Efficient Pipeline Parallelism for Reinforcement Learning

Faiz Sameer Ahmed
Computer Science
San Jose State University
San Jose, USA
faiz.ahmed@sjsu.edu

Dr. Genya Ishigaki
Computer Science
San Jose State University
San Jose, USA
genya.ishigaki@sjsu.edu

***Abstract*—Reinforcement learning trains agents through interaction with environments, alternating between a rollout phase where experiences are collected and a training phase where policy and value networks are updated. Traditional distributed RL architectures transfer collected experiences to a centralized server that trains the actor-critic network and returns updated parameters. However, when deploying large neural networks in resource-constrained environments that require model parallelism across multiple devices, communication overhead increases substantially. Pipeline parallelism necessitates transferring both forward activations and backward gradients between devices for each training epoch, which can be repeated 10 or more times per batch for sample efficiency. This paper presents a gradient accumulation technique that selectively transmits high-magnitude gradients above a specified percentile threshold during backward propagation. By accumulating gradients over time and only communicating the most significant values, we reduce network transfer by up to 90 percent while maintaining comparable training performance to full gradient communication.**

***Index Terms*—Reinforcement Learning, Pipeline Parallelism, Gradient Compression, Distributed Training, Model Parallelism**

I. Introduction

Reinforcement learning trains intelligent agents through environmental interaction, alternating between rollout phases where experiences are collected and training phases where policies are updated. As neural network architectures grow increasingly complex, computational and memory requirements often exceed single-device capabilities, necessitating distributed training approaches.

Traditional distributed RL systems use centralized architectures where edge devices collect experiences while cloud servers train models. However, when models become too large for a single device's memory, pipeline parallelism becomes essential, partitioning networks into sequential stages across multiple devices. Unlike centralized training where only experiences and parameters are transferred, pipeline parallelism requires continuous exchange of forward activations and backward gradients between devices. This communication occurs repeatedly for each training epoch, which can number 10 or more per batch for sample efficiency, creating substantial overhead in bandwidth-limited environments.

This paper addresses this communication challenge through selective gradient transmission. We present a gradient accumulation technique that identifies and communicates only the most significant gradients based on magnitude percentile thresholds. Section II reviews related work. Section III defines the problem and quantifies communication overhead. Section IV describes our method. Section V presents experimental results demonstrating network transfer reduction of up to 90 percent while maintaining comparable performance. Section VI discusses future work.

II. Related Work

A. Distributed Reinforcement Learning

Early work on distributed RL established foundational architectures for scaling training. Nair et al. [**?**] introduced massively parallel methods for deep RL using parallel actors generating behavior, parallel learners trained from stored experience, and distributed neural networks, achieving order-of-magnitude speedups on Atari games. Building on this, Espeholt et al. [**?**] proposed IMPALA, which decouples acting from learning using importance sampling for off-policy corrections, scaling to thousands of machines while achieving state-of-the-art results on multi-task benchmarks. Horgan et al. [**?**] introduced Ape-X, separating data collection from learning through distributed prioritized experience replay, where multiple actors generate experience stored in a shared replay buffer accessed by learners.

B. Communication-Efficient Distributed Training

Reducing communication overhead in distributed training has been extensively studied. Chen et al. [**?**] developed communication-efficient policy gradient methods specifically for distributed RL, proposing gradient compression techniques that reduce communication costs while maintaining convergence guarantees. In the supervised learning domain, Lin et al. [**?**] discovered that 99.9% of gradient exchange in distributed SGD is redundant, proposing Deep Gradient Compression (DGC) that achieves 270-600x bandwidth reduction through momentum correction and local gradient clipping. Alistarh et al. [**?**] introduced QSGD, a family of gradient quantization schemes providing convergence guarantees for both convex and non-convex optimization while allowing smooth trade-offs between communication bandwidth and convergence time.

C. Pipeline Parallelism for Large Models

Pipeline parallelism has emerged as a critical technique for training models exceeding single-device memory. Huang et al. [**?**] presented GPipe, which partitions neural networks across multiple accelerators and pipelines mini-batches through stages, introducing micro-batching and gradient accumulation to achieve near-linear speedup while training models 25x larger than single-accelerator capacity. Narayanan et al. [**?**] proposed PipeDream, which automatically partitions DNN models and schedules computation to minimize pipeline bubbles, maintaining multiple parameter versions to enable weight updates without stalling. For transformer models, Shoeybi et al. [**?**] developed Megatron-LM with efficient intra-layer model parallelism, achieving 76% scaling efficiency on 512 GPUs for 8.3 billion parameter models. Rajbhandari et al. [**?**] introduced ZeRO, eliminating redundant storage of optimizer states, gradients, and parameters across data-parallel processes, enabling training of 170 billion parameter models.

D. Resource-Constrained Environments

Recent work addresses distributed learning in resource-constrained settings. Lim et al. [**?**] applied federated learning principles to RL for IoT devices, enabling multiple agents to collaboratively learn optimal control policies while keeping data local, addressing device-specific dynamics variations. Feng et al. [**?**] designed IoTSL, an efficient split learning system for resource-constrained IoT devices, optimizing the split learning paradigm to handle non-IID data distribution and reduce communication requirements specific to IoT constraints.

Our work differs from prior approaches by specifically addressing the communication bottleneck in pipeline-parallel RL training through selective gradient accumulation. While existing gradient compression techniques focus on compressing all gradients uniformly or through quantization, our approach accumulates gradients over time and transmits only high-magnitude values above configurable percentile thresholds, tailored to the unique characteristics of RL training with repeated epochs.

III. Problem Statement

We consider a distributed RL setup where an agent learns to control the CarRacing-v3 environment through interaction. The agent consists of a convolutional neural network (CNN) for feature extraction and an actor-critic network for policy and value estimation. Given resource constraints, we partition this agent across two machines: Machine 0 hosts the CNN and environment interaction, while Machine 1 hosts the actor-critic network.

A. Communication Overhead Analysis

We compare data transfer requirements across two deployment scenarios over 100 training iterations with batch size 4096 (100 iterations $\times$ 4096 steps per iteration).

Cloud Setup: The local machine transmits raw observations to the cloud for complete training. Each observation is a $4 \times 96 \times 96$ tensor. Total data transfer:

$$100 \times 4096 \times (4 \times 96 \times 96) = 14.7\text{B tensors} \approx 28\text{ GB} \quad (1)$$

Pipeline-Parallel Setup: The CNN runs locally, transmitting activations to Machine 1 and receiving gradients. With 10 update epochs per batch and 32 minibatches per epoch (minibatch size 128), we transmit 4096-dimensional activation and gradient vectors. Total data transfer:

$$100 \times 10 \times 32 \times 128 \times 4096 \times 2 = 32.7\text{B tensors} \approx 66\text{ GB} \quad (2)$$

The factor of 2 accounts for bidirectional communication (forward activations and backward gradients). While cloud setup requires 28 GB, pipeline parallelism requires 66 GB—a 2.4$\times$ increase. Table **??** summarizes this comparison.

TABLE I
COMMUNICATION OVERHEAD COMPARISON

Setup	Data Transferred	Relative Cost
Cloud Setup	28 GB	1.0×
Pipeline-Parallel	66 GB	2.4×

This overhead stems from repeated gradient exchanges during multiple training epochs. Our goal is to reduce backward gradient communication while maintaining training performance, making pipeline parallelism viable for bandwidth-constrained environments.

B. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \quad (3)$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(**??**)", not "Eq. (**??**)" or "equation (**??**)", except at the beginning of a sentence: "Equation (**??**) is . . ."

C. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BIBTEX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BIBTEX to produce a bibliography you must send the .bib files.

LATEX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LATEX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

D. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [**?**].

E. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

F. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

G. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. **??**", even at the beginning of a sentence.

TABLE II
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of

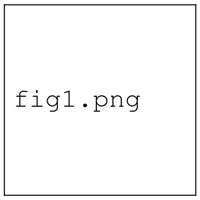

Fig. 1. Example of a figure caption.

quantities and units. For example, write "Temperature (K)", not "Temperature/K".

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks …". Instead, try "R. B. G. thanks…". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [**?**]. The sentence punctuation follows the bracket [**?**]. Refer simply to the reference number, as in [**?**]—do not use "Ref. [**?**]" or "reference [**?**]" except at the beginning of a sentence: "Reference [**?**] was the first …"

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [**?**]. Papers that have been accepted for publication should be cited as "in press" [**?**]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation.

REFERENCES

[1] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, et al., "Massively parallel methods for deep reinforcement learning," in *ICML Deep Learning Workshop*, 2015.

[2] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al., "IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures," in *Proc. ICML*, 2018, pp. 1407–1416.

[3] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, "Distributed prioritized experience replay," in *Proc. ICLR*, 2018.

[4] W. Chen, S. Wang, Y. Hong, and G. B. Giannakis, "Communication-efficient policy gradient methods for distributed reinforcement learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 2629–2641, 2021.

[5] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. ICLR*, 2018.

[6] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. NeurIPS*, 2017, pp. 1709–1720.

[7] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and Z. Chen, "GPipe: Efficient training of giant neural networks using pipeline parallelism," in *Proc. NeurIPS*, 2019, pp. 103–112.

[8] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia, "PipeDream: Generalized pipeline parallelism for DNN training," in *Proc. ACM SOSP*, 2019, pp. 1–15.

[9] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training multi-billion parameter language models using model parallelism," arXiv preprint arXiv:1909.08053, 2019.

[10] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "ZeRO: Memory optimizations toward training trillion parameter models," in *Proc. IEEE SC*, 2020, pp. 1–16.

[11] H. Lim, Y. J. Kim, and M. L. Sichitiu, "Federated reinforcement learning for training control policies on multiple IoT devices," *Sensors*, vol. 20, no. 5, p. 1359, 2020.

[12] X. Feng, C. Luo, J. Chen, Y. Huang, J. Zhang, W. Xu, J. Li, and V. C. M. Leung, "IoTSL: Toward efficient distributed learning for resource-constrained Internet of Things," *IEEE Internet Things J.*, vol. 10, no. 11, pp. 9892–9905, 2023.