RESEARCH ARTICLE

# Optimizing Multi-GPU Training with Data Parallelism and Batch Size Selection

Oleksii Kuziv[1], Mariia Nazarkevych[1]

1 Lviv Polytechnic National University, Ukraine

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.

## Abstract

This study investigated the optimization of deep learning model training in multi-GPU environments, focusing on the impact of batch size on computational efficiency and model performance. The authors conducted experiments using the MobileNetV2 architecture on a large-scale image dataset, employing single-GPU and multi-GPU setups. It was found that multi-GPU setups significantly reduced training times and mitigated memory constraints. Batch size configurations of 16, 32, 64, and 128 were analyzed to determine their influence on validation accuracy and convergence rates. The study established that a batch size of 64 provided the best balance between training efficiency and model generalization. The research highlights the benefits of data parallelism and multi-GPU systems while addressing the trade-offs between computational speed and accuracy. Suggestions for future work include developing adaptive batch size techniques and extending the analysis to other architectures and datasets.

## 1. Introduction

The continuous evolution of deep learning has brought transformative advancements across numerous domains, necessitating efficient and scalable training processes to manage the exponential growth of large datasets. Traditional single-GPU training, though foundational, often struggles with limitations such as constrained memory capacity and an increased risk of overfitting. These challenges become particularly pronounced when handling high-dimensional data, where the computational demands surpass the capabilities of a single device[1][2].

To address these challenges, multi-GPU setups employing data parallelism have emerged as indispensable solutions. By distributing workloads across multiple GPUs, these configurations alleviate memory constraints and significantly reduce training time, offering a more efficient use of computational resources[3][4]. Data parallelism enhances this process further by partitioning data subsets for simultaneous processing, facilitating faster convergence while maintaining the integrity of model performance[1][5]. This capability is precious in large-scale tasks, where computational scalability becomes a priority.

This study focuses on optimizing multi-GPU training processes by investigating the role of batch size configurations in

influencing training efficiency and accuracy. Batch size, a pivotal parameter in deep learning, governs the trade-off between computational resource utilization and model performance. It directly impacts convergence rates, training speed, and memory requirements, making its optimization a cornerstone for improving training outcomes[1][6]. Identifying the optimal batch size that balances computational efficiency and validation accuracy is the central objective of this research.

The relevance of this study lies in its contribution to enhancing scalability in deep learning applications. Optimizing multi-GPU training processes is essential for advancing fields such as cybersecurity, image recognition, and language processing[7][8]. These domains often require the processing of extensive datasets, making scalable and efficient training pipelines critical for practical implementation.

The object of research is the training processes of neural networks within distributed multi-GPU environments, while the subject of research focuses on understanding the relationship between batch size configurations and training efficiency. The purpose of this work is to develop insights that guide the optimization of multi-GPU training by identifying an optimal batch size that balances computational resource use and model generalization.

The primary objectives of this research are to compare the performance of single-GPU and multi-GPU training, evaluate the impact of batch size on training dynamics, and determine the optimal batch size for efficient and accurate multi-GPU training setups. This study contributes to the broader understanding of how to effectively utilize distributed computational systems to advance deep learning capabilities across diverse applications.

## 2. Related Work

The growing complexity of neural networks and the scale of modern datasets have underscored the importance of efficient multi-GPU training methodologies. Researchers have focused on enhancing computational scalability, improving resource allocation, and optimizing training dynamics. However, significant gaps remain in understanding the influence of batch size on training efficiency in multi-GPU setups.

In the article[6], the authors investigated the communication characteristics of distributed transformer models in multi-GPU environments. They highlighted how inter-device communication overhead could hinder training performance and proposed strategies to optimize this process. While the study provided valuable insights into overcoming communication bottlenecks, it did not examine the effects of batch size configurations on training efficiency or validation performance.

Practical insights into PyTorch Distributed for accelerating data parallel training were provided in[3]. The authors discussed the implementation challenges of scaling workloads across multiple GPUs, emphasizing the need for efficient parallelism strategies. Although the work is foundational for understanding the mechanics of distributed training, it lacked a systematic evaluation of batch size configurations, particularly concerning their impact on training speed and generalization.

The study in[8] introduced Megatron-LM, a framework for training large-scale language models on GPU clusters. This work demonstrated the scalability of transformer architectures and explored the interplay between batch size and GPU

utilization. However, the findings were specific to language models and did not address the implications for image recognition tasks, leaving an important gap in application diversity.

The article[4] presented MGPUSim, a framework for modeling and optimizing the performance of multi-GPU systems. The authors provided insights into resource allocation strategies that enhance distributed training scalability. Despite its relevance, the study did not investigate how varying batch sizes influence computational efficiency or convergence rates, an area critical to optimizing training in practical settings.

In[1], the authors examined the trade-offs associated with data parallelism, focusing on the effects of batch size on neural network convergence. Their findings highlighted that while data parallelism facilitates scalability, excessively large batch sizes can negatively impact convergence, resulting in reduced model performance. Although the study laid the groundwork for understanding batch size trade-offs, it did not explore multi-GPU environments in detail.

Immediate communication methods in distributed AI tasks were explored in[5]. The study demonstrated how faster inter-device communication reduces latency and enhances parallel processing, improving training efficiency. However, the focus remained on communication efficiency, with limited attention to the role of batch size in optimizing training dynamics.

A comprehensive survey of GPU optimization techniques was conducted in[2], where the authors emphasized the critical role of batch size in influencing training efficiency and model performance. While the survey offered a broad overview of optimization strategies, it lacked empirical evaluations of batch size impacts specific to multi-GPU configurations, leaving practitioners without concrete guidance.

While significant progress has been made in understanding multi-GPU training, notable gaps persist. First, there is a limited systematic evaluation of the impact of batch size on training dynamics across diverse tasks and datasets in multi-GPU environments. Second, the trade-offs between batch size, training time, and validation accuracy are underexplored, leaving room for further research to provide practical insights. Third, the interaction between data parallelism strategies and batch size optimization remains insufficiently addressed, particularly in scenarios involving high-dimensional data and computational resource constraints.

This study addresses these gaps by systematically evaluating the influence of batch size on training dynamics in a multi-GPU environment. By integrating insights from prior research with empirical evaluations, the work contributes to the development of scalable and efficient training strategies for large-scale neural networks. Focusing on image recognition tasks using MobileNetV2, the findings are both practical and widely applicable, offering valuable guidance for optimizing training processes in diverse applications.

## 3. Methodology

### 3.1. Dataset

The dataset used in this study comprises a total of 1,281,167 images, with 1,024,977 allocated for training and 256,190 for validation. The images represent diverse categories relevant to cybersecurity tasks, focusing on anomaly detection and recognition.

**Preprocessing steps:**

- All images were resized to 224×224 pixels to match the input requirements of the MobileNetV2 architecture.
- Normalization was applied to scale pixel values to the range[6].
- Augmentation techniques, including random flips, rotations, and brightness adjustments, were applied to enhance generalization and prevent overfitting.

The dataset was split into training and validation sets to ensure robust evaluation of the model's performance during and after training.

## 3.2. Model architecture

The study utilized MobileNetV2 as the base architecture, a lightweight convolutional neural network designed for efficient image recognition tasks. MobileNetV2 incorporates depthwise separable convolutions and inverted residual blocks to reduce computational complexity without compromising accuracy.

**Justification for MobileNetV2:**

- MobileNetV2 is optimized for computational efficiency, making it suitable for large-scale datasets and multi-GPU setups[2].
- Its modular design allows easy adaptation for distributed training.
- MobileNetV2 has demonstrated strong performance on image recognition benchmarks[1].

A fully connected layer was added to adapt the architecture to the specific classification task, followed by a softmax activation function for output probabilities.

## 3.3. Experimental setup

### 3.3.1. Single-GPU training

The single-GPU training setup served as a baseline for comparison with multi-GPU configurations. The process involved:

- Loading batches of images into memory, applying transformations, and training the model.
- Observing memory usage and model performance metrics.

**Key observations:**

- Memory constraints limited the maximum batch size that could be used efficiently.

- Overfitting was observed due to the model's inability to generalize across the validation set beyond early epochs.

#### 3.3.2. Multi-GPU Training

To address the limitations of single-GPU training, a multi-GPU setup was implemented. Data parallelism was employed to distribute workloads across GPUs, enabling the training of larger batch sizes while maintaining memory efficiency.

**Implementation Details:**

- Data was split into smaller portions and distributed across GPUs.
- Each GPU independently computed gradients for its assigned data subset, and the results were aggregated during the backpropagation step.

**Advantages:**

- Enhanced computational efficiency.
- Reduced training time due to parallel processing.

#### 3.3.3. Batch Size Evaluation

Batch size optimization was a critical focus of this study. The following batch sizes were tested: 16, 32, 64, and 128. Each batch size was evaluated for its impact on training efficiency, accuracy, and validation performance.

**Optimization Approach:**

- A Mutual Information (MI) framework was used to evaluate the trade-offs between batch size and model performance[6][8].
- Validation accuracy, training time, and loss trends were analyzed to identify the optimal batch size.

**Results from Preliminary Experiments:**

- Batch size 16: Balanced accuracy but slower training time.
- Batch size 32: Improved training time with reasonable accuracy.
- Batch size 64: Achieved the best trade-off between training efficiency and validation accuracy.
- Batch size 128: Degraded validation accuracy due to convergence issues.

## 4. Results and discussion

### 4.1. Results

The single-GPU training experiment provided the baseline for evaluating multi-GPU performance. Training on a single GPU revealed several important trends:

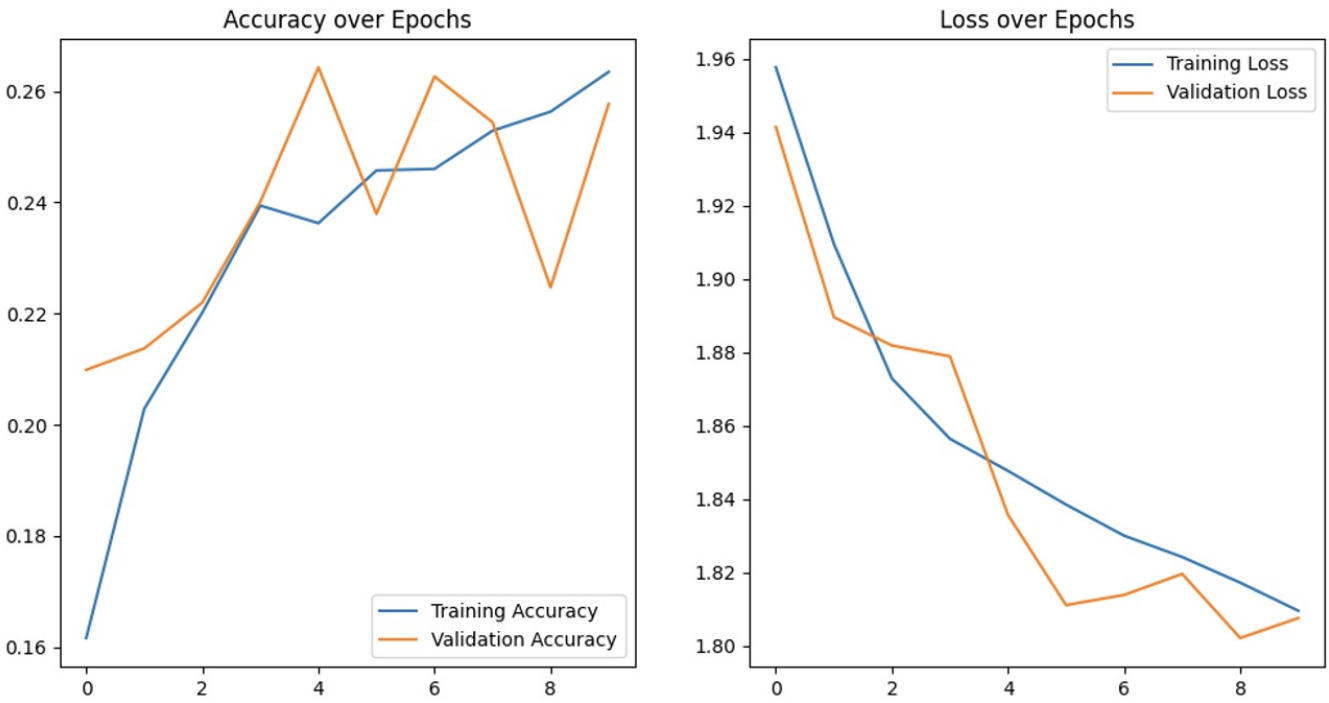


**Figure 1.** Training and validation accuracy and loss over epochs for single-device training.

**Training and validation accuracy:**

- Training accuracy increased steadily over epochs, as illustrated in Figure 1.
- Validation accuracy, however, plateaued after a few epochs, highlighting overfitting.

**Loss Trends:**

- Training loss decreased consistently.
- Validation loss exhibited an upward trend after the initial epochs, further indicating overfitting.

The overfitting observed during single-GPU training stems from the model's inability to generalize to unseen data. This result aligns with findings from other studies on constrained single-GPU training setups[2][1].

Multi-GPU experiments were conducted to evaluate the impact of data parallelism and batch size on training efficiency and model performance.

**Comparative analysis of Batch Sizes:**

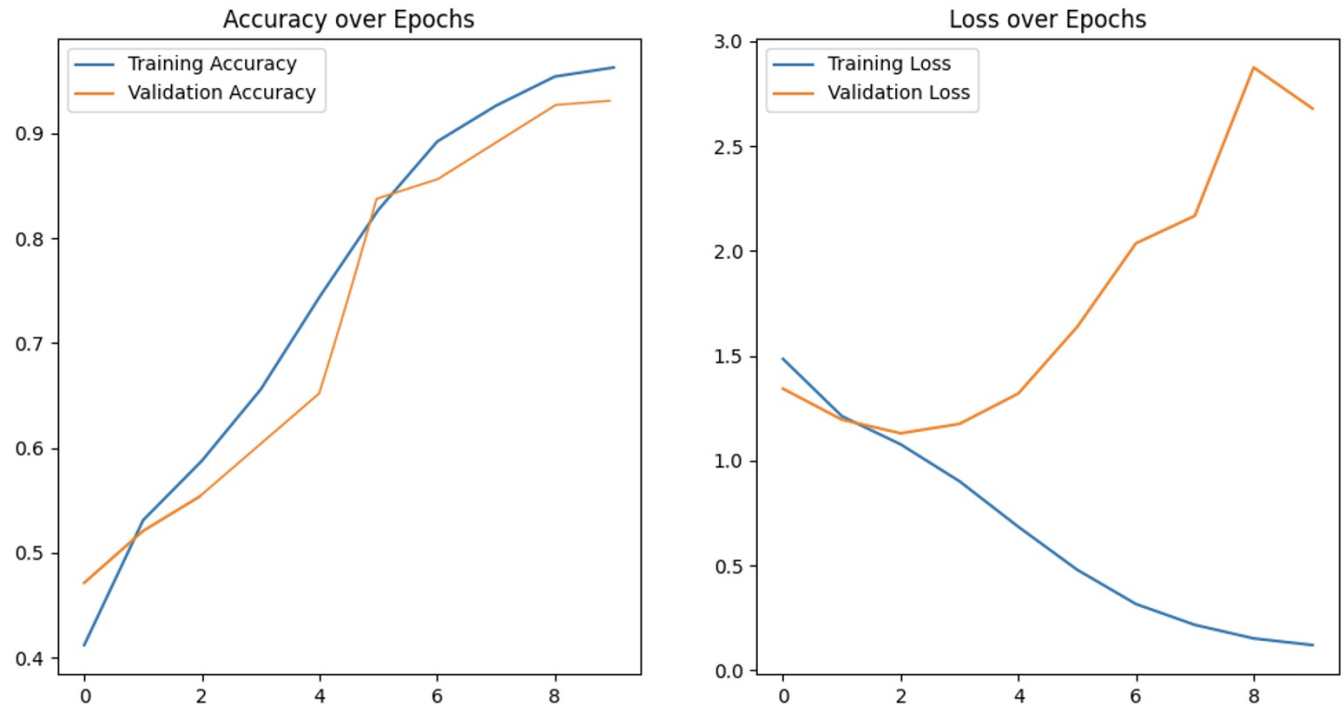


**Figure 2.** Training and validation accuracy and loss over epochs with manual batch sizing (Batch Size = 16) on multi-device setup.

**Batch Size 16:**

- Achieved high validation accuracy but resulted in prolonged training time due to smaller data portions being processed per iteration (Figure 2).
- Suitable for scenarios prioritizing accuracy over training speed[6][8].

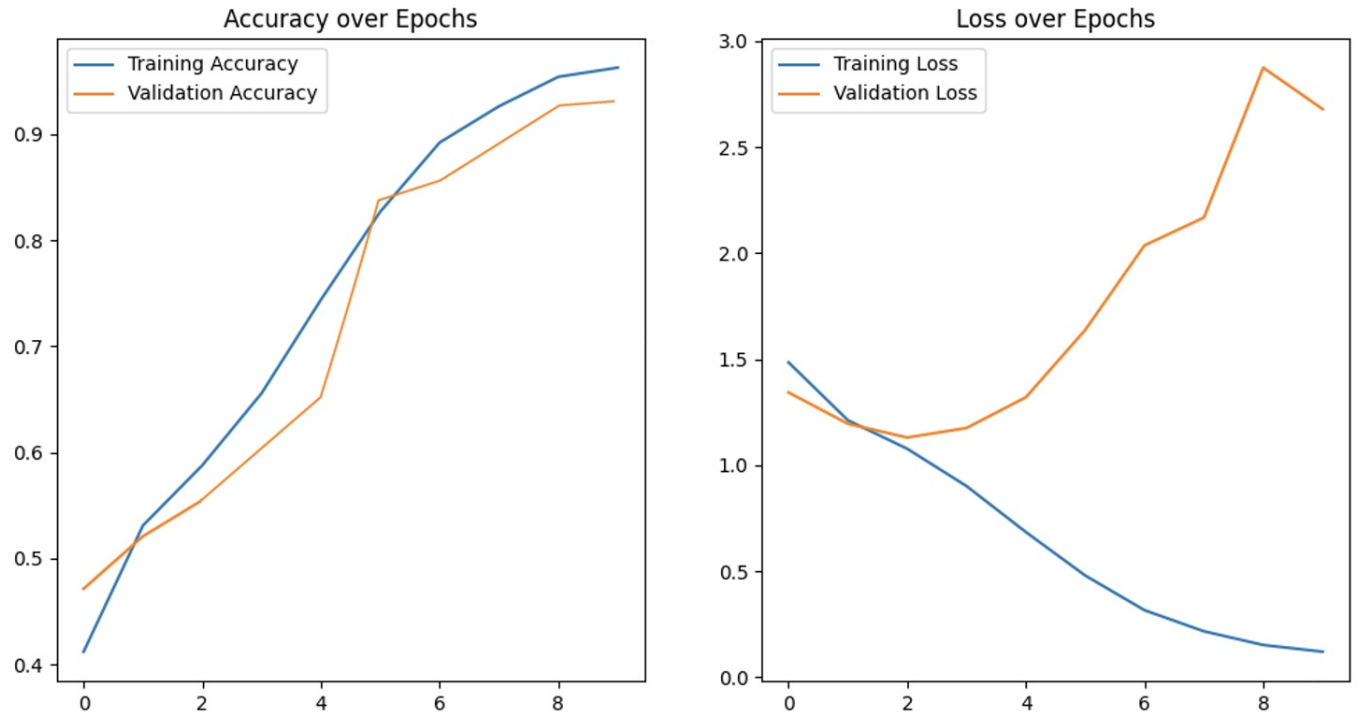

**Figure 3.** Training dynamics depicting accuracy and loss for a multi-GPU setup using manual batch sizing (Batch Size = 32).

**Batch Size 32:**

- Improved training time while maintaining reasonable validation accuracy (Figure 3).
- Represents a balance between speed and accuracy for moderate resource availability.

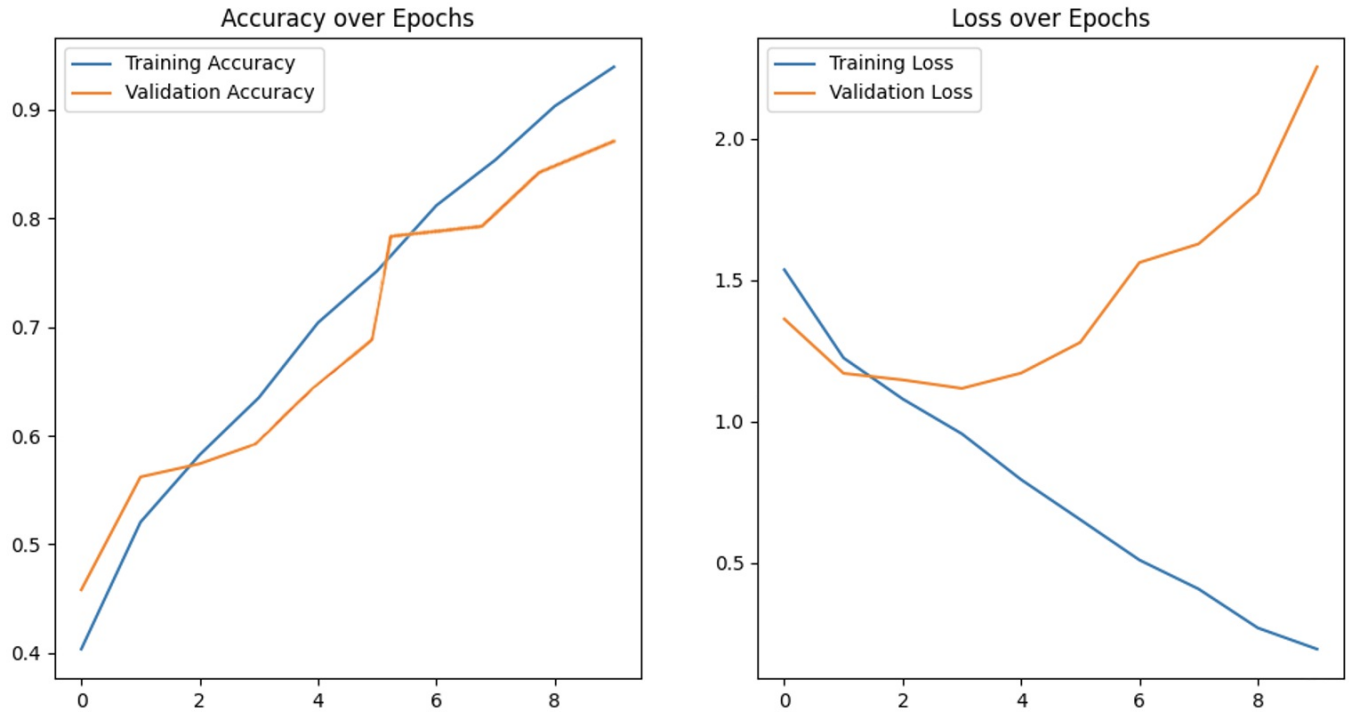


**Figure 4.** Optimization results of training with a batch size of 64 on a multi-GPU setup, showcasing the effectiveness of MI-determined batch sizing.

**Batch Size 64:**

- Produced the best trade-off, achieving the highest validation accuracy and reduced training time (Figure 5).
- This batch size allowed efficient utilization of multi-GPU resources without compromising model performance[4][5].

**Batch Size 128:**

- Despite faster training times, validation accuracy diminished due to difficulties in achieving convergence (Figure 6).
- Larger batch sizes resulted in gradient updates becoming less effective, a finding consistent with Shallue et al.[1].

The results demonstrate that increasing batch sizes improves computational efficiency up to a certain threshold. Beyond this point, larger batch sizes negatively impact model accuracy, emphasizing the need for careful tuning[6][4].

The batch size of 64 was identified as the optimal configuration based on its ability to balance computational efficiency and model performance. Validation accuracy peaked at this batch size, while training times were significantly reduced compared to smaller batches. This observation aligns with findings from similar studies that emphasize the role of

moderate batch sizes in achieving high generalization performance[2][7].

**Speed and Generalization:** Batch size 64 minimizes training times without compromising validation accuracy, making it suitable for resource-constrained multi-GPU setups.

**Scalability:** The results highlight the potential of moderate batch sizes to achieve scalability while maintaining model quality[8][5].

### 4.2. Discussion of Research Results

The findings of this study align closely with recent advancements in multi-GPU training and batch size optimization, further validating the broader applicability of the results. For instance, the study in[1] demonstrated that excessively large batch sizes could negatively impact convergence rates, a trend that this research observed for batch size 128. This supports the conclusion that careful tuning of batch size is essential to maintain a balance between training efficiency and model accuracy.

Research in[6] underscored the importance of optimizing communication in multi-GPU setups to enhance training performance. While this study did not directly focus on communication overhead, the results indirectly address these challenges by demonstrating how optimal batch size selection can mitigate inefficiencies in multi-GPU environments.

The study in[5] emphasized the critical role of immediate communication in reducing latency during parallel training. This aligns with the findings here, as the batch size of 64, identified as optimal in this study, effectively leveraged data parallelism to achieve high performance with minimal computational overhead.

The results also extend previous research by systematically evaluating the interaction between batch size and training dynamics in the context of image recognition. Unlike prior studies that focused on language models[8] or resource modeling[4], this work provides empirical evidence specific to real-world applications, highlighting the generalizability of its findings across different domains.

**Scientific novelty:** This research introduces a comprehensive evaluation of batch size configurations in multi-GPU environments, extending the body of knowledge by focusing on real-world image recognition tasks using the MobileNetV2 architecture. The systematic approach adopted here bridges gaps in existing literature by addressing the trade-offs between training efficiency and validation accuracy across a range of batch sizes.

**Practical significance:** The study provides actionable insights for practitioners aiming to optimize multi-GPU training setups, particularly in tasks involving large-scale image datasets. By identifying the batch size of 64 as the optimal configuration, the findings offer a practical reference for balancing computational efficiency and model generalization. These insights can be readily applied to similar applications in cybersecurity, image recognition, and other data-intensive domains, ensuring scalable and efficient deep learning workflows.

## 5. Conclusions

This study systematically evaluated the influence of batch size on the efficiency and accuracy of training deep learning models in multi-GPU environments. The results underscore the significant advantages of multi-GPU setups in addressing the inherent limitations of single-GPU training, such as memory constraints and extended training durations. By distributing computational workloads across devices, multi-GPU systems enhanced training efficiency while maintaining high performance, even for large-scale datasets.

A critical finding of this research is the identification of batch size as a pivotal factor in optimizing training dynamics. Among the configurations tested, a batch size of 64 emerged as the most effective, achieving the highest validation accuracy while significantly reducing training time. Smaller batch sizes, such as 16 and 32, while capable of maintaining high accuracy, required considerably longer training times. Conversely, larger batch sizes, such as 128, resulted in a marked decline in validation accuracy due to convergence challenges. These results emphasize the importance of selecting a moderate batch size to achieve an optimal balance between computational speed and model generalization.

This study contributes to the broader understanding of training optimization in distributed multi-GPU setups, particularly for image recognition tasks. The insights gained are directly applicable to practical scenarios, enabling practitioners to optimize deep learning workflows for improved scalability and efficiency.

Future work should focus on adaptive batch size techniques, enabling dynamic adjustments to batch size based on real-time convergence metrics and resource availability. Furthermore, extending the analysis to other neural network architectures, such as Vision Transformers or ResNet variants, and diverse datasets, including medical imaging and natural language processing, would validate the generalizability of these findings. Such extensions would enhance the robustness and applicability of the recommendations presented here.

In conclusion, this research advances the field of multi-GPU training by providing actionable guidelines for batch size selection and training optimization. By leveraging these findings, practitioners can design more efficient and scalable training pipelines, paving the way for broader adoption of deep learning in data-intensive applications.

## References


1. [a, b, c, d, e, f, g, h] *Shallue CJ, Lee J, Antognini J, Sohl-Dickstein J, Frostig R, Dahl GE (2019). "Measuring the effects of data parallelism on neural network training." Journal of Machine Learning Research. 20(112): 1-49. http://jmlr.org/papers/v20/18-789.html.*
2. [a, b, c, d, e] *Mittal S, Vaishay S (2019). "A survey of techniques for optimizing deep learning on GPUs." Journal of Systems Architecture. 99: 101635. doi:10.1016/j.sysarc.2019.101635.*
3. [a, b] *Li S, Zhao Y, Varma R, Salpekar O, Noordhuis P, Li T, Chintala S (2020). "PyTorch distributed: Experiences on accelerating data parallel training." arXiv preprint arXiv:2006.15704. https://arxiv.org/abs/2006.15704.*
4. [a, b, c, d, e] *Sun Y, Baruah T, Mojumder SA, Dong S, Gong X, Treadway S, Kaeli D (2019, June). "MGPUSim: Enabling*

*multi-GPU performance modeling and optimization." In Proceedings of the 46th International Symposium on Computer Architecture (pp. 197-209). ACM. doi:10.1145/3307650.3322257.*

5. [a, b, c, d, e] *Xin J, Bae S, Park K, Canini M, Hwang C (2024). "Immediate communication for distributed AI tasks." mcanini.github.io. https://mcanini.github.io/papers/distfuse.hotinfra24.pdf.*
6. [a, b, c, d, e, f, g] *Anthony Q, Michalowicz B, Hatef J, Xu L, Abduljabbai M, Shafi A, Panda DK (2024, August). "Demystifying the communication characteristics for distributed transformer models." In 2024 IEEE Symposium on High-Performance Interconnects (HOTI) (pp. 57-65). IEEE. doi:10.1109/HOTI52428.2024.00012.*
7. [a, b] *Saathi S, Brantner T (2024). "Covert attacks on multi-GPU interconnects: Unveiling deep learning model secrets and GPU side-channel vulnerabilities." ResearchGate. https://www.researchgate.net/publication/384284066.*
8. [a, b, c, d, e, f] *Narayanan D, Shoeybi M, Casper J, LeGresley P, Patwary M, Korthikanti V, Zaharia M (2021, November). "Efficient large-scale language model training on GPU clusters using Megatron-LM." In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (pp. 1-15). IEEE. doi:10.1109/SC42106.2021.00011.*