

Benchmarking and Optimizing Federated Learning with Hardware-related Metrics

BMVC 2024 Submission # 369

1 Supplementary Materials

In this section, we provide more information to illustrate the feasibility of our work. In summary, we provide our project code with a README file and a supplementary of our experimental data.

1.1 Project Code

We use the FedAvg algorithm and the FedOpt algorithm as examples to prove the effectiveness of FEDHW, and create two source codes for each algorithm, which are implemented using Python and PyTorch [1] and ran on the desktop PC. In the source code, we simulate the training process in federated learning by running the training process of the selected clients sequentially. Subsequently, the trained models are aggregated to update the global model. We present instructions for use in the README file.

1.2 Regression Models

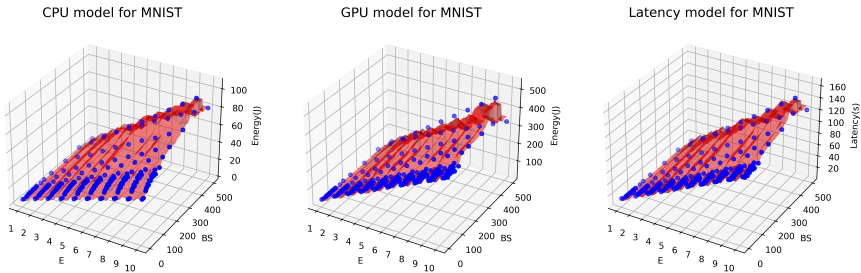


Fig. 1: Regression Model for MNIST

The FEDHW framework requires the utilization of client-side hardware consumption data throughout the client training process. Consequently, we monitor all of the hardware-related matrices on Jetson TX2 for each dataset, including the CPU energy consumption, GPU energy consumption, and training latency. For the CIFAR-10 and ESC50 datasets, we

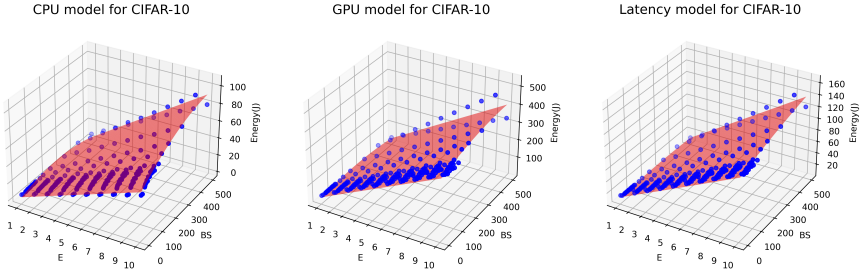


Fig. 2: Regressionn Model for CIFAR-10

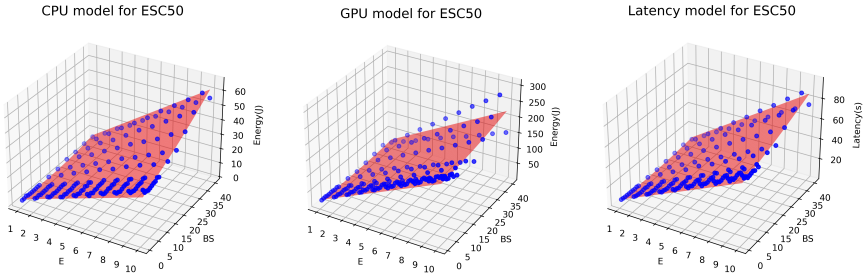


Fig. 3: Regressionn Model for ESC50

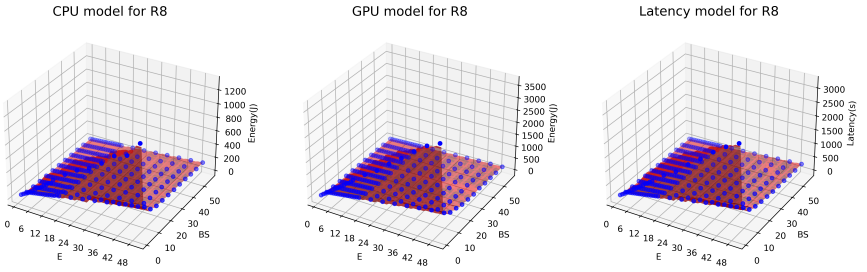


Fig. 4: Regressionn Model for R8

employ a polynomial regression model to simulate the consumption of the client training process. Similarly, for the MNIST and R8 datasets, we utilize a random forest regression model. We calculate the coefficient of determination (R^2 score) for each model. The lowest score is 0.89 for the GPU model of ESC50, while the other scores are above 0.96. From this aspect, we think our models are reliable for simulating the hardware condition of the client training.

1.3 Experimental Supplementation

Dataset	FedAvg	SA	GA	FedOpt	SA	GA
MNIST	6615.94	2859.18	6452.30	9181.47	2733.54	4401.54
CIFAR10	1537383.17	5224937.94	339416.31	588605.37	357513.33	373118.82
ESC50	449330.61	310696.58	404241.07	413345.01	257735.83	213490.55
R8	220048.99	18566.28	15999.36	1265090.98	95998.97	36084.14

Table 1: Energy consumption for each dataset, unit is Joule (J)

Dataset	FedAvg	SA	GA	FedOpt	SA	GA
MNIST	6.06	1.50	3.89	8.27	1.88	2.37
CIFAR10	621.61	212.73	175.77	230.91	161.71	180.36
ESC50	168.75	133.46	102.77	162.08	74.68	23.51
R8	277.48	58.25	48.24	1589.22	75.40	65.34

Table 2: Latency for each dataset, unit is minute (min)

Tbl. 1 and Tbl. 2 present the complete results of energy consumption and latency, respectively. As the configuration is identical for different methods of each dataset, we calculate the baseline consumption by multiplying the consumption of the first round in SA by the number of training rounds in the baseline.

When running client training and monitoring consumption data on Jetson TX2, we make use of the Darknet framework to implement the training process of MNSIT, CIFAR-10, and ESC50 datasets [4], while using PyTorch to implement the training process of R8 datasets. Consequently, the consumption of the R8 dataset training process is greater than that of the others due to the differing implementation.

References

[1] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalam-barkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xi-aodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit

Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366. URL <https://pytorch.org/assets/pytorch2-2.pdf>.

[2] Joseph Redmon. Darknet: Open source neural networks in c. <http://pjreddie.com/darknet/>, 2013–2016. Accessed on: May 1, 2024.