Optimizing Deep Learning Material Interface Reconstruction

VALEN YAMAMOTO*, University of California, Irvine

ACM Reference format:
Valen Yamamoto. 2021. Optimizing Deep Learning Material Interface Reconstruction. 1, 1, Article 1 (October 2021), 2 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Material interface reconstruction is the process of constructing boundaries between two or more immiscible materials from a discrete mesh. Current methods require trade-offs between the conservation of material and the continuity of the border. Using a convolutional autoencoder neural network as a surrogate model creates more accurate reconstructions.

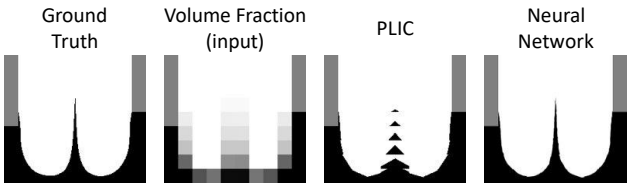

Fig. 1. Example material interface reconstruction

Figure 1 shows the discrete mesh created from volume fractions taken from the ground truth and compares the output from PLIC, a current reconstruction method, and the neural network. The neural network is able to reconstruct a continuous boundary that is much closer to the ground truth.

2 INITIAL RESULTS

Within the physics simulation, material interface reconstruction is embedded within the innermost loop of the workflow; in order to not impede physics calculations that use the model, the target throughput of the model is 100,000 samples per second. The initial model contained 5 convolution and deconvolution layers with 32 convolution channels each. When flattened, these convolutions produced fully connected layers of size 132k in between the convolution and deconvolution layers. Measurements of the initial model on Pascal P100 GPUs show that the model does not come close to the target throughput, as shown in Figure 2.

Profiling shows that the initial model is bottlenecked by the fully connected layers, which takes 50% of the compute time. Reducing the size of the fully connected layers is needed to increase the throughput.

*Advisors: Ian Karlin and Dan Fenn, Lawrence Livermore National Laboratory

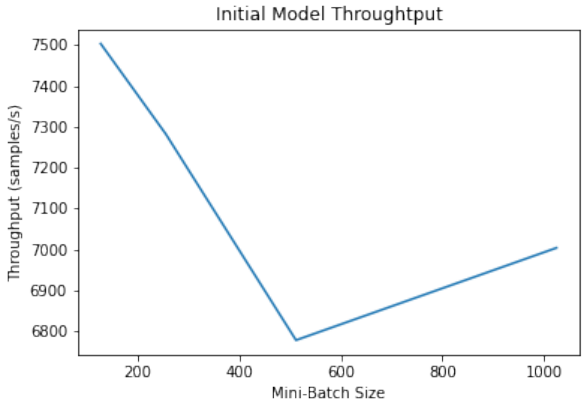

Fig. 2. Initial model measurements on Pascal NVIDIA P100 GPUs

3 REDUCING THE SIZE OF THE MODEL

3.1 Model Optimization

The variable that most impacts the size of the fully connected layer is the number of convolution channels. In addition, further optimization of model size can be accomplished by adding pooling layers to downsample after every convolution. The number of convolution channels, pooling kernel size, and pooling dilation were tuned to find the best accuracy. Models are trained with mean squared error (MSE) and additional penalties for the conversation of material and the continuity of the border.

Results of tuning model hyperparameters can be seen in Table 1. When changing the number of convolution channels, all models converge to the same error with about the same conservation and continuity penalties. Two convolution channels provided the best mean squared error and comparable conservation and continuity errors to the other options. Validation errors of the pooling models are also converge to around the same error. Of the pooling variables that were explored, a dilation of one and kernel size of five were the best combination of hyperparameters.

3.2 Choosing a Reduced Model

After applying the two optimizations, the models with the best validation error are a pooling model with a dilation of 1 and 2 convolution channels, and a fully connected layer size of 4608, and a model with 2 convolution channels and no pooling, with a fully connected layer size of 8192.

To choose between the two models, 32 of each model were trained to convergence. The mean error of the pooling model is 4.77% and the standard deviation is 0.1129%; the mean error of the model without pooling is 4.9548% and the standard deviation is 0.4145%. Doing a two-sample t-test gives a t-score of -2.0399 and a p-value of 0.02166, showing the the pooling model is the best reduced model.

Model Parameters	FC Layer Size	MSE	Conservation Error	Continuity Error
32 Channels (Original Model)	131072	0.0567	0.1160	0.0918
2 Channels	8192	0.0513	0.1098	0.0789
4 Channels	16384	0.0559	0.1099	0.0804
Pooling (Dilation 1 2 Channels)	4604	0.0466	0.1082	0.0783
Pooling (Dilation 2, 2 Channels)	2048	0.0535	0.1120	0.0877

Table 1. Comparison of errors of trained models with different hyperparameters

4 FINAL RESULTS

GPU measurements were collected on Lawrence Livermore National Laboratory's IPA testbed, which has A100 GPUs. Measurements are node-local inference times, where input data is generated and inference is conducted on the same node.

Running the reduced model reaches the target throughput of 100,000 samples per second on NVIDIA Tesla A100 GPUs, which is 15 times the throughput of the initial model on Pascal. Figure 3 compares the results from the reduced model on A100 GPUs with a naive PyTorch implementation and optimizations with CUDA Graphs, TensorRT, and a combination of both.

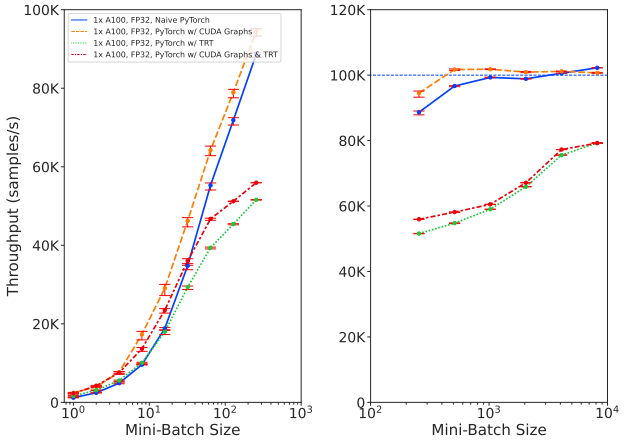

Fig. 3. Results from running the model on Nvidia A100 GPUs with various optimizations.

Reducing the size of the fully connected layer moves the computational bottleneck from the fully connected layers to the convolution layers. Further adding the pooling layers doubles the convolution time, which reduces the throughput of the model resulting in 70% of computation time spent in convolution kernels. Because the model with 2 convolutional channels and the model with pooling have very close validation errors, the model without pooling could be used in applications where more throughput is needed at the expense of accuracy.

The model was also run on SambaNova Systems' SN-10 hardware accelerator, which are connected to LLNL's Corona supercomputer. SambaNova optimized a version of the model without layernorm for their dataflow architecture. Results from SambaNova are for local inference; if used in the phsyics simulation workflow, some time would be needed for communication between SambaNova's systems and Corona. These results can be considered an upper bound for performance.

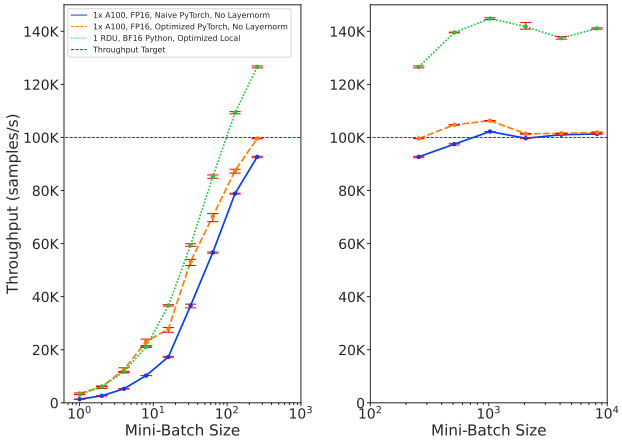

Fig. 4. Results from running the model without layernorm on Nvidia A100 GPUs and SambaNova Systems' SN-10.

SambaNova's results can be seen in Figure 4, along with those of the Nvidia A100 GPUs, both running the model without layernorm at half-precision. SambaNova exceeds the throughput target at a smaller mini-batch size than the GPUs and has a higher peak than the GPUs. At lower mini-batch sizes, Nvidia GPUs optimized with CUDA Graphs are competitive with SambaNova local inference times.

5 NEXT STEPS

Work is currently being done to port the full model to SambaNova Systems as well as to address TensorRT results by porting first to ONNX to use the onnx2trt converter.

6 ACKNOWLEDGEMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC. LLNL-ABS-826065