cases	doc_1		doc_2		decision	id
	authors	• Sergio Botelho • Ameya Joshi • Biswajit Khara • Soumik Sarkar • Chinmay Hegde • Santi Adavani • Baskar Ganapathysubramanian	**authors**	• Sergio Botelho • Ameya Joshi • Biswajit Khara • S. Sarkar • C. Hegde • Santi S. Adavani • B. Ganapathysubramanian	DUPLICATES	284
	title	Deep Generative Models that Solve PDEs: Distributed Computing for Training Large Data-Free Models	**title**	Deep Generative Models that Solve PDEs: Distributed Computing for Training Large Data-Free Models		
	publication_date	2020-07-24 22:42:35+00:00	**publication_date**	2020-07-24 00:00:00		
	source	SupportedSources.ARXIV	**source**	SupportedSources.SEMANTIC_SCHOLAR		
	journal	None	**journal**			
	volume		**volume**			
	doi		**doi**	10.1109/MLHPCAI4S51975.2020.00013		
	urls	• http://arxiv.org/pdf/2007.12792v1 • http://arxiv.org/abs/2007.12792v1 • http://arxiv.org/pdf/2007.12792v1	**urls**	• https://www.semanticscholar.org/paper/481813caadf69883205537cd47d7b6ad6572ea04		
	id	id2349258471689071492	**id**	id-7532338892019481910		
	abstract	Recent progress in scientific machine learning (SciML) has opened up the possibility of training novel neural network architectures that solve complex partial differential equations (PDEs). Several (nearly data free) approaches have been recently reported that successfully solve PDEs, with examples including deep feed forward networks, generative networks, and deep encoder-decoder networks. However, practical adoption of these approaches is limited by the difficulty in training these models, especially to make predictions at large output resolutions ($\geq 1024 \times 1024$). Here we report on a software framework for data parallel distributed deep learning that resolves the twin challenges of training these large SciML models - training in reasonable time as well as distributing the storage requirements. Our framework provides several out of the box functionality including (a) loss integrity independent of number of processes, (b) synchronized batch normalization, and (c) distributed higher-order optimization methods. We show excellent scalability of this framework on both cloud as well as HPC clusters, and report on the interplay between bandwidth, network topology and bare metal vs cloud. We deploy this approach to train generative models of sizes hitherto not possible, showing that neural PDE solvers can be viably trained for practical applications. We also demonstrate that distributed higher-order optimization methods are $2-3\times$ faster than stochastic gradient-based methods and provide minimal convergence drift with higher batch-size.	**abstract**	Recent progress in scientific machine learning (SciML) has opened up the possibility of training novel neural network architectures that solve complex partial differential equations (PDEs). Several (nearly data free) approaches have been recently reported that successfully solve PDEs, with examples including deep feed forward networks, generative networks, and deep encoder-decoder networks. However, practical adoption of these approaches is limited by the difficulty in training these models, especially to make predictions at large output resolutions (â‰¥ 1024 Ã— 1024).Here we report on a software framework for data parallel distributed deep learning that resolves the twin challenges of training these large SciML models training in reasonable time as well as distributing the storage requirements. Our framework provides several out of the box functionality including (a) loss integrity independent of number of processes, (b) synchronized batch normalization, and (c) distributed higher-order optimization methods.We show excellent scalability of this framework on both cloud as well as HPC clusters, and report on the interplay between bandwidth, network topology and bare metal vs cloud. We deploy this approach to train generative models of sizes hitherto not possible, showing that neural PDE solvers can be viably trained for practical applications. We also demonstrate that distributed higher-order optimization methods are 2â€“3 Ã— faster than stochastic gradient-based methods and provide minimal convergence drift with higher batch-size.		
	versions		**versions**			