Post Training in Deep Learning Reproducibility Report

Project 4: Reproducible Machine Learning. COMP-551: Applied Machine Learning, Fall 2017

Lita Fan
lita.fan@mail.mcgill.ca
McGill ID: 260787539

Rez GodarzvandChegini
ahmadreza.godarzvandchegini@mail.mcgill.ca
McGill ID: 260795422

Deeksha Arya
deeksha.arya@mail.mcgill.ca
McGill ID: 260786737

Project Repository Available at: https://github.com/deekshaarya4/Post_training.git

***Abstract*—Ensuring the reliability and reproducibility of published studies is one of the challenges faced by the machine learning research community. As part of an effort to encourage awareness and further studies into this challenge, we aimed to replicate the experiments in and evaluate the effort of reproducibility of one of the papers submitted to the ICLR 2018 Conference. 'Post Training in Deep Learning' proposes an additional training step following regular training of a network to improve performance. We compared the results in the paper with our own attempts. While we can confirm the improvements from post-training in all specified cases, certain inconsistencies and overlooked details in the paper and corresponding code complicated reproducibility and our confidence in the results.**

I. Introduction

Reproducibility is the ability of an existing experiment to be identically performed by the same or another researcher in order to verify results of a study. With more opportunities for research and new findings, it has become essential that claims of experimental results are valid in order for them to be used in further research. However, as researchers devote their time towards new fields, few are working on reproducing and ensuring the credibility of previous work. In 2016, a survey conducted by Nature revealed that over 50% of over 1,500 scientists had attempted to and failed at reproducing their own experiments and over 70% could not reproduce work by other scientists published in scientific journals [1]. In many cases, authors do not divulge important aspects of their experiments and their generated machine learning models. Hence, recreating their results becomes a difficult task.

In order to overcome the lack of work towards reproducibility, Professor Joelle Pineau from McGill University, in coordination with professors from five other universities, raised the ICLR 2018 Reproducibility Challenge, the goal of which was to assess the reproducibility of papers submitted to the ICLR 2018 Conference [1].

As participants of this challenge, we evaluated the reproducibility of the experiments presented in the paper 'Post Training in Deep Learning' [2]. We chose the paper because we found the concept and its applicability to all types of neural networks intriguing. Additionally, the paper was well-written and clear, and the experiments were well-explained. As well, the code for this project was made public by the authors, which facilitates the reproducibility process.

In this project, we attempt to run the experiments conducted in the paper, analyze the alignment of the results of these experiments against the claims of the paper, and assess the ease with which this could be done.

[1]http://www.cs.mcgill.ca/jpineau/ICLR2018-ReproducibilityChallenge.html

II. Post Training in Deep Learning

The motivation for this paper is the challenge of efficiently solving complex and non-convex problems in the training of artificial neural networks. Gradient descent methods result in convergence to the local minima of the error function but, as pointed out by the authors, depend greatly on weight initializations of the network. The authors study pre-training [4] as a good method to begin solving this problem and cite multiple successful experiments on different datasets using this concept.

The authors describe that a neural network can be divided into two parts where the first layers are responsible for data representation, and the last layers are inherently responsible for learning the best usage of this representation to solve a classification or regression problem.

This structure breakdown is the platform for the authors proposal for a method called post-training in order to improve the performance of a network. This algorithm focuses on improving and finding the best usage of the learned data representation. Although the authors do not specify that data representation is only done by the last layer, in their paper, they focus on improving the performance of the last layer particularly in the post-training phase.

Post-training happens as an additional step after regular training of the network. The authors state that there is no restriction on the regular training. Any number of layers, with any number of nodes, as well as any techniques such as Dropout[5] or Adagrad[6] can be included. The authors suggest that a fixed number of epochs can be used or an early stopping mechanism[7] can be implemented in the classical training process.

After a model is regularly trained, post-training is applied. Here, the first $L-1$ layers of the network are frozen, and only the weights $\mathbf{W}_L$ of the last layer Φ_L 1 are trained by solving the following optimization problem,

$$\underset{\mathbf{W}_L}{\text{argmin}} \frac{1}{N} \sum_{i=1}^{N} \tilde{\ell}(\Phi_{L-1}(x_i)\mathbf{W}_L^T, y_i) + \lambda \|\mathbf{W}_L\|_2^2 \quad (1)$$

where $\tilde{\ell}(x, y)$ is the loss function. The use of ℓ_2-regularization is to reduce over-fitting. In addition, since there is no back-propagation step in this process, the claim is that post-training is computationally faster than classical training. The optimization problem is convex given a softmax activation in the last layer and cross entropy loss[2]. The authors specifically mention that, during post-training, dropout should not be applied on the previous layers as this would change the feature function Φ_{L-1}. A visual representation can be seen in the figure 1.

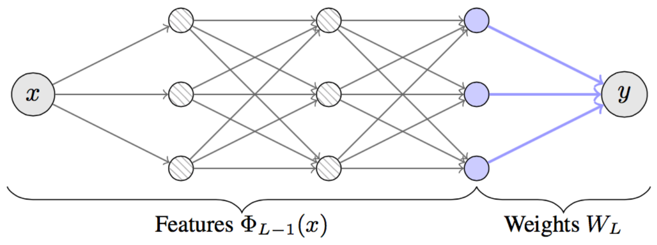

Fig. 1. Illustration of the post-training step on a neural network. During post-training, only the weights of the blue edges are updated.

The paper suggest that the optimal weights for the final layer can be solved using kernels, asserting that the minimization on W_L in the post training phase is similar to the Reproducing Kernel Hilbert Space (RKHS) H_k and is given by

$$\mathbf{W}_L{}^* = \sum_{i=1}^{N} \alpha^* \Phi_{L-1}(x_i) \quad (2)$$

where $\mathbf{W}_L$ is the optimal solution of $\mathbf{W}_L{}^*$ and $\alpha^* \in \mathcal{R}^N$.

The paper states that when the loss function is mean squared error and the activation function of the last layer is given by $f(x) = x$, the problem becomes a Kernel Ridge Regression problem and can be solved using a closed-form solution. Then $\mathbf{W}^*$ can be computed by combining (2) and

$$\alpha^* = (\Phi_{L-1}(D)^T \Phi_{L-1}(D) + \lambda I_N)^{-1} Y \quad (3)$$

where $\Phi_{L-1}(D) = [\Phi_{L-1}(x_1), ...\Phi_{L-1}(x_N)]$ is a representation of the input data $x_1, ...x_N$, Y is the matrix of output data $y_1, ...y_N$ and I_N is the identity matrix of $\mathcal{R}^N$.

As a result of kernelisation, the authors claim that post-training can be extended to multi-dimensional space.

III. Reproducibility Methodology

In order to verify their claims that post-training performs well in a variety of settings, the authors conduct experiments using three different networks and several well-known datasets.

In the paper, the authors present the results of applying post-training for the following experiments:

- Convolutional neural network on the CIFAR10, FACES, and MNIST datasets.
- Recurrent neural network on the Penn Tree Bank (PTB) dataset.
- Kernel ridge regression on the Parkinson Telemonitoring dataset and a synthetic dataset provided by the authors.

We aimed to reproduce the results of these experiments to confirm that post-training performs as reported.

The code for these experiments were made available by the authors and can be found on Github[2]. We ran the code with minor adjustments on the following system:

CPU	8 Core Intel(R) Xeon(R) (2.3GHz)
RAM	30GB
GPU	1 × Nvidia K80 (16GB)
Storage	500GB

A. Convolutional Neural Networks

To analyze the post-training methodology and its success, the authors applied the concept to a feedforward convolutional neural network and experimented on the following well-known image datasets:

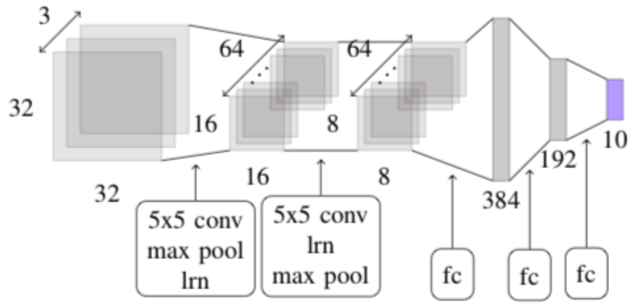

Fig. 2. Neural Network Structure used for CIFAR-10 classification experiment comprising of 5x5 convolutional layers (5x5 conv), max pooling activation (max pool), local response normalization (lrn) and fully connected linear layers (fc)

1) CIFAR10: The CIFAR10 dataset comprises of 60,000 images of size 3232. These images can be classified into objects from 10 classes. The network used to run this experiment is the default architecture proposed by Tensorflow[3], based off the original architecture proposed by Alex Krizhevsky[8]. Fig 2 illustrates the 5-layered CNN architecture used for this experiment. The blue highlighted layer is the one updated during post-training. This layer uses a softmax activation function and loss is measured using cross entropy.

For evaluation purposes, q iterations of classical training are compared with $q - 100$ iterations of classical training followed by 100 iterations of post-training. This is integral, as

[2]https://github.com/tomMoral/Post_training

it not only helps evaluate the performance of post-training but also whether or not the model is over-fitting. The 100 iterations of post-training are done using a regularization parameter of 0.001.

To perform the experiment, the dataset is already divided into training and test sets [8]. The network is trained for 90k iterations and at every 100 iterations, a comparison is made between loss using the classically trained model and that of the post-trained model. Weight initializations of the neural network are initialized using a normal function with standard deviation 0.05. The learning algorithm runs with 128-sized batches using stochastic gradient descent. It implements a dropout mechanism and exponential weight reduction for the learning rate.

The paper claims that though training loss is slightly higher in the case of post training than when normally trained, generalization tends to be better. Test error tends to be slightly lower than the normal case. The average time spent on post training too, is stated to be 4 times faster than classical training.

2) FACES: The FACES dataset by AT&T Laboratories Cambridge consists of 400 92×112 images: 10 each of the faces of 40 distinct individuals. The images have been resized to 64×64 and 102400 sub-images of size 32×32 are then extracted, 92160 of which used for training and 10240 for testing.

Two convolutional neural network architectures are specified in the paper:

- A small network: one convolutional layer (5×5 patches, 32 channels), one 2×2 pooling layer, and one fully connected hidden layer with 512 neurons.
- A large network: one convolutional layer (5×5 patches, 32 channels), one 2×2 pooling layer, one convolutional layer (5×5 patches, 64 channels), one 2×2 pooling layer, and one fully connected hidden layer with 1024 neurons.

Both networks are trained for 5000, 10000, and 20000 iterations with and without an additional 100 iterations of post-training. The application of post-training is expected to lower the classification error.

3) MNIST: A dataset of 65000 27×27 images of handwritten digits. 55000 of these images are used for training and 10000 are used for testing. The two convolutional neural network architectures specified above for the FACES dataset are also used on the MNIST dataset. The networks are trained for 1000, 2000, and 5000 iterations.

B. Recurrent Neural Network

To study post-training in Recurrent Neural Networks, the authors evaluated their model on Penn Tree Bank (PTB) dataset [10], composed of 929k training words and 82k test words. The dataset was not automatically downloaded by the code, unlike the other experiments. However, we verified that the dataset that was used for this experiment was obtained from Tomas Mikolovs webpage [3] as referenced by [9]. The objective of the model is to predict the next words in input samples given the word history.

[3]http://www.fit.vutbr.cz/ imikolov/rnnlm/simple-examples.tgz

The Recurrent Neural Network model uses the architecture proposed in [9] that employs two layers of 1500 LSTM units with tanh activation function and a softmax output layer. The number of unrolled steps of LSTM was chosen as 35. The network's weights are initialized uniformly in [0.04, 0.04] and a 65% dropout on the non-recurrent connections were implemented. The norm of the gradients (normalized by minibatch size) is capped at 10. The experiment takes 450 iterations on top of 100 iterations of training for the post-training model in batch sizes of 20. In other words, comparison made through the provided code is indeed between q iterations of regular training and q regularly trained iterations + 100 iterations of post-training. The learning rate starts at 1, and decays at rate of 1.15 per epoch after 14th epoch. Lastly, the regularization parameter was claimed to be 0.01 in the paper but in the provided code was 0.001.

C. Kernel Ridge Regression

The paper compares performance of classically trained models, post-trained models, as well as the closed form solution models.

To do this, a Feed Forward Neural Network with 2 hidden layers is evaluated using two different datasets, one real and one simulated. 70% of the data is used for training, while 30% is used for testing in both cases. Weight initialization is done from a uniform distribution, and the results reported in the paper use a seed of 1 for the initialization. All architecture details of the network were provided in the paper for each of the test cases. Batches of 50 were trained for 250, 500 and 750 iterations, and the post-training phase comprised of 200 iterations. Here, q iterations of classical training are compared with q iterations of classical training and additionally 200 iterations of post-training. ℓ_2 regularization with a fixed regularization parameter of 0.001 is used.

1) Real Data Set Regression: The Parkinson Telemonitoring dataset is a numerical representation of voice inputs. Comprising of 5,975 rows of 17-dimensional data, the output is a single real number. A 4-layer fully connected neural network was used for this data set. The two hidden layers were of size 17 with tanh activation and size 10 with RELU activation respectively.

2) Simulated Data Set Regression: For this experiment, a dataset was created by generating 10,000 pairs (x, y) where x was sampled from a uniformly distributed dataset on $[0,1]^10$. Outputs are computed as $Y = tanh(XW_1)W_2$ where $W_1 \in [-1,1]^10x5$ and $W_2 \in [-1,1]^5$ are randomly generated using a uniform law.

IV. Empirical Results

A. Convolutional Neural Networks

1) CIFAR10: To reproduce this experiment, we ran the same code provided by the author on the CIFAR-10 dataset once again. We began training the algorithm on our local machines, i.e a 4-core 1.8GHz processor, but soon realized that at an average of 0.732 seconds/batch for regular training and 0.233 seconds/batch, the model would take too long to

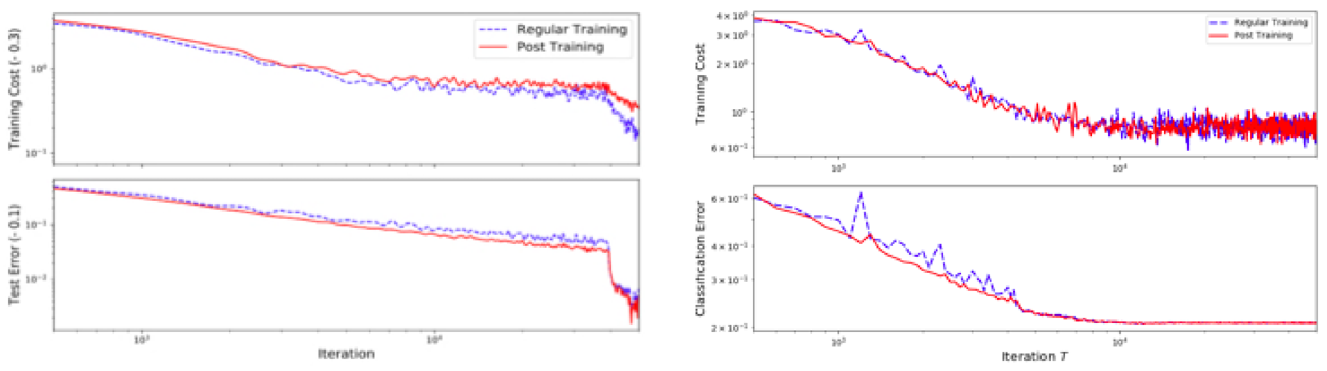

Fig. 3. The plots describe the error values for classification of CIFAR-10 data using models with and without post-training. The plot to the left is from the paper, while the plot to the right illustrates the results on reproducing the experiment

train and predict. As a result, we obtained access to a more powerful CPU III which reduced the evaluation time per batch to an average of 0.347 for regular training and 0.112 for post-training. Ultimately, the entire experiment took approximately 18 hours to complete even with the powerful server.

Though code for plotting training and classification error was provided, slight modifications had to be made in order to allow easy comparison with those presented in the paper. These included changing the scales of the graph to log scales, limiting the x-axis, and modifications to the visual representations. The paper vaguely mentioned that smoothing was done to the graph, but no details were given and no such function was found in the corresponding code.

3 compares the results of the experiment as reported in the paper and obtained by us on performing the experiment.

We do understand that neural networks converge to only the local optima and, with different initializations, the actual error values may be slightly different than those in the paper. However, we evaluate the success or failure of post training, and we find that training error is more or less the same between both training strategies, while post-training in general performs better on the test set in comparison to regular training.

However, as claimed by the paper, post-training does take a significantly less amount of time than regular training. The paper states that post-training performs 4 times faster than regular training. In this experiment, post-training was found to perform approximately 3 times faster than classical training. The paper claims that post-training generalizes well but, in the graphs, it appears that even regular training does not overfit. Hence, it is difficult to comment upon the generalization capabilities of post-training.

2) FACES: The paper's results and our results for the experiment are compared in Table I.

Code was provided by the authors to reproduce the experiment. Running the code for the small network gave us results very close to those presented in the paper.

However, in the case of the large network, we found that the errors from the provided code were over 10% higher than the reported errors for iterations 10000 and 20000, with and without post-training. We believe that the authors may have used different hyper-parameter values for the large network than the ones provided in the code. We noticed that the value of the regularization parameter λ in the code was set to 10 as opposed to the paper's reported 1×10^{-2} and, when we also set the learning rate decay and dropout rate (the values of which are not present in the paper) to those of the small network, we obtained results that were closer to the paper's. A more extensive hyper-parameter search may give us even closer error values.

Although our numbers differed from the authors', their conclusion that post-training gave an improvement in all cases is confirmed with our results. For all observed iterations and network complexities, applying post-training reduced, even if slightly, the classification error.

3) MNIST: As with FACES, code was made available for reproducibility. However, the code used the term 'steps' rather than 'iterations' like in the paper. The relationship between these terms is undefined by the authors. In all other code where we encountered this issue, we came to the conclusion that 'steps' and 'iterations' were used equivalently by the authors. This was supported by the similarity between our results and the paper's results following this assumption for the other experiments. Under this conclusion, however, the number of iterations run by the MNIST experiment code and the number of iterations stated in the paper differ by a factor of 10. To our understanding, 10 iterations of post-training are being run in place of the 100 stated in the paper.

In Table I, our results are generated from the authors' code with the number of iterations run and the regularization parameter λ set to those of the paper's. We observe that the values of our classification error differ significantly from the paper's and, because of the uncertainty in the alignment of terminology between the paper and the code, we are not confident that we are reproducing the experiment described in the paper. Nonetheless, our results still show that the application of post-training improved performance in all cases as claimed by the authors.

B. Recurrent Neural Network

As described earlier, the paper outlines a comparison between two networks of regular training and post-training over q and q+100 iteration, respectively. The post-training network

TABLE I. Classification error on FACES and MNIST at different iterations with and without post-training.

Dataset	Network	Iterations	Mean (Std) Error in %		Mean (Std) Error with post-training in %	
			Paper	Ours	Paper	Ours
FACES	Small	5000	21.5 (10)	27.7 (3.7)	19.1 (12)	24.8 (2)
		10000	20 (4)	20.3 (1.4)	19 (3.5)	19.7 (1.3)
		20000	18 (0.9)	18.6 (1.5)	16.5 (0.8)	18 (1.5)
	Large	5000	25 (15)	33.4 (16.9)	24 (15)	29.5 (13.5)
		10000	15 (5)	18.8 (2.5)	12 (5)	17.9 (2)
		20000	11 (0.5)	15.6 (1.1)	10 (0.5)	15.5 (1)
MNIST	Small	1000	10.7 (1)	2.6 (0.3)	9.2 (1.1)	2.1 (0.09)
		2000	7.5 (0.7)	2.1 (0.4)	6.7 (0.6)	1.8 (0.1)
		5000	4.1 (0.2)	2.0 (0.3)	3.9 (0.3)	1.9 (0.2)
	Large	1000	9.1 (1.3)	2.0 (0.5)	8.5 (1.4)	1.6 (0.3)
		2000	4.1 (0.2)	1.7 (0.4)	3.5 (0.2)	1.5 (0.3)
		5000	1.1 (0.01)	1.6 (0.2)	0.9 (0.01)	1.4 (0.1)

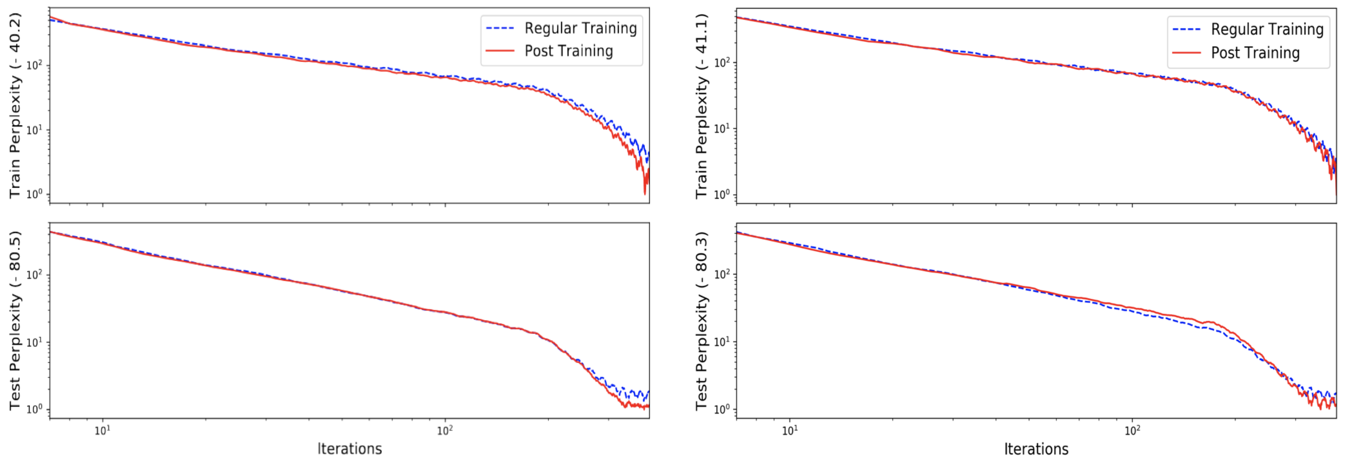

Fig. 4. The plots describe the perplexity values for the experiment on PTB dataset using models with and without post-training. The plot to the left is from the paper, while the plot to the right illustrates the results from reproducing the experiment. The regularization parameter is taken $\lambda = 0.001$.

is trained depending on the regular training that it is being compared to. In other words, at each iteration of the regular training, we branch out from the regularly trained network that we have so far, and train it over 100 iterations of post-training. However, the results of this process show that post-training does not in fact perform better than regular training before reaching a certain number of iterations. Despite our reservation that it would be advantageous to compare classically trained and post-trained networks trained with equal number of iterations, we confirmed that in the case presented in the paper, the network with post-training eventually outperforms the regular network with respect to test error as can be seen in Fig 4.

C. Kernel Ridge Regression

With the same random seed of 1 used to initialize weight parameters, the same error values as in the paper were observed. It was found that, as claimed in the paper, post-trained models perform better than the equivalent classically trained models and, though it approaches the accuracy of the closed form solution, it still does not perform as well. However, as the paper mentions, the closed form solution also tends to overfit on the Parkinson data, which can be seen in Table II. At iteration 500, the optimal solution produces a larger test error (0.140) than at iteration 250 (0.119), implying that it does not generalize well to previously unknown data.

Here, it seems that post-training does do significantly better than classical training without over-fitting.

In this case, q iterations of classical training were being compared with q iterations of classical training + 200 iterations of post-training. Surprisingly, we found if we classically train with an additional 200 iterations, the network in fact performs better than the optimal closed form solution and the post-trained model, and does not necessarily overfit. For example, as can be seen in Table II, at 250 iterations on the fully-connected network using the Parkinson Telemonitoring dataset, the error is 0.832. With an additional 200 iterations of post-training, the error is reduced to 0.433. However, if instead 200 iterations of regular training were performed, we found that the error would be reduced to 0.167.

To better evaluate the performance of post-training, a random seed of 2 was used as well, and we found a similar trend of error values for this case as well.

TABLE II. Kernel ridge regression results

Dataset	Iterations	Error with classical training	Error with additional classical training	Error with post-training	Error with optimal last layer
Parkinson	250	0.832	0.167	0.433	0.119
	500	0.147	0.134	0.147	0.140
	750	0.134	0.129	0.132	0.131
Simulated	250	1.185	0.535	1.122	1.075
	500	0.533	0.360	0.453	0.447
	750	0.322	0.259	0.299	0.296

V. Discussion

We evaluate the reproducibility of the paper according to the reproducibility metrics in Table III.

The datasets used for the experiments in the paper are CIFAR10, FACES, MNIST, PTB, Parkinson Telemonitoring, and a simulated dataset by the authors. With the exception of the last, all are popular benchmarks in machine learning and publicly available. The simulated dataset was well explained in the paper and could be generated accordingly. The partitioning of training and testing is specified in the paper for each dataset.

The code for all experiments was released by the authors. The versions of Python and Tensorflow used, however, were unspecified and caused us some difficulty when trying to run the code.

A number of the hyper-parameters values used were specified in the paper and those missing were often found in the corresponding code. In some cases, though, we found that the value in the code did not match the value in the paper. One such example of this is the experiment involving large convolutional neural networks and the FACES dataset. The regularization parameter λ is stated to be 1×10^{-2} in the paper but is set to 10 in the code. In this case and in others, changing the conflicting hyper-parameter(s) alone still did not achieve results in the range of the paper's. Thus, we believe some hyper-parameter values in the code which were not reported in the paper may not be the hyper-parameter values used to generate the results in the experiments.

For the most part, we found the paper and the code to be reasonably clear. One significant exception is that the terms 'epochs', 'iterations', and 'steps' and, specifically, their relation to one another have been left undefined by the authors. In some cases, at least two of these terms appear to be used interchangeably in the paper and the code while not so in others. This caused some confusion for reproducing experiments where these terms were used inconsistently between the paper and code, most particularly in the CNN-MNIST experiment. Another is that the experiment methodology is not explicitly stated for every case. In the CNN-CIFAR10 experiment, the paper is very clear that it is q iterations of regular training and $q - 100$ iterations of regular training with 100 additional iterations of post-training that is being compared. However, the subjects of comparison for the other experiments are less clear and had to be determined by looking at the code and the results of the code. It was found that in these other experiments, q iterations of regular training were compared with q iterations of regular training with p iterations of post training, where p was either 100 or 200 based on the experiment. Hence, within the code, there was discrepancy regarding the comparison and evaluation mechanism.

The use of a GPU was mentioned but the details of the computing infrastructure used by the authors for the experiments was not included in the paper. Without a GPU, we found the experiments computationally expensive to run. With one, we were able to run the experiments without issue and within a reasonable amount of time.

As the code for the experiments were provided by the authors, the effort of reimplementing the paper was low. Few changes to the code were necessary to obtain results, though some time was required to understand and correct misalignments from the paper. Some of our final results diverge from those in the paper but the general expected behavior is observed.

We have reached out to the authors regarding the points of confusion we have with the paper but unfortunately have not received a response at the time of writing this report.

Overall, the paper scores well by most of the reproducibility metrics and consequently we believe the paper to be mostly reproducible. Areas that can be worked on to further improve reproducibility are the availability of all hyper-parameters, consistency in terminology between the paper and the code, and details of the computing infrastructure used.

TABLE III. A summary of the reproducibility of the paper according to reproducibility metrics.

Availability of datasets, partitioning information.	Yes
Availability of code, names and version numbers of dependencies.	Yes [1]
Availability of random seed and all hyper-parameters.	Partial
Alignment between the paper and the code.	No
Clarity of code and paper (ease of understanding, language).	Strong
Details of computing infrastructure used.	No
Computation requirements (time, memory, number/type of machines).	Moderate
Reimplementation effort (time, expertise).	Low
Number and complexity of interactions with the authors.	N/A

VI. Conclusion

The paper 'Post-Training in Deep Learning' is overall well presented. Though we found discrepancies in the paper and the code, we find the concept extremely interesting and we believe that greater insight into the validity of post-training can be attained with more experimentation and a greater scope of comparison. For example, it would be helpful to be able to compare post-training results with that of proposed pre-training strategies. Another point to pursue with regard to this paper would be to examine whether post-training intrinsically does not overfit as claimed by the authors, or whether overfitting does not happen due to the addition of the regularization term to the loss function (1).

Overall, the paper does well in terms of reproducibility. While the numerical results for the experiments are not always a match, which can likely be improved by further details and attention to consistency in the paper and code, the general behavior of post-training can be replicated for all presented experiments. Our findings in this report show that the authors' claims regarding the improvements by post-training in the experiments are reliable and reproducible.

VII. Statement of Contributions

This project was the result of the combined effort of each of the group members.

Deeksha attempted to reproduce the post-training experiment using a CNN architecture on the CIFAR-10 dataset. The fully-connected network experiments on Parkinson Telemonitoring data and Simulated data were also handled by her.

Lita worked on reproducing the experiments using the CNN network on FACES and MNIST dataset. This included, but was

[1]https://github.com/tomMoral/Post_training

not limited to making changes in the code to align with the details of the experiment presented in the paper.

Rez had the responsibility of reproducing and analyzing the results of post-training on the specified RNN network using the Penn Tree Bank dataset. His contribution involved analysis of the results and comparison with the claims in the paper.

All three members of the group did a critical evaluation on their assigned experiments, as well as wrote about their work in the report. We hereby state that all the work presented in this report is that of the authors.

References

[1] Mpnya Baker. Is there a reproducibility crisis?, Nature, pp. 452-454, 2016.

[2] Anonymous. Post Training in Deep Learning, 2017. URL https://openreview.net/pdf?id=H1O0KGC6b.

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane , R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Vie gas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, software available from tensorflow.org. URL https://www.tensorflow.org/.

[4] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In International Conference on Machine Learning (ICML), pp. 473480, Corvallis, United States, 2007. ACM.

[5] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for LVCSR using rectified linear units and dropout. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 86098613, Vancouver, Canada, 2013. IEEE.

[6] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research (JMLR), 12:21212159, 2011. ISSN 15324435. URL http://jmlr.org/papers/v12/duchi11a.html.

[7] Nelson Morgan and Herv Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. International Computer Science Institute, Denver, United States, 1990.

[8] Alex Krizhevsky. Learning multiple layers of features from tiny images. Masters thesis, University of Toronto, 2009. URL http://www.cs.toronto.edu/ kriz/cifar.html.

[9] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. arXiv preprint, arXiv:1409(2329), 2014.

[10] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. Computational linguistics, 19(2):313330, 1993.

Appendix: Open Review

The paper on Post-Training in Deep Learning suggests another phase of training after a phase of regular training in neural networks. The second phase involves freezing all but the last layer and optimizing the loss function with respect to the weights at the last layer over several additional iterations of training. The authors assert that this additional phase can lead to an improvement of performance in neural networks.

We attempted to reproduce the experiments performed by the authors in order to verify the claims in the paper. The paper is crisp, clear, and well-written in its explanations and hence facilitated the understanding and reproducibility process. In addition, the authors have kindly made their code public, and all experiments were conducted on either well-known datasets or those that could be generated with details provided in the paper. We found that the clarity of the code is also reasonable and any technical details lacking in the report could, for the most part, be found in the code.

The authors performed experiments on three major classes of neural networks, namely CNNs, RNNs, and Feed Forward Neural Networks, using different datasets to compare the performance of additional post-training with classical training. We observed a discrepancy between the paper and its implementation in the setting of these iterations. In general, in the provided code, the accuracy or error comparison (based on the experiment) is made between q iterations of regular training and q regularly-trained iterations + p iterations of post-training. In only the experiment of the CIFAR-10 dataset using CNN, $q + 100$ iterations of regular-training were compared with q regularly-trained iterations + 100 iterations of post-training. Nonetheless, for all observed iterations and network complexities, we confirmed the authors findings that applying post-training, on average, improves test accuracy, and executes faster than regular training per iteration. That being said, through our own experimentation, we found that in the Kernel Ridge Regression experiment, an additional number of iterations of regular training would, in-fact, result in lower test error than the equal number of additional iterations of post-training, without necessarily overfitting. For example, at 250 iterations on the fully-connected network using the Parkinson Telemonitoring dataset, the error claimed by the authors in Table 2 of the paper is 0.832. With an additional 200 iterations of post-training, the error reduces to 0.433, however if instead 200 iterations of regular training were performed, we found that the error would have reduced to 0.167. Also, for the CNN experiment on the MNIST dataset, there was no clear relationship between the number of training iterations in the paper and those run in the code, which meant we could not reproduce the training conditions with certainty.

Overall, we believe that the paper is well presented and the experiments support the advantages of post-training stated by the authors. We conclude that the paper is reproducible.