Deep Learning Models for Road Pothole Detection

Subjects: Computer Science, Artificial Intelligence

Contributor: Hassam Tahir , Eun-Sung Jung

For self-driving cars, crack detection is crucial because these vehicles rely on sensors to perceive and navigate the environment. Crack visualization has certain methods, such as the use of a deep-learning architecture, capable of processing images at multiple scales. The inability to judge the difference between potholes or patches results in the sudden break or non-breaking elements at inappropriate places because of a confused state of the neural network. In this regard, detecting potholes in self-driving vehicles or road maintenance is vital for future intelligent transportation systems.

distributed deep-learning distributed edge AI/ML distributed hybrid model training

1. Introduction

A self-driving automobile employs an artificial intelligence (AI) system to evaluate data from sensors and make judgments while driving [1]. The disposition of smart cars assails as a fast catalyst for the revolutionary steps toward the future of intelligent transportation systems by decreasing pollution, reducing accidents, and decreasing traffic [2]. The self-driving vehicles will remarkably decrease road accidents in the future through human input integrated with AI programs. For self-driving cars, crack detection is crucial because these vehicles rely on sensors to perceive and navigate the environment. If cracks are not detected and repaired promptly, they can interfere with the vehicle's perception systems, leading to incorrect or incomplete information about the road ahead. Crack visualization has certain methods, such as the use of a deep-learning architecture, capable of processing images at multiple scales [3] and detecting strains in columns via the mark-free vision methodology [4]. However, substantial doubts about reliability, regulations, and predictive detection have been encountered and raised [5]. Most reported accidents of self-driving cars were due to inappropriate or heavy-weight neural network training on edge devices, resulting in heating issues [6]. The inability to judge the difference between potholes or patches results in the sudden break or non-breaking elements at inappropriate places because of a confused state of the neural network [7]. In this regard, detecting potholes in self-driving vehicles or road maintenance is vital for future intelligent transportation systems. The requirements of pothole-detecting AI systems include the following: (1) a lightweight distributed neural network, (2) high-quality input images for training in a distributed edge cloud environment, and (3) reliable communication for appropriate information exchange among distributed deep learning [8].

Over the past few years, distributed deep learning has emerged as a promising area of research, supported by scalable cutting-edge technology and driven by the need to tackle large-scale datasets and complex problems. Numerous state-of-the-art studies have been conducted to develop innovative techniques and frameworks for optimizing distributed deep-learning systems. For instance, exploring data and model parallelism has led to significant advancements in the scalability and efficiency of training large neural networks [9]. Additionally, researchers have been investigating the impact of communication strategies, such as gradient compression [10] and decentralized optimization [11], to reduce the communication overhead and latency associated with the distributed training process. Furthermore, novel approaches, such as federated learning [12], have been proposed to enable collaborative learning among multiple devices while preserving data privacy. These studies reflect ongoing efforts to develop more efficient, scalable, and privacy-preserving distributed deep-learning systems, ultimately contributing to the broader applicability of deep learning in various domains.

There are two main distributed learning strategies. The first strategy is data parallelism [13]. An extensive dataset is common for more accurate results in modern deep learning. Due to the extensive dataset, the memory fitting problem occurs vastly. To overcome this issue, the large dataset is divided into small batches, and their gradients are calculated individually on different GPUs; the final result is the weighted average of all the calculated gradients. Furthermore, the second technique is model parallelism [14]. Model parallelism is required when the model (layers of the model) or parameters are too large to fit in the memory. Therefore, deep-learning models could be divided into pieces; a few consecutive layers could be transferred to a single node, and the gradients could be calculated in the forward direction. Synchronous [15] and asynchronous training [16] is a typical method to solve data/model parallelism.

Majorly, two main libraries support distributed learning: TensorFlow and PyTorch. TensorFlow is used vastly in industrial products and provides distributed APIs for data distribution across multiple devices (e.g., GPU and TPU). Users can distribute data and create a training pipeline with minimal changes in the code. One of the significant drawbacks of TensorFlow's distributed APIs is that they support model distribution but with many limitations. On the other hand, PyTorch's distributed APIs are fastly growing in model distribution and data distribution [17]. PyTorch contains various model and data parallelism options according to the user's requirements [18]. PyTorch also provides flexibility to develop its model and data distribution training pipeline.

However, the increase in computational capabilities is significantly outpaced by the expansion of the datasets and models [19], which is why, even after achieving the distribution scenarios of training, the production deployment of these networks remains premature [20]. Consequently, the memory capacity and communication overhead can limit the scaling of data parallelism.

2. Deep Learning Models for Road Pothole Detection

The detection of road potholes using computer vision and machine learning approaches can be a valuable tool to assist with visual challenges [21]. Potholes can pose a significant risk to autonomous vehicles, potentially causing damage to their sensors or suspensions and can lead to accidents or disruptions in traffic flow. Similarly, the automatic detection of pothole distress in asphalt pavement using improved convolutional neural networks (CNNs) is a promising approach for identifying and addressing potholes on time. Potholes can cause significant damage to vehicles, disrupt traffic flow, and pose safety hazards to drivers and pedestrians alike [22].

Similarly, rethinking road surface 3D reconstruction and pothole detection from perspective transformation to disparity map segmentation is a novel approach to detecting and addressing potholes on the road. The traditional method of pothole detection involves using cameras to capture images of the road surface, followed by perspective transformation to create a 3D surface model. However, this method can be time-consuming and computationally expensive [23].

The system known as 3Pod is a federated learning-based system for 3D pothole detection in smart transportation. The system uses a distributed approach where data is collected from various sensors installed in the vehicles and then sent to a centralized server for processing using federated learning techniques. This approach helps improve the accuracy and efficiency of pothole detection while ensuring data privacy. One drawback of this system is that it requires a large amount of computational power and data storage to process and store the 3D point clouds [24].

Traditional distributed deep-learning pothole detection systems may not be accurate or reliable enough for use in self-driving cars, as they may be affected by various factors, such as lighting conditions, weather, and road surface variations. Moreover, the system's reliability is dependent on both hardware and software. The system's hardware components, such as the sensors and processors, must be able to accurately capture and process data for the software to analyze and interpret it effectively. Therefore, it is essential to ensure that the hardware is high-quality and meets the necessary specifications. To achieve cutting-edge development, high-end distributed strategies should be developed. Developing a high-end distributed environment for pothole and road distress detection, as a use case of self-driving cars, requires an in-depth understanding of distributed deep learning.

The distributed model analysis is thought to be the foundation of an Oracle tool that can help to identify limitations and bottlenecks of various parallelism approaches during their scaling scenario. This methodology assesses Oracle using six parallelization algorithms, four CNN models, and different datasets (2D and 3D) on up to 1024 GPUs. Compared to empirical results, the Oracle tool has an average accuracy of roughly 86.74% and data parallelism accuracy of up to 97.57% [25]. However, GPU processing performance and training throughput are severely limited because of the excessive memory consumption mentioned before.

To tackle the issue mentioned above, a model named Hippie was proposed [26]. Hippie is a hybrid parallel training framework that combines pipeline and data parallelism to increase the memory economy and scalability of massive DNN training. Hippie uses a hybrid parallel approach based on hiding gradient communication to boost training throughput and scalability. Hippie was created utilizing the PyTorch and NCCL platforms. According to tests on diverse models, Hippie achieved above 90% scaling efficiency on a 16-GPU architecture. Hippie also boosts performance by up to 80% while reducing memory overhead by 57%, resulting in a memory efficiency of 4.18×. However, significant speed-up issues were observed in inherently sequential tasks.

HyPar-Flow is a single API for processing data, model, and hybrid parallel training at scale for any Keras model. To accumulate/average gradients across model replicates, the all-reduce algorithm is employed. HyPar-Flow presents a significant advancement in distributed training, as it provides several notable benefits. First, it offers up to 1.6 times the speed of Horovod-based (Horovod is an open-source package that overcomes both scaling challenges in inter-GPU communication [27]) data-parallel training in sequential tasks, demonstrating its superior efficiency. Second, HyPar-Flow can achieve 110 times the speed of a single node when deployed, showcasing its impressive scalability. Lastly, for ResNet-1001, an ultra-deep model, HyPar-Flow boasts an astounding 481 times the speed of single-node performance when implemented on 512 Frontera nodes, further emphasizing its remarkable capabilities in handling complex and resource-intensive tasks. While the aforementioned information highlights the impressive performance and scalability of HyPar-Flow, it does not address the potential increase in communication overhead due to the combination of data and model parallelism in HyPar-Flow, which could impact its overall efficiency in specific scenarios.

Communication overhead is one of the most significant roadblocks to training big deep-learning models at scale. Gradient sparsification is a promising technique for reducing the amount of data transmitted. First, developing a scalable and efficient sparse all-reduce method has proven to be complex. Secondly, the sparsification overhead is crucial in limiting the potential for speed improvement.

The aforementioned issues were addressed for big and distributed deep-learning models by Ok-TOPK, a distributed training system with sparse gradients [28]. Ok-TOPK combines a decentralized parallel stochastic gradient descent (SGD) optimizer with a unique sparse all-reduce technique (less than 6k communication volume and asymptotically optimal). Ok-TOPK achieves model accuracy comparable to dense all-reduce, according to empirical results. Ok-TOPK is more scalable and boosts training performance significantly compared to the optimized dense and state-of-the-art sparse all-reduces (e.g., 3.29×–12.95× improvement for BERT on 256 GPUs).

Furthermore, a distributed framework was introduced for air quality prediction featuring Busan, Republic of Korea as its model city. To forecast the intensity of particle pollution, a deep-learning model was trained on a distributed system known as data parallelism (PM2.5 and PM10) [29]. To determine how the air quality particles are connected in space and time with the dataset distribution, multiple one-dimensional CNN layers are combined with a stacked attention-based BiLSTM layer to extract local spatial features.

The hybrid approach observed in the mentioned research involved asynchronously distributing data and the model within the same algorithm. For instance, the data was initially distributed and trained with the undistributed model, followed by distributing the undistributed model and training it with undistributed data. Additionally, the communication overhead between the GPUs was a more significant concern than the training and epoch time. The research also lacked practical comparisons, as the developed algorithms' training times were analyzed but not compared to state-of-the-art APIs.

References

1. Li, L.; Ota, K.; Dong, M. Humanlike Driving: Empirical Decision-Making System for Autonomous Vehicles. IEEE Trans. Veh. Technol. 2018, 67, 6814–6823.

2. Retallack, A.E.; Ostendorf, B. Current Understanding of the Effects of Congestion on Traffic Accidents. Int. J. Environ. Res. Public Health 2019, 16, 3400.
3. Tang, Y.; Huang, Z.; Chen, Z.; Chen, M.; Zhou, H.; Zhang, H.; Sun, J. Novel visual crack width measurement based on backbone double-scale features for improved detection automation. Eng. Struct. 2023, 274, 115158.
4. Tang, Y.; Zhu, M.; Chen, Z.; Wu, C.; Chen, B.; Li, C.; Li, L. Seismic performance evaluation of recycled aggregate concrete-filled steel tubular columns with field strain detected via a novel mark-free vision method. Structures 2022, 37, 426–441.
5. Takács, Á.; Rudas, I.; Bösl, D.; Haidegger, T. Highly Automated Vehicles and Self-Driving Cars . IEEE Robot. Autom. Mag. 2018, 25, 106–112.
6. Verhelst, M.; Moons, B. Embedded Deep Neural Network Processing: Algorithmic and Processor Techniques Bring Deep Learning to IoT and Edge Devices. IEEE-Solid-State Circuits Mag. 2017, 9, 55–65.
7. Ni, Z.; Yuksel, A.C.; Ni, X.; Mandel, M.I.; Xie, L. Confused or Not Confused? Disentangling Brain Activity from EEG Data Using Bidirectional LSTM Recurrent Neural Networks. In Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB'17), Boston, MA, USA, 20–23 August 2017; pp. 241–246.
8. Jin, P.H.; Yuan, Q.; Iandola, F.N.; Keutzer, K. How to scale distributed deep learning? arXiv 2016, arXiv:1611.04581.
9. Yuan, Z.; Xue, H.; Zhang, C.; Liu, Y. Hulk: Graph Neural Networks for Optimizing Regionally Distributed Computing Systems. arXiv 2023, arXiv:2302.13741.
10. Alimohammadi, M.; Markov, I.; Frantar, E.; Alistarh, D. L-GreCo: An Efficient and General Framework for Layerwise-Adaptive Gradient Compression. arXiv 2022, arXiv:2210.17357.
11. Song, Z.; Shi, L.; Pu, S.; Yan, M. Compressed gradient tracking for decentralized optimization over general directed networks. IEEE Trans. Signal Process. 2022, 70, 1775–1787.
12. Charles, Z.; Bonawitz, K.; Chiknavaryan, S.; McMahan, B.; Agüera y Arcas, B. Federated select: A primitive for communication-and memory-efficient federated learning. arXiv 2022, arXiv:2208.09432.
13. Lessley, B.; Childs, H. Data-parallel hashing techniques for GPU architectures. IEEE Trans. Parallel Distrib. Syst. 2019, 31, 237–250.
14. Lai, Z.; Li, S.; Tang, X.; Ge, K.; Liu, W.; Duan, Y.; Qiao, L.; Li, D. Merak: An Efficient Distributed DNN Training Framework with Automated 3D Parallelism for Giant Foundation Models. IEEE Trans. Parallel Distrib. Syst. 2023, 34, 1466–1478.

15. Zhang, J.; Tu, H.; Ren, Y.; Wan, J.; Zhou, L.; Li, M.; Wang, J. An adaptive synchronous parallel strategy for distributed machine learning. IEEE Access 2018, 6, 19222–19230.
16. Wu, W.; He, L.; Lin, W.; Mao, R.; Maple, C.; Jarvis, S. SAFA: A semi-asynchronous protocol for fast federated learning with low overhead. IEEE Trans. Comput. 2020, 70, 655–668.
17. Riba, E.; Mishkin, D.; Ponsa, D.; Rublee, E.; Bradski, G. Kornia: An Open Source Differentiable Computer Vision Library for PyTorch. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Snowmass Village, CO, USA, 1–5 March 2020.
18. Li, S.; Zhao, Y.; Varma, R.; Salpekar, O.; Noordhuis, P.; Li, T.; Paszke, A.; Smith, J.; Vaughan, B.; Damania, P.; et al. PyTorch Distributed: Experiences on Accelerating Data Parallel Training. arXiv 2020, arXiv:2006.15704.
19. Hao, Y.; Wang, S.; Cao, P.; Gao, X.; Xu, T.; Wu, J.; He, X. Attention in Attention: Modeling Context Correlation for Efficient Video Classification. IEEE Trans. Circuits Syst. Video Technol. 2022, 32, 7120–7132.
20. Yan, M.; Meisburger, N.; Medini, T.; Shrivastava, A. Distributed SLIDE: Enabling Training Large Neural Networks on Low Bandwidth and Simple CPU-Clusters via Model Parallelism and Sparsity. arXiv 2022, arXiv:2201.12667.
21. Devi, U.A.; Arulanand, N. Detection of Road Potholes Using Computer Vision and Machine Learning Approaches to Assist the Visually Challenged. In Smart Computer Vision; EAI/Springer Innovations in Communication and Computing (EAISICC); Kumar, B.V., Sivakumar, P., Surendiran, B., Ding, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2023; pp. 61–79.
22. Wang, D.; Liu, Z.; Gu, X.; Wu, W.; Chen, Y.; Wang, L. Automatic Detection of Pothole Distress in Asphalt Pavement Using Improved Convolutional Neural Networks. Remote Sens. 2022, 14, 3892.
23. Fan, R.; Özgünalp, U.; Wang, Y.; Liu, M.; Pitas, I. Rethinking Road Surface 3-D Reconstruction and Pothole Detection: From Perspective Transformation to Disparity Map Segmentation. IEEE Trans. Cybern. 2022, 52, 5799–5808.
24. Musa, A.; Hassan, M.; Hamada, M.; Kakudi, H.A.; Amin, M.F.I.; Watanobe, Y. A Lightweight CNN-Based Pothole Detection Model for Embedded Systems Using Knowledge Distillation. In Proceedings of the 21st International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques (SoMeT’22), Kitakyushu, Japan, 20–22 September 2022.
25. Kahira, A.N.; Nguyen, T.T.; Gomez, L.B.; Takano, R.; Badia, R.M.; Wahib, M. An Oracle for Guiding Large-Scale Model/Hybrid Parallel Training of Convolutional Neural Networks. In Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing (HPDC’21), Stockholm, Sweden, 21–25 June 2021; pp. 161–173.

26. Ye, X.; Lai, Z.; Li, S.; Cai, L.; Sun, D.; Qiao, L.; Li, D. Hippie: A Data-Paralleled Pipeline Approach to Improve Memory-Efficiency and Scalability for Large DNN Training. In Proceedings of the 50th International Conference on Parallel Processing (ICPP 2021), Lemont, IL, USA, 9–12 August 2021.
27. Sergeev, A.; Balso, M.D. Horovod: Fast and easy distributed deep learning in TensorFlow. arXiv 2018, arXiv:1802.05799.
28. Li, S.; Hoefler, T. Near-Optimal Sparse Allreduce for Distributed Deep Learning. arXiv 2022, arXiv:2201.07598.
29. Mengara Mengara, A.G.; Park, E.; Jang, J.; Yoo, Y. Attention-Based Distributed Deep Learning Model for Air Quality Forecasting. Sustainability 2022, 14, 3269.