MERIT Internship Report

October 4, 2023
Department of Physics, the University of Tokyo
MERIT-WINGS 11th
HIRAKIDA, Ryosuke

Overview

Recipient: Preferred Networks, Inc.
Period: August 10, 2023 – September 27, 2023
Theme: Development of frameworks for implementation of deep learning models

Background

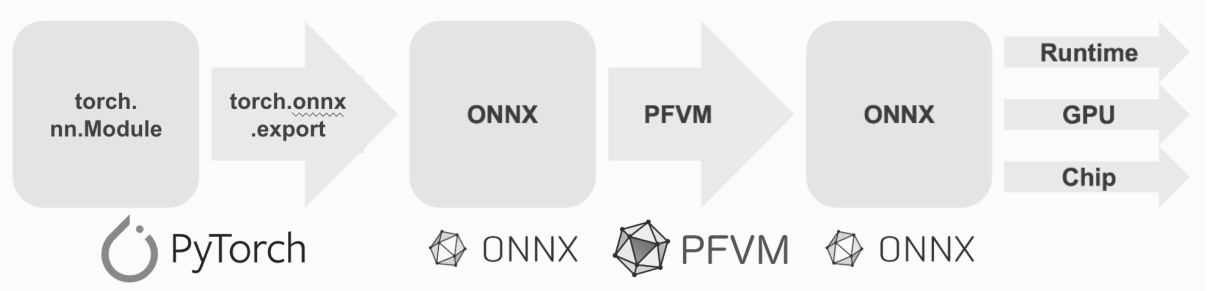

Figure 1. Typical deep learning inference workload

Figure 1 provides a schematic overview of a typical deep learning inference workload. In the workload, models built in frameworks such as PyTorch are executed at runtime or on a chip, but before that, they are often converted to a unified format such as ONNX and optimized.
PFN has an in-house optimizing compiler runtime for deep learning computational graphs called PFVM, which uses a model format called ONNX for its input and output.
The DL Ecosystem team is involved in the development and utilization of these optimization frameworks and needed a more flexible exporter of ONNX to reduce the cost of exporting ONNX.

Issues of existing exporters

One existing ONNX exporter is torch.onnx.export, developed by PyTorch. However, torch.onnx.export has the following problems

1. To get a computational graph of a deep learning model, the model must actually be run. In many cases the exporter needs to be run on the CPU, and getting the computed graph takes a long time
2. The C++ API of torch.jit, which can be used via pybind11, is heavily used to develop exporters. Therefore, it is not easy to extend or debug

To solve these issues, the following innovations were made in the development of the new exporter

3. Leverage PyTorch's new graph acquisition method (torch.fx.symbolic_trace/torch.compile)
4. Leverage the PyTorch and ONNX ecosystem (especially torch.fx.Graph and ONNX Script) implemented in Python

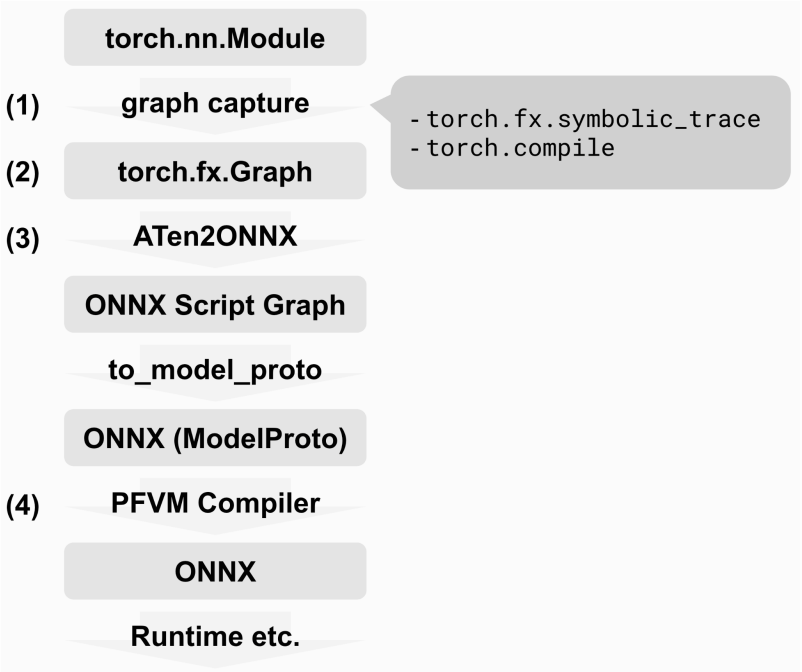

Figure 2. Processing performed inside the exporter

Figure 2 shows a schematic representation of the process performed inside the exporter: the computed graph is obtained from the torch.nn.Module constructed in PyTorch as torch.fx.Graph. The ONNX model is then constructed through the process of converting ATen functions into ONNX operators. The constructed ONNX model is optimized by a framework such as PFVM, and then executed at runtime or on the actual device. Points (1) through (4) of this flow have developed in this internship.

Evaluation of the developed exporter

1. Exportable models

Using the newly developed exporter, we were able to output ONNX models for inference and training of MNIST and torchvision's ResNet. We also achieved output (inference only) of T5, a language model developed by Google, as a relatively practical model. However, the T5 output was only achieved using the custom fx.Tracer supported by Hugging Face, and it appears that modifications would need to be made not only to the ONNX exporter but also to the acquisition of the computed graph to allow a wider range of models to be output using symbolic_trace. ONNX exporter as well as the acquisition of the computed graphs.

2. Time required for export

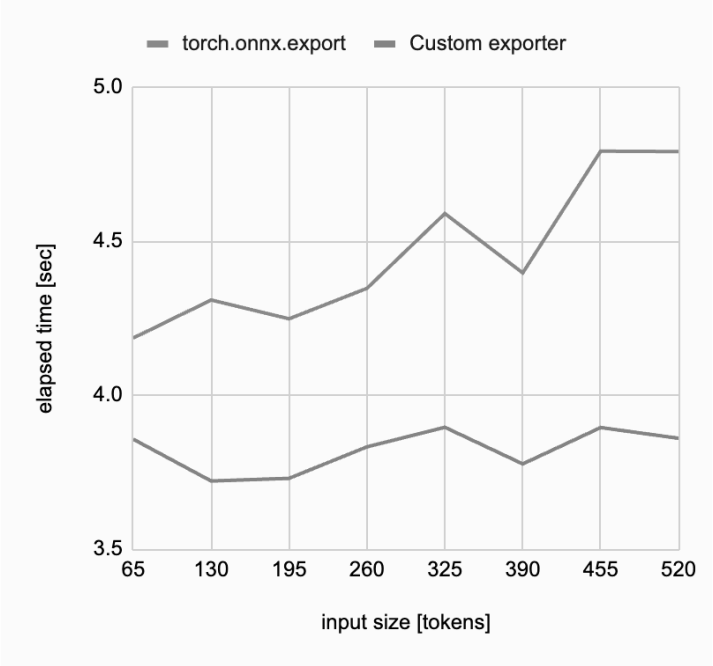

Figure 3. Sample input size dependence of the time taken to output the model. (Blue) For the existing ONNX exporter torch.onnx.export. (red) For the newly developed exporter. This profile was obtained for the transformers implementation of the Hugging face of T5, a language model developed by Google. The sample input represents the example input to be passed to the exporter and the input size represents the number of tokens of tokenized text

Figure 3 plots the time it takes to export a model versus the sample input size, using the language model T5. In the existing ONNX exporter, torch.onnx.export, the export time increases with the input size because the model needs to be executed to obtain the computed graph. On the other hand, the newly developed exporter does not directly execute the model, so it can output the model in a time independent of the input size, and the time required for export is about 10% faster than the existing implementation.

3. Proportion of time required for each method in the export

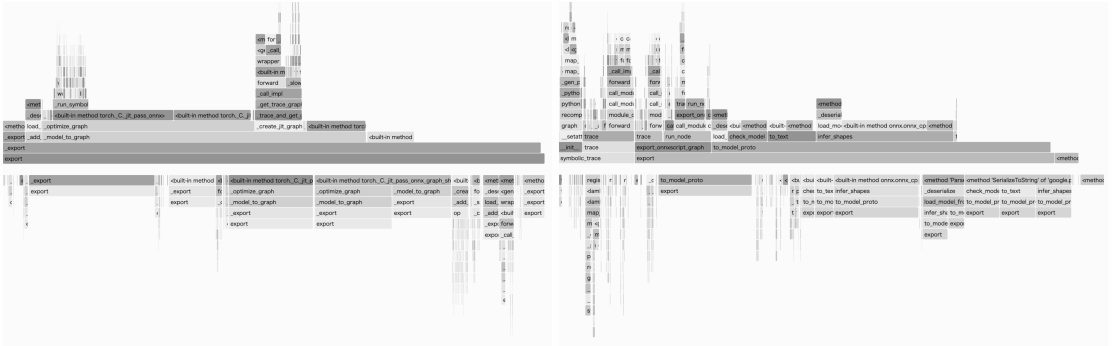

Figure 4. Percentage of time each method takes to export. (Left) Existing ONNX exporter, torch.onnx.export. (Right)The case of a newly developed exporter.

Figure 4 shows the frame graphs obtained in the ONNX exports, again using T5. For both the existing exporter, torch.onnx.export, and the newly developed exporter, the time required to retrieve the

computed graph is only about 20% of the total time. This is due to the fact that not only the acquisition of the computed graph, but also the tensor shape inference, which is proportional to the number of nodes in the model, is also accelerated (however, this may change for larger models). Therefore, we can conclude that to speed up the entire export, not only the acquisition of the computed graph, but also these processes need to be sped up.

Acknowledgement

I would like to express my sincere gratitude to everyone at Preferred Networks, Inc. for their support during this internship, and especially to the DL Ecosystem team members who took time out of their busy schedules to kindly discuss and consult with me.
Finally, I would like to thank my supervisor, Professor Masao Ogata, my secondary supervisor, Professor Yukitoshi Motome, and everyone at MERIT-WINGS for their kind cooperation and support in my participation in this internship.