Do Regularizers improve "Reproducibility" in Deep Neural Networks?

Abstract

A known problem with training deep neural networks, mostly parameterized by connection weights at each layer, is that of finding an appropriate model complexity under the empirical risk minimization setting. Ideally, the goal is for the model to produce a satisfactory indication of high accuracy while retaining the attractive property of performance reproducibility (generalization), at least, on a test-set assumed to be drawn from the same distribution as the train-set. In an attempt to address such problems, a lot of regularization methods have been proposed. In this work, we empirically study regularization and reproducibility from the perspective of test-set prediction consistency across several training runs. We then introduce ***effective test accuracy*** *as a trust-measure on the predictions made during training, which can supplement conventional* ***test accuracy*** *as a metric for reporting performance. Specifically, we use the metric to quantify the effect that common categories of regularizers used in training deep feed-forward neural networks have on consistent (reproducible) predictions. Our findings across multiple benchmark image recognition datasets indicate that the structure of a feed-forward neural network is a higher weighted factor than other explicitly added regularization methods for improving reproducibility.*

1. Introduction

Intuitively, we start with the hypothesis that regularizers induce some amount of stability in the learning process, the learning process determines the weights (representations) in the neural network, and the weights determine the predictions of the network. Then if we were to attenuate most sources of variability during training, by controlling the random generator seeds used, such as: those used in the random initialization and sampling process; we would expect that compared to when the explicitly added regularizers were absent, we should obtain some amount of increased reproducibility on the network's predictions (less prediction difference) on the test set across training runs, irrespective of the number of times we were to train the network.

In this work, we attempt to prove the aforementioned intuitive expectation. Our empirical findings show that such hypothesis can be significantly rejected. We find that, despite recommended seed control, the model (representations) found by the neural network learning process varies each time it is trained. Interestingly, even though the test accuracy across training runs seem close with respect to some decimal places. The test prediction difference most always varies. To capture the effect of such variability when training deep neural networks, we introduce a newer metric, *Effective Test Accuracy*, as a better measure of trusting test performance results compared to using only the popular *Test Accuracy* metric.

1.1. Background

In the empirical risk minimization (maximum likelihood estimation (MLE)) setting [40], a common introduction to the problem of underfitting, overfitting and generalization in machine learning theory, most always starts with a background concept on the trade-off between bias error and variance error [12]. A high bias error indicates underfitting on the train dataset. A high variance error indicates underfitting on the test dataset or overfitting on the train dataset. Generalization error is a combination of both the bias and variance errors. If both errors are large, generalization is poor [7]. The goal is to find a model that appropriately balances this bias-variance trade-off or in other words, a model with an appropriate complexity. Learning theory is usually concerned with confidence bounds on the generalization error [6, 11, 14] and how to construct algorithms that control the generalization ability of learning [38]. Generally, in learning with neural networks as a class of prediction rules, the model complexity is influenced by a number of factors.

Particularly, for feed-forward neural networks [1], as the number of layers, hidden neurons, initial input neurons, and final output neurons in the network increase, the number of connection-weight parameters also increase. Therefore, bias error has a tendency of reducing, while increasing variance error (overfitting), hence increased generalization error during training. Therefore, such models have been referred to as high capacity models (models that can memorize the training data), with extreme tendency to overfit.

In a sense, the number of learnable parameters is fixed. Therefore, the task of reducing overfitting, then simplifies to controlling the complexity of the selected model class for a particular training dataset by constraining the values

of the connection-weight parameters [8]. It has been shown in [5] that the magnitude values of the weights determine the generalization performance of neural networks, instead of the number of weights. Approaches that control the model complexity of learning in neural networks and other machine learning models are generally known under the name of *regularization*. The aim here is to try to reduce the bias error and at the same time reduce the variance error, without changing the number of connection-weight parameters input to each layer.

The first general approach is to identify an appropriate complexity (or select a best model) by varying (increasing or reducing) or changing an hyper-parameter that is linked to regularization and then monitoring performance on both train and validation data, during training. This is known as *model selection* and appears in techniques of early-stopping, cross-validation or structural risk minimization [3, 16, 39, 40].

The second general approach attempts to add an *estimated measure of the complexity* of the connection-weight parameters or other appropriate parameters as a form of noise to the training error (empirical risk (loss) or objective function), changing the unconstrained risk minimization problem to a constrained optimization problem or regularized loss function [25, 39]. Methods in this approach can also be viewed as injecting a form of adversarial noise (perturbations) to the neurons (input and hidden) and then minimizing either a constrained or unconstrained empirical risk [7]. Some methods that standalone in this approach are the use of *dropout* [41], and *data augmentation* [16]. In bayesian terms, such measures of complexity can be viewed as representing a form of *prior knowledge* over the connection-weights and can be viewed as network stabilizers [42]. In fact, relatively recent methods in this loss regularization approach are presented as smoothening the loss landscape of the neural network [9, 13, 15].

The third general approach is to take regularization as a structural problem of the neural network. The word *structure* or *structural*, here is used to mean an add-on component to the end-to-end feed-forward network architecture. Four popular methods in this category are: the use of convolutions leading to *convolutional layers* [27], residual connections leading to *residual layers* [19], normalization leading to *normalization layers* and weight initialization methods [4]. The last three methods can be viewed as adding useful structural noise (perturbations) to the neurons (input and hidden) at each layer.

The fourth general approach is to create an ensemble of separate neural networks, with different connection-weights and then average their predictions, also known as *model combination* [16].

The third and fourth approaches are often known to empirically reduce variance error without increasing bias error.

Almost all regularization methods have hyper-parameter(s) that need to be configured before the learning process begins. Also, several methods can be viewed as belonging to one or more of these general approaches or can be used together. Regularization effectively smoothens the non-convex loss landscape of the deep network to learn appropriate weight values within a certain range, helping the network converge faster and more stably during training [23, 34]. The overarching effect of regularization is simply then to control the model complexity by constraining the values of the connection-weight parameters to appropriate ranges, in order to improve the predictive performance of the model on the train and test datasets, such that the effective model complexity becomes lesser than the number of learnable parameters in the model.

1.2. Related works

In the following, without being exhaustive, we expand on some of the common and explicit regularization methods used in neural networks as discussed above and then branch on to a discussion on reproducibility.

Loss or Norm Regularization Compared to $\mathcal{L}_1$ norm regularization (laplace prior) which is useful for feature selection, $\mathcal{L}_2$ norm regularization (gaussian prior) is a recommended loss regularization method in deep learning. It is also known as weight-decay or Tikohonov regularization [7]. It is also a form of bayesian regularization (maximum a posteriori estimation (MAP)) [42]. Like all constraint based techniques [16], at the cost of increasing the bias error, it controls the complexity of the neural network model by the $\mathcal{L}_2$ norm constraint objective that the weight parameters should decay close to zero values. To achieve this, a constrain or lagrangian hyperparameter λ has to be set leading to a trade-off between minimizing the empirical risk and minimizing the weight norm, which is an estimate of the largeness of the individual weight elements in the connection weight tensor. Notwithstanding, one main selling-point of norm regularizers is that they are cheap to add, and often work well, as a form of strong convexity, if an appropriate constrain parameter λ is found. Also, it is easy to combine with other regularization approaches. A tighter constrain, can lead to underfitting, and a very lax constrain can lead to overfitting.

While not reducing the hyper-parameters to be tuned, from the perspective of smoother loss surfaces, there are other much more recent way of doing $\mathcal{L}_2$ norm regularization that encourages the learning process (stochastic gradient descent (SGD)) to find weights that correspond to a flatter minima instead of a sharper minima such as [9, 13, 26]

Another norm regularization method, used especially in convolutional layers, is to constrain the norm of the gram matrix of the connection weights to be orthogonal [4].

Input Normalization Normalizing the flow of batch in-

put information in the neural network, is another way to improve the generalization performance of neural networks. Arguably the most popularized technique in this method category is batch normalization [22]. However, normalizing across the batch dimension is not without its flaws leading to other preferred alternatives such as layer normalization [2], instance normalization [37], and more generally group normalization [43].

Typically, the act of normalization influences the values of the connection weights at each layer as proposed by SGD, by introducing useful noise in the layers of the network, and during optimization by inducing a smoothening effect on the loss landscape [34].

Model Combination or Averaging This technique generally involves averaging by the convex combination of the output predictions of a finite m number of separate neural networks with the same training data input [10]. Interestingly, most state-of-the-art results have been reported using ensembles [19, 22]. When the convex combination of output predictions is through a posterior weight probability distribution, this is known as *bayesian voting or averaging* [44] which is based on the bayesian notion that the connection-weight parameter is a random variable with a distribution. Although guaranteed to give better performance as the number of combined models gets larger (an infinite ensemble), this method of model combination involves the use of subjective prior probability distributions on the connection weights and is computationally more expensive both from a training and testing perspective, making them undesirable in applications where prediction time needs to be quick [28, 31, 35]. However, a single neural network model, can itself be viewed as an ensemble [16, 39], therefore other approaches that exploit this fact such as dropout and *stochastic weight averaging* (SWA) are more preferred [23] and are standard components in mainstream deep learning frameworks like Torch.

Generalization and Reproducibility We note that the use of the word *reproducibility* here, indicates consistency in a neural network's predictions at different training-runs. Variable predictions of (or explanations by) the network on the same test-data are undesirable leading to untrustworthiness in practice [18, 20, 33]. Interestingly, it turns out that generalization and reproducibility have the same basic implication of generality [17].

Statements on generalization of deep neural networks are usually benchmarked on improving prediction consistency with respect to the ground truth values of the test set (Test Accuracy) across several training runs (even if it is just by 2%) [3, 4]. A closely related study on generalization is out-of-distribution robustness, where trust measures on a neural network's confidence about its predictions can be used to evaluate (or calibrate) its predictive uncertainty [21, 24]. Such works involve the use of loss regularizers and ensembles [24, 36], and are benchmarked on test-set accuracy. Notably, out-of-distribution confidence scores tell us nothing about consistency on test predictions across multiple training runs. More recently, [3] argues that regularization by weight decay and data augmentation methods in deep neural networks fundamentally increases test accuracy performance over all classes at the cost of reduced test accuracy over certain classes, hence generalization obtained by these methods are at the cost of increased class-dependent bias error. This can be viewed as a form of irreproducibility [33].

On the other hand, compared to the statistical perspective of replication (as we imply in this work), statements on reproducibility, are mostly considered from a technical perspective of replicating experiments carried out under certain configurations [30, 32]. The summary of all this reduces to the implicit fact that reproducibility can be viewed as a regularization problem. Consequently, we would like to quantify, how much trust we can have on using one regularizer over the other one, in helping us train networks with much more consistent predictions. One practical motivation for this is that: there are many applications that care about safety, trust and consistency in diagnosis, detections and recommendations [29] that make use of feed-forward networks. In such domains, a flip in prediction from 0 to 1, could cost lives and money.

1.3. This work

In this work, we empirically study the relationship that certain regularizers have with reproducibility in deep feed-forward neural networks. Particularly (omitting data-augmentation and the combination of separate models), we will restrict our study to image recognition problems, and expect to move from fully connected network to convolutional networks. Our objectives are as follows:

1. identify certain appropriate and robust measures of reproducibility, such as prediction difference or variance with respect to test accuracy.
2. study and compare the influence of common regularizers on reproducibility in benchmark image recognition datasets, such as MNIST, FMNIST, CIFAR-10, CIFAR-100, SVHN.
3. make statements on the trade-off between increase in test accuracy and the reproducibility obtained by using such regularizers both in isolation and in combination with other regularizers.

2. Training Metrics

In supervised learning tasks, the most popular training metric used as an unbiased estimation of generalization performance is the $0/1$ test-set accuracy. Given a test dataset with samples of size N. Assume, $\mathcal{P}$ number of training runs are carried out, such that $q = 1, \ldots, \mathcal{P}$. The total

number of paired combinations of predictions $\mathcal{C}$, such that $c = 1, \ldots, \mathcal{C}$, obtained will be $\mathcal{C} = {}^{\mathcal{P}}\mathsf{C}_2$.

Test Accuracy The expectation on the similarity between any $\hat{y}^q$ and the ground-truth y for a given test-set of sample size N.

$$a_q = \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}\big[y_n = \hat{y}_n\big] = \mathbb{E}\big[y_n = \hat{y}_n\big] \tag{1}$$

Consequently, when considering the consistency of predictions across several training runs, we can modify the test-accuracy equation to the following metrics defined below:

Test Prediction Difference The expectation on the difference between any $u, v \in q$, such that $u \neq v$, for a given test-set of sample size N.

$$z_c = \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}\big[\hat{y}_n^u \neq \hat{y}_n^v\big] = \mathbb{E}\big[\hat{y}_n^u \neq \hat{y}_n^v\big] \tag{2}$$

Effective Test Accuracy The difference between the mean test accuracy and mean test prediction difference across $\mathcal{P}$ training runs.

$$\mathcal{A}_e = \frac{1}{\mathcal{P}}\sum_{q=1}^{\mathcal{P}} a_q - \frac{1}{\mathcal{C}}\sum_{c=1}^{\mathcal{C}} z_c = \mathbb{E}_{\sim q}\big[a_q\big] - \mathbb{E}_{\sim c}\big[z_c\big] \tag{3}$$

This metric tells us how much we should trust the test accuracy. If the mean test prediction difference is far higher than the mean test accuracy, then the effective test accuracy $\mathcal{A}_e$ would move close to -1. On the other hand, if the mean test prediction difference is far lower than the mean test accuracy, then $\mathcal{A}_e$ would move close to 1.

3. Experimental Setup

Our experiment design for this analysis is described, as follows, below. Given time constraints for this project, we omitted convolutional layers, group-normalization, other types of loss regularizers, and omitted the use of drop-out and stochastic depth.

This left for each dataset, a total of 36 experiments, comparing different explicit regularizers under the use of three different optimizers, leading to total maximum of 108 experiments. Our *Baseline*: 2-layer fully connected network with only SGD (no explicit regularization).

Before running the main experiments, we found learning-rate hyper-parameter settings that worked very well, for the base optimizers involved in our experimental setup. The other optimizer hyper-parameters were left at their recommended default values. For all datasets considered, the learning rate for each optimizer was set as follows: 10^{-1} for SGD, 10^{-2} for SGD-MOM, 3×10^{-4} for ADAM, except for FMNIST where 10^{-3} was used for ADAM. Also, we selected the number of epochs for each experiment to be around the epoch at which we found overfitting to occur (set to 10 for SVHN, CIFAR-10, and CIFAR-100; set to 25 for MNIST and FMNIST). The total number of runs was set to $\mathcal{P} = 5$ for each experiment. This value was selected as a trade-off between time to run experiments and number of prediction combinations $\mathcal{C} = 10$ obtainable. Batch-size was fixed to 128. The weight-decay was set to $\lambda = 10^{-4}$. A seed number of 0 was used for all experiments.

We computed a statistical T-test (Wilcoxon signed-rank test) on the prediction difference combinations across the 5 runs. to determine if there is a statistically significant difference between two independent sample groups. The null hypothesis is that: the paired z_c come from the same distribution (that is: not significantly different). The alternate hypothesis is that: the paired z_c do not come from the same distribution (that is: significantly different). A p-value of less than 0.05 would indicate that this test rejects the null hypothesis at the 5% significance level. This means that the prediction difference distribution falls within the range of what would happen 95% of the time, described by the alternate-hypothesis.

4. Main Results

The main results of our analysis, illustrated in Figures 1–5 are arranged by the dataset considered. Each of these figures is partitioned to illustrate six (6) main results. The first result (top-left) shows the statistical test significance on the prediction differences obtained in each experiment. The second result (top-middle) shows the influence of the optimizers on $\mathcal{A}_e$ for that dataset. The third result (top-right) shows the influence of depth on $\mathcal{A}_e$ for that dataset. The fourth result (bottom-left) shows the influence of the absence or presence of layer or batch normalization on $\mathcal{A}_e$ for that dataset. The fifth result (bottom-middle) shows the influence of the absence or presence of weight-decay on $\mathcal{A}_e$ for that dataset. The sixth result (bottom-right) shows the influence of the absence or presence of residual connections on $\mathcal{A}_e$ for that dataset.

Result 1. For all experiments, we found the test prediction differences across training runs to be significant for all datasets.

Result 2. The use of SGD variants like SGD-MOM and ADAM slightly help to reduce the variance in the observed effective test accuracy, possibly through increased test-accuracy. Other than that, particularly for CIFAR-100, there is no much difference in the use of the three optimizers.

Result 3. Increasing fully-connected multi-layer depth has significant effect on increasing test prediction difference. For MNIST, this effect is slightly mini-

Dataset	Optimizer	Layers	Normalization	Weight Decay	Residual Connection	$\mathcal{A}_e$
MNIST	SGD	6	LAYER	TRUE	TRUE	0.970
FMNIST	SGD-MOM	2	FALSE	FALSE	FALSE	0.818
SVHN	ADAM	2	LAYER	TRUE	FALSE	0.731
CIFAR-10	ADAM	2	BATCH	TRUE	FALSE	0.262
CIFAR-100	ADAM	2	BATCH	TRUE	FALSE	-0.276

Table 1. Best Results in Combination.

Dataset	Optimizer	Layers	Normalization	Weight Decay	Residual Connection	$\mathcal{A}_e$
MNIST	SGD	20	LAYER	FALSE	FALSE	-0.382
FMNIST	SGD-MOM	20	FALSE	ANY	FALSE	-0.9
SVHN	SGD	20	FALSE	TRUE	TRUE	-0.396
CIFAR-10	ADAM	20	FALSE	TRUE	FALSE	-0.9
CIFAR-100	ANY	20	FALSE	ANY	FALSE	-0.99

Table 2. Worst Results in Combination.

mal.

Result 4. Normalizing information flow through the network layers, across dimensions (layer or batch), most always help to significantly reduce test prediction difference, and increase test-accuracy. For SVHN, layer normalization helped best in reducing the test prediction difference, while for the other datasets, batch normalization, was the general best in reducing test prediction difference.

Result 5. Weight decay seems to have little or no significant effect on reducing the test prediction difference

Result 6. For SVHN, using residual connections in the fully-connected network, although increased test accuracy, yet had little effect on improving the effective test accuracy. For the other datasets, residual connections contributed significantly in stabilizing the effective accuracy across experiments.

From the perspective of influence on $\mathcal{A}_e$, the six results above are the standalone performance of each explicit regularizer considered. However, as illustrated in Tables 1 and 2, we might also want to know, which regularizer combinations gave the overall best and worst performance respect to $\mathcal{A}_e$.

5. Conclusion

In this paper, we focused on addressing the question: "do the use of explicit regularizers in a deep feed-forward neural network aid reproducibility (test prediction consistency or generalization) across multiple training runs?" Our empirical investigation across multiple image recognition datasets showed that compared to the network structure and the nature of the dataset, explicit regularizers have little effect on stabilizing test prediction differences. We also found that the predictions of fully-connected networks are seldom reproducible across training runs. While limitations on time and computing resources constrained a comprehensive analysis with larger image datasets and network architectures, we believe the analysis in this work is insightful. It would be interesting to see what answers similar analysis under such unconstrained settings would bring forth.

Technical Reproducibility Our source-code can be found at https://github.com/apurva94/Reproducibility, and our results are publicly available at https://drive.google.com/drive/folders/1VoKYXQwUDhXdoW3XaD8kNzdnvxVGSmQL?usp=sharing.

References

[1] Martin Anthony, Peter L. Bartlett, and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*, volume 9. Cambridge University Press, Cambridge, 1999. 1

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv:1607.06450 [cs, stat]*, July 2016. 3

[3] Randall Balestriero, Leon Bottou, and Yann LeCun. The Effects of Regularization and Data Augmentation are Class Dependent. *arXiv:2204.03632 [cs, stat]*, Apr. 2022. 2, 3

[4] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can We Gain More from Orthogonality Regularizations in Training Deep Networks? In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2, 3

[5] Peter Bartlett. For valid generalization the size of the weights is more important than the size of the network. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996. 2

[6] Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002. 1

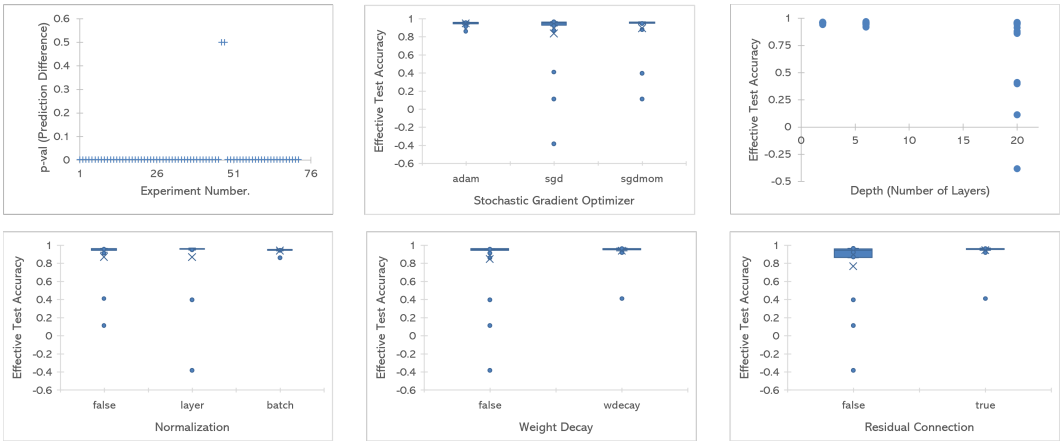

Figure 1. MNIST.

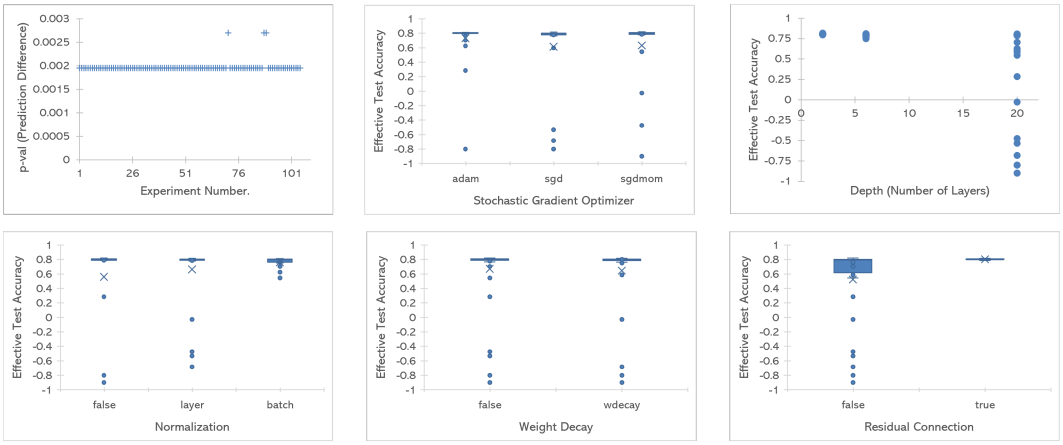

Figure 2. FMNIST.

[7] Chris M. Bishop. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 7(1):108–116, Jan. 1995. 1, 2

[8] Ovidiu Calin. Cost Functions. In *Deep Learning Architectures: A Mathematical Approach*, Springer Series in the Data Sciences, pages 41–68. Springer International Publishing, Cham, 2020. 2

[9] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. Nov. 2016. 2

[10] Thomas G. Dietterich. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 1–15, Berlin, Heidelberg, 2000. Springer. 3

[11] Ke-Lin Du and M. N. S. Swamy. Elements of Computational Learning Theory. In *Neural Networks and Statistical Learning*, pages 65–79. Springer, London, 2019. 1

[12] Ke-Lin Du and M. N. S. Swamy. Fundamentals of Machine Learning. In *Neural Networks and Statistical Learning*, pages 21–63. Springer, London, 2019. 1

[13] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware Minimization for Efficiently Improving Generalization. In *International Conference on Learning Representations*, 2020. 2

[14] Yoav Freund, Yishay Mansour, and Robert E Schapire. Generalization bounds for averaged classifiers. *The annals of statistics*, 32(4):1698–1722, 2004. 1

[15] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, Cambridge, Massachusetts, illustrated edition edition, Nov. 2016. 2, 3

[17] Odd Erik Gundersen. The fundamental principles of reproducibility. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2197):20200210, May 2021. 3

[18] Matthew Hartley and Tjelvar S.G. Olsson. dtoolAI: Reproducibility for Deep Learning. *Patterns*, 1(5):100073, Aug.

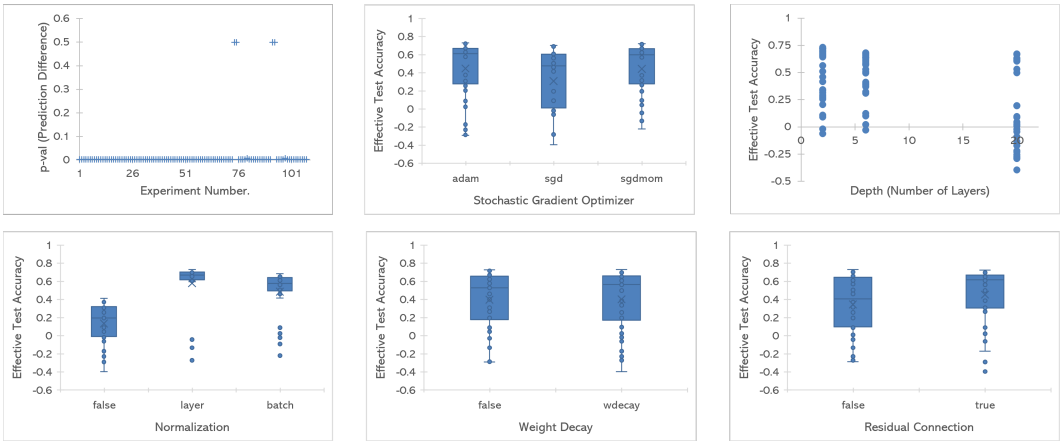

Figure 3. SVHN.

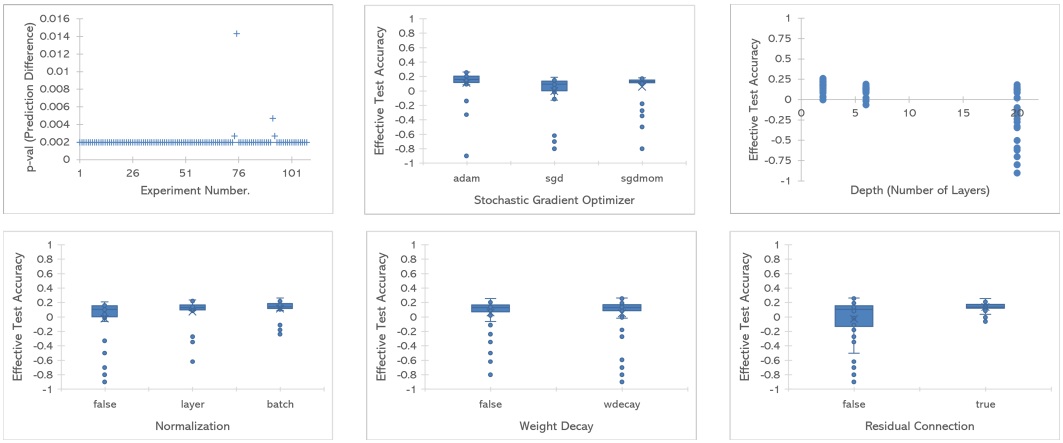

Figure 4. CIFAR10.

2020. 3

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 2, 3

[20] Benjamin J. Heil, Michael M. Hoffman, Florian Markowetz, Su-In Lee, Casey S. Greene, and Stephanie C. Hicks. Reproducibility standards for machine learning in the life sciences. *Nature Methods*, 18(10):1132–1135, Oct. 2021. 3

[21] Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations*, 2018. 3

[22] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456. PMLR, June 2015. 3

[23] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization: 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018. *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885, 2018. 2, 3

[24] Heinrich Jiang, Been Kim, Melody Y. Guan, and Maya Gupta. To trust or not to trust a classifier. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 5546–5557, Red Hook, NY, USA, Dec. 2018. Curran Associates Inc. 3

[25] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Regularization Techniques for Learning with Matrices. *Journal of Machine Learning Research*, 13(59):1865–1890, 2012. 2

[26] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks. In *Proceedings of the 38th International Conference on Machine Learning*, pages 5905–5914. PMLR, July 2021. 2

[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov. 1998. 2

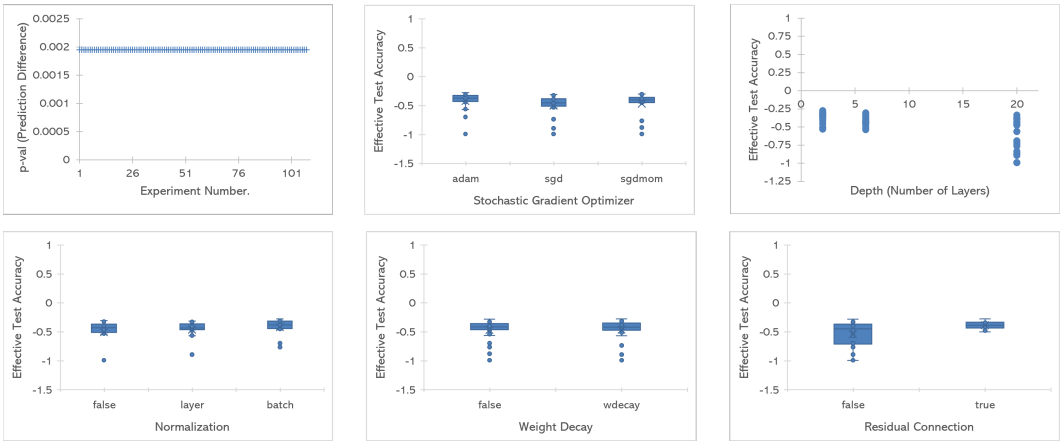

Figure 5. CIFAR100.

[28] Lucas Liebenwein. *Efficient Deep Learning: From Theory to Practice*. Thesis, Massachusetts Institute of Technology, Sept. 2021. 3

[29] Tianming Liu, Eliot Siegel, and Dinggang Shen. Deep Learning and Medical Image Analysis for COVID-19 Diagnosis and Prediction. *Annual Review of Biomedical Engineering*, 24(1):annurev–bioeng–110220–012203, June 2022. 3

[30] Matthew B. A. McDermott, Shirly Wang, Nikki Marinsek, Rajesh Ranganath, Luca Foschini, and Marzyeh Ghassemi. Reproducibility in machine learning for health research: Still a ways to go. *Science Translational Medicine*, 13(586):eabb1655, Mar. 2021. 3

[31] Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction*. The MIT Press, Feb. 2022. 3

[32] Gabriele Piantadosi, Stefano Marrone, and Carlo Sansone. On Reproducibility of Deep Convolutional Neural Networks Approaches. In Bertrand Kerautret, Miguel Colom, Daniel Lopresti, Pascal Monasse, and Hugues Talbot, editors, *Reproducible Research in Pattern Recognition*, Lecture Notes in Computer Science, pages 104–109, Cham, 2019. Springer International Publishing. 3

[33] Félix Renard, Soulaimane Guedria, Noel De Palma, and Nicolas Vuillerme. Variability and reproducibility in deep learning for medical image segmentation. *Scientific Reports*, 10(1):13724, Aug. 2020. 3

[34] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How Does Batch Normalization Help Optimization? In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2, 3

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 3

[36] Christian Tomani and Florian Buettner. Towards Trustworthy Predictions from Deep Neural Networks with Fast Adversarial Calibration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):9886–9896, May 2021. 3

[37] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 3

[38] Vladimir N. Vapnik. Constructing Learning Algorithms. In Vladimir N. Vapnik, editor, *The Nature of Statistical Learning Theory*, pages 119–166. Springer, New York, NY, 1995. 1

[39] Vladimir N. Vapnik. Controlling the Generalization Ability of Learning Processes. In Vladimir N. Vapnik, editor, *The Nature of Statistical Learning Theory*, pages 89–118. Springer, New York, NY, 1995. 2, 3

[40] Vladimir N. Vapnik. Setting of the Learning Problem. In *The Nature of Statistical Learning Theory*, pages 15–32. Springer, New York, NY, 1995. 1, 2

[41] Stefan Wager, Sida Wang, and Percy S Liang. Dropout Training as Adaptive Regularization. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. 2

[42] Peter M. Williams. Bayesian Regularization and Pruning Using a Laplace Prior. *Neural Computation*, 7(1):117–143, Jan. 1995. 2

[43] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. 3

[44] Yuling Yao, Aki Vehtari, Daniel Simpson, and Andrew Gelman. Using Stacking to Average Bayesian Predictive Distributions (with Discussion). *Bayesian Analysis*, 13(3):917–1007, Sept. 2018. 3