Discrepancies among Pre-trained Deep Neural Networks: A New Threat to Model Zoo Reliability

Diego Montes
Purdue University, USA
montes10@purdue.edu

Pongpatapee Peerapatanapokin
Purdue University, USA
ppeerapa@purdue.edu

Jeff Schultz
Purdue University, USA
schul203@purdue.edu

Chengjun Guo
Purdue University, USA
guo456@purdue.edu

Wenxin Jiang
Purdue University, USA
jiang784@purdue.edu

James C. Davis
Purdue University, USA
davisjam@purdue.edu

ABSTRACT

Training deep neural networks (DNNs) takes significant time and resources. A practice for expedited deployment is to use pre-trained deep neural networks (PTNNs), often from model zoos—collections of PTNNs; yet, the reliability of model zoos remains unexamined. In the absence of an industry standard for the implementation and performance of PTNNs, engineers cannot confidently incorporate them into production systems. As a first step, discovering potential discrepancies between PTNNs across model zoos would reveal a threat to model zoo reliability. Prior works indicated existing variances in deep learning systems in terms of accuracy. However, broader measures of reliability for PTNNs from model zoos are unexplored. This work measures notable discrepancies between accuracy, latency, and architecture of 36 PTNNs across four model zoos. Among the top 10 discrepancies, we find differences of 1.23%–2.62% in accuracy and 9%–131% in latency. We also find mismatches in architecture for well-known DNN architectures (*e.g.,* ResNet and AlexNet). Our findings call for future works on empirical validation, automated tools for measurement, and best practices for implementation.

CCS CONCEPTS

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

KEYWORDS

Neural networks, Model zoos, Software reuse, Empirical software engineering

ACM Reference Format:
Diego Montes, Pongpatapee Peerapatanapokin, Jeff Schultz, Chengjun Guo, Wenxin Jiang, and James C. Davis. 2022. Discrepancies among Pre-trained Deep Neural Networks: A New Threat to Model Zoo Reliability. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3540250.3560881

1 INTRODUCTION

With the growing energy consumption of training DNNs [26], taking advantage of the re-usability of PTNNs can significantly reduce the costs of training [13]. In particular, transfer learning can result in shorter training times and higher asymptotic accuracies compared to other weight initialization methods [22, 36]. This kind of technique accelerates model reuse and development. The history of PTNNs and their impact on the development of artificial intelligence has been extensively documented [13, 25]. As such, collections of PTNNs have been created, referred to as *model zoos*. Notably, maintainers of popular machine learning frameworks, such as TensorFlow [2], maintain corresponding model zoos developed with their framework, such as the TensorFlow Model Garden [38].

There are many model zoos [1, 18, 23, 38] and an expanding use of PTNNs in production systems [13]. Past work has emphasized the difficulties in adopting software engineering practices in machine learning, and specifically, the challenges with reproducing machine learning research papers [4, 17]. These reproducibility issues may affect PTNNs, leading to variations across model zoos [28]. Disparities in the accuracy, latency, or architecture of a PTNN could negatively affect a deep learning system, threatening PTNNs' reuse potential. Consider a model zoo that has an incorrect implementation of a well-known DNN architecture, which has increased its latency significantly. If an engineer were to use the PTNN from this zoo, they would unknowingly be receiving a lower quality PTNN than they might otherwise have from a different model zoo. The engineer's effort to enable a quick turnaround time with a PTNN would have become harmful. Discovering discrepancies would shine a light on the reliability of model zoos.

To explore the reliability of model zoos, we performed a measurement study to identify discrepancies among 36 image classification PTNN architectures across four model zoos: *TensorFlow Model Garden* (TFMG) [38], *ONNX Model Zoo* (ONNX) [1], *Torchvision Models* (Torchvision) [23], and *Keras Applications* (Keras) [18]. The PTNNs were measured along three dimensions: accuracy, latency, and architecture. We find the differences in accuracies on *ILSVRC-2012-CLS* dataset (ImageNet) can be as large as 2.62% [10].[1] Similarly, over 20% of the PTNNs measured had latency differences (FLOPs) of 10% or more when comparing PTNNs of the same name across the model zoos. Lastly, we discover architectural differences in several PTNNs, including implementations of *AlexNet* and *ResNet V2*. We conclude with an agenda for future research on further empirical validation, automated tools for measurement, and best practices for implementing model zoo PTNNs.

[1]The *ILSVRC-2012-CLS* image dataset has 50,000 validation images. A 1% accuracy difference is equivalent to 500 images.

ESEC/FSE '22, November 14–18, 2022, Singapore, Singapore
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9413-0/22/11.
https://doi.org/10.1145/3540250.3560881

2 BACKGROUND AND RELATED WORK

PTNNs are applied in a wide variety of domains [13]. With the demand for engineers far exceeding supply [32], companies are looking for best practices that can boost the productivity of their engineers. Major companies (*e.g.,* Google and Microsoft) have shared best practices on machine learning development and informed future directions on model reuse [3, 8]. A case study from SAP indicates possible compatibility, portability, and scalability challenges in machine learning model deployment, which may affect their performance [30]. There have been many efforts to improve the quality of model zoos. For example, IBM has developed a tool to extract model metadata [37] to support better model management. Banna *et al.* promote best practices for reproducing and publishing high-quality PTNNs [4]. However, the reliability of model zoos has not been validated by prior works.

The ability to replicate the accuracy of a DNN in identical training environments is hindered by non-deterministic factors. Accuracy differences of up to 10.8%, stemming purely from non-determinism, have been reported with popular DNN architectures [28]. Closely related, research has investigated and benchmarked the performance variances tied to deep learning frameworks [21, 33]. This variability threatens the reliability of new deep learning techniques. As such, automated variance testing [27] has been proposed to assure the validity of these comparisons. However, PTNNs in model zoos may also suffer from varying architectural implementations, affecting more than just accuracy. Our work measures the disparities in PTNNs across different model zoos as opposed to attempting to improve the standard in just one [4]. Our results enlighten future works validating the quality and promoting the standardization of model zoos.

3 METHODOLOGY

We perform a measurement study to assess our problem statement: whether discrepancies exist between the accuracy, latency, and architecture of PTNNs across different model zoos.

3.1 Subjects

A *model zoo* is a collection of PTNNs for various tasks. We carry out a selection process for four model zoos. Our selection criteria included the model zoo being maintained alongside a machine learning framework: this increases the likelihood of the model zoo being actively maintained. Furthermore, to ensure the popularity of the model zoo, the zoo must have a public GitHub repository with at least three thousand stars [7]. Using GitHub search[2] to identify potential model zoo candidates, 11 model zoos were selected that met the criteria.[3] The PTNNs within the 11 model zoos were categorized into deep learning tasks, including image classification, object detection, and natural language processing. We focused on image classification models because it is the most common type in 8 of the 11 model zoos.

A PTNN availability analysis was done on the candidate model zoos to assess how many model zoos offered the same image classification PTNN architectures. Based on the largest shared availability, we chose *TensorFlow Model Garden*, *ONNX Model Zoo*, *Torchvision Models*, and *Keras Applications*. Within these model zoos, we selected all the image classification PTNN architectures that were present in at least two of the four model zoos, yielding 36 PTNN architectures. The selected PTNNs are either directly downloadable from the model zoos' GitHub repositories or can be pulled using the model zoos' APIs.

[2]https://github.com/search

[3]The 11 identified potential model zoos are as follows: TensorFlow Model Garden, ONNX Model Zoo, Torchvision Models, Keras Applications, TensorFlow Model Hub, PyTorch Model Zoo, MXNet Model Zoo, Gluon Model Zoo, Deeplearning4j Model Zoo, Caffe Model Zoo, and OpenVINO Model Zoo.

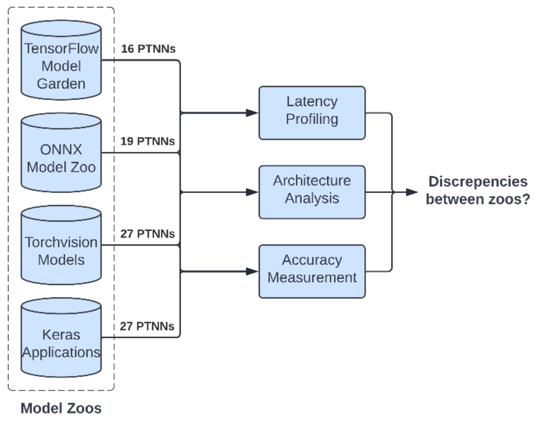

Figure 1: Overview of the measurement process. We gather PTNNs from the model zoos with the same name, perform measurements on each PTNN, and compare for discrepancies.

3.2 Evaluation Metrics

Accuracy. Image classification DNNs' effectiveness is measured in accuracy, which is a critical component of a PTNN. We are measuring discrepancies between the claims of model zoos as opposed to verifying them. *Top-1 accuracy* is the conventional accuracy where model prediction must exactly match the expected label, while *top-5 accuracy* measures the fraction of images where any of the top five predicted labels matches the target label [9, 10]. 35 image classification PTNN architectures reported top-1 ImageNet classification accuracies, meanwhile only 32 reported top-5 ImageNet classification accuracies.

Latency. The latency of a DNN is a key factor that engineers consider [11]. For example, *MobileNet* is a DNN image classification architecture that prioritizes low latency on mobile and embedded systems [16]. We used open-source tools [5, 34, 35] to measure the latency by counting the floating point operations (FLOPs) [6]. FLOPs are framework and hardware-agnostic, allowing for unbiased comparisons.

Architecture. PTNNs are trained weights based on research papers that propose DNN architectures. As a result, model zoos advertise PTNNs by their architecture name. The observed accuracy differences and past work on DNN vulnerabilities motivated us to examine architecture [12]. Qualitative observations of discrepancies in the descriptions, source code, and visualizations of PTNN architectures were employed. Specifically, netron, an open-source neural network visualizer, was used to inspect the architecture of the PTNNs [31]. However, not all neural network weight formats are supported, so all PTNNs were converted to the ONNX format for architectural analysis using an appropriate tool for each framework [24, 29]. The source code for the implementations of the PTNNs are present in the model zoos' GitHub repositories and was used as an additional form of PTNN inspection.

Table 1: Frequency at which each model zoo had the most or least accurate model ordered by highest top-1 accuracy.

	Highest Top-1	Lowest Top-1	Highest Top-5	Lowest Top-5
Torchvision Models	48%	41%	52%	36%
TF Model Garden	40%	33%	36%	43%
Keras Applications	37%	44%	36%	40%
ONNX Model Zoo	35%	41%	31%	44%

4 RESULTS AND ANALYSIS

4.1 Accuracy

We compared the top-1 accuracy of 35 PTNN architectures and the top-5 accuracy of 32 PTNN architectures by using ImageNet. Notably, 12 of the 35 profiled PTNN architectures had top-1 accuracy differences greater than 0.96%. For top-5 accuracies, 6 of the 32 PTNN architectures had differences greater than 0.94%. The large differences present in Figure 2 have significant consequences. For example, *ResNet V1 152* from Keras is noticeably less accurate than the PTNN by the same name from Torchvision, with top-1 accuracies of 76.6% and 78.31%, respectively. This difference is pronounced enough that *ResNet V1 101* from Torchvsion with top-1 accuracy of 77.37% is more accurate than *ResNet V1 152* from Keras.[4]

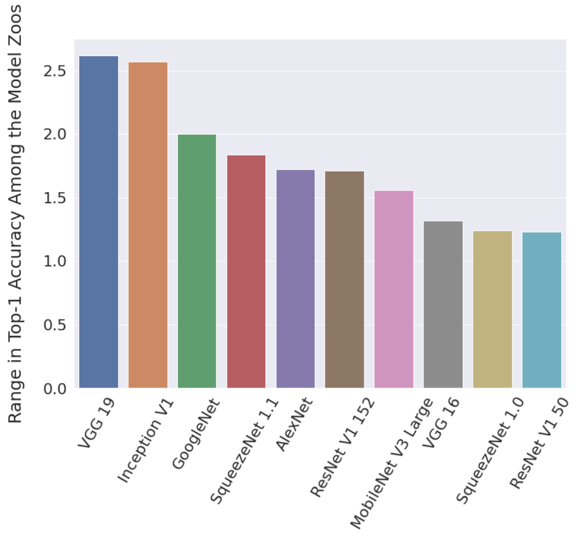

Figure 2: Top 10 largest top-1 accuracy differences. For a PTNN architecture, the accuracy of the PTNN with the lowest reported top-1 accuracy is subtracted from that of the PTNN with the largest top-1 accuracy.

Table 1 shows the aggregation of accuracy differences across model zoos, highlighting how often a model zoo had the highest or lowest top-1 or top-5 accuracy for a given PTNN architecture. As seen, 48% of the PTNNs that were available on Torchvision had the highest top-1 accuracy among the model zoos. On the other hand, Keras had the lowest top-1 accuracy 44% of the time for its selection of PTNNs.

4.2 Latency

36 PTNN architectures were measured for their FLOPs. Figure 3 shows that there are 8 PTNN architectures where the PTNN with the highest amount of FLOPs had greater than 10% more FLOPs than the PTNN with the lowest FLOP count. At the extreme, Torchvision's *SqueezeNet 1.0*, sitting at 819.08 million FLOPs, had 2.31× the FLOPs of ONNX's *SqueezeNet 1.0*. Likewise, the three PTNN architectures from the *ResNet V2* family all had greater than 85% more FLOPs than the lowest FLOPs PTNN. All the high FLOP-count *ResNet V2* come from TFMG.

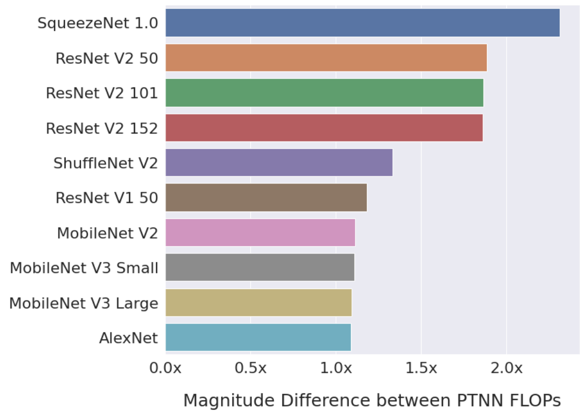

Figure 3: Top 10 largest FLOPs differences. For a PTNN architecture, the FLOP count of the PTNN with the most FLOPs is divided by the FLOPs of the PTNN with the fewest.

We discuss the possible explanations for the FLOPs differences seen in Figure 3. The high FLOPs difference measured in *SqueezeNet 1.0* can be explained by looking at its successor, *SqueezeNet 1.1*. *SqueezeNet 1.1* is advertised by ONNX to contain 2.4× less computation than the former. However, *SqueezeNet 1.1* from ONNX has the same number of measured FLOPs as the *1.0* PTNN offered. ONNX has been advertising *SqueezeNet 1.1* as its *1.0* counterpart. Similarly, looking at the *ResNet V2* from TFMG: a primary contributor to the large amount of FLOPs is the input shape. *ResNet V2* architectures, according to the origin paper, accept 224×224 inputs [15]; however, TFMG states that the *ResNet V2* PTNNs it provides use Inception pre-processing and an input image size of 299×299. A trade-off between accuracy and throughput, FLOPs, was potentially made here by the model zoo maintainers.

Across all FLOP-counted PTNNs, Torchvision had the highest FLOPs PTNNs for 78% of the PTNNs it offered. Close behind, TFMG had 69%. Pointedly, Keras never had the highest FLOPs PTNN and had the lowest FLOPs implementation 81% of the time.

4.3 Architecture

We frame our results for architecture in terms of the discrepancies we discovered in our analysis. Specifically, we discuss differences among PTNNs for *AlexNet*, *ResNet V1 101*, *ResNet V2 50*, and *ResNet V2 101* and against the PTNNs' origin papers.

The *AlexNet* from Torchvision cites a different origin paper than other model zoos [19, 20]. Both papers contain the same first author; however, only the latter contains an explicit description of a DNN architecture. As such, our analysis pegs the PTNN against the latter paper [20]. We notice two main discrepancies: the PTNN is missing the response normalization layers and the kernel-size and number of kernels for the convolution layers are incorrect. For instance, Torchvision's PTNN has 64 kernels in the first convolution layer as opposed to the 96 that are described in the origin paper.

[4]ResNet V1 101 was originally reported to be 0.32% less accurate than *ResNet V1 152* [14].

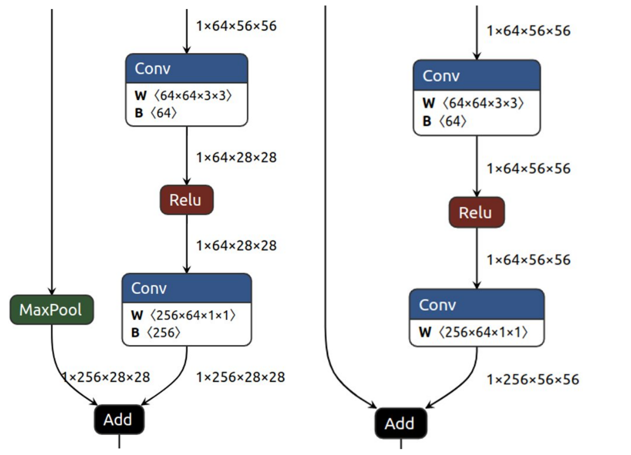

Figure 4: *ResNet V2 50* architecture differences between *Keras Applications* (left) and *ONNX Model Zoo* (right). The top-right convolution on the left has a stride size of 2, while the top-right convolution on the right has a stride size of 1.

The *ResNet V1 101* from ONNX and Keras contain convolution shortcuts, which were only introduced in the *ResNet V2* paper, but not in the *ResNet V1* origin paper [14, 15]. Torchvision's and TFMG's *ResNet V1 101* do not include this shortcut. Also in the *ResNet* family, both the *ResNet V2 50* and *ResNet V2 101* have a shared discrepancy. As seen in Figure 4, Keras' *ResNet V2 50* implementation contains max pool skip connections, which are not present in the paper, and uses convolutions with larger strides in these residual blocks [15].

The observed discrepancies in architecture may affect the accuracy and latency. For example, the larger convolution strides and max pool skip connection in the *ResNet V2 50* from Keras allows the network to use less compute, FLOPs, compared to the PTNN from ONNX. This can be seen in the FLOP measurements of the *ResNet V2 50* from Keras and ONNX. ONNX's *ResNet V2 50* has 4.12 billion FLOPs while Keras' PTNN only has 3.49 billion FLOPs, an 18.1% difference. Moreover, the Keras PTNN did not sacrifice accuracy through this implementation, reporting a 76% top-1 accuracy, which is greater than ONNX's *ResNet V2 50* top-1 accuracy of 75.81%. While the Keras maintainers did not implement *ResNet V2 50* faithfully to the origin paper, they produced a more accurate PTNN with lower latency.

5 DISCUSSION AND FUTURE WORK

Empirical Validation. The top-1 accuracy differences depicted in Figure 2 suggest that the choice of model zoo matters. Specifically, 34% of the PTNN architectures having top-1 accuracy differences greater than 0.96% is not easily overlooked. An engineer may receive a PTNN that incorrectly classifies greater than 500 validation images on ImageNet more than a PTNN from a different model zoo. Model zoo choice should not result in a noticeable impact on the accuracy of PTNNs that engineers receive. Although model zoos currently report the accuracy of the PTNNs they offer, our work has shown that this does not guarantee that there is not another model zoo with the same PTNN at a higher accuracy. Publicly available and actively maintained comparisons of model zoo PTNNs would allow engineers to be more informed when choosing a model zoo. Furthermore, we only studied the accuracies of image classification models at face value. We recommend future works focus on empirical validation on the claims of PTNNs in model zoos to check for the existence of false advertising.

New Metrics and Automated Tools. The measured FLOP disparities seen in Figure 3 have consequences, especially in edge devices with limited compute. For example, ONNX incorrectly listing *SqueezeNet 1.1* as *SqueezeNet 1.0* may lead to confusion when an engineer switches to *SqueezeNet 1.1* from *SqueezeNet 1.0* expecting a drop in latency. Similar confusion may arise from instances like the one seen in TFMG's selection of *ResNet V2*. While the increased input size is stated, the impact on latency is not made clear. To effectively inform engineers of the latency of PTNNs, model zoos should report FLOP counts alongside accuracy. Also of interest is the energy usage of these PTNNs, another important property for edge devices. The lack of reporting of these properties may make choosing PTNNs more difficult. We recommend future works create new metrics to measure the reliability and quality of PTNNs from model zoos and develop tools for automatically measuring these properties. Publishing updated results frequently can support easier decision-making of models for deployment.

Naming Conventions. The differences in the architectures of PTNNs may indicate an underlying improper documentation standard and a need for improved naming conventions in model zoos. As indicated in §4.3, Torchvision's *AlexNet* did not adhere to the origin paper while still claiming to be *AlexNet*. Seemingly, model zoos are advertising PTNNs labeled as well-known DNN architectures, like *ResNet* and *AlexNet*, but when they do this, they really mean that the PTNNs are based on the DNN architecture and are not strict implementations. This inadequate naming convention leads to a false sense of equality and thus confusion. We recommend the community comprehensively document PTNN naming conventions to increase cohesion among model zoos. Likewise, we suggest future works investigate the expectations of engineers with regards to the PTNNs from model zoos to see whether they prefer exact reproductions or more accurate and lower latency PTNNs. The result of such a study would inform model zoo maintainers on how to best implement and train PTNNs.

6 CONCLUSION

We present an investigation of the discrepancies between 36 image classification PTNN architectures from four popular model zoos through accuracy, latency, and architecture analyses. We find several significant discrepancies among these three axes that challenge the well-established use of PTNNs from model zoos, suggesting that an engineer will receive a PTNN with different characteristics based on the model zoo. The PTNN's goal of shortening model deployment time is diminished because of the time investment needed to verify the properties of the PTNN. We discuss the importance of future works to validate the claims of model zoos, develop automated tools for measurement, and explore best practices for implementing model zoo PTNNs.

ACKNOWLEDGMENTS

This work was supported by gifts from Google and Cisco and by NSF-OAC award #2107230. We thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

REFERENCES

[1] 2019. ONNX | Home. https://onnx.ai/

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://doi.org/10.5281/zenodo.4724125

[3] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software Engineering for Machine Learning: A Case Study. In *International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. https://doi.org/10.1109/ICSE-SEIP.2019.00042

[4] Vishnu Banna, Akhil Chinnakotla, Zhengxin Yan, Anirudh Vegesana, Naveen Vivek, Kruthi Krishnappa, Wenxin Jiang, Yung-Hsiang Lu, George K. Thiruvathukal, and James C. Davis. 2021. An Experience Report on Machine Learning Reproducibility: Guidance for Practitioners and TensorFlow Model Garden Contributors. https://doi.org/10.48550/arXiv.2107.00821

[5] blacklong28. 2022. *onnx-opcounter*. https://github.com/blacklong28/onnx-opcounter

[6] Michaela Blott, Lisa Halder, Miriam Leeser, and Linda Doyle. 2019. QuTiBench: Benchmarking Neural Networks on Heterogeneous Hardware. *Journal on Emerging Technologies in Computing Systems (JETC)* (2019). https://doi.org/10.1145/3358700

[7] Hudson Borges and Marco Valente. 2018. What's in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform. *Journal of Systems and Software* (2018). https://doi.org/10.1016/j.jss.2018.09.016

[8] Eric Breck, Shanqing Cai, Eric Nielsen, Michael Salib, and D. Sculley. 2017. The ML test score: A rubric for ML production readiness and technical debt reduction. In *International Conference on Big Data (BigData)*. https://doi.org/10.1109/BigData.2017.8258038

[9] Anh T. Dang. 2021. Accuracy and Loss: Things to Know about The Top 1 and Top 5 Accuracy. https://towardsdatascience.com/accuracy-and-loss-things-to-know-about-the-top-1-and-top-5-accuracy-1d6beb8f6df3

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition (CVPR)*. https://doi.org/10.1109/CVPR.2009.5206848

[11] Nikhil Krishna Gopalakrishna, Dharun Anandayuvaraj, Annan Detti, Forrest Lee Bland, Sazzadur Rahaman, and James C. Davis. 2022. "If security is required": Engineering and Security Practices for Machine Learning-based IoT Devices. In *2022 IEEE/ACM 4th International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT)*. 1–8. https://doi.org/10.1145/3528227.3528565

[12] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. https://doi.org/10.48550/arXiv.1708.06733

[13] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. 2021. Pre-trained models: Past, present and future. *AI Open* (2021). https://doi.org/10.1016/j.aiopen.2021.08.002

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/CVPR.2016.90

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. In *European conference on computer vision (ECCV)*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). https://doi.org/10.1007/978-3-319-46493-0_38

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017). https://doi.org/10.48550/arXiv.1704.04861

[17] Matthew Hutson. 2018. Artificial intelligence faces reproducibility crisis. *Science* (2018). https://doi.org/10.1126/science.359.6377.725

[18] Keras. 2022. *Keras Applications*. https://keras.io/api/applications/

[19] Alex Krizhevsky. 2014. One weird trick for parallelizing convolutional neural networks. *arXiv* (2014). https://doi.org/10.48550/arXiv.1404.5997

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *International Conference on Neural Information Processing Systems (NeurIPS)*. https://doi.org/10.1145/3065386

[21] Ling Liu, Yanzhao Wu, Wenqi Wei, Wenqi Cao, Semih Sahin, and Qi Zhang. 2018. Benchmarking Deep Learning Frameworks: Design Considerations, Metrics and Beyond. In *International Conference on Distributed Computing Systems (ICDCS)*. https://doi.org/10.1109/ICDCS.2018.00125

[22] Tong Liu, Shakeel Alibhai, Jinzhen Wang, Qing Liu, Xubin He, and Chentao Wu. 2019. Exploring Transfer Learning to Reduce Training Overhead of HPC Data in Machine Learning. In *International Conference on Networking, Architecture and Storage (NAS)*. https://doi.org/10.1109/NAS.2019.8834723

[23] Sébastien Marcel and Yann Rodriguez. 2010. Torchvision the Machine-Vision Package of Torch. In *ACM international conference on Multimedia*. https://doi.org/10.1145/1873951.1874254

[24] ONNX. 2022. *tf2onnx*. https://github.com/onnx/tensorflow-onnx

[25] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* (2010). https://doi.org/10.1109/TKDE.2009.191

[26] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon Emissions and Large Neural Network Training. *arXiv*. https://doi.org/10.48550/arXiv.2104.10350

[27] Hung Viet Pham, Mijung Kim, Lin Tan, Yaoliang Yu, and Nachiappan Nagappan. 2021. DEVIATE: A Deep Learning Variance Testing Framework. In *International Conference on Automated Software Engineering (ASE)*. https://doi.org/10.1109/ASE51524.2021.9678540

[28] Hung Viet Pham, Shangshu Qian, Jiannan Wang, Thibaud Lutellier, Jonathan Rosenthal, Lin Tan, Yaoliang Yu, and Nachiappan Nagappan. 2020. Problems and Opportunities in Training Deep Learning Software Systems: An Analysis of Variance. In *International Conference on Automated Software Engineering (ASE)*. https://doi.org/10.1145/3324884.3416545

[29] PyTorch. 2022. *pytorch*. https://github.com/pytorch/pytorch

[30] Saidur Rahman, Emilio River, Foutse Khomh, Yann Gal Guhneuc, and Bernd Lehnert. 2019. Machine learning software engineering in practice: An industrial case study. In *International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. https://doi.org/10.1109/ICSE-SEIP.2019.00042

[31] Lutz Roeder. 2022. *netron*. https://netron.app/

[32] Lucas Sakurada, Carla A. S. Geraldes, Florbela P. Fernandes, Joseane Pontes, and Paulo Leitao. 2020. Analysis of New Job Profiles for the Factory of the Future. *International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing* (2020). https://doi.org/10.1007/978-3-030-69373-2_18

[33] Shayan Shams, Richard Platania, Kisung Lee, and Seung-Jong Park. 2017. Evaluation of Deep Learning Frameworks Over Different HPC Architectures. In *International Conference on Distributed Computing Systems (ICDCS)*. https://doi.org/10.1109/ICDCS.2017.259

[34] Facebook AI Research Team. 2022. *fvcore*. Facebook Research. https://github.com/facebookresearch/fvcore

[35] Google Brain Team. 2022. *TensorFlow*. https://github.com/tensorflow/tensorflow

[36] Sebastian Thrun and Lorien Pratt. 1998. *Learning to Learn: Introduction and Overview*. https://doi.org/10.1007/978-1-4615-5529-2_1

[37] Jason Tsay, Alan Braz, Martin Hirzel, Avraham Shinnar, and Todd Mummert. 2020. AIMMX: Artificial Intelligence Model Metadata Extractor. In *International Conference on Mining Software Repositories (MSR)*. https://doi.org/10.1145/3379597.3387448

[38] Hongkun Yu, Chen Chen, Xianzhi Du, Yeqing Li, Abdullah Rashwan, Le Hou, Pengchong Jin, Fan Yang, Frederick Liu, Jaeyoun Kim, and Jing Li. 2020. TensorFlow Model Garden. https://github.com/tensorflow/models.