White Paper

Oil and Gas
Deep Learning on Seismic Data

Accelerated Deep Learning on Seismic Data with Larger Tile and int8 Precision

Larger tile inference with lower precision is the most optimal way to perform deep learning for seismic interpretation

Contributors

Flaviano Christian Reyes
Manas Pathak
Ravi Panchumarthy
Alexey Khorkin
Alexey Gruzdev

Table of Contents

Introduction

Deep learning for seismic interpretation is used by geophysicists and earth scientists for quick interpretation of the vast amount of seismic data that can vary in range from tens of gigabytes all the way to terabytes. Recent research[1,2] has shown that a CNN model well trained on synthetic seismic data can do the inference/interpretation on new data with acceptable accuracy. This leads to a possibility where a pretrained CNN model can fly by a large amount of seismic data for quick insights to help exploration scientists accelerate finding oil and gas deposits. The application of seismic interpretation is also relevant for geothermal systems where the presence of faults in the subsurface can impact their feasibility.

Following the research on the adverse effect of tiling performed on MRI data and satellite data, we started to investigate int8 convolutions for seismic data that use int8 data–type acceleration powered by Intel® Deep Learning Boost (Intel® DL Boost) while performing inference on larger tile sizes to boost accuracy in automatic seismic interpretation. Limitations in memory size in deep learning accelerator cards prevent relatively large images—such as those from medical, satellite, or seismic—from being processed in their original resolution. As a result, a typically fully convolutional topology, such as U-Net, is trained on down-sampled images and inferred on images of their original size and resolution by simply dividing the larger image into smaller (typically overlapping) tiles, making predictions on these tiles, and stitching them back together as the prediction for the whole image.[3] In this study, we show that this tiling technique may have a huge impact on prediction accuracy from the model. We used 2nd Generation Intel® Xeon® Scalable processors and the Intel® Distribution of OpenVINO™ toolkit to enable fast inference on a 3D seismic data set.

Method

A modified public U-Net–based model[4] was used to classify facies—a task typically needed in the seismic interpretation that can be regarded as a segmentation task in a typical computer vision. We made two notable modifications in the decoder part of the topology: We swapped 2D convolution transpose layers with regular convolution layers and 2D max unpool layers with interpolate layers. These changes did not have a major impact on the accuracy but yielded better latency performance than the original model proposed by the author. Facies classification can occur by two inference paradigms: 1) patch inference, and 2) section inference. Perhaps intuitively, patch inference denotes inference done on smaller tiles of seismic data while section inference denotes inference done on larger tiles. The test script for benchmarking accuracy provided by the author of the facies classification model[4] was used to produce metrics such as pixel accuracy, mean class accuracy, and IoU metrics. Their test script conducts inference on both patches and sections. The sizes of these sections range from 99x99 to 255x701. The seismic data used is from F3 Dutch block[5] in the North Sea.

To test performance benchmarking, we use a suite of benchmarking scripts that give randomly generated input tensors to the models. With these randomly generated matrices, models are tested on two inference tasks: 1) single-section inference of 255x701, and 2) patch-based inference (99x99) to complete the full section of 255x701. The sizes were selected based on the size extremes presented within the data set introduced by the author.[4] In other words, 99x99 was the smallest input size encountered, and 255x701 was the largest input size encountered. From here on, the section would mean a section of tile sized 255x701, and the patch would mean a patch of tile sized 99x99. As different scenarios of tiling are possible in deep learning, a section can be considered a larger tile, while a patch is a smaller tile.

Result

The matrix below shows all possible configurations of tiling in deep learning, due to different sizes used in the training and inference phase, i.e., section vs. patch training and section vs. patch inference. We introduced another hybrid category that does 80 percent training on sections and the remaining 20 percent training on patches. The accuracy metric used was Intersection over Union (IoU). IoU achieved in each scenario is displayed in Table 1, and the color coding denotes the top three scenarios, the darker green shade being the best. We discovered that the best accuracy in inference is achieved when the model is trained on a section, and inference is also performed on a section. The industry standard scenario—where the model is trained on patches and inference is performed on patches—is the second-worst performer. It is worth noting that these are mean IoU values for all classes, and differences are even more drastic in a few of the individual classes. The hybrid approach provides the next-best accuracy after section trained—section inference. These inferences were performed in full precision, i.e., FP32 data type.

Model: Aug skip interpolate conv (accuracy)	Patch inf (FP32)	Section inf (FP32)
Patch trained	0.547	0.646
Section trained	0.475	0.758
Hybrid trained (80% epochs on patch and 20% epochs on section)	0.718	0.748

Table 1: Mean class accuracy (IoU) for different scenarios with inference in FP32 precision.

Recommended configurations

Following the determination of accuracies of different configurations in FP32, we decided to quantize the model to perform convolutions in int8 precision to leverage Intel Deep Learning Boost (Intel DL Boost), a new set of embedded processor technologies on 2nd Generation Intel Xeon Scalable processors (Cascade Lake), designed to accelerate deep learning inference. Intel DL Boost includes new Vector Neural Network Instructions (VNNI) that enable int8 deep learning inference. int8's lower precision increases power efficiency by decreasing compute and memory bandwidth requirements, which produces significant performance benefits. The quantization pipeline workflow was published in a prior blog post.[6]

Table 2 shows the drop in accuracy of the int8 model in section trained. Section inference was almost negligible and was still far better than the worst-case scenario, patch trained and patch inference.

Model: Aug skip interpolate conv (accuracy)	Patch inf (int8)	Section inf (int8)
Patch trained	0.518	0.638
Section trained	0.454	0.758
Hybrid trained (80% epochs on patch and 20% epochs on section)	0.72	0.747

Table 2: Mean class accuracy (IoU) for different scenarios with inference in int8 precision.

Based on the above observation, the following were determined to be the recommended ways to perform deep learning:

1. Train on sections, perform inference on sections with int8 precision
2. Hybrid training,* perform inference on sections with int8 precision
3. Hybrid training,* perform inference on patches with int8 precision

*Hybrid training: Train 80 percent epochs on patches and remaining 20 percent epochs on sections.

Performance

Table 3 shows latency (time to infer) for different scenarios with inference in int8 precision. The 3372.2 ms in completing inference of full section of 255x701 via patches of size 99x99 is way too large compared with inference time of 18.9 ms section in a section train, section inference (int8) configuration. The reason is that the number of iterations required to complete the inference of a full section via patches is very high. To understand the change in accuracy with variable stride, we performed some follow-up experiments with the results shown in Figure 1. It was clear that a lower stride (more overlaps in patches) is needed for maintaining good accuracy in patch-based inference, which unfortunately increases the number of iterations needed to complete the full-section inference via patches.

Model configuration	Batch size	Intel® Xeon® Gold 6252N with the Intel® Distribution of OpenVINO™ toolkit (2021.1)
Section inference FP32	1	65 ms
Section inference int8	1	18.9 ms
Patches (99x99) to complete 1 full section (255x701) inference int8	1	3372.2* ms *Each 99x99 patch latency is 2.1 ms and took 277,200 iterations at stride 10 to complete full-section inference

Table 3: Latency (time to infer) for different scenarios with inference in int8 precision.

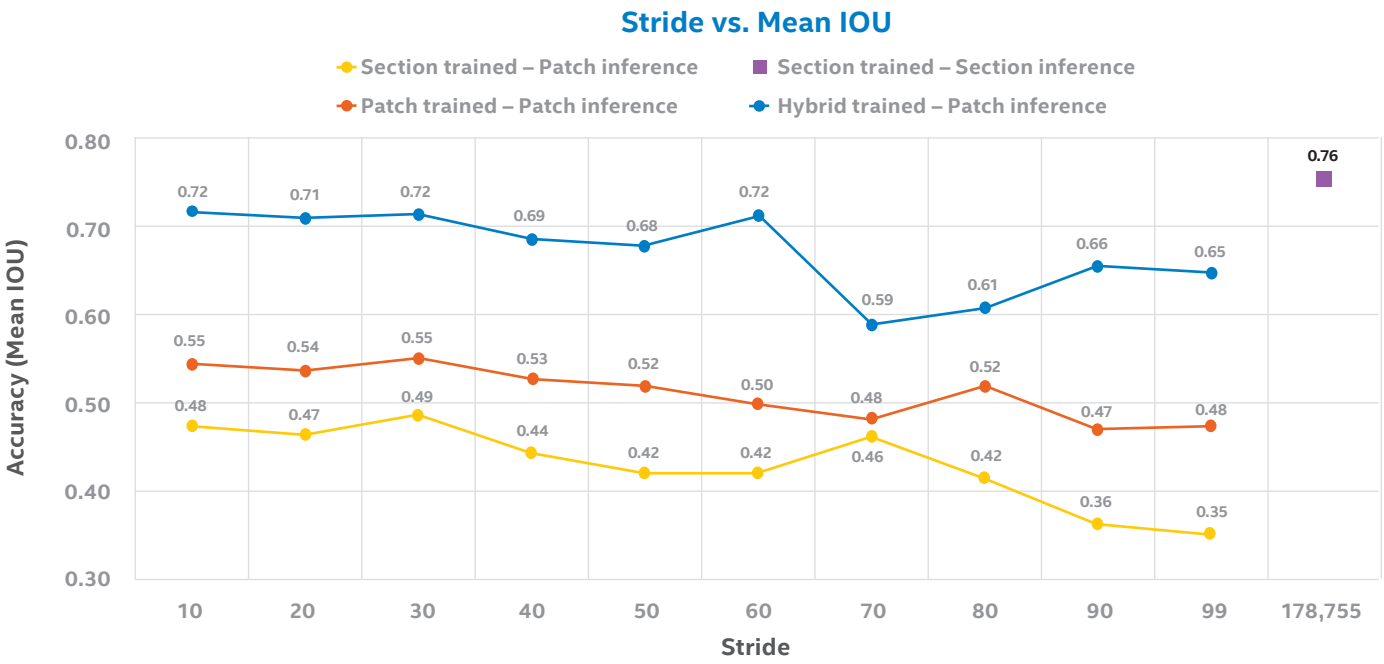

Figure 1: Stride vs. accuracy for patch inference. See the last page for configuration details. Results may vary.

Figure 1 shows that it is important to have a lower stride to have good accuracy. The drawback is that more iterations are needed in lower stride (Figure 2), which increases the total time to infer full section via patch-based inference. Thus, it makes much more sense from both accuracy and performance points to perform the same task of inference with section.

Table 3 also shows that the int8 performance for section train, section inference configurations has drastically better latency (65 ms > 18.9 ms) due to software- and hardware-based combined optimization using OpenVINO toolkit int8 model on Intel® Xeon® processors leveraging VNNI technology. Figure 3 shows the comparison of FP32 vs. int8 precision layer–wise execution time of the Intel Distribution of OpenVINO toolkit on Intel® Xeon® Gold 6252N.

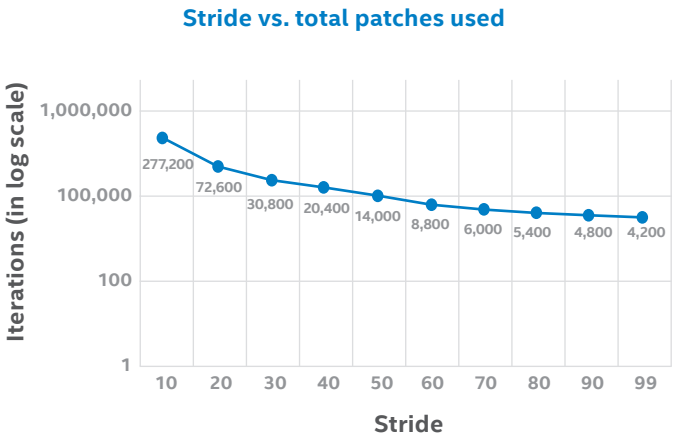

Figure 2: Stride vs. iterations for patch inference. See the last page for configuration details. Results may vary.

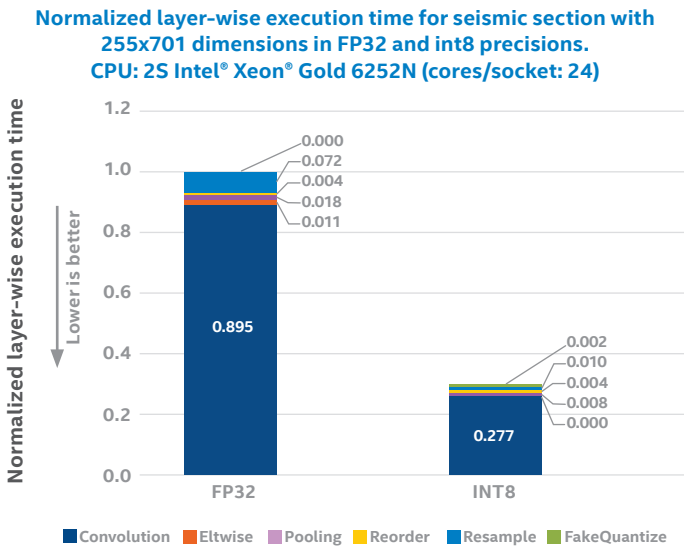

Figure 3: The graphs show the comparison of FP32 vs. int8 precision layer–wise execution time of the Intel® Distribution of OpenVINO™ toolkit on Intel® Xeon® Gold 6252N. See the last page for configuration details. Results may vary.

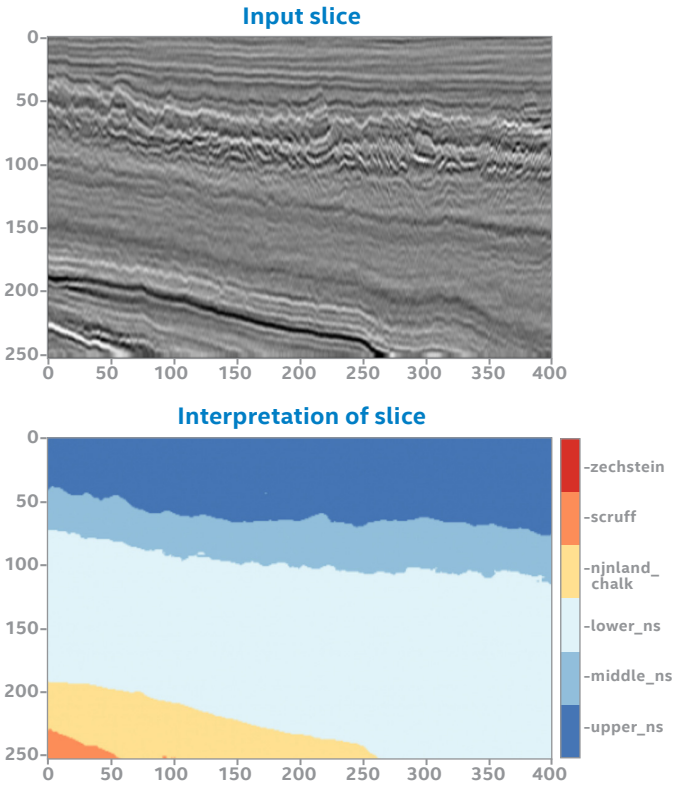

Figure 4: This shows the output of the modified U-Net Topology on seismic data. The top part is a shifted and scaled version of the inline slice input.

Conclusion

In summary, we recommend that the best accuracy is achieved when both training and inference are performed on sections. If a pretrained model trained using a patch is already available, the next-best accuracy can be achieved by retraining the model for 20 percent of the epochs on sections. All inference in int8 precision is faster and almost equally as accurate as their FP32 counterparts. The section-based inference approach requires higher memory, which may be a limiting factor in deep learning accelerator cards. However, these limits should not prevent inference on relatively large images for the sake of better accuracy achieved in inference on larger tiles.

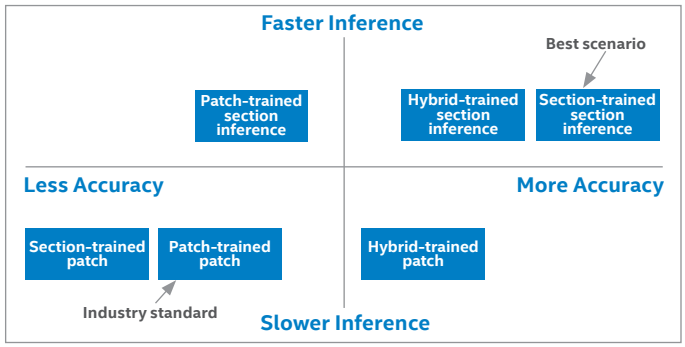

Figure 5: Accuracy vs. latency vs. memory requirement (larger bubble indicates higher memory).

Figure 5 clearly shows the different configurations possible. The recommended configurations are:

1. Section trained – Section inference int8
2. Hybrid trained – Section inference int8
3. Hybrid trained – Patch inference int8

References

1. Xinming Wu, Luming Liang, Yunzhi Shi, and Sergey Fomel, (2019), "FaultSeg3D: Using synthetic data sets to train an end-to-end convolutional neural network for 3D seismic fault segmentation," GEOPHYSICS 84: IM35-IM45. https://library.seg.org/doi/10.1190/geo2018-0646.1
2. York Zheng, Qie Zhang, Anar Yusifov, and Yunzhi Shi, (2019), "Applications of supervised deep learning for seismic interpretation," The Leading Edge 38: 526–533. https://library.seg.org/doi/10.1190/tle38070526.1
3. Reina, G. A., Panchumarthy, R., Thakur, S. P., Bastidas, A., and Bakas, S. (2020). "Systematic Evaluation of Image Tiling Adverse Effects on Deep Learning Semantic Segmentation," Frontiers in Neuroscience, 14, 65. https://doi.org/10.3389/fnins.2020.00065
4. Yazeed Alaudah, Patrycja Michałowicz, Motaz Alfarraj, and Ghassan AlRegib, (2019), "A machine-learning benchmark for facies classification," Interpretation 7: SE175-SE187. https://doi.org/10.1190/INT-2018-0 249.1 or https://arxiv.org/pdf/1901.07659.pdf
5. https://terranubis.com/datainfo/Netherlands-Offshore-F3-Block-Complete
6. https://www.intel.com/content/www/us/en/artificial-intelligence/posts/openvino-low-precision-fault-segmentation.html

Configuration	**Config1**
Test by:	Intel
Test date:	09/08/2020
Platform:	Intel® Xeon® Gold 6252N CPU @ 2.30 GHz
# Nodes:	1
# Sockets:	2
CPU:	96
Cores/socket, threads/socket:	24/48
ucode:	0x5002f01
HT:	On
Turbo:	On
BIOS version:	4.1.13, 0x5002f01
System DDR mem config:	
slots/cap/run speed:	DDR4: 12/16GB/2933 MHz
Total memory/node (DDR+DCPMM):	192 GB
Storage – application drives:	439.56 GB
OS:	Ubuntu 18.04.5 LTS
Kernel:	4.15.0-112-generic
Mitigation variants (1,2,3,3a,4, L1TF):	Mitigated

https://github.com/speed47/spectre-meltdown-checker