More is Not Always Better: Exploring Early Repair of DNNs

Andrei Mancu
Technical University of Munich
Munich, Germany
andrei.mancu@tum.de

Thomas Laurent
National Institute of Informatics, JSPS International Research Fellow
Tokyo, Japan
thomas-laurent@nii.ac.jp

Franz Rieger
Max Planck Institute for Biological Intelligence
Munich, Germany
franz.rieger@bi.mpg.de

Paolo Arcaini
National Institute of Informatics
Tokyo, Japan
arcaini@nii.ac.jp

Fuyuki Ishikawa
National Institute of Informatics
Tokyo, Japan
f-ishikawa@nii.ac.jp

Daniel Rueckert
Technical University of Munich
AI in Medicine
Munich, Germany
daniel.rueckert@tum.de

ABSTRACT

DNN repair is an effective technique applied after training to enhance the class-specific accuracy of classifier models, where a low failure rate is required on specific classes. The repair methods introduced in recent studies assume that they are applied to fully trained models. In this paper, we argue that this could not always be the best choice. We analyse the performance of DNN models under various training times and repair combinations. Through meticulously designed experiments on two real-world datasets and a carefully curated assessment score, we show that applying DNN repair earlier in the training process, and not only at its end, can be beneficial. Thus, we encourage the research community to consider the idea of *when* to apply DNN repair in the model development.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Software and its engineering** → **Software testing and debugging**.

KEYWORDS

Deep Neural Networks, DNN Repair, DNN Training, Safety-critical

ACM Reference Format:
Andrei Mancu, Thomas Laurent, Franz Rieger, Paolo Arcaini, Fuyuki Ishikawa, and Daniel Rueckert. 2024. More is Not Always Better: Exploring Early Repair of DNNs. In *2024 IEEE/ACM International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest '24), April 20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION AND BACKGROUND

Deep Neural Networks (DNNs) have entered every field of action, from creative ones such as film-making and game development, to safety-critical ones like medical diagnosis and autonomous driving. Especially in the latter case, it is crucial to have low error rates to avoid human harm. Therefore, DNNs have become an important object of study for the field of Software Engineering [10, 14, 21].

Safety-critical systems often rely on DNN classifiers. These systems, in addition to the overall accuracy, put additional *class-specific requirements* on the DNNs; for example, the classification accuracy for a specific critical class should be higher than a given threshold.

To achieve these class-specific requirements, some *DNN repair* techniques [6, 7, 15, 16, 18–20] have been proposed, adapting fault localisation and automated program repair techniques originally developed for classic software. These techniques try to identify faulty DNN components (e.g., *weights*) using a localisation technique [1–4, 8, 16] and fix them to improve the accuracy of some specific classes, while still preventing degradation of the performance of the other classes. Specifically, DNN repair targeting weights [6, 7, 15, 16] consists of two steps: *Fault Localisation* (FL), where the weights responsible for the false predictions are identified, and *weight optimisation*, where the identified weights are modified to fix the false predictions while still preserving the overall model performance. These two steps rely on the predictions of the model on a *repair dataset*, different from the training, validation, and test sets.

These techniques have been applied on converged models. However, exploration into repairing non-converged models has been overlooked. Our assumption is that sometimes repairing non-converged models could allow to obtain a better final repaired model.

To validate this idea, we analyse how repair performance changes with training progress by repairing the same model at different training stages. Specifically, we repair different checkpoints, taken at different training epochs by targeting the misclassifications of a specific class. This process is visualised in Fig. 1, where we take a model at three different stages of training and apply repair to it. We obtain different final models (models 1, 2, and 3), having the same model architecture, but different sets of weights.

A model can be evaluated using a *quality* metric, that is defined based on the model's performance requirements. For safety-critical systems, this metric should consider the overall performance of the model, as well as the performance on the class under repair. Sect. 2 describes the metric used in this work. Repairing the model at different points in its training results in different final models (e.g., models 1, 2, and 3 in Fig. 1) that exhibit different quality. Fig. 2 illustrates the quality of the repaired model for two different repair problems (e.g., different model architectures and classes to repair).

DeepTest '24, April 20, 2024, Lisbon, Portugal

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

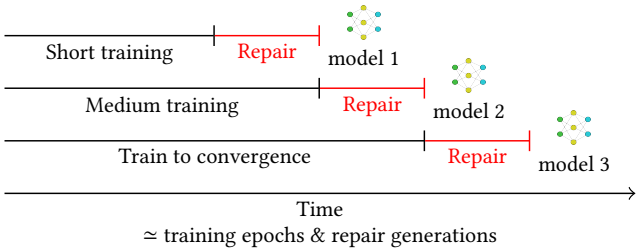
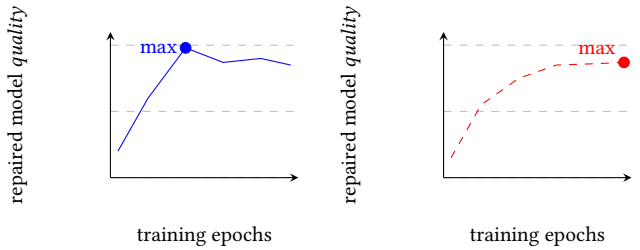


Figure 1: Repair applied to models obtained at different training epochs (non-converged and converged models)



(a) Pre-convergence pattern **(b) Post-convergence pattern**

Figure 2: Patterns of best train-repair combination

Specifically, it reports the quality of the model obtained by repairing the original model at different training stages (i.e., stopping training at different epochs). Each plot also shows the epoch from which repairing the model obtains the best final quality (*max*). By looking at the two plots, we realise that two *patterns* can occur:

***pre-convergence pattern*:** the best repaired model is obtained from a non-converged model (Fig. 2a);

***post-convergence pattern*:** the best repaired model is obtained from the fully-converged model (Fig. 2b); state-of-the-art repair approaches have assumed this pattern in their experiments.

Insights on when to start repair could provide a significant advantage in the development of safety-critical systems, where even small percentages of performance gain matter. In this paper, we investigate how the *quality* of the repaired model changes over training time, and how often the two patterns occur.

2 EXPERIMENT DESIGN

To explore how the training time can influence repair results, we utilised a systematic approach for repairing models at varying stages of training. Specifically, given a dataset and a model architecture, we trained the model until the validation loss did not improve for 5 epochs. The epoch at which the lowest validation loss occurred was defined as *convergence*, and, in the following, any reference to a converged model will always mean the checkpoint from that epoch. This process also involved saving checkpoints every five epochs (i.e., non-converged models) and at convergence. We repair all the saved checkpoints, targeting the misclassification of inputs in the classes for which the model performed the worst across all epochs. All resources to reproduce these experiments and detailed analyses (including all results) are made available online [9].

Training Datasets and Models. We conducted experiments using two well-known datasets in two different application domains. *German Traffic Sign Recognition Benchmark* (GTSRB) [17] consists of traffic sign images that correspond to 43 classes. It contains 32x32 pixels RGB images, from which 39,209 are train images and 12,630 are test images that constitute a highly imbalanced dataset. We split the original train data, using 70% of images to train the model and 30% as the repair set. During training, 20% of our training set is used as validation data.

RSNA Pneumonia Detection Challenge [12] contains chest X-rays that show or do not show signs of pneumonia. The dataset consists of 26,684 256x256 pixels greyscale images divided into two classes with a class imbalance of approximately 2:1 (pneumonia positive being underrepresented). We use 85% of the data as training set, 10% of the images as test set, and the remaining 5% as repair set. During training, 20% of the training data is used as validation data.

For GTSRB, we used VGG16 pre-trained on ImageNet, and for RSNA we used DenseNet121. We trained the models with a categorical cross-entropy loss with equal class weights and no weight decay, using an SGD optimiser with an initial learning rate of 10^{-2} and momentum of 0.9 and exponential learning rate decay of 10^{-1} every 10 epochs. Both models were trained with a batch size of 128.

Repair. We use Arachne [15] as a DNN repair method, with default hyper-parameters. The FL phase only targets the last layer of the DNN and the optimisation phase uses PSO with 100 particles, velocity 4.1, and 50 iterations. For both datasets, the repair process targets a single type of fault (misclassifications of inputs in a target class) at a time. Namely, for GTSRB we performed repair targeting classes 20, 27, and 30; and for RSNA we performed repair targeting class 1 (the pneumonia-positive class). The repair method used 200 samples that the model classifies correctly from all classes and all samples the model misclassified from the target class.

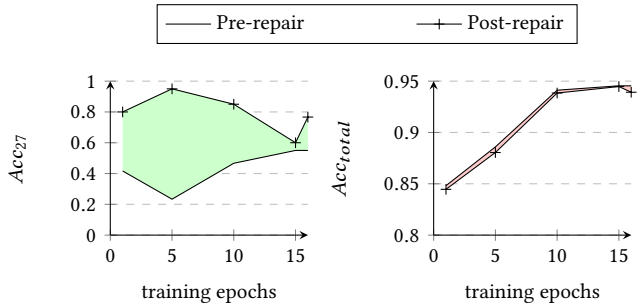
To account for the stochasticity of the training and repair processes, we trained the same model five times. Then, for each version of the model, we applied repair on the classes mentioned before.

Quality metric. Various metrics are typically employed to measure the quality of classification systems. In this study, we have chosen accuracy as our preferred metric. As DNN repair methods are applied in cases where a class is of particular importance (i.e., the class targeted by repair), the overall accuracy (Acc_{total}) of the model is not a sufficient quality indicator, and the particular accuracy (Acc_{cls}) of the targeted class must also be taken into account. Consequently, the quality metric applied to the repaired model must consider both Acc_{total} and Acc_{cls}. This consideration should be balanced by considering the requirements stipulated by the system's stakeholders. To achieve this, we used a parameterized metric (parameter α) mimicking a weighted accuracy that can be tuned according to stakeholder requirements. The metric is computed as:

$$Qual_{\alpha,cls}(M,D) = \alpha * Acc_{cls}(M,D) + (1-\alpha) * Acc_{total}(M,D) \quad (1)$$

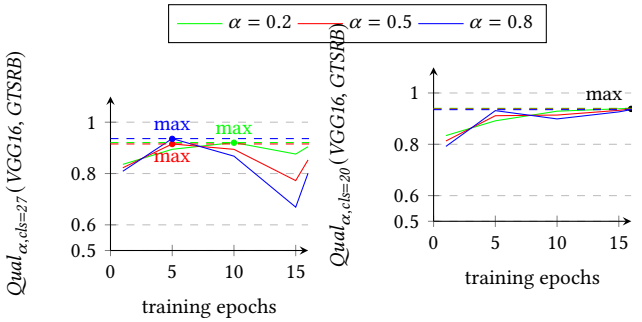
where $\alpha \in [0,1]$ is a parameter that balances out the class accuracy Acc_{cls} and the total accuracy Acc_{total} of model M over dataset D.

Pattern classification. In the experiments, given a training run for a model M and a dataset D, we classified it into either the *post-convergence pattern* or the *pre-convergence pattern* (see Sect. 1 and



(a) Target class accuracy Acc_{cls} **(b) Total accuracy Acc_{total}**

Figure 3: Pre- and post-repair test accuracies for a run in the GTSRB dataset and VGG16 model when repair is applied after different training epochs targeting class 27



(a) Class 27 **(b) Class 20**

Figure 4: Repaired model quality over time showing pre-convergence pattern (a) and post-convergence pattern (b)

Fig. 2), depending on whether the best repaired model quality is obtained by repairing the converged model (last epoch as in Fig. 2b) or a non-converged one (before last epoch as in Fig. 2a).

3 PRELIMINARY EXPERIMENTAL RESULTS

This section provides evidence of the existence of the two above-mentioned patterns through a series of examples. All results are available in the companion GitHub repository [9].

We begin with an example demonstrating how the components Acc_{cls} and Acc_{total}, that contribute to the computation of the repair quality metric $Qual_{\alpha,cls}$, are improved by repair at different training epochs. Fig. 3 shows Acc_{cls} and Acc_{total} of VGG16 before repair (line —) and after repair (line -+-) on class 27 over training epochs for one run. In Fig. 3a, notice the significant increase of Acc_{cls} after repair in early training epochs. This trend is visible across all experiments, regardless of the class under repair, dataset, or model. This hints at why some repairs on non-converged models might yield superior results, possibly as the repair algorithm may have more freedom to look for the best model that fits the target function without breaking many correct predictions (as the weights are still untrained).

However, this analysis alone is not enough to determine whether the performance of an early repair is better. As explained in Sect. 2, the *quality* metric $Qual_{\alpha,cls}$ can be tuned by stakeholders using hyperparameter α to specify their preference between Acc_{total} and Acc_{cls}. We want to observe whether using different values of α leads to different patterns. We consider three cases, $\alpha = 0.2$, $\alpha = 0.5$ and $\alpha = 0.8$, meaning that more or less importance is put on Acc_{total} or Acc_{cls}. In Fig. 4 we can see how $Qual_{\alpha,cls}$ changes over training time for the three values of α. The two sub-figures present the results for one run of the repair of classes 27 and 20 of GTSRB. These two examples are used to describe the occurrences of the two patterns.

Pre-convergence pattern. As shown in Fig. 4a, the optimal performance after repair occurs at either epoch 5 or 10 (i.e., prior to epoch 16 where the model converges), depending on the value of α. These peaks in the repair quality metric **before** training convergence characterise the *pre-convergence pattern*, suggesting the model does not need to be fully trained to achieve superior repair quality.

Post-convergence pattern. The *post-convergence pattern* is distinguished by a repair quality peak **at** training convergence. This indicates that prolonged training prior to repair can enhance the resulting repair quality. This is the case of Fig. 4b, in which, for any value of α, the optimal performance after repair is obtained using the converged model.

Table 1: Number of experiment runs following each pattern for both datasets and different α values

Dataset	RSNA			GTSRB		
α	0.2	0.5	0.8	0.2	0.5	0.8
pre-convergence pattern	3	2	4	9	9	9
post-convergence pattern	2	3	1	6	6	6

Frequency of patterns. We count the pre- and post-convergence patterns across the benchmarks described in Sect. 2. Table 1 shows the number of runs following each pattern for each value of α used.

The RSNA results reveal that the repair performance pattern depends on the repair requirements (choice of α), implying that repair timing can be impacted by the system requirements. For instance, prioritising the repair class ($\alpha = 0.8$), the DNN system would benefit more by initiating repair earlier. Instead, if we emphasise the class accuracy less ($\alpha = 0.2$, $\alpha = 0.5$), it is less definitive whether we should stop the training earlier or not for the best repair outcome.

For GTSRB, the distribution of the patterns is stable across different requirements settings, with a predominance of the pre-convergence pattern. However, when we look at more detailed results of GTSRB as reported in the online repository [9], we observe that classes 20 and 30, which represent signs of a dangerous right turn and of a dangerous snowy section, exhibit different patterns in different runs for the same values of alpha. This re-emphasizes that repairing the converged model is not always effective.

We also observe that class 27 of GTSRB is predominantly in the pre-converge pattern, while other classes are spread among the two patterns. Class 27, which represents the attention sign for pedestrians, is an important yet challenging class that is difficult to optimise for and, therefore, it is beneficial to start repair earlier. For easier classes, the repair process's effectiveness is not influenced by the epoch at which it begins.

These results highlight the intricate relation between training time and the repaired model's quality. Factors like requirements (symbolised by α in the metric $Qual_{\alpha,cls}$), or problem definition (dataset, architecture, repair target class) appear to influence whether the best repaired model is obtainable early on (pre-convergence pattern) or post full training (post-convergence pattern).

4 CONCLUSION AND FUTURE WORK

The above-mentioned results clearly show that the training time influences performance of the repair applied to the trained model. This is particularly important as, for safety-critical systems, even small improvements have high impact. Thus, establishing an online method to determine a model's repair readiness during training could enhance quality. Although this is not trivial, it opens the door to new avenues for research, described in the following.

Different repair methods. In this work, we validated our intuition using the Arachne repair approach [15]. However, other repair methods have been proposed in the literature [7, 11, 16, 18–20] that use either different techniques in their FL step, or other optimisation methods (e.g., producing multiple repaired models); we plan to investigate whether the pre-convergence pattern could also occur with these approaches. Moreover, different from the methods in our work, a new class of methods try to repair the model at the architecture level [5, 13]. We will investigate whether the problem of *when* to do repair is also relevant in the context.

Multiple classes. Repair methods can also target multiple faults (e.g., classification of different classes) at once. In such a setting, it could be that the repair of some classes require to have a non-converged model, while the repair of other classes work well on the converged model. As future research, we plan to investigate which types of patterns occur for this type of repair.

Online switching to repair. Our investigation has shown that, in some cases, stopping the training before convergence allows to obtain a better final repaired model quality. Thus, an *online method* to determine if a model is ready for repair during training would be highly desirable. This would allow to not only obtain the best achievable repaired model quality, but also to cut the cost of training. In order to do this, we plan to conduct studies to understand which are the characteristics of the model during training, that make it suitable for repair. Indeed, our current investigation has shown that the pre-convergence pattern exists, but does not explain *why* some model is better repairable when it is non-converged.

Datasets. Our investigation has been performed on two datasets. A more systematic exploration of the occurrence of the two patterns, using different datasets is needed. More specifically, safety-critical datasets that need high performance to be used in real-world applications. This targets especially autonomous driving datasets and medical datasets, which are prone to significant class imbalances or anomalies, and present critical classes whose correct classification is particularly important, in addition to the total accuracy.

ACKNOWLEDGMENTS

P. Arcaini and F. Ishikawa are supported by Engineerable AI Techniques for Practical Applications of High-Quality Machine Learning-based Systems Project (Grant Number JPMJMI20B8), JST-Mirai.

REFERENCES

[1] Jialun Cao, Meiziniu Li, Xiao Chen, Ming Wen, Yongqiang Tian, Bo Wu, and Shing-Chi Cheung. 2022. DeepFD: Automated Fault Diagnosis and Localization for Deep Learning Programs. In *Proceedings of the 44th International Conference on Software Engineering (ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 573–585.

[2] Matias Duran, Xiao-Yi Zhang, Paolo Arcaini, and Fuyuki Ishikawa. 2021. What to Blame? On the Granularity of Fault Localization for Deep Neural Networks. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. 264–275.

[3] Hasan Ferit Eniser, Simos Gerasimou, and Alper Sen. 2019. DeepFault: Fault Localization for Deep Neural Networks. In *Fundamental Approaches to Software Engineering*. Springer International Publishing, Cham, 171–191.

[4] Ali Ghanbari, Deepak-George Thomas, Muhammad Arbab Arshad, and Hridesh Rajan. 2023. Mutation-based Fault Localization of Deep Neural Networks. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 1301–1313.

[5] Jinhan Kim, Nargiz Humbatova, Gunel Jahangirova, Paolo Tonella, and Shin Yoo. 2023. Repairing DNN Architecture: Are We There Yet?. In *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*. 234–245.

[6] Davide Li Calsi, Matias Duran, Thomas Laurent, Xiao-Yi Zhang, Paolo Arcaini, and Fuyuki Ishikawa. 2023. Adaptive Search-Based Repair of Deep Neural Networks. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Lisbon, Portugal) *(GECCO '23)*. Association for Computing Machinery, New York, NY, USA, 1527–1536.

[7] Davide Li Calsi, Matias Duran, Xiao-Yi Zhang, Paolo Arcaini, and Fuyuki Ishikawa. 2023. Distributed Repair of Deep Neural Networks. In *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*. 83–94.

[8] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, and Ananth Grama. 2018. MODE: Automated Neural Network Model Debugging via State Differential Analysis and Input Selection. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018)*. Association for Computing Machinery, New York, NY, USA, 175–186.

[9] Andrei Mancu, Thomas Laurent, Paolo Arcaini, Franz Rieger, Fuyuki Ishikawa, and Daniel Rueckert. 2024. https://github.com/jst-qaml/early_DNN_Repair_DeepTest2024.

[10] Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, and Stefan Wagner. 2022. Software engineering for AI-based systems: a survey. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 2 (2022), 1–59.

[11] Takao Nakagawa, Susumu Tokumoto, Shogo Tokui, and Fuyuki Ishikawa. 2023. An Experience Report on Regression-Free Repair of Deep Neural Network Model. In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 778–782.

[12] Radiological Society of North America. 2018. *RSNA Pneumonia Detection Challenge*.

[13] Hua Qi, Zhijie Wang, Qing Guo, Jianlang Chen, Felix Juefei-Xu, Fuyuan Zhang, Lei Ma, and Jianjun Zhao. 2023. ArchRepair: Block-Level Architecture-Oriented Repairing for Deep Neural Networks. *ACM Trans. Softw. Eng. Methodol.* 32, 5, Article 129 (jul 2023), 31 pages.

[14] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. 2020. Testing Machine Learning Based Systems: A Systematic Mapping. *Empirical Softw. Engg.* 25, 6 (nov 2020), 5193–5254.

[15] Jeongju Sohn, Sungmin Kang, and Shin Yoo. 2019. Search Based Repair of Deep Neural Networks. *CoRR* abs/1912.12463 (2019). arXiv:1912.12463

[16] Jeongju Sohn, Sungmin Kang, and Shin Yoo. 2023. Arachne: Search-Based Repair of Deep Neural Networks. *ACM Trans. Softw. Eng. Methodol.* 32, 4, Article 85 (may 2023), 26 pages.

[17] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* 0 (2012), –.

[18] S. Tokui, S. Tokumoto, A. Yoshii, F. Ishikawa, T. Nakagawa, K. Munakata, and S. Kikuchi. 2022. NeuRecover: Regression-Controlled Repair of Deep Neural Networks with Training History. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE Computer Society, Los Alamitos, CA, USA, 1111–1121.

[19] Bing Yu, Hua Qi, Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Lei Ma, and Jianjun Zhao. 2022. DeepRepair: Style-Guided Repairing for Deep Neural Networks in the Real-World Operational Environment. *IEEE Transactions on Reliability* 71, 4 (2022), 1401–1416.

[20] Hao Zhang and W. K. Chan. 2020. Apricot: A Weight-Adaptation Approach to Fixing Deep Learning Models. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE '19)*. IEEE Press, 376–387.

[21] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2022. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Trans. Softw. Eng.* 48, 1 (jan 2022), 1–36.