Vol: 2, Issue: 2, 2024
Pages:108-139
Received: 01 December 2024
Accepted: 25 December 2024

CURRENT TRENDS IN COMPUTING
Research Article
DOI: 10.71074/CTC.1594291

ENHANCING GREEN COMPUTING THROUGH ENERGY-AWARE TRAINING: AN EARLY STOPPING PERSPECTIVE

ABDULKADIR TAŞDELEN *

Department of Software Engineering, Faculty of Engineering and Natural Sciences, Ankara Yıldırım Beyazıt University, Ankara 06010, Türkiye

ABSTRACT. **This study delves into energy-efficient training strategies, emphasizing their alignment with green computing principles. In particular, it highlights the utility of early stopping mechanisms in optimizing the training process of deep learning models. Early stopping works by monitoring performance metrics, such as validation accuracy or loss, and halting the training process once these metrics stabilize or show no improvement over a predefined number of epochs. This approach eliminates redundant computations, leading to significant reductions in energy consumption and computational costs while preserving model accuracy. The research is centered on transfer learning models, specifically MobileNetV2, InceptionV3, ResNet50V2, and Xception, which are well-regarded for their versatility and performance in image classification tasks. By systematically varying patience criteria, the study explores their impact on training duration, model accuracy, and computational efficiency. Each patience criteria determine how many epochs the training continues without improvement before stopping, allowing for a nuanced examination of its effects across different architectures. Additionally, the Rock Paper Scissors dataset, used for this study, is thoroughly described, including its structure, size, and pre-processing steps applied. The findings reveal that early stopping not only streamlines the training process but also aligns well with the broader goals of sustainable artificial intelligence development. Supported by statistical analyses, such as Kruskal-Wallis H and Conover-Iman tests, the results demonstrate that early stopping significantly reduces training time without compromising accuracy. By effectively balancing computational efficiency with performance optimization, this strategy exemplifies how environmentally responsible practices can be integrated into AI workflows. This study contributes valuable insights into how adopting such techniques can mitigate the environmental impact of AI model training, highlighting their importance in the context of advancing green computing initiatives.**

1. INTRODUCTION

The development and application of technology increasingly demand environmentally conscious approaches, particularly in response to the global challenges of sustainability and energy consumption [1],

E-mail address: abdulkadirtasdelen@aybu.edu.tr (*).
Key words and phrases. Artificial Intelligence, Green Computing, Green AI, Early Stop Strategy .

[2]. In this context, green computing emerges as a critical paradigm, aiming to minimize the environmental footprint of computational processes by enhancing energy efficiency and reducing operational costs. Its applications span diverse areas, including power management, server virtualization, data center optimization, and energy-efficient resource utilization, significantly influencing fields such as business, environmental management, and artificial intelligence (AI) [3], [4].

AI, while transformative across numerous industries, presents a considerable environmental challenge due to its high energy demands. Since the success of AlexNet in the 2012 ImageNet competition [5], the computational requirements for model training have grown exponentially, leading to substantial energy consumption [4], [6]. Addressing these challenges necessitates strategies that align AI development with sustainability goals. Recent advancements in AI training have highlighted the critical role of energy-efficient strategies. For instance, studies on large-scale language models such as GPT-3 have shown significant energy demands, emphasizing the necessity of integrating green computing principles into AI workflows [7]. Among these, early stopping—a technique that halts training once performance metrics stabilize—has shown promise. Early stopping is a regularization technique used to terminate training when performance on a validation set ceases to improve. This prevents over-fitting by halting before the model memorizes training data. By curbing unnecessary computations, early stopping not only enhances computational efficiency but also embodies the principles of green computing, offering a pathway toward sustainable AI practices [8].

Another critical aspect of green computing is the efficient management of energy distribution and usage, as exemplified by smart grid technologies. These systems manage renewable energy sources more effectively, boosting energy efficiency while reducing greenhouse gas emissions. However, ensuring reductions in energy costs and CO_2 emissions remains a priority. Research in commercial and institutional buildings demonstrates that human intervention, supported by energy-saving techniques and information systems, can significantly minimize energy losses. Similarly, sorting tasks based on time and power requirements exemplifies strategies for reducing power consumption during decision-making processes [6], [9], [10].

In AI, early stopping stands out as a pivotal approach to mitigating the environmental impact of training processes. This method halts the training process when performance metrics, such as validation accuracy, plateau or decline. By reducing computational demands, early stopping directly addresses the sustainability challenges of AI, balancing computational efficiency with model performance [11], [12]. Such optimization techniques illustrate how green computing principles can be effectively integrated into AI workflows, fostering a future where technological progress aligns with environmental responsibility.

The increasing demand for energy-efficient solutions in AI underscores the need for interdisciplinary approaches that prioritize sustainability without sacrificing performance. As global concerns about climate change and resource scarcity grow, embedding green computing principles into emerging technologies becomes imperative. Innovations like energy-efficient consensus mechanisms, smart grids, and energy-aware training strategies enable technological advancements to align with the objectives of sustainable development, ensuring that progress benefits both society and the environment [4], [6], [12], [13].

This study specifically investigates the application of early stopping mechanisms in the training of deep learning models, focusing on their energy efficiency and impact on model performance. Transfer learning architectures—MobileNetV2, InceptionV3, ResNet50V2, and Xception—were selected for their versatility and strong performance in image classification tasks. These architectures are systematically evaluated by varying patience criteria (PC), which define thresholds for halting training in the absence of performance improvements, facilitating an analysis of training duration, model accuracy, and computational efficiency [11].

The Rock Paper Scissors dataset [14] serves as the basis for evaluation, with detailed descriptions of its structure, size, and pre-processing steps ensuring clarity and reproducibility. Statistical analyses, including Kruskal-Wallis H [15], Mann-Whitney U [16], and Conover-Iman [17] tests, validate the results rigorously.

The novelty of this work lies in its comparative analysis of multiple deep learning architectures, underpinned by statistical rigor. By employing Kruskal-Wallis H and Conover-Iman tests, the study ensures the reliability and generalizability of its findings. This approach establishes a benchmark for future research on energy-aware AI methodologies.

Furthermore, this research underscores the interdisciplinary nature of integrating green computing principles into AI model training. By fostering collaboration across fields such as computer science, environmental science, and data engineering, it paves the way for innovations that prioritize sustainability without compromising technological advancements.

2. Background

The rising demand for machine learning (ML)-enabled systems has significantly increased energy consumption across various computational tasks. As ML applications proliferate, their environmental impact becomes a growing concern. To address this, researchers and practitioners have emphasized green computing practices, which focus on minimizing energy usage while maintaining model performance [13].

Xu *et al.* [12], empirically evaluated the impact of experimental design on the energy efficiency of the training process by analyzing three different convolutional neural network (CNN) architectures across two large image classification datasets. The training sessions were assessed using three efficiency metrics: CO_2 emissions produced, total energy consumed, and the number of floating-point operations (FLOPs) required. Statistical evidence revealed that carbon emissions and energy consumption are closely linked to the experimental design of neural network architectures. Furthermore, external factors, such as the geographical location of cloud-hosted services, also influence the computational impact, highlighting challenges beyond the researcher's control. These findings emphasize the importance of incorporating energy-efficient strategies into deep learning research to ensure that advancements in model performance align with sustainable computing practices.

A catalog of green architectural tactics for ML-enabled systems highlights a structured approach to energy efficiency. These tactics span multiple dimensions, including data-centric methods, algorithm design, model optimization, model training, deployment, and management. Among these, energy-aware

training strategies, such as quantization-aware training, leveraging checkpoints, and designing for memory constraints, are pivotal in reducing computational overhead during the training phase [8], [13].

Recent studies emphasize the importance of addressing over-parameterization in pre-trained CNN models to enhance energy efficiency and computational performance. For example, a systematic analysis of 27 pre-trained models identified EfficientNetB0 as the most energy-efficient candidate for Eimeria parasite detection, reducing parameter counts by up to 8% through pruning without sacrificing classification accuracy. This approach not only saves energy but also demonstrates the potential for holistic model design, combining multiple species into a single model for improved efficiency [18].

In this context, early stopping emerges as a key energy-aware training method that halts model training once a predefined performance threshold is met. By preventing unnecessary computations, this approach not only conserves energy but also accelerates the development cycle of ML models [13]. While early stopping has been widely studied in the context of energy-aware machine learning, its comparative effects across diverse transfer learning architectures remain underexplored. This study addresses this gap by systematically evaluating multiple architectures using a standardized dataset and statistical rigor.

Early exit strategies in deep learning have emerged as an effective approach to balancing model performance and computational efficiency. These strategies allow intermediate predictions within neural networks, enabling the termination of computations for certain inputs when a confident prediction is achieved. This approach has been extensively studied across various domains and tasks, including image classification, machine translation, text ranking, and quality enhancement [11], [19].

Recent studies on early existing, as summarized in [20], highlight the diversity of applications and metrics employed to evaluate these strategies. For instance, Teerapittayanon *et al.* [21], [22] applied early exit strategies to image classification tasks using datasets such as MNIST and CIFAR-10, with base models like LeNet-5, AlexNet, and ResNet, focusing on metrics like accuracy and latency. Similarly, Wang *et al.* [23] and Li *et al.* [24] investigated the efficiency of early exits in large-scale datasets like ImageNet, employing models such as ResNet, DenseNet, and MSDNet. Notably, energy consumption was explicitly considered in several works, such as those by Laskaridis *et al.* [25] and Wang *et al.* [23], underscoring the role of early exits in energy-efficient computing.

By reducing latency, computational complexity, and energy consumption, early exit strategies align with green computing principles, offering a pathway to sustainable deep learning practices. Integrating these strategies into model design and training processes could significantly reduce the environmental footprint of machine learning applications. Future research in this area is expected to expand the scope of early exits to additional tasks and domains, further advancing the synergy between performance optimization and energy efficiency in deep learning [11].

In line with the objectives outlined in the introduction, this study underscores the necessity of interdisciplinary efforts to integrate green computing principles into the life-cycle of emerging technologies. By prioritizing energy efficiency and sustainability, we pave the way for innovations that are not only technically advanced but also environmentally responsible. A key focus of this research is the application of early stopping strategies within the context of deep learning, specifically exploring how these strategies influence model performance across various deep learning architectures. By investigating different PC,

this study provides a comparative analysis that highlights the impact of early stopping on both training time and model accuracy.

The primary goal of this study is to explore how early stopping strategies can affect the training process of deep learning models, offering a comprehensive analysis of different model architectures and their performance under varying parameters. While this study does not focus on directly measuring energy consumption or computational resources, it does provide valuable insights into how early stopping can influence training time and model performance. By halting training when the performance metrics, such as validation accuracy or loss, no longer show significant improvement, early stopping reduces unnecessary epochs, which in turn decreases training time.

Validation loss measures the model's error on the validation dataset after each training epoch. It serves as a key indicator of over-fitting, where a rising validation loss suggests that the model is no longer generalizing well to unseen data. Similarly, validation accuracy measures the proportion of correctly predicted instances in the validation dataset after each training epoch, providing insight into the model's generalization performance. A stagnating or declining validation accuracy, despite improvements in training accuracy, may suggest over-fitting.

This research examines the relationship between the number of epochs before stopping, the final test accuracy, and the time taken for training. By analyzing different PC (such as 3, 5, 7, 10, and 15 epochs), we determine the optimal stopping point for achieving a good balance between model performance and training duration. The comparative analysis focuses on how these parameters impact the test accuracy and the training time, ultimately showing that early stopping can be a practical technique to improve the efficiency of deep learning processes.

Ultimately, this study aims to demonstrate how early stopping can optimize deep learning workflows by reducing unnecessary training time, allowing researchers and practitioners to achieve efficient models without over-fitting. By offering a framework for understanding how different PC influence model accuracy and training duration, this research supports the integration of efficiency-oriented strategies into machine learning practices, contributing to more sustainable development cycles.

3. Materials and Methods

3.1. **Transfer Learning Methods:**

To investigate the impact of early stopping strategies on different model architectures, we utilize transfer learning with four well-established transfer learning models [26]: MobileNetV2, InceptionV3, ResNet50V2, and Xception. These pre-trained models are fine-tuned on the Rock Paper Scissors Dataset [14]. Transfer learning is particularly beneficial for tasks with limited labeled data, as the models have already learned relevant features from large-scale datasets such as ImageNet. The fine-tuning process adapts the pre-trained models to the specific classification task, thus accelerating the training process and potentially enhancing model performance. Each of these models has been selected due to its proven effectiveness in image classification tasks, providing a diverse range of model architectures to assess the generalizability and performance of early stopping strategies.

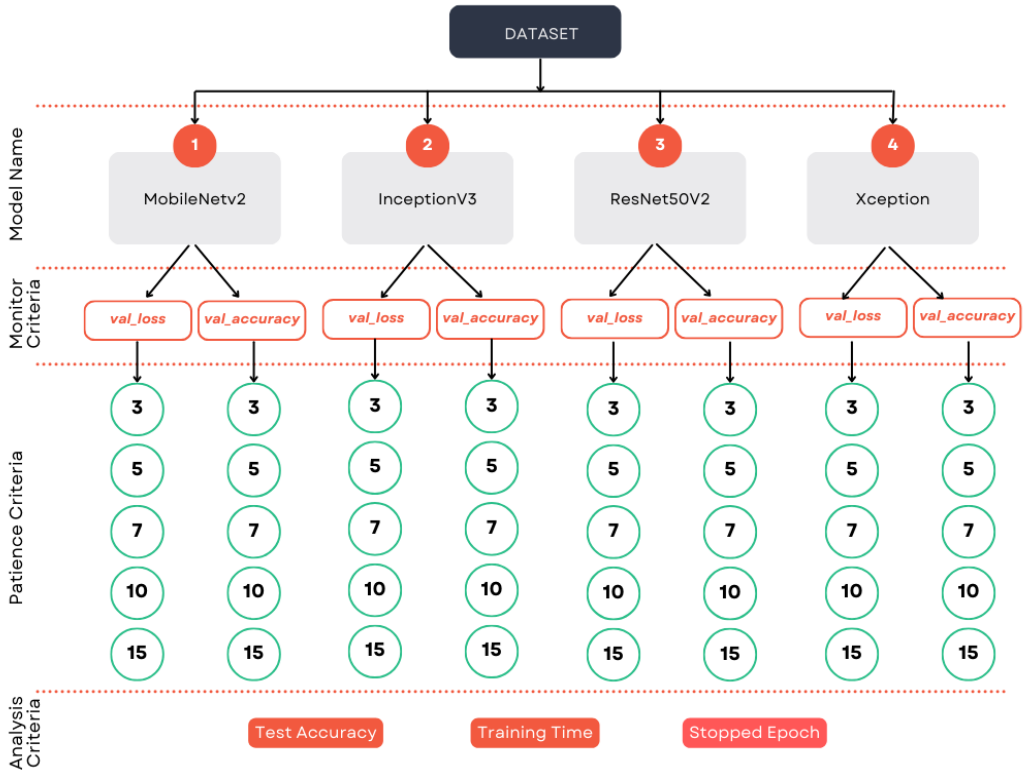

FIGURE 1. **Early stopping parameter configuration and monitoring across transfer learning models.**

Figure 1 illustrates the experimental setup for evaluating the impact of different PC on four transfer learning models: MobileNetV2, InceptionV3, ResNet50V2, and Xception. The monitored criteria, including validation loss and validation accuracy, are analyzed to determine the effects of early stopping strategies on key performance metrics: test accuracy, training time, and stopped epoch. The visual framework emphasizes the systematic evaluation process to identify optimal PC settings for efficient and effective model training.

3.2. **Dataset:**

The Rock Paper Scissors Dataset [14] is used for this study and consists of labeled images categorized into different types. Each category contains 975 images, resulting in a total of 2,925 images. The dataset is divided into two main subsets: 80% (2,340 images) for the training dataset (70% for training and 10% for validation), and 20% (585 images) for the test dataset (Figure 2). This division ensures that the model is trained on a substantial portion of the data, while also leaving a portion for validation and testing to evaluate its performance on unseen data. Moreover, to account for potential variability and to ensure that

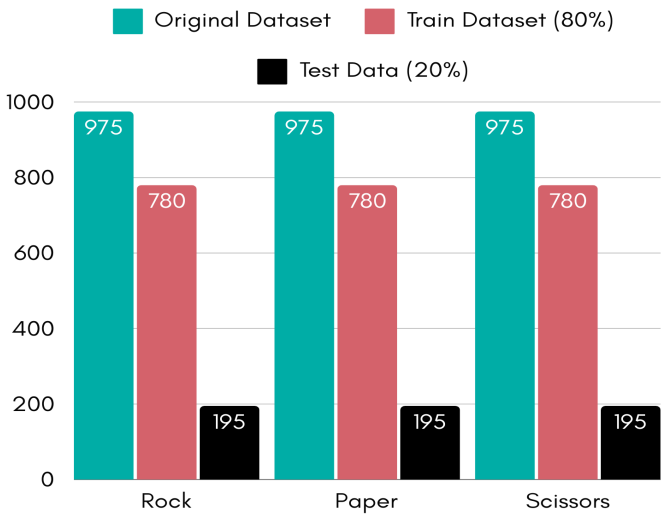

FIGURE 2. **The distribution of each class.**

the models are assessed fairly, 5-fold cross-validation (CV) is employed. CV is a statistical method used to evaluate a model's generalization performance by splitting the dataset into complementary subsets for training and testing. This ensures that the evaluation is robust and not overly dependent on a single split. This method ensures that each fold of the CV process maintains a proportional distribution of the classes, providing a more reliable estimate of the model's generalization ability. In each fold, a different partition is used for training, validation, and testing, and the process is repeated for each fold, allowing the model to be trained and tested on all available data [27], [28].

The dataset serves as the basis for evaluation, with detailed descriptions of its structure, size, and pre-processing steps ensuring clarity and reproducibility. In selecting the dataset, this study aims to analyze model performance on a medium-scale dataset that allows for efficient experimentation with various early stopping strategies. The dataset is well-suited for evaluating energy-efficient training techniques, as it provides a manageable size for computational experiments while maintaining enough complexity to demonstrate key differences in performance. This choice also facilitates reproducibility and comparability with other studies in the field. Future research will consider larger and more complex datasets, enabling the examination of early stopping strategies on a broader range of tasks and more demanding computational settings.

Figure 3 displays samples from different classes of the dataset, which represent the classic game of Rock, Paper, or Scissors. Each class corresponds to one of the three possible moves in the game: Rock, Paper, or Scissors. These samples are crucial for training machine learning models, as they help the

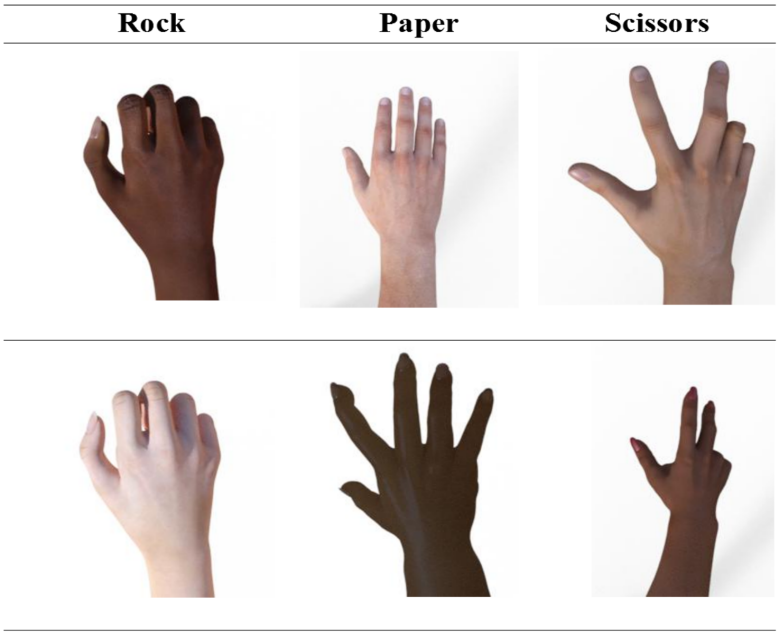

FIGURE 3. **Samples from different classes of the dataset.**

algorithm distinguish between the various input categories. The dataset likely includes images or representations of each move, enabling the model to learn patterns and make predictions based on the visual features associated with Rock, Paper, or Scissors.

3.3. **Statistical Analysis:**

The performance of early stopping strategies is evaluated using several metrics, including classification accuracy, training time, and validation loss. To determine whether early stopping strategies lead to statistically significant differences in performance, a Kruskal-Wallis H test [15] is employed. This non-parametric test is appropriate for comparing multiple independent groups—such as different PC—across performance metrics, as it does not assume the data to be normally distributed. If the Kruskal-Wallis H test yields a significant result, indicating differences in performance across groups, a post-hoc Conover-Iman Test [17] is conducted. This test is particularly suitable when sample sizes within groups are small, providing a robust method for pairwise comparisons while controlling for the family-wise error rate. Additionally, for pairwise comparisons between groups of PC, the Mann-Whitney U test [16] is utilized. This test is a non-parametric alternative to the t-test and is applied when comparing two independent groups, such as specific pairs. It assesses whether the distributions of the two groups differ significantly. The Mann-Whitney U test provides a complementary approach to the Conover-Iman Test by offering

more granular insights into pairwise differences [17]. All statistical tests are conducted with a significance level of and $a = 0.05$. If the p-value from any test is less than 0.05, the null hypothesis is rejected, indicating statistically significant differences between the groups being compared. This rigorous statistical approach ensures that the conclusions drawn from the analysis are reliable and valid. By evaluating the impact of early stopping strategies on model performance through Kruskal-Wallis H, Conover-Iman, and Mann-Whitney U tests, this study aims to identify optimal training configurations that balance computational efficiency with model accuracy.

3.4. **Common Specifications:**

The common specifications for the model are outlined in Table 1. CV is employed with a fold size of 5 to ensure balanced class representation. Data shuffling is enabled with a random state set to 1 for reproducibility. The base model is configured with an input shape of 96x96, utilizes pre-trained weights from the ImageNet dataset. and is set as non-trainable. The model architecture includes a dense layer with 64 units, using the Rectified Linear Unit (ReLU) activation function [29] followed by a dropout layer with a rate of 0.05 to mitigate over-fitting and improve generalization. The Adam optimizer is used during compilation, with a learning rate of 0.0001. Training is conducted using a batch size of 32 and spans a maximum of 50 epochs.

TABLE 1. **Common specifications for each transfer learning models**

Category	**Name**	**Value**
Monitor Criteria	Validation Loss Metric	val_loss
	Validation Accuracy Metric	val_accuracy
Patience Criteria	Stopped Epochs	3, 5, 7, 10, 15
CV	Number of Folds	5
	Shuffle	True
	Random Seed	1
Base Model	Input Shape	96*96
	Weights	ImageNet
	Trainable	False
Dense Layer	Units	64
	Activation Function	ReLU
Dropout Layer	Dropout Rate	0.50
Compile	Optimizer	Adam
	Learning Rate	0.0001
Model Training	Batch Size	32
	Number of Epochs	50

To improve efficiency and avoid over-fitting, early stopping is applied based on the monitoring criteria (MC). Specifically, the training process monitors validation loss and validation accuracy, stopping automatically if no improvement is observed across consecutive epochs. The evaluation is performed at specific checkpoints (epochs 3. 5. 7. 10. and 15), offering insights into model performance at different stages of training. These configurations aim to balance computational efficiency, model generalization, and robustness. ensuring reliable results across all folds of the CV process.

4. RESULTS

The performance of four deep learning models—MobileNetV2, InceptionV3, ResNet50V2, and Xception—was evaluated based on several key metrics, including training accuracy, training loss, validation accuracy, validation loss, test accuracy, test loss, and training time. These metrics were averaged over various early stop criteria, such as *val_loss* and *val_accuracy*, with results reported for different PC. Table 2 summarizes the average performance across these models for each validation criterion.

In general, MobileNetV2 consistently demonstrated high training and test accuracies with relatively low training and test losses. Similarly, InceptionV3 and ResNet50V2 achieved competitive results, although with some variations in performance depending on the validation criteria. Xception, on the other hand, showed a robust performance, especially in terms of validation accuracy and test accuracy, though with higher training and test losses in comparison to the other models. The average training times for all models were also considered, providing an indication of computational efficiency.

The analysis of model performance (Table 3), grouped by MC and PC, revealed distinct patterns in the test accuracy, training time, and early stopping behavior for each architecture.

For MobileNetv2, the test accuracy did not show a statistically significant difference ($p=0.382$, $p>0.05$), indicating consistent performance across groups. However, both training time ($p=0.000$, $p<0.05$) and the stopped epoch count ($p=0.000$, $p<0.05$) exhibited significant variations, suggesting the model's sensitivity to these parameters.

Similarly, InceptionV3 displayed no significant difference in test accuracy ($p=0.403$, $p>0.05$), while training time ($p=0.000$, $p<0.05$) and stopped epoch count ($p=0.000$, $p<0.05$) were significantly impacted, reflecting comparable trends to MobileNetv2.

For ResNet50V2, test accuracy remained consistent ($p=0.269$, $p<0.05$) without notable variation. However, as observed with the other models, both training time ($p=0.000$, $p<0.05$) and stopped epoch count ($p=0.000$, $p<0.05$) varied significantly, reinforcing the importance of these parameters in model training dynamics.

In contrast, Xception demonstrated a significant difference in test accuracy ($p=0.030$, $p<0.05$). indicating variability in performance across groups. Additionally, training time ($p=0.000$, $p<0.05$) and stopped epoch count ($p=0.000$, $p<0.05$) also showed significant variation, further emphasizing the model's sensitivity to MC and PC.

The Kruskal-Wallis H test results, summarized in Table 4, show the comparison of different monitor parameters across various deep learning architectures. For most models, including MobileNetV2, InceptionV3, and ResNet50V2, the test results for test accuracy did not show significant differences ($p > 0.05$),

TABLE 2. **Average performance metrics for MobileNetV2, InceptionV3, ResNet50V2, and Xception models**

MN	MC	PC	SE	TE	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	TT (s)
MobileNetV2	val_loss	3	9.4	50	0.995	0.028	0.841	0.414	0.984	0.056	25.055
	val_loss	5	13.4	50	0.998	0.017	0.848	0.414	0.983	0.051	33.112
	val_loss	7	18	50	0.999	0.011	0.870	0.390	0.986	0.046	42.814
	val_loss	10	19	50	0.999	0.009	0.863	0.429	0.987	0.049	48.824
	val_loss	15	25	50	0.999	0.007	0.827	0.518	0.981	0.061	61.902
	val_acc	3	8.8	50	0.994	0.034	0.831	0.419	0.979	0.068	22.719
	val_acc	5	11.6	50	0.997	0.021	0.857	0.390	0.986	0.054	32.113
	val_acc	7	15	50	0.997	0.017	0.863	0.421	0.985	0.064	39.614
	val_acc	10	18.4	50	0.998	0.011	0.829	0.461	0.980	0.062	48.845
	val_acc	15	25.2	50	0.999	0.006	0.859	0.440	0.986	0.063	75.320
	Original	**No**	**No**	**50**	**1**	**0.002**	**0.840**	**0.580**	**0.978**	**0.070**	**121.674**
InceptionV3	val_loss	3	8.4	50	0.986	0.057	0.758	0.743	0.965	0.125	33.622
	val_loss	5	17	50	0.996	0.023	0.798	0.720	0.976	0.093	70.758
	val_loss	7	17.2	50	0.994	0.027	0.814	0.683	0.972	0.117	82.924
	val_loss	10	30.6	50	0.998	0.010	0.825	0.678	0.982	0.077	182.771
	val_loss	15	30.6	50	0.998	0.009	0.823	0.726	0.981	0.077	181.658
	val_acc	3	10.8	50	0.988	0.049	0.814	0.659	0.969	0.128	68.307
	val_acc	5	19.4	50	0.996	0.017	0.799	0.767	0.979	0.086	125.599
	val_acc	7	27.8	50	0.999	0.008	0.816	0.735	0.979	0.077	191.087
	val_acc	10	31.8	50	0.999	0.008	0.857	0.595	0.983	0.075	223.856
	val_acc	15	36.2	50	0.999	0.005	0.847	0.695	0.982	0.075	256.737
	Original	**No**	**No**	**50**	**0.999**	**0.004**	**0.835**	**0.772**	**0.979**	**0.0991**	**416.361**
ResNet50V2	val_loss	3	17.8	50	0.995	0.025	0.958	0.121	0.993	0.026	139.693
	val_loss	5	16	50	0.993	0.032	0.918	0.194	0.991	0.044	127.430
	val_loss	7	26.6	50	0.997	0.018	0.923	0.166	0.993	0.026	197.787
	val_loss	10	26.2	50	0.997	0.016	0.891	0.232	0.987	0.040	201.758
	val_loss	15	41.6	50	0.998	0.008	0.934	0.133	0.996	0.019	364.172
	val_acc	3	11.6	50	0.985	0.053	0.939	0.1401	0.983	0.056	151.258
	val_acc	5	16.2	50	0.993	0.031	0.939	0.132	0.990	0.037	176.910
	val_acc	7	19.2	50	0.994	0.028	0.941	0.137	0.991	0.040	238.396
	val_acc	10	23.4	50	0.997	0.018	0.955	0.106	0.991	0.036	303.144
	val_acc	15	36.4	50	0.997	0.009	0.919	0.166	0.992	0.025	520.313
	Original	**No**	**No**	**50**	**0.999**	**0.004**	**0.911**	**0.209**	**0.985**	**0.033**	**723.026**
Xception	val_loss	3	9.4	50	0.975	0.116	0.691	0.653	0.942	0.184	71.368
	val_loss	5	15.4	50	0.990	0.065	0.719	0.660	0.958	0.138	118.122
	val_loss	7	16.4	50	0.990	0.068	0.704	0.663	0.951	0.150	125.397
	val_loss	10	22.8	50	0.995	0.042	0.719	0.699	0.958	0.137	175.587
	val_loss	15	28.8	50	0.998	0.025	0.741	0.702	0.965	0.113	233.605
	val_acc	3	10	50	0.974	0.125	0.701	0.648	0.940	0.218	95.685
	val_acc	5	16	50	0.988	0.066	0.716	0.664	0.956	0.140	129.949
	val_acc	7	21.6	50	0.992	0.052	0.698	0.757	0.956	0.159	171.013
	val_acc	10	26.8	50	0.997	0.029	0.722	0.752	0.963	0.116	219.265
	val_acc	15	37.4	50	0.998	0.019	0.728	0.823	0.966	0.105	313.890
	Original	**No**	**No**	**50**	**0.999**	**0.010**	**0.732**	**0.874**	**0.965**	**0.120**	**429.153**

*MN: Transfer Learning Model Name. MC: Monitor Criteria. PC: Patience Criteria. SE: Stopped epoch. TE: Total Number of Epochs. TT: Training Times in Seconds. Note: All results except TE are mean of each fold. See **Appendix A** for details.

TABLE 3. **Summary of each model by groups**

#	Group Name	Test Statistic	*p*-value	Result
	Test Accuracy	10.693	0.382	+
MobileNetV2	Training Time	42.263	0.000	**x**
	Stopped Epoch	40.643	0.000	**x**
	Test Accuracy	10.434	0.403	+
InceptionV3	Training Time	42.982	0.000	**x**
	Stopped Epoch	43.074	0.000	**x**
	Test Accuracy	12.251	0.269	+
ResNet50V2	Training Time	43.426	0.000	**x**
	Stopped Epoch	41.092	0.000	**x**
	Test Accuracy	19.895	0.030	**x**
Xception	Training Time	41.517	0.000	**x**
	Stopped Epoch	41.719	0.000	**x**

Note: "+": H_0 is accepted ($p > 0.05$), "x": H_0 is rejected ($p < 0.05$).

meaning the null hypothesis (H_0) was accepted. However, for training time and the number of stopped epochs, significant differences were observed ($p < 0.05$), leading to the rejection of the null hypothesis (H_0) for those parameters, indicating that early stopping significantly affected these factors. Only Xception model's test accuracy did show a significant result ($p < 0.05$), while other models exhibited no significant difference in test accuracy. Overall, the results suggest that early stopping strategies had a noticeable impact on training time and number of epochs, but did not always affect the test accuracy in a significant way.

The Mann-Whitney U Test was conducted to evaluate the pairwise differences in performance metrics across four transfer learning architectures: MobileNetV2, ResNet50V2, InceptionV3, and Xception. The metrics analyzed included test accuracy, training time, and stopped epoch under various early stopping PC (Table 5).

The test accuracy results across all comparisons yielded $p > 0.05$, indicating that the null hypothesis (H_0) could not be rejected. This signifies that there were no statistically significant differences in test accuracy between the models across the considered PC. The early stopping strategies employed do not significantly affect the accuracy of the models. This consistency suggests that the transfer learning architectures are robust to changes in PC with respect to test accuracy.

For most pairwise comparisons, the $p > 0.05$, indicating no significant differences in training time across architectures and PC. At a PC of 7, the comparison involving MobileNetV2 resulted in a p-value = 0.008, which is statistically significant ($p < 0.05$). This suggests that the training time for MobileNetV2 is significantly different compared to another model for this specific PC. While early stopping does not generally lead to significant differences in training time, the significant result for MobileNetV2 indicates that certain architectures may exhibit more sensitivity to specific PC in terms of computational efficiency.

TABLE 4. **Summary of Kruskal-Wallis Test results for two different MC**

		MC= *val_loss*			MC= *val_accuracy*		
#	**Group Name**	**Test Statistic**	***p*-value**	**Result**	**Test Statistic**	***p*-value**	**Result**
	Test Accuracy	5.515	0.356	+	7.881	0.163	+
MobileNetV2	Training Time	23.885	0.000	**x**	24.474	0.000	**x**
	Stopped Epoch	25.137	0.000	**x**	23.540	0.000	**x**
	Test Accuracy	6.147	0.292	+	4.715	0.452	+
InceptionV3	Training Time	22.925	0.000	**x**	24.025	0.000	**x**
	Stopped Epoch	23.655	0.000	**x**	24.159	0.000	**x**
	Test Accuracy	8.418	0.135	+	4.632	0.462	+
ResNet50V2	Training Time	24.561	0.000	**x**	23.622	0.000	**x**
	Stopped Epoch	23.698	0.000	**x**	22.883	0.000	**x**
	Test Accuracy	12.703	0.026	x	8.430	0.134	+
Xception	Training Time	25.552	0.000	**x**	22.533	0.000	**x**
	Stopped Epoch	25.493	0.000	**x**	22.943	0.000	**x**

Note: "+": H_0 is accepted ($p > 0.05$), "x": H_0 is rejected ($p < 0.05$).

For most pairwise comparisons of stopped epochs, the $p > 0.05$, meaning no significant differences were observed between the models. At a PC of 3, one comparison yielded a $p = 0.020$, which is statistically significant ($p < 0.05$). This indicates a significant difference in the number of epochs at which training is halted for smaller PC between the models. Early stopping with smaller PC may result in varying training dynamics across architectures, influencing the point at which training is terminated. The boxplot (Figure 4) illustrates the test accuracy of the Xception model across various MC and PC. Each group represents a specific monitoring criterion (e.g. validation loss or validation accuracy) evaluated at different epochs, as well as a no-monitoring baseline. The mean and median test accuracies are annotated for each group, providing insights into consistency and performance variations.

The results highlight that the *val_accuracy_15* and *no_monitoring_No* groups achieved the highest median and mean test accuracies (0.97), while *val_loss_3* and *val_accuracy_3* showed comparatively lower performance, with median values around 0.94. This indicates that MC and the choice of PC significantly influence the model's ability to generalize effectively.

Although the test accuracy results for the Xception model suggest that there is a statistically significant difference between at least one pair of monitoring groups ($p < 0.05$), further pairwise analysis using the Conover-Iman test reveals a nuanced interpretation (Table 6). Specifically, while some groups, such as *no_strategy* vs. *val_loss_3* ($p = 0.046$) and *val_accuracy_3* vs. *val_accuracy_15* ($p = 0.046$) show *p*-values below the significance threshold of 0.05, these values are very close to the threshold. Additionally, other comparisons, such as *val_loss_3* vs. *val_loss_5* ($p = 0.297$) and *val_accuracy_5* vs. *val_accuracy_7* (p=0.687), yield much higher *p*-values, indicating no statistically significant difference.

TABLE 5. **Summary of Mann-Whitney U Test results for pair comparison by monitoring strategy**

PC	Group Name	Test Statistic	*p*-value	Result	Test Statistic	*p*-value	Result
	Test Accuracy	8.5	0.462	+	6	0.206	+
3	Training Time	10	0.690	+	15	0.690	+
	Stopped Epoch	10.5	0.747	+	1	**0.020**	**x**
	Test Accuracy	16	0.523	+	11	0.829	+
5	Training Time	12	1.000	+	22	0.056	+
	Stopped Epoch	9	0.522	+	10.5	0.750	+
	Test Accuracy	13	1.000	+	8	0.395	+
7	Training Time	10	0.690	+	16	0.548	+
	Stopped Epoch	8.5	0.461	+	5	0.151	+
	Test Accuracy	4	0.090	+	10	0.671	+
10	Training Time	12	1.000	+	20	0.151	+
	Stopped Epoch	12.5	1.000	+	7.5	0.340	+
	Test Accuracy	15.5	0.600	+	5	0.134	+
15	Training Time	15	0.690	+	19	0.222	+
	Stopped Epoch	10	0.674	+	8.5	0.462	+
	Test Accuracy	16	0.530	+	10.5	0.753	+
3	Training Time	20	0.151	+	14	0.841	+
	Stopped Epoch	17	0.421	+	10	0.675	+
	Test Accuracy	13	1.000	+	16	0.530	+
5	Training Time	22	0.056	+	17	0.421	+
	Stopped Epoch	17	0.402	+	15.5	0.599	+
	Test Accuracy	14	0.832	+	17.5	0.344	+
7	Training Time	25	**0.008**	**x**	18	0.310	+
	Stopped Epoch	22	0.059	+	16.5	0.458	+
	Test Accuracy	12.5	1.000	+	16	0.530	+
10	Training Time	17	0.421	+	18	0.310	+
	Stopped Epoch	14	0.841	+	16.5	0.463	+
	Test Accuracy	15.5	0.599	+	14	0.834	+
15	Training Time	21	0.095	+	21	0.095	+
	Stopped Epoch	16	0.530	+	17.5	0.344	+

Note: • *MobileNetV2,* • *ResNet50V2,* • *InceptionV3,* • *Xception*, PC: Patience Criteria, "+": H_0 is accepted ($p > 0.05$), "x": H_0 is rejected ($p < 0.05$).

Given these findings, while some differences appear significant at first glance, the proximity of the *p*-values to the threshold in certain cases, along with the consistency across most comparisons, suggests that

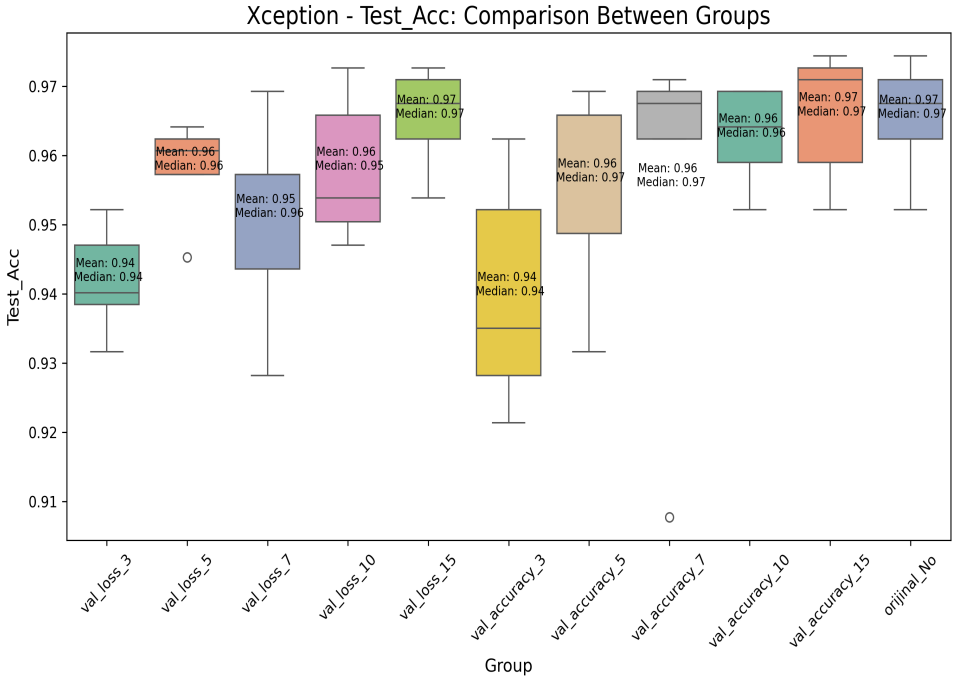

FIGURE 4. **Boxplot of test accuracy for different PC with Xception model.**

these results may not represent meaningful differences in practical terms. As such, the overall outcomes can be interpreted as lacking substantial evidence for significant performance differences between the monitoring strategies. This indicates that the choice of MC might not critically impact the Xception model's test accuracy under the conditions evaluated.

5. DISCUSSION

The results from the application of early stopping strategies to various transfer learning models reveal significant insights into balancing computational efficiency and model performance. In this section, we interpret the findings, their implications for green computing.

5.1. **Impact on Training Time and Computational Efficiency:**

Hussein *et al.* [11] highlighted the critical interplay between early stopping PC and the number of epochs in deep learning models, demonstrating that higher PC values often require more epochs to achieve optimal validation accuracy. Conversely, lower PC values can lead to premature stopping and suboptimal model performance. Importantly, they also noted that prolonged training times do not necessarily enhance validation accuracy, reinforcing the utility of early stopping in mitigating over-fitting and conserving computational resources. These findings underscore the necessity of carefully calibrating PC

TABLE 6. **Pairwise comparisons of monitoring strategies for the Xception model Using Conover-Iman Test**

Group Name	**no strategy**	**val_acc_3**	**val_accuracy_5**	**val_accuracy_7**	**val_accuracy_10**	**val_accuracy_15**	**val_loss_3**	**val_loss_5**	**val_loss_7**	**val_loss_10**	**val_loss_15**
no strategy (*original_no*)	1.000	**0.049**	0.425	0.687	0.690	0.935	**0.046**	0.383	0.236	0.425	0.991
val_accuracy_3	**0.049**	1.000	0.297	0.140	0.136	**0.046**	0.934	0.383	0.565	0.297	**0.049**
val_accuracy_5	0.425	0.297	1.000	0.687	0.687	0.383	0.236	0.934	0.687	0.991	0.425
val_accuracy_7	0.687	0.140	0.687	1.000	0.991	0.685	0.110	0.685	0.406	0.687	0.687
val_accuracy_10	0.690	0.136	0.687	0.991	1.000	0.687	0.110	0.653	0.383	0.687	0.687
val_accuracy_15	0.935	**0.046**	0.383	0.685	0.687	1.000	0.046	0.342	0.202	0.383	0.936
val_loss_3	**0.046**	0.934	0.236	0.110	0.110	**0.046**	1.000	0.297	0.434	0.236	**0.046**
val_loss_5	0.383	0.383	0.934	0.685	0.653	0.342	0.297	1.000	0.712	0.934	0.383
val_loss_7	0.236	0.565	0.687	0.406	0.383	0.202	0.434	0.712	1.000	0.687	0.236
val_loss_10	0.425	0.297	0.991	0.687	0.687	0.383	0.236	0.934	0.687	1.000	0.425
val_loss_15	0.991	**0.049**	0.425	0.687	0.687	0.936	0.046	0.383	0.236	0.425	1.000

to achieve a balance between computational efficiency and model performance. Building upon this, our results further confirm that early stopping consistently reduced training time across all four models (MobileNetV2, InceptionV3, ResNet50V2, and Xception), with statistically significant reductions observed in most cases. This demonstrates that early stopping effectively prevents unnecessary computations, contributing to energy conservation and aligning with the principles of green computing. However, the sensitivity of training time reduction to PC underscores the importance of carefully tuned parameters. For instance, shorter PCs drastically reduce epochs but may compromise the achievement of optimal weights for certain architectures, as evidenced by the significant test accuracy variations observed in Xception. These findings align with Chen's [8] assertion that "Algorithms aimed at early termination of training or dynamic adjustment of model complexity based on performance can prevent over-computation and conserve resources."

5.2. **Consistency of Model Accuracy:**

The findings underline that early stopping does not significantly degrade test accuracy for most models. Models like MobileNetV2 and ResNet50V2 exhibited stable performance across varying PCs, reaffirming the robustness of these architectures. However, Xception's test accuracy displayed significant

variability, indicating that the model's performance is more sensitive to the choice of PC and monitoring criteria. These results resonate with Patterson *et al.*'s perspective that prioritizing metrics beyond accuracy, such as training efficiency and carbon footprint, could foster innovations in ML algorithms, systems, and hardware, ultimately advancing both performance consistency and sustainability [7].

5.3. **Implications for Green AI:.**

The study supports the hypothesis that early stopping strategies contribute to reducing the environmental impact of AI training. By halting the training process earlier, models consume less energy, thereby reducing carbon footprints. This aligns with sustainable development goals and emphasizes the role of energy-efficient practices in AI research.

5.4. **Limitations and Challenges:**

While the findings are promising, several challenges remain:

Dataset Dependency: The experiments were conducted on the Rock Paper Scissors dataset. The generalizability of these findings to more complex datasets with higher feature dimensionality requires further exploration.

Model Sensitivity: The variability in performance for Xception suggests that some architectures might require additional adaptive mechanisms to ensure consistent accuracy while leveraging early stopping.

Energy Measurement: Although training time reductions imply energy savings, the study does not provide direct measurements of energy consumption, limiting the precision of conclusions about environmental impact.

6. Conclusion and Future Studies

The results of this study underscore the effectiveness of early stopping as a practical approach to enhancing the energy efficiency of deep learning model training. By preventing unnecessary epochs, early stopping significantly reduces training time and computational resources. The comparative analysis across multiple transfer learning architectures demonstrated consistent performance in accuracy, with substantial gains in energy savings. These outcomes highlight the role of energy-aware strategies in addressing the environmental challenges posed by the growing computational demands of artificial intelligence. Early stopping represents a straightforward yet impactful optimization that aligns well with global sustainability objectives, such as the United Nations' Sustainable Development Goals. This research provides a foundation for integrating such strategies into standard deep learning practices, paving the way for environmentally responsible AI development. To further advance the insights from this study, future research could focus on the following areas:

Broader Dataset Evaluation: Expand experiments to include larger and more diverse datasets, ensuring the generalizability of early stopping strategies across various domains and data types.

Direct Energy Measurement: Develop and employ methodologies to directly measure energy consumption during training, enabling a more precise assessment of energy savings.

Advanced Early Stopping Criteria: Investigate more sophisticated stopping mechanisms, such as adaptive PC thresholds and hybrid criteria combining multiple performance metrics.

Renewable Energy Integration: Evaluate the performance and energy efficiency of early stopping in data centers powered by renewable energy sources, providing insights into its sustainability impact under different energy scenarios.

Interdisciplinary Approaches: Collaborate across fields to incorporate principles of green computing into AI research, exploring synergies with smart grid technologies and sustainable data center designs.

Optimization for Real-Time Systems: Explore early stopping strategies in real-time AI applications, such as edge computing and Internet of Things (IoT) systems, where energy constraints are critical.

By addressing these areas, future studies could expand the application of energy-aware training strategies and enhance their effectiveness in promoting sustainable computing practices. This will not only benefit the AI community but also contribute to broader efforts toward reducing the environmental impact of advanced technologies.

DECLARATIONS

- **Contribution Rate Statement:** Abdulkadir TAŞDELEN has conducted this study as a single author.
- **Conflict of Interest:** The author has no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.
- **Data Availability:** The dataset has been properly cited, and a comprehensive analysis is included in the Appendices.
- **Statement of Support and Acknowledgment:** None.

REFERENCES

[1] K. I. Ibekwe, A. A. Umoh, Z. Q. S. Nwokediegwu, E. A. Etukudoh, V. I. Ilojianya, A. Adefemi, Energy efficiency in industrial sectors: A review of technologies and policy measures, Engineering Science & Technology Journal 5 (1) (2024) 169–184. doi:10.51594/estj.v5i1.742.

[2] A. Tasdelen, M. H. Habaebi, M. R. Islam, Exploring blockchain technologies: Insights into consensus mechanisms, mining pool dynamics, and energy consumption patterns, in: 2024 9th International Conference on Mechatronics Engineering (ICOM), IEEE, 2024, p. 95–100. doi:10.1109/icom61675.2024.10652588.

[3] V. Bolón-Canedo, L. Morán-Fernández, B. Cancela, A. Alonso-Betanzos, A review of green artificial intelligence: Towards a more sustainable future, Neurocomputing 599 (2024) 128096. doi:10.1016/j.neucom.2024.128096.

[4] Z. Vale, L. Gomes, D. Ramos, P. Faria, Green computing: a realistic evaluation of energy consumption for building load forecasting computation, Journal of Smart Environments and Green Computing 2 (2) (2022) 34–45. doi:10.20517/jsegc.2022.06.

[5] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. V. Esesn, A. A. S. Awwal, V. K. Asari, The history began from alexnet: A comprehensive survey on deep learning approaches (2018). arXiv:1803.01164. URL https://arxiv.org/abs/1803.01164

[6] Y. Zhou, X. Lin, X. Zhang, M. Wang, G. Jiang, H. Lu, Y. Wu, K. Zhang, Z. Yang, K. Wang, Y. Sui, F. Jia, Z. Tang, Y. Zhao, H. Zhang, T. Yang, W. Chen, Y. Mao, Y. Li, D. Bao, Y. Li, H. Liao, T. Liu, J. Liu, J. Guo, X. Zhao, Y. WEI, H. Qian, Q. Liu, X. Wang, W. Kin, Chan, C. Li, Y. Li, S. Yang, J. Yan, C. Mou, S. Han, W. Jin, G. Zhang, X. Zeng, On

the opportunities of green computing: A survey (2023). arXiv:2311.00447.
URL https://arxiv.org/abs/2311.00447

[7] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, J. Dean, Carbon emissions and large neural network training (2021). arXiv:2104.10350.
URL https://arxiv.org/abs/2104.10350

[8] X. Chen, Optimization strategies for reducing energy consumption in ai model training, ACS 6 (1) (Mar. 2023).

[9] L. Lannelongue, J. Grealey, M. Inouye, Green algorithms: Quantifying the carbon footprint of computation, Advanced Science 8 (12) (May 2021). doi:10.1002/advs.202100707.

[10] S. Georgiou, M. Kechagia, T. Sharma, F. Sarro, Y. Zou, Green ai: do deep learning frameworks have different costs?, in: Proceedings of the 44th International Conference on Software Engineering, ICSE '22, ACM, 2022, p. 1082–1094. doi:10.1145/3510003.3510221.

[11] B. M. Hussein, S. M. Shareef, An empirical study on the correlation between early stopping patience and epochs in deep learning, ITM Web of Conferences 64 (2024) 01003. doi:10.1051/itmconf/20246401003.

[12] Y. Xu, S. Martínez-Fernández, M. Martinez, X. Franch, Energy efficiency of training neural network architectures: An empirical study (Feb. 2023). arXiv:2302.00967.

[13] H. Järvenpää, P. Lago, J. Bogner, G. Lewis, H. Muccini, I. Ozkaya, A synthesis of green architectural tactics for ml-enabled systems, in: Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Society, ICSE-SEIS'24, ACM, 2024, p. 130–141. doi:10.1145/3639475.3640111.

[14] L. Maroney, Rock paper scissors classification dataset, https://laurencemoroney.com/datasets.html, accessed: 2024-10-17.

[15] W. H. Kruskal, W. A. Wallis, Use of ranks in one-criterion variance analysis, Journal of the American Statistical Association 47 (260) (1952) 583–621. doi:10.1080/01621459.1952.10483441.

[16] H. B. Mann, D. R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, The Annals of Mathematical Statistics 18 (1) (1947) 50–60.

[17] W. Conover, R. Iman, Multiple-comparisons procedures. Informal report, 1979. doi:10.2172/6057803.

[18] S. S. Acmali, Y. Ortakci, H. Seker, Green ai-driven concept for the development of cost-effective and energy-efficient deep learning method: Application in the detection ofeimeriaparasites as a case study, Advanced Intelligent Systems 6 (7) (Jun. 2024). doi:10.1002/aisy.202300644.

[19] D. Reguero, S. Martínez-Fernández, R. Verdecchia, Energy-efficient neural network training through runtime layer freezing, model quantization, and early stopping, Computer Standards & Interfaces 92 (2025) 103906. doi:10.1016/j.csi.2024.103906.

[20] Y. Matsubara, M. Levorato, F. Restuccia, Split computing and early exiting for deep learning applications: Survey and research challenges, ACM Computing Surveys 55 (5) (2022) 1–30. doi:10.1145/3527155.

[21] S. Teerapittayanon, B. McDanel, H. Kung, Branchynet: Fast inference via early exiting from deep neural networks, in: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, 2016, p. 2464–2469. doi:10.1109/icpr.2016.7900006.

[22] S. Teerapittayanon, B. McDanel, H. Kung, Distributed deep neural networks over the cloud, the edge and end devices, in: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2017, p. 328–339. doi:10.1109/icdcs.2017.226.

[23] Y. Wang, J. Shen, T.-K. Hu, P. Xu, T. Nguyen, R. Baraniuk, Z. Wang, Y. Lin, Dual dynamic inference: Enabling more efficient, adaptive, and controllable deep inference, IEEE Journal of Selected Topics in Signal Processing 14 (4) (2020) 623–633. doi:10.1109/jstsp.2020.2979669.

[24] H. Li, H. Zhang, X. Qi, R. Yang, G. Huang, Improved techniques for training adaptive deep networks (2019).
URL https://arxiv.org/abs/1908.06294

[25] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, N. D. Lane, Spinn: synergistic progressive inference of neural networks over device and cloud, in: Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, MobiCom '20, ACM, 2020, p. 1–15. doi:10.1145/3372224.3419194.
[26] Keras applications, https://keras.io/api/applications/, accessed: 2024-10-15.
[27] T. Fontanari, T. C. Fróes, M. Recamonde-Mendoza, Cross-validation Strategies for Balanced and Imbalanced Datasets, Springer International Publishing, 2022, p. 626–640. doi:10.1007/978-3-031-21686-2_43.
[28] D. Berrar, Cross-Validation, Elsevier, 2019, p. 542–545. doi:10.1016/b978-0-12-809633-8.20349-x.
[29] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: G. Gordon, D. Dunson, M. Dudík (Eds.), Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Vol. 15 of Proceedings of Machine Learning Research, PMLR, Fort Lauderdale, FL, USA, 2011, pp. 315–323.
URL https://proceedings.mlr.press/v15/glorot11a.html

APPENDIX (A)

TABLE A.1. Detailed results for MobileNetV2 with various MC and PC (Appendix A.1)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
3	loss	1	10	0.998046	0.02271	0.716724	0.61063	0.97265	0.07408	24.89913
3	loss	2	8	0.989741	0.039663	0.897611	0.407444	0.991453	0.046021	21.42021
3	loss	3	10	0.998046	0.02114	0.829352	0.383397	0.977778	0.058188	31.18301
3	loss	4	12	0.996092	0.0236	0.863481	0.39339	0.984615	0.052646	31.40667
3	loss	5	7	0.994138	0.034424	0.897611	0.27528	0.991453	0.046647	16.36674
5	loss	1	12	0.994626	0.022884	0.836177	0.407061	0.977778	0.061897	40.35201
5	loss	2	13	0.998046	0.017082	0.8157	0.505994	0.991453	0.037977	31.8802
5	loss	3	13	0.998534	0.017046	0.805461	0.482879	0.97265	0.070374	27.50178
5	loss	4	12	0.999023	0.018486	0.887372	0.335181	0.981197	0.054642	30.86905
5	loss	5	17	1	0.011072	0.894198	0.338853	0.989744	0.031267	34.95643
7	loss	1	20	0.999023	0.008709	0.829352	0.448269	0.982906	0.050633	48.03739
7	loss	2	13	0.998534	0.014379	0.853242	0.470612	0.984615	0.048726	29.33452
7	loss	3	15	0.998534	0.013665	0.921502	0.284624	0.984615	0.0479	35.74343
7	loss	4	26	1	0.006078	0.856655	0.471619	0.982906	0.059258	54.84729
7	loss	5	16	0.999023	0.010669	0.887372	0.27569	0.996581	0.025798	46.10693
10	loss	1	18	0.999023	0.009387	0.866894	0.40527	0.982906	0.061318	44.18391
10	loss	2	16	0.999511	0.009909	0.866894	0.479844	0.996581	0.032309	40.4024
10	loss	3	16	1	0.009493	0.846416	0.489362	0.982906	0.068769	48.54023
10	loss	4	22	0.999511	0.007761	0.860068	0.441097	0.981197	0.054182	56.33341
10	loss	5	23	0.999511	0.008148	0.87372	0.329433	0.991453	0.029512	54.65796
15	loss	1	34	1	0.003683	0.836177	0.465539	0.979487	0.054873	90.97743
15	loss	2	23	0.999023	0.006659	0.788396	0.644614	0.991453	0.044725	51.89995
15	loss	3	24	0.999511	0.007204	0.713311	0.718376	0.957265	0.098596	51.59951
15	loss	4	25	0.998046	0.007837	0.918089	0.349919	0.986325	0.04279	69.64649
15	loss	5	19	0.999023	0.00891	0.877133	0.413882	0.989744	0.062436	45.38881
3	accuracy	1	8	0.995115	0.035462	0.812287	0.452551	0.977778	0.075911	19.19615
3	accuracy	2	7	0.991695	0.03949	0.750853	0.558056	0.969231	0.086702	22.83207
3	accuracy	3	8	0.990718	0.044302	0.890785	0.298727	0.986325	0.06379	22.52161
3	accuracy	4	10	0.995603	0.028642	0.883959	0.357354	0.981197	0.063921	23.85584
3	accuracy	5	11	0.996092	0.022401	0.8157	0.428575	0.982906	0.049576	25.19012
5	accuracy	1	9	0.997557	0.02813	0.924915	0.235847	0.989744	0.052855	26.95682
5	accuracy	2	14	0.997069	0.01464	0.897611	0.355202	0.991453	0.031014	35.12124
5	accuracy	3	14	0.999023	0.014261	0.778157	0.556217	0.981197	0.0605	34.65687
5	accuracy	4	9	0.995115	0.028258	0.887372	0.304796	0.984615	0.071368	23.06869
5	accuracy	5	12	0.995603	0.019862	0.798635	0.499943	0.981197	0.054467	40.76033
7	accuracy	1	10	0.995603	0.027834	0.843003	0.472188	0.989744	0.073563	30.75511
7	accuracy	2	10	0.99658	0.025613	0.849829	0.40949	0.988034	0.07837	26.90913
7	accuracy	3	21	0.998534	0.007147	0.880546	0.373008	0.982906	0.051573	47.40013
7	accuracy	4	21	0.999511	0.00838	0.87372	0.394475	0.977778	0.060484	48.73546
7	accuracy	5	13	0.997069	0.016314	0.866894	0.455652	0.986325	0.056186	44.27132
10	accuracy	1	13	0.99658	0.018139	0.808874	0.418347	0.974359	0.100734	39.47001
10	accuracy	2	17	0.997069	0.011614	0.78157	0.638499	0.988034	0.051562	40.93397
10	accuracy	3	25	0.999511	0.00633	0.866894	0.357318	0.981197	0.046204	57.84523
10	accuracy	4	18	0.999511	0.010149	0.83959	0.434623	0.976068	0.061002	41.20016
10	accuracy	5	19	0.999511	0.006501	0.849829	0.45403	0.981197	0.050926	64.77641
15	accuracy	1	40	0.999511	0.002785	0.856655	0.440164	0.981197	0.055093	123.3092
15	accuracy	2	23	1	0.005775	0.822526	0.555798	0.993162	0.035381	66.58583
15	accuracy	3	24	1	0.005983	0.836177	0.522272	0.976068	0.063149	65.34113

(Continued on next page)

(Table A.1 Continued from previous page)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
15	accuracy	4	17	0.998534	0.009715	0.911263	0.307966	0.988034	0.120658	42.9821
15	accuracy	5	22	0.999511	0.006387	0.870307	0.375544	0.991453	0.04107	78.37715
-	no monitoring	1	N/A	1	0.001575	0.880546	0.457774	0.977778	0.075568	148.2634
-	no monitoring	2	N/A	1	0.002043	0.771331	0.749586	0.981197	0.048413	117.9772
-	no monitoring	3	N/A	1	0.001433	0.822526	0.563311	0.97265	0.091081	113.3214
-	no monitoring	4	N/A	1	0.001629	0.866894	0.492891	0.977778	0.079798	114.794
-	no monitoring	5	N/A	1	0.001867	0.856655	0.635862	0.982906	0.056227	114.0149

TABLE A.2. Descriptive statistics for MobileNetV2 model performance (Appendix A.2)

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_loss	3	**count**	5	5	5	5	5	5	5
val_loss	3	**mean**	0.995	0.028	0.841	0.414	0.984	0.056	25.055
val_loss	3	**std**	0.003	0.008	0.075	0.122	0.008	0.011	6.454
val_loss	3	**min**	0.990	0.021	0.717	0.275	0.973	0.046	16.367
val_loss	3	**25%**	0.994	0.023	0.829	0.383	0.978	0.047	21.420
val_loss	3	**50%**	0.996	0.024	0.863	0.393	0.985	0.053	24.899
val_loss	3	**75%**	0.998	0.034	0.898	0.407	0.991	0.058	31.183
val_loss	3	**max**	0.998	0.040	0.898	0.611	0.991	0.074	31.407
val_loss	5	**count**	5	5	5	5	5	5	5
val_loss	5	**mean**	0.998	0.017	0.848	0.414	0.983	0.051	33.112
val_loss	5	**std**	0.002	0.004	0.041	0.079	0.008	0.016	4.844
val_loss	5	**min**	0.995	0.011	0.805	0.335	0.973	0.031	27.502
val_loss	5	**25%**	0.998	0.017	0.816	0.339	0.978	0.038	30.869
val_loss	5	**50%**	0.999	0.017	0.836	0.407	0.981	0.055	31.880
val_loss	5	**75%**	0.999	0.018	0.887	0.483	0.990	0.062	34.956
val_loss	5	**max**	1.000	0.023	0.894	0.506	0.991	0.070	40.352
val_loss	7	**count**	5	5	5	5	5	5	5
val_loss	7	**mean**	0.999	0.011	0.870	0.390	0.986	0.046	42.814
val_loss	7	**std**	0.001	0.003	0.036	0.101	0.006	0.012	10.181
val_loss	7	**min**	0.999	0.006	0.829	0.276	0.983	0.026	29.335
val_loss	7	**25%**	0.999	0.009	0.853	0.285	0.983	0.048	35.743
val_loss	7	**50%**	0.999	0.011	0.857	0.448	0.985	0.049	46.107
val_loss	7	**75%**	0.999	0.014	0.887	0.471	0.985	0.051	48.037
val_loss	7	**max**	1.000	0.014	0.922	0.472	0.997	0.059	54.847
val_loss	10	**count**	5	5	5	5	5	5	5
val_loss	10	**mean**	1.000	0.009	0.863	0.429	0.987	0.049	48.824
val_loss	10	**std**	0.000	0.001	0.010	0.065	0.007	0.018	6.763
val_loss	10	**min**	0.999	0.008	0.846	0.329	0.981	0.030	40.402
val_loss	10	**25%**	1.000	0.008	0.860	0.405	0.983	0.032	44.184
val_loss	10	**50%**	1.000	0.009	0.867	0.441	0.983	0.054	48.540
val_loss	10	**75%**	1.000	0.009	0.867	0.480	0.991	0.061	54.658
val_loss	10	**max**	1.000	0.010	0.874	0.489	0.997	0.069	56.333
val_loss	15	**count**	5	5	5	5	5	5	5
val_loss	15	**mean**	0.999	0.007	0.827	0.518	0.981	0.061	61.902
val_loss	15	**std**	0.001	0.002	0.080	0.157	0.014	0.023	18.603
val_loss	15	**min**	0.998	0.004	0.713	0.350	0.957	0.043	45.389
val_loss	15	**25%**	0.999	0.007	0.788	0.414	0.979	0.045	51.600
val_loss	15	**50%**	0.999	0.007	0.836	0.466	0.986	0.055	51.900
val_loss	15	**75%**	1.000	0.008	0.877	0.645	0.990	0.062	69.646
val_loss	15	**max**	1.000	0.009	0.918	0.718	0.991	0.099	90.977

Table A.2 Continued from previous page

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_accuracy	3	**count**	5	5	5	5	5	5	5
val_accuracy	3	**mean**	0.994	0.034	0.831	0.419	0.979	0.068	22.719
val_accuracy	3	**std**	0.002	0.009	0.058	0.099	0.007	0.014	2.228
val_accuracy	3	**min**	0.991	0.022	0.751	0.299	0.969	0.050	19.196
val_accuracy	3	**25%**	0.992	0.029	0.812	0.357	0.978	0.064	22.522
val_accuracy	3	**50%**	0.995	0.035	0.816	0.429	0.981	0.064	22.832
val_accuracy	3	**75%**	0.996	0.039	0.884	0.453	0.983	0.076	23.856
val_accuracy	3	**max**	0.996	0.044	0.891	0.558	0.986	0.087	25.190
val_accuracy	5	**count**	5	5	5	5	5	5	5
val_accuracy	5	**mean**	0.997	0.021	0.857	0.390	0.986	0.054	32.113
val_accuracy	5	**std**	0.002	0.007	0.065	0.134	0.005	0.015	7.048
val_accuracy	5	**min**	0.995	0.014	0.778	0.236	0.981	0.031	23.069
val_accuracy	5	**25%**	0.996	0.015	0.799	0.305	0.981	0.053	26.957
val_accuracy	5	**50%**	0.997	0.020	0.887	0.355	0.985	0.054	34.657
val_accuracy	5	**75%**	0.998	0.028	0.898	0.500	0.990	0.060	35.121
val_accuracy	5	**max**	0.999	0.028	0.925	0.556	0.991	0.071	40.760
val_accuracy	7	**count**	5	5	5	5	5	5	5
val_accuracy	7	**mean**	0.997	0.017	0.863	0.421	0.985	0.064	39.614
val_accuracy	7	**std**	0.002	0.010	0.016	0.042	0.005	0.011	10.067
val_accuracy	7	**min**	0.996	0.007	0.843	0.373	0.978	0.052	26.909
val_accuracy	7	**25%**	0.997	0.008	0.850	0.394	0.983	0.056	30.755
val_accuracy	7	**50%**	0.997	0.016	0.867	0.409	0.986	0.060	44.271
val_accuracy	7	**75%**	0.999	0.026	0.874	0.456	0.988	0.074	47.400
val_accuracy	7	**max**	1.000	0.028	0.881	0.472	0.990	0.078	48.735
val_accuracy	10	**count**	5	5	5	5	5	5	5
val_accuracy	10	**mean**	0.998	0.011	0.829	0.461	0.980	0.062	48.845
val_accuracy	10	**std**	0.001	0.005	0.034	0.106	0.005	0.022	11.659
val_accuracy	10	**min**	0.997	0.006	0.782	0.357	0.974	0.046	39.470
val_accuracy	10	**25%**	0.997	0.007	0.809	0.418	0.976	0.051	40.934
val_accuracy	10	**50%**	1.000	0.010	0.840	0.435	0.981	0.052	41.200
val_accuracy	10	**75%**	1.000	0.012	0.850	0.454	0.981	0.061	57.845
val_accuracy	10	**max**	1.000	0.018	0.867	0.638	0.988	0.101	64.776
val_accuracy	15	**count**	5	5	5	5	5	5	5
val_accuracy	15	**mean**	1.000	0.006	0.859	0.440	0.986	0.063	75.319
val_accuracy	15	**std**	0.001	0.002	0.034	0.102	0.007	0.034	29.723
val_accuracy	15	**min**	0.999	0.003	0.823	0.308	0.976	0.035	42.982
val_accuracy	15	**25%**	1.000	0.006	0.836	0.376	0.981	0.041	65.341
val_accuracy	15	**50%**	1.000	0.006	0.857	0.440	0.988	0.055	66.586
val_accuracy	15	**75%**	1.000	0.006	0.870	0.522	0.991	0.063	78.377
val_accuracy	15	**max**	1.000	0.010	0.911	0.556	0.993	0.121	123.309
no_monitoring	No	**count**	5	5	5	5	5	5	5
no_monitoring	No	**mean**	1.000	0.002	0.840	0.580	0.978	0.070	121.674
no_monitoring	No	**std**	0.000	0.000	0.044	0.117	0.004	0.018	14.970
no_monitoring	No	**min**	1.000	0.001	0.771	0.458	0.973	0.048	113.321
no_monitoring	No	**25%**	1.000	0.002	0.823	0.493	0.978	0.056	114.015
no_monitoring	No	**50%**	1.000	0.002	0.857	0.563	0.978	0.076	114.794
no_monitoring	No	**75%**	1.000	0.002	0.867	0.636	0.981	0.080	117.977
no_monitoring	No	**max**	1.000	0.002	0.881	0.750	0.983	0.091	148.263

TABLE B.1. Detailed results for InceptionV3 with various MC and PC (Appendix B.1)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
3	loss	1	9	0.983878851	0.063424252	0.788395882	0.691795886	0.962393165	0.137821734	34.0100146
3	loss	2	8	0.989252567	0.040872496	0.733788371	0.860006809	0.979487181	0.088228665	29.8984313
3	loss	3	13	0.994137764	0.030255197	0.825938582	0.536911786	0.974358976	0.092576943	47.7669581
3	loss	4	7	0.986809969	0.060364712	0.679180861	0.84068495	0.960683763	0.123133942	28.6042797
3	loss	5	5	0.974596977	0.090318508	0.761092126	0.78642565	0.948717952	0.182484463	27.8326678
5	loss	1	32	0.999022961	0.00739672	0.853242338	0.577991843	0.984615386	0.070216581	117.0095957
5	loss	2	9	0.988764048	0.043512646	0.709897637	1.017861128	0.960683763	0.128654212	37.7095276
5	loss	3	14	0.997557402	0.02145461	0.726962447	0.860041678	0.970940173	0.120089293	53.3798204
5	loss	4	15	0.996580362	0.021638783	0.815699637	0.625344396	0.982905984	0.062162023	79.2635385
5	loss	5	15	0.997068882	0.019070214	0.883959055	0.518541694	0.982905984	0.08363767	66.42613
7	loss	1	19	0.996091843	0.019235797	0.771331072	0.681577265	0.977777779	0.080463678	78.0538619
7	loss	2	8	0.982901812	0.066090435	0.791808903	0.86845696	0.938461542	0.24873282	36.247579
7	loss	3	15	0.995114803	0.025213875	0.832764506	0.57030046	0.981196582	0.105363898	64.5173208
7	loss	4	15	0.997068882	0.019120131	0.849829376	0.585848749	0.979487181	0.069590084	88.7693096
7	loss	5	29	1	0.005879726	0.825938582	0.708134949	0.982905984	0.080251314	147.0300085
10	loss	1	20	0.997557402	0.015320398	0.788395882	0.742725492	0.979487181	0.077499501	78.6837716
10	loss	2	17	0.995603323	0.016541081	0.713310599	1.11474967	0.969230771	0.117452495	68.8851666
10	loss	3	35	0.99951148	0.005891902	0.829351544	0.576556921	0.981196582	0.094332993	153.3307557
10	loss	4	47	0.99951148	0.004547769	0.883959055	0.550591946	0.988034189	0.03828479	375.8302146
10	loss	5	34	0.999022961	0.005897738	0.91126281	0.403910398	0.98974359	0.057163749	237.1262536
15	loss	1	31	0.999022961	0.008081561	0.866894186	0.523374856	0.981196582	0.068272196	184.3225278
15	loss	2	26	0.999022961	0.010207516	0.761092126	1.104023457	0.976068377	0.084269352	121.4915938
15	loss	3	33	0.99951148	0.006300624	0.825938582	0.654499173	0.981196582	0.093615212	140.6734394
15	loss	4	26	0.994137764	0.017620478	0.781569958	0.797062635	0.974358976	0.081692502	121.4853378
15	loss	5	37	1	0.004083774	0.877133131	0.553094745	0.991452992	0.059600405	340.3184283
3	accuracy	1	17	0.995114803	0.0206411	0.808873713	0.725548565	0.970940173	0.102678411	140.3074823
3	accuracy	2	11	0.992183685	0.034354091	0.76791811	0.876152992	0.976068377	0.092387527	65.2484093
3	accuracy	3	4	0.965315104	0.124897584	0.815699637	0.536990285	0.933333337	0.271629155	25.8223064
3	accuracy	4	12	0.991695166	0.028814746	0.829351544	0.577801764	0.986324787	0.071021616	58.943908
3	accuracy	5	10	0.993649244	0.034925677	0.846416354	0.57682842	0.976068377	0.10173586	51.2127445
5	accuracy	1	20	0.995603323	0.015065268	0.829351544	0.646792293	0.981196582	0.082639068	154.8791887
5	accuracy	2	21	0.996091843	0.014288138	0.713310599	1.207827926	0.979487181	0.074940607	145.3536055
5	accuracy	3	22	0.997068882	0.01275574	0.798634827	0.676660776	0.972649574	0.118803278	137.5336144
5	accuracy	4	24	0.998534441	0.009583861	0.883959055	0.497871906	0.98974359	0.032772083	133.823382
5	accuracy	5	10	0.991206646	0.035472624	0.771331072	0.80417043	0.972649574	0.119832106	56.4036108
7	accuracy	1	28	0.998045921	0.007867916	0.798634827	0.770256519	0.981196582	0.089092769	154.7163437
7	accuracy	2	31	0.998534441	0.006880392	0.771331072	0.958002925	0.974358976	0.072101355	162.4083133
7	accuracy	3	22	0.997557402	0.010819421	0.860068262	0.450244308	0.974358976	0.093822978	190.629765
7	accuracy	4	36	1	0.003621859	0.819112599	0.797214568	0.981196582	0.05228468	278.316256
7	accuracy	5	22	0.99951148	0.009172016	0.829351544	0.701360941	0.986324787	0.077255376	169.3633921
10	accuracy	1	40	1	0.003867939	0.846416354	0.548633695	0.979487181	0.082988761	284.900779
10	accuracy	2	25	0.999022961	0.010678066	0.808873713	0.794085383	0.977777779	0.069033876	173.9952877
10	accuracy	3	39	0.99951148	0.0046171	0.849829376	0.599616289	0.977777779	0.11157576	247.5588835
10	accuracy	4	24	0.998045921	0.010440554	0.873720109	0.534663856	0.98974359	0.043696053	141.4675116
10	accuracy	5	31	0.998045921	0.009225765	0.907849848	0.497515827	0.98974359	0.068952538	271.3575743
15	accuracy	1	28	1	0.005100978	0.832764506	0.671540976	0.977777779	0.089632653	226.8121977
15	accuracy	2	50	1	0.002206131	0.778156996	1.169290781	0.976068377	0.07546217	379.2302903
15	accuracy	3	30	0.998534441	0.007315609	0.863481224	0.531567454	0.982905984	0.089072227	207.6188399

(Continued on next page)

(Table B.1 Continued from previous page)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
15	accuracy	4	44	1	0.00308408	0.877133131	0.580116212	0.986324787	0.046814781	282.2350076
15	accuracy	5	29	0.998534441	0.009574798	0.883959055	0.522773325	0.988034189	0.071746752	187.7881733
-	no monitoring	1	N/A	0.998534441	0.00314847	0.907849848	0.450603038	0.982905984	0.085012116	317.1966538
-	no monitoring	2	N/A	0.999022961	0.003437786	0.812286675	1.013028383	0.977777779	0.075830355	344.6338879
-	no monitoring	3	N/A	0.99951148	0.003941682	0.78498292	0.774207115	0.976068377	0.136283711	544.4648693
-	no monitoring	4	N/A	0.999022961	0.004351126	0.788395882	1.109158397	0.970940173	0.12248259	494.416349
-	no monitoring	5	N/A	0.99951148	0.002637709	0.883959055	0.515361607	0.98974359	0.076655284	381.0932598

TABLE B.2. Descriptive statistics for InceptionV3 model performance (Appendix B.2)

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_loss	3	**count**	5	5	5	5	5	5	5
val_loss	3	**mean**	0.986	0.057	0.758	0.743	0.965	0.125	33.622
val_loss	3	**std**	0.007	0.023	0.056	0.132	0.012	0.038	8.258
val_loss	3	**min**	0.975	0.030	0.679	0.537	0.949	0.088	27.833
val_loss	3	**25%**	0.984	0.041	0.734	0.692	0.961	0.093	28.604
val_loss	3	**50%**	0.987	0.060	0.761	0.786	0.962	0.123	29.898
val_loss	3	**75%**	0.989	0.063	0.788	0.841	0.974	0.138	34.010
val_loss	3	**max**	0.994	0.090	0.826	0.860	0.979	0.182	47.767
val_loss	5	**count**	5	5	5	5	5	5	5
val_loss	5	**mean**	0.996	0.023	0.798	0.720	0.976	0.093	70.758
val_loss	5	**std**	0.004	0.013	0.077	0.211	0.010	0.030	30.102
val_loss	5	**min**	0.989	0.007	0.710	0.519	0.961	0.062	37.710
val_loss	5	**25%**	0.997	0.019	0.727	0.578	0.971	0.070	53.380
val_loss	5	**50%**	0.997	0.021	0.816	0.625	0.983	0.084	66.426
val_loss	5	**75%**	0.998	0.022	0.853	0.860	0.983	0.120	79.264
val_loss	5	**max**	0.999	0.044	0.884	1.018	0.985	0.129	117.010
val_loss	7	**count**	5	5	5	5	5	5	5
val_loss	7	**mean**	0.994	0.027	0.814	0.683	0.972	0.117	82.924
val_loss	7	**std**	0.007	0.023	0.032	0.120	0.019	0.075	40.881
val_loss	7	**min**	0.983	0.006	0.771	0.570	0.938	0.070	36.248
val_loss	7	**25%**	0.995	0.019	0.792	0.586	0.978	0.080	64.517
val_loss	7	**50%**	0.996	0.019	0.826	0.682	0.979	0.080	78.054
val_loss	7	**75%**	0.997	0.025	0.833	0.708	0.981	0.105	88.769
val_loss	7	**max**	1.000	0.066	0.850	0.868	0.983	0.249	147.030
val_loss	10	**count**	5	5	5	5	5	5	5
val_loss	10	**mean**	0.998	0.010	0.825	0.678	0.982	0.077	182.771
val_loss	10	**std**	0.002	0.006	0.079	0.272	0.008	0.031	127.375
val_loss	10	**min**	0.996	0.005	0.713	0.404	0.969	0.038	68.885
val_loss	10	**25%**	0.998	0.006	0.788	0.551	0.979	0.057	78.684
val_loss	10	**50%**	0.999	0.006	0.829	0.577	0.981	0.077	153.331
val_loss	10	**75%**	1.000	0.015	0.884	0.743	0.988	0.094	237.126
val_loss	10	**max**	1.000	0.017	0.911	1.115	0.990	0.117	375.830
val_loss	15	**count**	5	5	5	5	5	5	5
val_loss	15	**mean**	0.998	0.009	0.823	0.726	0.981	0.077	181.658
val_loss	15	**std**	0.002	0.005	0.051	0.237	0.007	0.013	92.332
val_loss	15	**min**	0.994	0.004	0.761	0.523	0.974	0.060	121.485
val_loss	15	**25%**	0.999	0.006	0.782	0.553	0.976	0.068	121.492
val_loss	15	**50%**	0.999	0.008	0.826	0.654	0.981	0.082	140.673
val_loss	15	**75%**	1.000	0.010	0.867	0.797	0.981	0.084	184.323
val_loss	15	**max**	1.000	0.018	0.877	1.104	0.991	0.094	340.318

Table B.2 Continued from previous page

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_accuracy	3	**count**	5	5	5	5	5	5	5
val_accuracy	3	**mean**	0.988	0.049	0.814	0.659	0.969	0.128	68.307
val_accuracy	3	**std**	0.013	0.043	0.029	0.141	0.020	0.081	42.948
val_accuracy	3	**min**	0.965	0.021	0.768	0.537	0.933	0.071	25.822
val_accuracy	3	**25%**	0.992	0.029	0.809	0.577	0.971	0.092	51.213
val_accuracy	3	**50%**	0.992	0.034	0.816	0.578	0.976	0.102	58.944
val_accuracy	3	**75%**	0.994	0.035	0.829	0.726	0.976	0.103	65.248
val_accuracy	3	**max**	0.995	0.125	0.846	0.876	0.986	0.272	140.307
val_accuracy	5	**count**	5	5	5	5	5	5	5
val_accuracy	5	**mean**	0.996	0.017	0.799	0.767	0.979	0.086	125.599
val_accuracy	5	**std**	0.003	0.010	0.064	0.270	0.007	0.036	39.515
val_accuracy	5	**min**	0.991	0.010	0.713	0.498	0.973	0.033	56.404
val_accuracy	5	**25%**	0.996	0.013	0.771	0.647	0.973	0.075	133.823
val_accuracy	5	**50%**	0.996	0.014	0.799	0.677	0.979	0.083	137.534
val_accuracy	5	**75%**	0.997	0.015	0.829	0.804	0.981	0.119	145.354
val_accuracy	5	**max**	0.999	0.035	0.884	1.208	0.990	0.120	154.879
val_accuracy	7	**count**	5	5	5	5	5	5	5
val_accuracy	7	**mean**	0.999	0.008	0.816	0.735	0.979	0.077	191.087
val_accuracy	7	**std**	0.001	0.003	0.033	0.185	0.005	0.016	50.563
val_accuracy	7	**min**	0.998	0.004	0.771	0.450	0.974	0.052	154.716
val_accuracy	7	**25%**	0.998	0.007	0.799	0.701	0.974	0.072	162.408
val_accuracy	7	**50%**	0.999	0.008	0.819	0.770	0.981	0.077	169.363
val_accuracy	7	**75%**	1.000	0.009	0.829	0.797	0.981	0.089	190.630
val_accuracy	7	**max**	1.000	0.011	0.860	0.958	0.986	0.094	278.316
val_accuracy	10	**count**	5	5	5	5	5	5	5
val_accuracy	10	**mean**	0.999	0.008	0.857	0.595	0.983	0.075	223.856
val_accuracy	10	**std**	0.001	0.003	0.037	0.117	0.006	0.025	62.886
val_accuracy	10	**min**	0.998	0.004	0.809	0.498	0.978	0.044	141.468
val_accuracy	10	**25%**	0.998	0.005	0.846	0.535	0.978	0.069	173.995
val_accuracy	10	**50%**	0.999	0.009	0.850	0.549	0.979	0.069	247.559
val_accuracy	10	**75%**	1.000	0.010	0.874	0.600	0.990	0.083	271.358
val_accuracy	10	**max**	1.000	0.011	0.908	0.794	0.990	0.112	284.901
val_accuracy	15	**count**	5	5	5	5	5	5	5
val_accuracy	15	**mean**	0.999	0.005	0.847	0.695	0.982	0.075	256.737
val_accuracy	15	**std**	0.001	0.003	0.043	0.272	0.005	0.017	77.001
val_accuracy	15	**min**	0.999	0.002	0.778	0.523	0.976	0.047	187.788
val_accuracy	15	**25%**	0.999	0.003	0.833	0.532	0.978	0.072	207.619
val_accuracy	15	**50%**	1.000	0.005	0.863	0.580	0.983	0.075	226.812
val_accuracy	15	**75%**	1.000	0.007	0.877	0.672	0.986	0.089	282.235
val_accuracy	15	**max**	1.000	0.010	0.884	1.169	0.988	0.090	379.230
no_monitoring	No	**count**	5	5	5	5	5	5	5
no_monitoring	No	**mean**	0.999	0.004	0.835	0.772	0.979	0.099	416.361
no_monitoring	No	**std**	0.000	0.001	0.057	0.292	0.007	0.028	98.394
no_monitoring	No	**min**	0.999	0.003	0.785	0.451	0.971	0.076	317.197
no_monitoring	No	**25%**	0.999	0.003	0.788	0.515	0.976	0.077	344.634
no_monitoring	No	**50%**	0.999	0.003	0.812	0.774	0.978	0.085	381.093
no_monitoring	No	**75%**	1.000	0.004	0.884	1.013	0.983	0.122	494.416
no_monitoring	No	**max**	1.000	0.004	0.908	1.109	0.990	0.136	544.465

TABLE C.1. Detailed results for ResNet50V2 with various MC and PC (Appendix C.1)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
3	loss	1	18	0.998046	0.02271	0.716724	0.61063	0.97265	0.07408	24.89913
3	loss	2	18	0.989741	0.039663	0.897611	0.407444	0.991453	0.046021	21.42021
3	loss	3	18	0.998046	0.02114	0.829352	0.383397	0.977778	0.058188	31.18301
3	loss	4	20	0.996092	0.0236	0.863481	0.39339	0.984615	0.052646	31.40667
3	loss	5	15	0.994138	0.034424	0.897611	0.27528	0.991453	0.046647	16.36674
5	loss	1	16	0.994626	0.022884	0.836177	0.407061	0.977778	0.061897	40.35201
5	loss	2	16	0.998046	0.017082	0.8157	0.505994	0.991453	0.037977	31.8802
5	loss	3	21	0.998534	0.017046	0.805461	0.482879	0.97265	0.070374	27.50178
5	loss	4	12	0.999023	0.018486	0.887372	0.335181	0.981197	0.054642	30.86905
5	loss	5	15	1	0.011072	0.894198	0.338853	0.989744	0.031267	34.95643
7	loss	1	18	0.999023	0.008709	0.829352	0.448269	0.982906	0.050633	48.03739
7	loss	2	37	0.998534	0.014379	0.853242	0.470612	0.984615	0.048726	29.33452
7	loss	3	19	0.998534	0.013665	0.921502	0.284624	0.984615	0.0479	35.74343
7	loss	4	23	1	0.006078	0.856655	0.471619	0.982906	0.059258	54.84729
7	loss	5	36	0.999023	0.010669	0.887372	0.27569	0.996581	0.025798	46.10693
10	loss	1	27	0.999023	0.009387	0.866894	0.40527	0.982906	0.061318	44.18391
10	loss	2	27	0.999511	0.009909	0.866894	0.479844	0.996581	0.032309	40.4024
10	loss	3	27	1	0.009493	0.846416	0.489362	0.982906	0.068769	48.54023
10	loss	4	18	0.999511	0.007761	0.860068	0.441097	0.981197	0.054182	56.33341
10	loss	5	32	0.999511	0.008148	0.87372	0.329433	0.991453	0.029512	54.65796
15	loss	1	50	1	0.003683	0.836177	0.465539	0.979487	0.054873	90.97743
15	loss	2	48	0.999023	0.006659	0.788396	0.644614	0.991453	0.044725	51.89995
15	loss	3	33	0.999511	0.007204	0.713311	0.718376	0.957265	0.098596	51.59951
15	loss	4	49	0.998046	0.007837	0.918089	0.349919	0.986325	0.04279	69.64649
15	loss	5	28	0.999023	0.00891	0.877133	0.413882	0.989744	0.062436	45.38881
3	accuracy	1	17	0.995115	0.035462	0.812287	0.452551	0.977778	0.075911	19.19615
3	accuracy	2	8	0.991695	0.03949	0.750853	0.558056	0.969231	0.086702	22.83207
3	accuracy	3	9	0.990718	0.044302	0.890785	0.298727	0.986325	0.06379	22.52161
3	accuracy	4	10	0.995603	0.028642	0.883959	0.357354	0.981197	0.063921	23.85584
3	accuracy	5	14	0.996092	0.022401	0.8157	0.428575	0.982906	0.049576	25.19012
5	accuracy	1	16	0.997557	0.02813	0.924915	0.235847	0.989744	0.052855	26.95682
5	accuracy	2	14	0.997069	0.01464	0.897611	0.355202	0.991453	0.031014	35.12124
5	accuracy	3	14	0.999023	0.014261	0.778157	0.556217	0.981197	0.0605	34.65687
5	accuracy	4	25	0.995115	0.028258	0.887372	0.304796	0.984615	0.071368	23.06869
5	accuracy	5	12	0.995603	0.019862	0.798635	0.499943	0.981197	0.054467	40.76033
7	accuracy	1	17	0.995603	0.027834	0.843003	0.472188	0.989744	0.073563	30.75511
7	accuracy	2	20	0.99658	0.025613	0.849829	0.40949	0.988034	0.07837	26.90913
7	accuracy	3	16	0.998534	0.007147	0.880546	0.373008	0.982906	0.051573	47.40013
7	accuracy	4	30	0.999511	0.00838	0.87372	0.394475	0.977778	0.060484	48.73546
7	accuracy	5	13	0.997069	0.016314	0.866894	0.455652	0.986325	0.056186	44.27132
10	accuracy	1	26	0.99658	0.018139	0.808874	0.418347	0.974359	0.100734	39.47001
10	accuracy	2	15	0.997069	0.011614	0.78157	0.638499	0.988034	0.051562	40.93397
10	accuracy	3	18	0.999511	0.00633	0.866894	0.357318	0.981197	0.046204	57.84523
10	accuracy	4	25	0.999511	0.010149	0.83959	0.434623	0.976068	0.061002	41.20016
10	accuracy	5	33	0.999511	0.006501	0.849829	0.45403	0.981197	0.050926	64.77641
15	accuracy	1	50	0.999511	0.002785	0.856655	0.440164	0.981197	0.055093	123.3092
15	accuracy	2	26	1	0.005775	0.822526	0.555798	0.993162	0.035381	66.58583
15	accuracy	3	26	1	0.005983	0.836177	0.522272	0.976068	0.063149	65.34113

(Continued on next page)

(Table C.1 Continued from previous page)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
15	accuracy	4	45	0.998534	0.009715	0.911263	0.307966	0.988034	0.120658	42.9821
15	accuracy	5	35	0.999511	0.006387	0.870307	0.375544	0.991453	0.04107	78.37715
-	no monitoring	1	N/A	1	0.001575	0.880546	0.457774	0.977778	0.075568	148.2634
-	no monitoring	2	N/A	1	0.002043	0.771331	0.749586	0.981197	0.048413	117.9772
-	no monitoring	3	N/A	1	0.001433	0.822526	0.563311	0.97265	0.091081	113.3214
-	no monitoring	4	N/A	1	0.001629	0.866894	0.492891	0.977778	0.079798	114.794
-	no monitoring	5	N/A	1	0.001867	0.856655	0.635862	0.982906	0.056227	114.0149

TABLE C.2. Descriptive statistics for ResNet50V2 model performance (Appendix C.2)

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_loss	3	**count**	5	5	5	5	5	5	5
val_loss	3	**mean**	0.995	0.026	0.958	0.121	0.994	0.026	139.693
val_loss	3	**std**	0.002	0.005	0.007	0.010	0.004	0.006	14.402
val_loss	3	**min**	0.993	0.017	0.949	0.108	0.988	0.020	126.999
val_loss	3	**25%**	0.994	0.026	0.952	0.117	0.990	0.020	132.744
val_loss	3	**50%**	0.996	0.026	0.959	0.123	0.995	0.023	133.583
val_loss	3	**75%**	0.996	0.029	0.962	0.123	0.997	0.031	141.364
val_loss	3	**max**	0.998	0.030	0.966	0.136	0.998	0.033	163.777
val_loss	5	**count**	5	5	5	5	5	5	5
val_loss	5	**mean**	0.993	0.032	0.918	0.194	0.991	0.044	127.430
val_loss	5	**std**	0.004	0.012	0.034	0.069	0.006	0.018	26.791
val_loss	5	**min**	0.988	0.018	0.884	0.107	0.983	0.029	89.900
val_loss	5	**25%**	0.991	0.024	0.887	0.153	0.986	0.029	112.390
val_loss	5	**50%**	0.995	0.031	0.925	0.183	0.995	0.037	134.345
val_loss	5	**75%**	0.997	0.041	0.928	0.256	0.995	0.060	141.700
val_loss	5	**max**	0.997	0.046	0.966	0.271	0.995	0.066	158.813
val_loss	7	**count**	5	5	5	5	5	5	5
val_loss	7	**mean**	0.997	0.015	0.923	0.166	0.993	0.026	197.787
val_loss	7	**std**	0.002	0.007	0.031	0.046	0.002	0.008	60.948
val_loss	7	**min**	0.996	0.007	0.870	0.131	0.990	0.017	140.188
val_loss	7	**25%**	0.996	0.007	0.922	0.142	0.991	0.021	150.817
val_loss	7	**50%**	0.996	0.017	0.935	0.151	0.993	0.022	173.366
val_loss	7	**75%**	0.999	0.020	0.939	0.162	0.995	0.033	247.801
val_loss	7	**max**	1.000	0.022	0.949	0.246	0.995	0.037	276.762
val_loss	10	**count**	5	5	5	5	5	5	5
val_loss	10	**mean**	0.997	0.016	0.891	0.232	0.987	0.040	201.758
val_loss	10	**std**	0.001	0.006	0.091	0.204	0.017	0.038	38.513
val_loss	10	**min**	0.996	0.010	0.730	0.115	0.957	0.020	142.963
val_loss	10	**25%**	0.996	0.011	0.908	0.127	0.993	0.023	201.046
val_loss	10	**50%**	0.997	0.016	0.932	0.139	0.993	0.025	203.378
val_loss	10	**75%**	0.999	0.018	0.939	0.184	0.995	0.025	210.735
val_loss	10	**max**	0.999	0.024	0.949	0.593	0.997	0.109	250.669
val_loss	15	**count**	5	5	5	5	5	5	5
val_loss	15	**mean**	0.999	0.008	0.934	0.133	0.996	0.019	364.172
val_loss	15	**std**	0.001	0.003	0.025	0.033	0.003	0.008	78.947
val_loss	15	**min**	0.996	0.003	0.894	0.104	0.991	0.013	273.448
val_loss	15	**25%**	0.999	0.006	0.928	0.112	0.995	0.015	287.525
val_loss	15	**50%**	0.999	0.007	0.939	0.121	0.997	0.015	392.789
val_loss	15	**75%**	1.000	0.011	0.949	0.141	0.997	0.016	419.770
val_loss	15	**max**	1.000	0.011	0.959	0.187	1.000	0.034	447.328

Table C.2 Continued from previous page

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_accuracy	3	**count**	5	5	5	5	5	5	5
val_accuracy	3	**mean**	0.985	0.053	0.939	0.141	0.983	0.056	151.258
val_accuracy	3	**std**	0.009	0.025	0.042	0.062	0.011	0.023	33.598
val_accuracy	3	**min**	0.972	0.027	0.891	0.074	0.974	0.029	120.028
val_accuracy	3	**25%**	0.982	0.034	0.904	0.093	0.976	0.040	133.953
val_accuracy	3	**50%**	0.986	0.053	0.939	0.126	0.976	0.062	139.552
val_accuracy	3	**75%**	0.991	0.065	0.980	0.197	0.990	0.063	155.989
val_accuracy	3	**max**	0.995	0.088	0.983	0.214	0.998	0.088	206.765
val_accuracy	5	**count**	5	5	5	5	5	5	5
val_accuracy	5	**mean**	0.993	0.031	0.939	0.132	0.990	0.037	176.910
val_accuracy	5	**std**	0.003	0.013	0.021	0.043	0.003	0.009	53.507
val_accuracy	5	**min**	0.989	0.012	0.915	0.095	0.986	0.027	142.929
val_accuracy	5	**25%**	0.992	0.031	0.922	0.097	0.988	0.035	146.108
val_accuracy	5	**50%**	0.993	0.031	0.945	0.113	0.990	0.037	147.014
val_accuracy	5	**75%**	0.994	0.035	0.949	0.165	0.991	0.037	179.683
val_accuracy	5	**max**	0.998	0.049	0.966	0.189	0.995	0.050	268.818
val_accuracy	7	**count**	5	5	5	5	5	5	5
val_accuracy	7	**mean**	0.994	0.028	0.941	0.137	0.991	0.040	238.396
val_accuracy	7	**std**	0.004	0.013	0.046	0.084	0.005	0.019	100.550
val_accuracy	7	**min**	0.988	0.012	0.860	0.075	0.986	0.021	162.213
val_accuracy	7	**25%**	0.991	0.019	0.949	0.084	0.986	0.030	190.180
val_accuracy	7	**50%**	0.995	0.028	0.962	0.110	0.991	0.032	193.103
val_accuracy	7	**75%**	0.996	0.034	0.962	0.134	0.991	0.045	234.178
val_accuracy	7	**max**	0.999	0.046	0.969	0.280	0.998	0.071	412.303
val_accuracy	10	**count**	5	5	5	5	5	5	5
val_accuracy	10	**mean**	0.997	0.018	0.955	0.106	0.991	0.036	303.144
val_accuracy	10	**std**	0.002	0.009	0.010	0.021	0.006	0.020	103.501
val_accuracy	10	**min**	0.994	0.010	0.945	0.073	0.981	0.020	185.947
val_accuracy	10	**25%**	0.997	0.013	0.945	0.102	0.991	0.023	217.471
val_accuracy	10	**50%**	0.999	0.015	0.956	0.113	0.991	0.028	315.892
val_accuracy	10	**75%**	0.999	0.020	0.959	0.121	0.993	0.038	356.140
val_accuracy	10	**max**	0.999	0.034	0.969	0.124	0.998	0.069	440.271
val_accuracy	15	**count**	5	5	5	5	5	5	5
val_accuracy	15	**mean**	0.999	0.009	0.919	0.166	0.992	0.025	520.313
val_accuracy	15	**std**	0.002	0.007	0.040	0.071	0.003	0.006	152.293
val_accuracy	15	**min**	0.997	0.002	0.874	0.089	0.988	0.019	356.875
val_accuracy	15	**25%**	0.998	0.005	0.891	0.092	0.991	0.020	384.393
val_accuracy	15	**50%**	0.999	0.007	0.911	0.189	0.991	0.025	514.190
val_accuracy	15	**75%**	1.000	0.015	0.952	0.227	0.993	0.030	651.946
val_accuracy	15	**max**	1.000	0.018	0.969	0.234	0.997	0.033	694.159
no_monitoring	No	**count**	5	5	5	5	5	5	5
no_monitoring	No	**mean**	1.000	0.004	0.911	0.209	0.985	0.033	723.026
no_monitoring	No	**std**	0.000	0.001	0.057	0.147	0.010	0.026	41.621
no_monitoring	No	**min**	0.999	0.004	0.829	0.102	0.969	0.014	694.063
no_monitoring	No	**25%**	1.000	0.004	0.874	0.103	0.985	0.016	705.354
no_monitoring	No	**50%**	1.000	0.004	0.942	0.128	0.988	0.021	707.042
no_monitoring	No	**75%**	1.000	0.005	0.949	0.271	0.991	0.035	712.132
no_monitoring	No	**max**	1.000	0.005	0.962	0.440	0.993	0.076	796.537

TABLE D.1. Detailed results for Xception with various MC and PC (Appendix D.1)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
3	loss	1	9	0.972642899	0.129837066	0.648464143	0.726279438	0.94700855	0.195763454	64.5046287
3	loss	2	9	0.965803623	0.145221606	0.713310599	0.582341194	0.940170944	0.190947622	66.8170586
3	loss	3	8	0.969711781	0.121941559	0.662116051	0.679573655	0.931623936	0.209745377	62.8957683
3	loss	4	11	0.98534441	0.084378332	0.737201393	0.631100476	0.938461542	0.164128602	86.5575404
3	loss	5	10	0.979482174	0.099705689	0.69624573	0.643902957	0.952136755	0.161598474	76.0659985
5	loss	1	14	0.98583293	0.070808701	0.692832768	0.699655771	0.95726496	0.147542983	108.2439275
5	loss	2	12	0.989252567	0.076574892	0.709897637	0.663573325	0.964102566	0.136994839	91.7473245
5	loss	3	16	0.990229607	0.064519256	0.69624573	0.740498126	0.960683763	0.13576889	122.3358252
5	loss	4	21	0.993649244	0.04727627	0.757679164	0.623574436	0.945299149	0.132990971	162.5682723
5	loss	5	14	0.989741087	0.066146471	0.740614355	0.570958138	0.962393165	0.135764197	105.715716
7	loss	1	19	0.994137764	0.043756392	0.720136523	0.658100367	0.95726496	0.122135416	145.7735281
7	loss	2	12	0.979482174	0.105163231	0.668941975	0.66026026	0.928205132	0.212660953	96.5790529
7	loss	3	17	0.992672205	0.058356903	0.648464143	0.803342044	0.95726496	0.13514547	127.8505467
7	loss	4	15	0.989741087	0.064322673	0.757679164	0.545565844	0.943589747	0.165480867	119.8481848
7	loss	5	19	0.995603323	0.042332146	0.726962447	0.64823091	0.969230771	0.11589212	136.9359018
10	loss	1	23	0.996580362	0.032822847	0.716723561	0.705110371	0.965811968	0.111317508	172.6875957
10	loss	2	25	0.996091843	0.037626501	0.692832768	0.806456029	0.972649574	0.096915461	192.6734316
10	loss	3	16	0.991206646	0.064496234	0.713310599	0.656079113	0.950427353	0.195622265	122.5274311
10	loss	4	33	0.998045921	0.023534197	0.757679164	0.660158396	0.953846157	0.11040625	262.3697482
10	loss	5	17	0.992183685	0.052593071	0.713310599	0.668919325	0.94700855	0.171621963	127.6774122
15	loss	1	29	0.998534441	0.023043793	0.757679164	0.580849946	0.967521369	0.108793311	214.6257377
15	loss	2	23	0.996580362	0.036374774	0.713310599	0.73689121	0.962393165	0.131636575	174.2537773
15	loss	3	32	0.997557402	0.022166649	0.747440279	0.742685497	0.970940173	0.097075753	252.1042346
15	loss	4	30	0.999022961	0.022557253	0.754266202	0.713742137	0.953846157	0.121158503	250.3157561
15	loss	5	30	0.998534441	0.023030076	0.730375409	0.734522462	0.972649574	0.107649416	276.7263841
3	accuracy	1	15	0.993649244	0.061796885	0.69624573	0.670105934	0.962393165	0.130404502	142.4387054
3	accuracy	2	5	0.957498789	0.191257089	0.621160388	0.741892576	0.928205132	0.319094092	50.9358952
3	accuracy	3	7	0.966780663	0.159885839	0.737201393	0.562528431	0.935042739	0.244990811	65.6215291
3	accuracy	4	17	0.994137764	0.045600958	0.774744034	0.577316344	0.952136755	0.127913699	154.4101072
3	accuracy	5	6	0.959941387	0.16798079	0.675767899	0.688473523	0.921367526	0.265415251	65.0164075
5	accuracy	1	16	0.990718126	0.060059696	0.692832768	0.812040746	0.965811968	0.127139732	149.1743443
5	accuracy	2	10	0.975085497	0.113430224	0.651877105	0.691287875	0.931623936	0.21590881	76.2116969
5	accuracy	3	15	0.989741087	0.065249637	0.744027317	0.619207919	0.965811968	0.122518174	123.6166012
5	accuracy	4	22	0.994137764	0.040192954	0.761092126	0.607570052	0.948717952	0.125213459	170.1709438
5	accuracy	5	17	0.991695166	0.052661575	0.730375409	0.589609325	0.969230771	0.109279864	130.5702227
7	accuracy	1	25	0.996091843	0.032614194	0.69624573	0.786663473	0.967521369	0.111560121	190.436003
7	accuracy	2	17	0.991206646	0.05449627	0.682593882	0.746500134	0.969230771	0.112729013	130.1836983
7	accuracy	3	19	0.992672205	0.047206469	0.672354937	0.820098698	0.962393165	0.125569016	162.6656513
7	accuracy	4	9	0.980459213	0.111395694	0.69624573	0.665015817	0.907692313	0.355802476	74.0074007
7	accuracy	5	38	0.999022961	0.015742909	0.740614355	0.766431868	0.970940173	0.089645736	297.7733321
10	accuracy	1	34	0.997557402	0.020063197	0.675767899	0.878209054	0.969230771	0.105710268	277.3612384
10	accuracy	2	31	0.999022961	0.020484067	0.686006844	0.948435187	0.969230771	0.086290933	252.9607732
10	accuracy	3	22	0.995114803	0.037949301	0.737201393	0.683958352	0.958974361	0.119537391	187.3716481
10	accuracy	4	30	0.99951148	0.022190876	0.78498292	0.569951594	0.952136755	0.123341084	241.4252356
10	accuracy	5	17	0.995114803	0.043765735	0.726962447	0.678048849	0.964102566	0.145041451	137.2071482
15	accuracy	1	46	0.998534441	0.010064815	0.720136523	0.874718249	0.974358976	0.100223176	365.9279238
15	accuracy	2	50	0.999022961	0.010573568	0.709897637	0.986595809	0.972649574	0.073904678	388.8055961
15	accuracy	3	29	0.998045921	0.023716006	0.703071654	0.846732259	0.958974361	0.121186987	273.9335451

(Continued on next page)

(Table D.1 Continued from previous page)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
15	accuracy	4	26	0.997068882	0.028543573	0.76791811	0.628445148	0.952136755	0.132668525	234.2733131
15	accuracy	5	36	0.998534441	0.022074558	0.740614355	0.778797984	0.970940173	0.098766908	306.510873
-	no monitoring	1	N/A	0.999022961	0.010553496	0.706484616	0.941983461	0.974358976	0.118602358	439.8904622
-	no monitoring	2	N/A	1	0.01098085	0.686006844	1.034381628	0.967521369	0.094622567	437.3137553
-	no monitoring	3	N/A	0.99951148	0.010696846	0.75085324	0.825193703	0.962393165	0.125199303	396.7657198
-	no monitoring	4	N/A	0.999022961	0.011571754	0.764505148	0.73913306	0.952136755	0.154828563	463.3622841
-	no monitoring	5	N/A	0.99951148	0.008148649	0.75085324	0.829228759	0.970940173	0.104553312	408.4304722

TABLE D.2. Descriptive statistics for Xception model Performance (Appendix D.2)

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_loss	3	**count**	5	5	5	5	5	5	5
val_loss	3	**mean**	0.975	0.116	0.691	0.653	0.942	0.184	71.368
val_loss	3	**std**	0.008	0.024	0.036	0.054	0.008	0.021	9.904
val_loss	3	**min**	0.966	0.084	0.648	0.582	0.932	0.162	62.896
val_loss	3	**25%**	0.970	0.100	0.662	0.631	0.938	0.164	64.505
val_loss	3	**50%**	0.973	0.122	0.696	0.644	0.940	0.191	66.817
val_loss	3	**75%**	0.979	0.130	0.713	0.680	0.947	0.196	76.066
val_loss	3	**max**	0.985	0.145	0.737	0.726	0.952	0.210	86.558
val_loss	5	**count**	v	.5	.5	.5	.5	.5	.5
val_loss	5	**mean**	0.990	0.065	0.719	0.660	0.958	0.138	118.122
val_loss	5	**std**	0.003	0.011	0.028	0.066	0.008	0.006	27.112
val_loss	5	**min**	0.986	0.047	0.693	0.571	0.945	0.133	91.747
val_loss	5	**25%**	0.989	0.065	0.696	0.624	0.957	0.136	105.716
val_loss	5	**50%**	0.990	0.066	0.710	0.664	0.961	0.136	108.244
val_loss	5	**75%**	0.990	0.071	0.741	0.700	0.962	0.137	122.336
val_loss	5	**max**	0.994	0.077	0.758	0.740	0.964	0.148	162.568
val_loss	7	**count**	5	5	5	5	5	5	5
val_loss	7	**mean**	0.990	0.063	0.704	0.663	0.951	0.150	125.397
val_loss	7	**std**	0.006	0.025	0.045	0.092	0.016	0.040	18.812
val_loss	7	**min**	0.979	0.042	0.648	0.546	0.928	0.116	96.579
val_loss	7	**25%**	0.990	0.044	0.669	0.648	0.944	0.122	119.848
val_loss	7	**50%**	0.993	0.058	0.720	0.658	0.957	0.135	127.851
val_loss	7	**75%**	0.994	0.064	0.727	0.660	0.957	0.165	136.936
val_loss	7	**max**	0.996	0.105	0.758	0.803	0.969	0.213	145.774
val_loss	10	**count**	5	5	5	5	5	5	5
val_loss	10	**mean**	0.995	0.042	0.719	0.699	0.958	0.137	175.587
val_loss	10	**std**	0.003	0.016	0.024	0.063	0.011	0.044	56.882
val_loss	10	**min**	0.991	0.024	0.693	0.656	0.947	0.097	122.527
val_loss	10	**25%**	0.992	0.033	0.713	0.660	0.950	0.110	127.677
val_loss	10	**50%**	0.996	0.038	0.713	0.669	0.954	0.111	172.688
val_loss	10	**75%**	0.997	0.053	0.717	0.705	0.966	0.172	192.673
val_loss	10	**max**	0.998	0.064	0.758	0.806	0.973	0.196	262.370
val_loss	15	**count**	5	5	5	5	5	5	5
val_loss	15	**mean**	0.998	0.025	0.741	0.702	0.965	0.113	233.605
val_loss	15	**std**	0.001	0.006	0.019	0.068	0.008	0.013	39.886
val_loss	15	**min**	0.997	0.022	0.713	0.581	0.954	0.097	174.254
val_loss	15	**25%**	0.998	0.023	0.730	0.714	0.962	0.108	214.626
val_loss	15	**50%**	0.999	0.023	0.747	0.735	0.968	0.109	250.316
val_loss	15	**75%**	0.999	0.023	0.754	0.737	0.971	0.121	252.104
val_loss	15	**max**	0.999	0.036	0.758	0.743	0.973	0.132	276.726

Table D.2 Continued from previous page

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_accuracy	3	**count**	5	5	5	5	5	5	5
val_accuracy	3	**mean**	0.974	0.125	0.701	0.648	0.940	0.218	95.685
val_accuracy	3	**std**	0.018	0.067	0.059	0.076	0.017	0.085	48.686
val_accuracy	3	**min**	0.957	0.046	0.621	0.563	0.921	0.128	50.936
val_accuracy	3	**25%**	0.960	0.062	0.676	0.577	0.928	0.130	65.016
val_accuracy	3	**50%**	0.967	0.160	0.696	0.670	0.935	0.245	65.622
val_accuracy	3	**75%**	0.994	0.168	0.737	0.688	0.952	0.265	142.439
val_accuracy	3	**max**	0.994	0.191	0.775	0.742	0.962	0.319	154.410
val_accuracy	5	**count**	5	5	5	5	5	5	5
val_accuracy	5	**mean**	0.988	0.066	0.716	0.664	0.956	0.140	129.949
val_accuracy	5	**std**	0.008	0.028	0.044	0.091	0.016	0.043	35.056
val_accuracy	5	**min**	0.975	0.040	0.652	0.590	0.932	0.109	76.212
val_accuracy	5	**25%**	0.990	0.053	0.693	0.608	0.949	0.123	123.617
val_accuracy	5	**50%**	0.991	0.060	0.730	0.619	0.966	0.125	130.570
val_accuracy	5	**75%**	0.992	0.065	0.744	0.691	0.966	0.127	149.174
val_accuracy	5	**max**	0.994	0.113	0.761	0.812	0.969	0.216	170.171
val_accuracy	7	**count**	5	5	5	5	5	5	5
val_accuracy	7	**mean**	0.992	0.052	0.698	0.757	0.956	0.159	171.013
val_accuracy	7	**std**	0.007	0.036	0.026	0.058	0.027	0.111	83.055
val_accuracy	7	**min**	0.980	0.016	0.672	0.665	0.908	0.090	74.007
val_accuracy	7	**25%**	0.991	0.033	0.683	0.747	0.962	0.112	130.184
val_accuracy	7	**50%**	0.993	0.047	0.696	0.766	0.968	0.113	162.666
val_accuracy	7	**75%**	0.996	0.054	0.696	0.787	0.969	0.126	190.436
val_accuracy	7	**max**	0.999	0.111	0.741	0.820	0.971	0.356	297.773
val_accuracy	10	**count**	5	5	5	5	5	5	5
val_accuracy	10	**mean**	0.997	0.029	0.722	0.752	0.963	0.116	219.265
val_accuracy	10	**std**	0.002	0.011	0.044	0.156	0.007	0.022	56.463
val_accuracy	10	**min**	0.995	0.020	0.676	0.570	0.952	0.086	137.207
val_accuracy	10	**25%**	0.995	0.020	0.686	0.678	0.959	0.106	187.372
val_accuracy	10	**50%**	0.998	0.022	0.727	0.684	0.964	0.120	241.425
val_accuracy	10	**75%**	0.999	0.038	0.737	0.878	0.969	0.123	252.961
val_accuracy	10	**max**	1.000	0.044	0.785	0.948	0.969	0.145	277.361
val_accuracy	15	**count**	5	5	5	5	5	5	5
val_accuracy	15	**mean**	0.998	0.019	0.728	0.823	0.966	0.105	313.890
val_accuracy	15	**std**	0.001	0.008	0.026	0.132	0.010	0.023	63.855
val_accuracy	15	**min**	0.997	0.010	0.703	0.628	0.952	0.074	234.273
val_accuracy	15	**25%**	0.998	0.011	0.710	0.779	0.959	0.099	273.934
val_accuracy	15	**50%**	0.999	0.022	0.720	0.847	0.971	0.100	306.511
val_accuracy	15	**75%**	0.999	0.024	0.741	0.875	0.973	0.121	365.928
val_accuracy	15	**max**	0.999	0.029	0.768	0.987	0.974	0.133	388.806
no_monitoring	No	**count**	5	5	5	5	5	5	5
no_monitoring	No	**mean**	0.999	0.010	0.732	0.874	0.965	0.120	429.153
no_monitoring	No	**std**	0.000	0.001	0.034	0.115	0.009	0.023	26.601
no_monitoring	No	**min**	0.999	0.008	0.686	0.739	0.952	0.095	396.766
no_monitoring	No	**25%**	0.999	0.011	0.706	0.825	0.962	0.105	408.430
no_monitoring	No	**50%**	1.000	0.011	0.751	0.829	0.968	0.119	437.314
no_monitoring	No	**75%**	1.000	0.011	0.751	0.942	0.971	0.125	439.890
no_monitoring	No	**max**	1.000	0.012	0.765	1.034	0.974	0.155	463.362