Porting AI/ML Models to Intelligence Processing Units (IPUs)

Abhinand S. Nasari*†
HPRC, Texas A&M University, College Station TX
abhinand@tamu.edu

Lujun Zhai†
Electrical and Computer Engineering, Prairie View A&M University
lzhai@pvamu.edu

Zhenhua He
HPRC, Texas A&M University, College Station TX
happidence1@tamu.edu

Hieu T. Le
Department of Electrical and Computer Engineering, Texas A&M University
hieult@tamu.edu

Jian Tao
Department of Visualization, Texas A&M University, College Station TX
jtao@tamu.edu

Suxia Cui
Electrical and Computer Engineering, Prairie View A&M University
sucui@pvamu.edu

Dhruva K. Chakravorty
HPRC, Texas A&M University, College Station TX
chakravorty@tamu.edu

Lisa M. Perez
HPRC, Texas A&M University, College Station TX
perez@tamu.edu

Honggao Liu
HPRC, Texas A&M University, College Station TX
honggao@tamu.edu

ABSTRACT

Intelligence processing units (IPUs) are specifically designed accelerators that are dedicated to support artificial intelligence (AI) and machine learning (ML) workflows. Here, we report on the performance characteristics and code-porting experiences on Graphcore IPUs offered on the new National Science Foundation (NSF)-funded Accelerating Computing for Emerging Sciences (ACES) testbed. Our benchmarks compared performance of AI/ML frameworks on ACES IPUS to similar runs on the Graphcloud environment, a commercial IPU cloud service offered by Graphcore. We also ported two PyTorch neural network models from Graphics Processing Units (GPUs) to IPUs to ensure the efficacy of the software environment. The ported models include the TransCycleGAN model that is used in reconstructing high-resolution images from low-resolution images, and the Hierarchical Autoencoder that is for large-scale high-resolution scientific data compression in climate models. These models were successfully ported on mulitple IPUs using utilities in the Graphcore Poplar software development kit. Increasing the number of IPUs resulted in a considerable enhancement in the model's throughput.

CCS CONCEPTS

• **Machine Learning**; • **Accelerators**; • **Performance Benchmarks**;

KEYWORDS

Accelerating Computing for Emerging Sciences (ACES), Intelligence Processing Unit (IPU), ResNet50, Image Super-Resolution, Data Compression

ACM Reference Format:
Abhinand S. Nasari, Lujun Zhai, Zhenhua He, Hieu T. Le, Jian Tao, Suxia Cui, Dhruva K. Chakravorty, Lisa M. Perez, and Honggao Liu. 2023. Porting AI/ML Models to Intelligence Processing Units (IPUs). In *Practice and Experience in Advanced Research Computing (PEARC '23), July 23–27, 2023, Portland, OR, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3569951.3603632

1 INTRODUCTION

Purpose-built accelerators for AI/ML applications offer opportunities to reduce the time taken for inference and training. The NSF ACES cyberinfrastructure (CI) testbed at Texas A&M University will offer various types of computing technologies that are designed to support specific research workflows. accelerators including co-processors, vector engines, GPUs and IPUs [1]. These technologies will be offered over a software-defined composable network, in anticipation of research workflows that can mix and match CI technologies of choice. In a previous work [2], we have shown the respective strengths of GPUs and IPUs in supporting AI/ML frameworks. In that work we leveraged GPUs at several campus and national cyberinfrastructure sites, and IPUs from the Graphcloud service. We also offered a description of how IPU benefits from an architecture that supports specific AI/ML applications [2]. Briefly, in an IPU, the Arithmetic Logic Units (ALUs) are divided into smaller units called tiles. Each IPU has 1472 tiles. The tiles act independently of one another, having their own local memory and enough compute power to prepare their own instructions. Tiles communicate with each other directly over high-bandwidth channels, even across different IPUs. The IPUs incorporates a large amount of in-processor memory composed of SRAM, which exists in the form of smaller independent distributed memory units. Additionally, the IPU is equipped with a set of DRAM chips (streaming

*High Performance Research Computing.
†Two authors contribute equally.

PEARC '23, July 23–27, 2023, Portland, OR, USA

ACM ISBN 978-1-4503-9985-2/23/07…$15.00
https://doi.org/10.1145/3569951.3603632

memory) that can transfer to the in-processor memory. This memory architecture enables rapid data communication within the IPU, allowing it to hold larger data compared to other processors [3]. IPU achieves massive parallelism with multiple instruction, multiple data (MIMD) technique, which is ideal for machine learning workloads such as image super-resolution and data compression models. To facilitate transitioning code to the IPUs, Graphcore offers a software development kit (SDK), named Poplar. Among others, the Poplar SDK includes performance tuning utilities, support for notebooks and application support for PyTorch models via the PopTorch utilities. By coupling the hardware environment to an extensive software applications and development suite, these IPUs are positioned to support scientific AI/ML applications that process large datasets. As part of deploying the ACES testbed, we discuss our results from testing the performance of IPUs at Texas A&M using standard AI/ML benchmarks. These IPUs are available for researchers via the NSF ACCESS program [28]. Here, we also present the results from porting two scientific AI/ML models used in data compression and image super resolution workflows from GPUs to IPUs.

1.1 TransCycleGAN Network Architecture

Image super-resolution (SR) is a challenging task that aims to reconstruct high-resolution (HR) images from low-resolution (LR) inputs while preserving clean content. However state-of-the-art SR methods face two major challenges: the lack of paired training data for most real-world images, and the difficulty in capturing long-range pixel dependencies using unpaired image restoration models based on convolution. Inspired by the image-to-image translation application of Cycle-Consistent Adversarial Network (CycleGAN [10]), and the capability of the Transformer to capture long-range pixel dependencies between image patches by leveraging the self-attention mechanism, TransCycleGAN was developed to address the above issues and tackle real-world image super-resolution, particularly for scenarios involving multiple degradations. It leverages pseudo- supervision based on unpaired learning samples to super-resolve real-world images. Specifically, TransCycleGAN builds upon CycleGAN architecture and incorporates an efficient, light-weight degradation-removal Transformer module to enhance the image quality. The TransCycleGAN network consists of an unpaired blur/noise correction network integrated with a degradation–removal Transformer in the Low-resolution domain and a pseudo-paired SR network for reconstructing High-resolution from a low resolution image. These networks perform two mappings: a) unpaired LR $\leftrightarrow$ clean LR translation for denoising and deblurring the LR image, and b) paired clean LR$\rightarrow$HR mapping for Image SR. The degradation-removal Generator is based on a multiscale, light-weight Transformer module that uses a 3-level encoder-decoder architecture based on Restormer [9]. Our TransCycleGAN model was trained on the Wider Face LR dataset [8] (with images of size 16x16 pixels) and CelebA HR dataset [5] (with images of size 64x64 pixels) in an unaligned manner.

1.2 Hierarchical Autoencoder Architecture for Climate Modeling

Another aspect of computing that will impact data availability is data compression. There are two types of data compression: lossless and lossy. Lossless compression allows for exact reconstructions of the original data, while lossy compression techniques offer a much better compression ratio within an acceptable reconstruction error. A deep learning technique for lossy compression, AutoEncoder is widely used to learn a representation of data in lower dimensions and reconstruct it from that representation. [16-18]. Our AutoEncoder model consists of an encoder (E), a quantizer (Q), and a decoder (D) as shown in Figure 1. The encoder learns the data representation and reduces the data dimension to output latent representation. Following the work by Razavi et al. on Vector Quantized Variational Autoencoder (VQ-VAE) [20], our encoder is quantized to produce integer quantized values, which are further compressed by a lossless algorithm, e.g. Huffman coding [15]. In this paper, we present our work to port a neural network model to IPUs. The model is based on the Hierarchical AutoEncoder (HAE) architecture proposed by Kim et al. [19]. The HAE architecture is an extension of the standard autoencoder with multiple layers of encoding and decoding. The input data is passed through two or more encoding layers, each of which produces a compressed representation of the input. The output of one encoding layer is the input to the next, creating a hierarchy of representations. Compared to a standard Autoencoder, the HAE allows the model to learn more complex patterns and dependencies in the data and construct a more compact and informative representation of the input. We have fully tested our model using SDRBench [11, 12], a publicly available benchmark dataset that is commonly employed in the scientific community [21]. SDRBench offers an array of simulation data covering a wide variety of domains – from atomic and molecular electronic structures to meteorological and cosmological data. We also tried our model on large-scale high-resolution climate modeling data sets generated by the High-Resolution Community Earth System Model (CESM) Version 1.3 [13, 14]. Our model currently achieves a compression ratio of ~200 with reconstruction error negligible for scientific analysis.

2 METHODS

2.1 Test Environment

Graphcore IPUs were deployed as part of the ACES testbed at the West Campus Data Center at Texas A&M University. These include the previous generation of 16 Colossus GC200 IPUs, and the current generation of 16 Bow IPUs that are situated on a dual AMD Rome CPU server on a 100 GbE RoCE fabric. The IPUs are one of several accelerators that are orchestrated using the Slurm Scheduler on the ACES testbed. To check the performance of the IPUs on the ACES computing system, we use the Graphcore GitHub repo [6], which provides different deep learning models for benchmarking purposes. In this study, we choose the popular and well-established deep learning model - ResNet50 in PyTorch deep learning framework using the Colossus GC200 Graphcore IPUs. The Graphcore github repository contains various application examples that have been optimized to run for both training and inference on the Ubuntu 20.04 operating system [6]. This was followed by porting the TransCyleGAN Network model for super resolution and the Hierarchical Autoencoder Architecture for Climate Modeling. These models used the PyTorch framework and were initially trained on GPUs. They were ported to run on IPUs using the Poplar Software Development Kit (SDK)

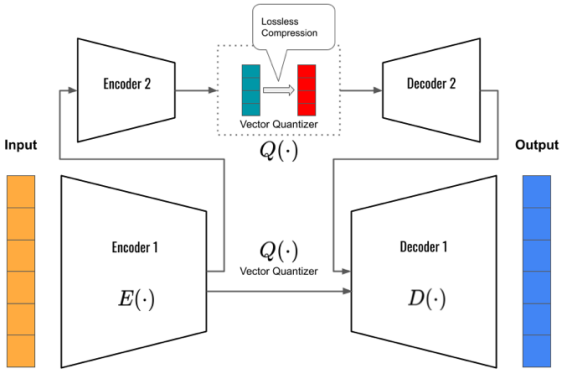

Figure 1: The architecture of Hierarchical Autoencoder. The input of the neural network is the original scientific data, and the output is the reconstructed data. In the ideal case, the input and output data are identical. The compression ratio achieved by the neural network is calculated by dividing the size of the input data (represented with orange blocks) by the size of the compressed latent space (represented by red blocks).

Table 1: Comparing the quality of images produced by superresolution using different models on the Set14x4 dataset. Higher PSNR and SSIM values indicate higher quality images. The TranCycleGAN model was implemented on NVIDIA A100 GPUs.

Method	PSNR	SSIM
SRGAN [24]	26.02	0.7397
EDSR [25]	28.80	0.7876
RCAN [26]	28.87	0.7889
HAT-L [27]	29.47	0.8015
TranCycleGAN	29.78	0.8103

3.2.0 with Poptorch 3.2.0. These represent the current generations of Poplar and Poptorch at the time of writing this manuscript.

2.2 Converting PyTorch models to Poptorch for Graphcore IPUs

Details about converting PyTorch models to poptorch have been described in detail on our training modules available at [22, 23]. To convert a PyTorch model to IPU code, we first need to import the Poptorch module, which is a PyTorch extension in the Poplar SDK. Next, we create a data loader using the poptorch.DataLoader class for efficient data batching with respect to the PopTorch framework. We then construct the model and include the loss computation in the forward function. Fourth, we utilize the poptorch.Options for compilation and execution on IPU. Finally, we train and evaluate the model with the poptorch.trainingModel and poptorch.inferenceModel wrappers, respectively. The Steps are shown in Fig 2.

For the TransCycleGAN model, we first migrated the TransCycleGAN model from CPUs to GPUs to ensure that the code worked on accelerators. Here we find that it outperforms other super resolution methods in terms of PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index), two metrics used to evaluate the quality of images produced by super resolution techniques. These values were compared to other established SR methods using the commonly used Set14x4 dataset of low resolution images.

The TransCycleGAN model was ported from GPUs to IPUs using the Poplar SDK as shown in Figure 2. While porting the code for Hierarchical Autoencoder to IP, we encountered a few issues related to coding styles permissible in Pytorch for GPU training, but incompatible with poptorch for IPUs. The common neural network model architecture patterns worked seamlessly after porting, but we needed to pay closer attention to some aspects of the model architecture to make it compatible with poptorch. For example, the vector quantizer module has a custom architecture and the backpropagation was a bit different compared to a general module. When we ran the model on IPUs, we got a Runtime Error of a leaf Variable that requires grad is being used in an in-place operation. The resolution we used was to do an explicit copy of data of the nn.Parameter instead of just calling a copy of nn.Parameter. More details can be found in the project GitHub [4]. Fixing some errors required a deeper understanding of the poptorch framework, and required assistance from scientists at Graphcore. Porting the inference part of the code was relatively straightforward compared to the training. In the following sections, we discuss the tuning required to achieve the performance on the hardware in the results sections.

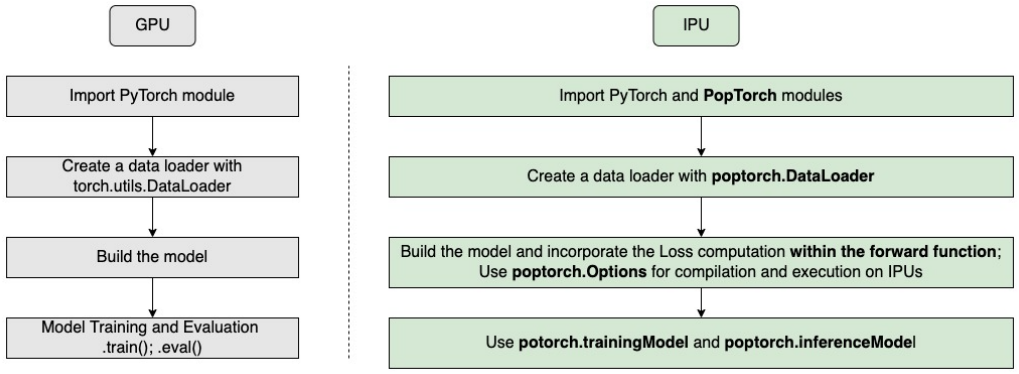

Figure 2: Steps taken to convert TransCycleGAN built on PyTorch from GPUs to poptorch on Graphcore IPUs.

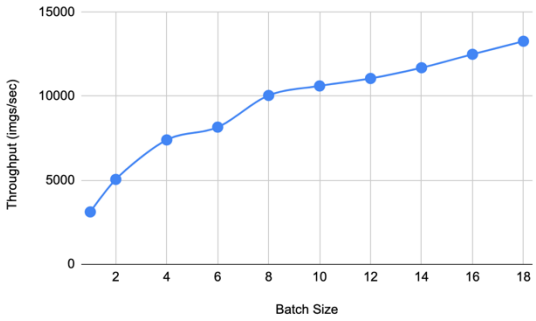

Figure 3: The image throughput of PyTorch ResNet 50 model increases with batch size. Smaller batch sizes (less than 4) had a greater increase in image throughput with batch size compared to larger batch sizes (greater than 8).

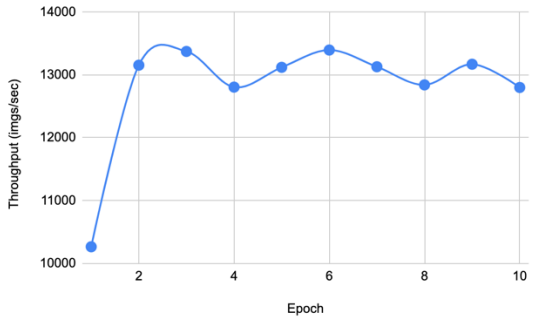

Figure 4: The image throughput with the number of epochs over 8 IPUs. The first epoch has lower performance due to graph compilation. The performance stabilizes after the second epoch.

3 RESULTS

3.1 Performance of IPUs on ACES for PyTorch ResNet50

To ensure that the Graphcore IPUs and their software environment were deployed appropriately we performed a scaling study on the popular ResNet 50 model. IPU architecture has limitations on batch sizes and needs tuning of some poptorch options like gradient accumulation and device iterations to speed up the training. With gradient accumulation, the gradients computed across multiple batches are accumulated before updating weights and this helps in speeding up the training. As part of this, we also noted the impact of characteristics such as batch size, precision on throughput. Batch size is an important hyperparameter that affects the image throughput and time to train. We first examined the effect of varying batch size on the image throughput of the PyTorch ResNet50 model on the ACES IPUs. The model is trained using image data from the ImageNet 2012 machine learning challenge [7]. In the study, the precision of the model is half precision and every experiment runs for two epochs. The throughput is from the second epoch. Our results indicate that image throughput increases as batch size increases, with batch size having a lesser effect as it becomes greater than 8. (Figure 3).

The performance of the PyTorch ResNet50 model changes with the number of epochs. As such we identify the appropriate number of epochs at which to report performance by identifying the number of epochs at which the image throughput stabilizes. As shown in Figure 4, we find that the PyTorch ResNet50 model on 8 IPUs with a batch size of 16 using half precision stabilizes after 2 epochs. The image throughput of the first epoch is lower than other epochs, which is due to the inclusion of graph compilation in this epoch and also observed in previous study [2]. Therefore, we report the throughput on the following experiments over 2 epochs.

We next investigate the effect of precision on throughput on the PyTorch ResNet50 model. Precision, i.e., the level of floating-point representation affects computations and memory operations and is an important configuration option that affects image throughput and possibly the model's accuracy. These calculations were performed on 1 IPU and 8 IPUs with a batch size of 2, using both half precision (16 bits) and full precision (32 bits). A small batch size (i.e., 2) was selected because a larger batch size caused an "out of

Table 2: Training performance of IPUs on the TransCycleGAN

Accelerator	Count	Images/sec
IPU	1	438
IPU	16	6552

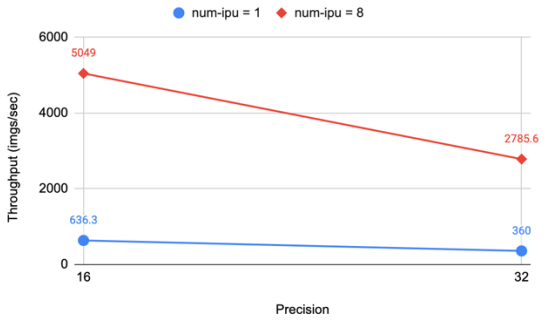

Figure 5: Impact of precision on the image throughput of PyTorch ResNet 50 model for 1 and 8 IPUs. Batch size was set at 2., and a similar performance drop (~44%) was observed in switching from half to full precision, for both 1 and 8 IPUs.

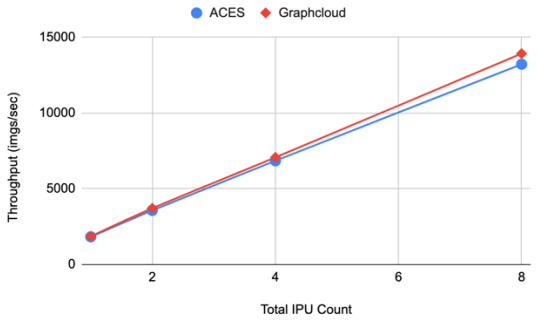

Figure 6: The image throughput increases almost linearly with the number of IPUs on the NSF ACES cluster (blue line) and the Graphcloud service (red line). To ensure maximum throughput, these calculations were performed using a batch size of 16 with half precision.

memory error" for full precision experiments. Our results indicate that there is a decrease in image throughput when switching from half precision to full precision. For 1 IPU, the throughput decreased by approximately 43.4%, and for 8 IPUs, the throughput decreased by approximately 44.8% as shown in Figure 5.

With a view towards ensuring maximum throughput, we next analyzed the scalability of the PyTorch ResNet50 model with 16 bits (i.e., half precision) and the batch size is set as 16 on the NSF ACES IPUs. Image throughput calculated from 1 to 8 IPUs finds almost linear behavior (Fig 6). Scalability calculations performed on the Graphcloud service [6], showed similar characteristics, helping validate the IPU set up and software environment for the ACES cluster.

In addition to exploring the trade-offs between batch size, and precision on the ResNet50 model, we explore the performance of the TransCycleGAN - Image super resolution model on IPUs. We find that the code scales appropriately to 16 IPUs and offers new opportunities for low energy consumption computing (Table 2).

In a similar effort, we also trained the Hierarchical AutoEncoder-based model with simulation data of Earth's sea surface temperature on IPUs and GPU platforms (Table 3). The data set consists of the ~100GB 2D sea surface temperature data with a size of 3600x1800 grid points each. The code was trained under different configurations of batch size, gradient accumulation, and deviceIterations parameters, and it scaled well from 1 to 16 IPUs with almost a 10x speedup. In a head-to-head comparison we note that the model gets comparable performance on a single NVIDIA A100 GPU compared to a single IPU. Our study shows valuable insights into the development and deployment of efficient data compression models on IPU, paving the way for the development of more efficient and optimized applications in the future.

4 CONCLUSIONS

This study is a part of a concerted effort to study the scaling and performance characteristics of accelerators on the ACES system by the SWEETER CyberTeam [29–31]. In our study, we show that the PyTorch ResNet50 model has an almost linear scaling behavior on the IPUs on the ACES system, ensuring that researchers will experience the same performance characteristics as on the Graphcloud cloud service. Similar performance changes are observed for different precision studies, regardless of the number of IPUs being used. Our study evaluated the performance of IPUs on the TransCycleGAN model and found that increasing the number of IPUs significantly improved the model's throughput. The Hierarchical Autoencoder model was found to scale well on IPUs, motivating us to explore and optimize the model's performance on other hardware platforms available on ACES. IPU gave a comparable performance and the code scaled seamlessly with the number of IPUs used. In the future, we anticipate studying the scaling and performance of other research applications on to IPUs and other accelerator technologies on the ACES testbed.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (NSF) award number 2112356 ACES - Accelerating Computing for Emerging Sciences, NSF award number 1925764 SWEETER - SouthWest Expertise in Expanding, Training, Education and Research, NSF award number 2019129 FASTER - Fostering Accelerated Scientific Transformations, Education, and Research, staff and students at Texas AM High-Performance Research Computing.

Table 3: Performance of the Hierarchical Autoencoder on IPUs.

Accelerator	Stage	Count	Images/sec
IPU	Training	1	1.69
IPU	Training	16	16.58
IPU	Inference	1	2.0

REFERENCES

[1] NSF ACES | Texas A&M High Performance Research Computing. Retrieved June 2, 2023 from https://hprc.tamu.edu/aces/

[2] Abhinand Nasari, Hieu Le, Richard Lawrence, Zhenhua He, Xin Yang, Mario Krell, Alex Tsyplikhin, *et al.* 2022. Benchmarking the Performance of Accelerators on National Cyberinfrastructure Resources for Artificial Intelli- gence/Machine Learning Workloads. *Practice and Experience in Advanced Research Computing* (2022), 1–9.https://doi.org/10.1145/3491418.3530772

[3] Graphcore IPU hardware overview. Retrieved June 2, 2023 from https://docs.graphcore.ai/projects/ipu-overview/en/latest/about_ipu.html

[4] Compression Vector-quantized Variational Autoencoder. Retrieved June 2, 2023 from https://github.com/abhinand5ai/tccs_torch

[5] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV).*

[6] Graphcore IPU example GitHub repository. Retrieved June 2, 2023 from https://github.com/graphcore/examples.

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. arXiv:1409.0575 [cs.CV]

[8] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. 2016. WIDER FACE: A Face Detection Benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[9] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. 2022. Restormer: Efficient Transformer for High-Resolution Image Restoration. arXiv:2111.09881 [cs.CV] *International conference on curves and surfaces.* Springer, 711–730.

[10] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2020. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. arXiv:1703.10593 [cs.CV]

[11] SDRBench. Retrieved June 2, 2023 from https://sdrbench.github.io

[12] K. Zhao, S. Di, X. Liang, S. Li, D. Tao, J. Bessac, Z. Chen, and F. Cappello, "SDRBench: Scientific Data Reduction Benchmark for Lossy Compressors", International Workshop on Big Data Reduction (IWBDR2020), in conjunction with IEEE Bigdata20.

[13] iHESP Archives. Retrieve June 2, 2023 from https://ihesp.github.io/archive

[14] Ping Chang, Shaoqing Zhang, Gokhan Danabasoglu, Stephen G. Yeager, Haohuan Fu, Hong Wang, Frederic S. Castruccio, Yuhu Chen, James Edwards, Dan Fu, Yinglai Jia, Lucas C. Laurindo, Xue Liu, Nan Rosenbloom, R. Justin Small, Gaopeng Xu, Yunhui Zeng, Qiuying Zhang, Julio Bacmeister, David A. Bailey, Xiaohui Duan, Alice K. DuVivier, Dapeng Li, Yuxuan Li, Richard Neale, Achim Stössel, Li Wang, Yuan Zhuang, Allison Baker, Susan Bates, John Dennis, Xiliang Diao, Bolan Gan, Abishek Gopal, Dongning Jia, Zhao Jing, Xiaohui Ma, R. Saravanan, Warren G. Strand, Jian Tao, Haiyuan Yang, Xiaoqi Wang, Zhiqiang Wei, and Lixin Wu. 2020. An Unprecedented Set of High-Resolution Earth System Simulations for Understanding Multiscale Interactions in Climate Variability and Change. Journal of Advances in Modeling Earth Systems 12, 12 (2020), e2020MS002298. https://doi.org/10.1029/2020MS002298 arXiv:https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2020MS002298 e2020MS002298 2020MS002298

[15] David A Huffman. 1952. A method for the construction of minimum-redundancy codes. Proceedings of the IRE 40, 9 (1952), 1098–1101.

[16] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation." , Parallel Distributed Processing. Vol 1: Foundations. MIT Press, Cambridge, MA, 1986.

[17] Ballard, "Modular learning in neural networks," Proceedings AAAI (1987)

[18] LeCun, Y. (1987). Modèles connexionistes de l'apprentissage. Ph.D. thesis, Université de Paris VI. 17, 499, 511

[19] Doyub Kim, Minjae Lee, and Ken Museth. 2022. NeuralVDB: High-resolutionSparse Volume Representation using Hierarchical Neural Networks. arXiv preprint arXiv:2208.04448 (2022).

[20] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with vq-vae-2. Advances in neural information processing systems 32 (2019).

[21] Hieu Le, Hernan Santos, and Jian Tao. 2023. Hierarchical Autoencoder-based Lossy Compression for Large-scale High-resolution Scientific Data. Manuscript submitted for publication.

[22] Texas AM High Performance Research Computing. Accessed on April 21, 2023. High Performance Research Computing Training. https://hprc.tamu.edu/files/training/2022/Fall/IPU_Training_Labs_Fall2022.pdf.

[23] Zhenhua He, Sandra Nite, Abhinand, Hieu Le, Jian Tao, Dhruva Chakravorty, Lisa Perez, Honggao Liu. 2023. Development of a Training Framework for Novel Accelerators. Manuscript submitted for publication.

[24] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, *et al.* Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4681–4690, 2017

[25] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution, 2017.

[26] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu.Image super-resolution using very deep residual channel attention networks. In Proceedings of the European conference on computer vision (ECCV), pages 286–301, 2018

[27] Xiangyu Chen, Xintao Wang, Jiantao Zhou, and Chao Dong. Activating more pixels in image super-resolution transformer.arXiv preprint arXiv:2205.04437, 2022

[28] NSF - Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support. Retrieved June 2, 2023 from https://access-ci.org/

[29] Zhenhua He, Aditi Saluja, Richard Lawrence, Dhruva K. Chakravorty, Francis Dang, Lisa M. Perez, and Honggao Liu. 2023 (Accepted). Performance of Distributed Deep Learning Workloads on a Composable Cyberinfrastructure. In *Practice and Experience in Advanced Research Computing (PEARC '23), Portland, OR, USA.* ACM, New York, NY, USA, 12 pages.

[30] Edmundo Medina-Gurrola, Dhruva K. Chakravorty, Diana V. Dugas, Tim Cockerill, Lisa M. Perez, Emily Hunt. (2022). Regional Collaborations Supporting Cyberinfrastructure-Enabled Research During a Pandemic: The Structure and Support Plan of the SWEETER CyberTeam. In Practice and Experience in Advanced Research Computing (PEARC '22), 4 pages. https://doi.org/10.1145/3491418.3535186

[31] Richard Lawrence, Dhruva K. Chakravorty, Francis Dang, Lisa M. Perez, Wesley Brashear, Zhenhua He, and Honggao Liu. 2023 (Accepted). Developing Synthetic Applications Benchmarks on Composable Cyberinfrastructure: A Study of Scaling Molecular Dynamics Applications on GPUs. In *Practice and Experience in Advanced Research Computing (PEARC '23), Portland, OR, USA.* ACM, New York, NY, USA, 6 pages.