, H. He, A. Mitrokhin, T. Goldstein, J. A. Yorke, C. Fermüller, and Y. Aloimonos. "Network Deconvolution." In: **ICLR** (2020). https://openreview.net/pdf?id=rkeu30EtvS.
2. C. Ye. **GitHub - yechengxi/deconvolution — github.com**. https://github.com/yechengxi/deconvolution. [Accessed 18-04-2024]. 2019.
3. S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: **International conference on machine learning**. pmlr. 2015, pp. 448–456.
4. A. Krizhevsky, V. Nair, and G. Hinton. **CIFAR-10 (Canadian Institute for Advanced Research)**. http://www.cs.toronto.edu/~kriz/cifar.html. 2009.
5. A. Krizhevsky, V. Nair, and G. Hinton. **CIFAR-100 (Canadian Institute for Advanced Research)**. http://www.cs.toronto.edu/~kriz/cifar.html. 2023.
6. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database." In: **2009 IEEE conference on computer vision and pattern recognition**. Ieee. 2009, pp. 248–255.
7. J. Cao, R. M. Anwer, H. Cholakkal, F. S. Khan, Y. Pang, and L. Shao. **SipMask: Spatial Information Preservation for Fast Image and Video Instance Segmentation**. 2020. arXiv: 2007.14772 [cs.CV].
8. L. S. Passarella, S. Mahajan, A. Pal, and M. R. Norman. "Reconstructing high resolution ESM data through a novel fast super resolution convolutional neural network (FSRCNN)." In: **Geophysical Research Letters** 49.4 (2022), e2021GL097571.
9. S. Gidaris and N. Komodakis. "Locnet: Improving localization accuracy for object detection." In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016, pp. 789–798.
10. S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos. "Image segmentation using deep learning: A survey." In: **IEEE transactions on pattern analysis and machine intelligence** 44.7 (2021), pp. 3523–3542.
11. R. M. Kinney et al. **The Semantic Scholar Open Data Platform**. https://api.semanticscholar.org/CorpusID:256194545. 2023.
12. R. R. Obadage, S. M. Rajtmajer, and J. Wu. "SHORT: Can citations tell us about a paper's reproducibility? A case study of machine learning papers." In: **Proceedings of the 2nd ACM Conference on Reproducibility and Replicability**. ACM REP '24. Rennes, France: Association for Computing Machinery, 2024, pp. 96–100.
13. T. Contributors. **torchvision.datasetsx2014; Torchvision 0.8.1 documentation — pytorch.org**. https://pytorch.org/vision/0.8/datasets.html. [Accessed 03-09-2024]. 2017.
14. O. Ronneberger, P. Fischer, and T. Brox. **U-Net: Convolutional Networks for Biomedical Image Segmentation**. 2015. arXiv: 1505.04597 [cs.CV].
15. V. Badrinarayanan, A. Kendall, and R. Cipolla. **SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation**. 2016. arXiv: 1511.00561 [cs.CV].
16. I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. **Generative Adversarial Networks**. 2014. arXiv: 1406.2661 [stat.ML].
17. J. Guo, X. Zou, Y. Chen, Y. Liu, J. Hao, J. Liu, and Y. Yan. **AsConvSR: Fast and Lightweight Super-Resolution Network with Assembled Convolutions**. 2023. arXiv: 2305.03387 [eess.IV].
18. P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. **Image-to-Image Translation with Conditional Adversarial Networks**. 2018. arXiv: 1611.07004 [cs.CV].
19. L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. **DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs**. 2017. arXiv: 1606.00915 [cs.CV].