Collage: Automated Integration of Deep Learning Backends

Byungsoo Jeon*¹, Sunghyun Park*², Peiyuan Liao¹,⁴, Sheng Xu³, Tianqi Chen¹,², Zhihao Jia¹

¹Carnegie Mellon University, ²OctoML, ³Amazon Web Services, ⁴Praxis Pioneering
Deep Learning (DL) Backend

Backend
- a software library or a runtime framework that takes DL workloads as inputs and generates an optimized low-level target code

Frontend
- XLA
- GLOW
- TVM
- TC
- nGraph

Backend
- MKL
- DNNL
- OpenVINO
- cuBLAS
- cuDNN
- TensorRT

Hardware
- Intel
- AMD
- Apple
- Arm
- Xilinx
- Qualcomm
- NVIDIA
Deep Learning (DL) Backend

Backend
- a software library or a runtime framework that takes DL workloads as inputs and generates an optimized low-level target code

Frontend

| DL Compiler | XLA | GLOW | TVM | TC | nGraph |
|-------------|-----|------|-----|----|--------|
|             | MKL | DNNL | OpenVINO | cuBLAS | cuDNN | TensorRT |

Hardware

- Intel
- AMD
- Apple
- ARM
- Xilinx
- Qualcomm
- NVIDIA
Deep Learning (DL) Backend

Backend
- a software library or a runtime framework that takes DL workloads as inputs and generates an optimized low-level target code

Frontend
- XLA
- GLOW
- TVM
- TC
- nGraph

Vendor Lib
- MKL
- DNNL
- OpenVINO
- cuBLAS
- cuDNN
- TensorRT

Hardware
- Intel
- AMD
- Apple
- ARM
- Xilinx
- Qualcomm
- NVIDIA
Observation: Diversified DL Backends

DL backends are highly diversified and evolving fast
- Each backend has its own coverage (e.g., HW, DL operators) and strength
Problem: Backend Integration

Backend Integration = Backend Register + Backend Placement

Computation Graph

Diverse Backends
- cuBLAS
- cuDNN
- TensorRT
- TVM

Optimized Backend Placement

DL System
Problem: Backend Integration

Backend Integration = **Backend Register** + Backend Placement

Computation Graph

Diverse Backends
- cuBLAS
- cuDNN
- TensorRT
- TVM

Backend Register

DL System

Optimized Backend Placement
Problem: Backend Integration

Backend Integration = Backend Register + **Backend Placement**

**Computation Graph**

**Diverse Backends**
- cuBLAS
- cuDNN
- TensorRT
- TVM

**DL System**

**Optimized Backend Placement**

**Backend Placement**
Existing Approach: Manual Backend Integration

- Heuristics are often sub-optimal and susceptible to be outdated
- Direct code modification to the DL framework is required

Rule-based Heuristics

```
If op == "conv2d":
    If cudnn_enabled:
        lower_to_cudnn_kernel
    Else if ...
Else if op == "batch_matmul":
    If cublas_enabled:
        lower_to_cublas_kernel
    ...
```
Our Approach: Automated Backend Integration

- It eliminates manual efforts to design heuristics and change codes
- It provides fast and stable performance across different models and hardwares
System Overview

Backend Pattern Abstraction (Sec 3)
1) Op pattern
   - conv = is_op("nn.conv2d")(*, *)
   - conv + element-wise operator (e.g., ReLU)
   - fused = conv.has_attr("OpPattern": K_ELEM)
   - add_pattern(backend= 'cudnn', pattern= fused)
2) Op pattern rule
   - tvm_pattern_rule is a func that checks if the
     # pattern rule can be applied on the input IR
   - add_pattern_rule(backend= 'tvm',
     rule= tvm_pattern_rule)

Backend Pattern Generators (Sec 3)
- Built-in Pattern Rules
- User-defined Pattern Rules

Op-level Placement Optimizer (Sec 4.2) – Optimize backend placements with DP
- \( C_{\text{OPT}}(G) = \min(M((\text{conv1}, \text{cuDNN}) + M((\text{conv2}, \text{relu}), \text{TRT}) + M((\text{conv3} + \text{add}), \text{TVM}) + M((\text{dense}), \text{TRT})) + M((\text{conv1}), \text{TRT}) + M((\text{conv3} + \text{add}), \text{TVM}) + M((\text{conv}2 + \text{relu}, \text{cuDNN}) + M((\text{dense}), \text{TRT}), ...)\)

Op-level optimized placement

Graph-level Placement Optimizer (Sec 4.3) – Fine-tune placements with evolutionary search

Optimized Backend Placement
Overview

**Computation Graph (G)**

- Conv1 → Conv2 → ReLU → Conv3 → Add → Dense

**Backend Pattern Abstraction (Sec 3)**

1. **Op pattern**
   - `conv = is_op("nn.conv2d")(*, *)`
   - `# conv + element-wise operator (e.g., ReLU)`
   - `fused = conv.has_attr("OpPattern": K_ELEM)`
   - `add_pattern(backend= 'cudnn', pattern= fused)`

2. **Op pattern rule**
   - `
     # tvm_pattern_rule is a func that checks if the
     # pattern rule can be applied on the input IR
     add_pattern_rule(backend= 'tvm',
     rule= tvm_pattern_rule)

**Backend Pattern Generators (Sec 3)**

- Built-in Pattern Rules
- User-defined Pattern Rules

**Op-level Placement Optimizer (Sec 4.2)** – Optimize backend placements with DP

- $C_{OPT}(G) = \min(M(\text{conv1}, \text{cuDNN}) + M(\text{conv2, relu}, \text{TRT}) + M(\text{conv3 + add}, \text{TVM}) + M(\text{dense, TVM}),$
  
  $M(\text{conv1}, \text{TRT}) + M(\text{conv3 + add}, \text{TVM}) + M(\text{conv2 + relu}, \text{cuDNN}) + M(\text{dense, TRT}), \ldots)$

**Op-level optimized placement**

**Graph-level Placement Optimizer (Sec 4.3)** – Fine-tune placements with evolutionary search

- Seed
- Pick best
- Op-level optimized placement

**Optimized Backend Placement**

---

TVM Conf 2021
Overview

~ 70 LoC to integrate one backend
Overview

Collage

Backend Pattern Abstraction (Sec 3)
1) Op pattern
   conv = is_op("nn.conv2d\(\)\(\)\(*\,*\)\)
   # conv + element-wise operator (e.g., ReLU)
   fused = conv.has_attr("OpPattern": K_ELEM)
   add_pattern(backend='cudnn', pattern=fused)

2) Op pattern rule
   # tvm_pattern_rule is a func that checks if the
   # pattern rule can be applied on the input IR
   add_pattern_rule(backend='tvm', rule=tvm_pattern_rule)

Backend Pattern Registry (Sec 3)
- cuDNN
- TVM
- TensorRT

Op-level Placement Optimizer (Sec 4.2)
- Optimize backend placements with DP
  \[ C_{OPT}(G) = \min(M((\text{conv1}, \text{cuDNN})) + M((\text{conv2}, \text{relu}), \text{TRT})) + M((\text{conv3} + \text{add}), \text{TVM}) + M((\text{dense}, \text{TRT}), \text{TRT}), \ldots) \]

Op-level optimized placement

Graph-level Placement Optimizer (Sec 4.3)
- Fine-tune placements with evolutionary search

Graph Cost
- Measurer (M)
- Single Op
- Op Cost

Seed
Pick best

Optimized Backend Placement
Overview

**Built-in patterns support most of popular backends** (e.g., cuDNN, cuBLAS, TensorRT, TVM, MKL, etc.)
Overview

Automated Two-level Optimizer

Op-level Placement Optimizer (Sec 4.2) – Optimize backend placements with DP

\[ C_{OPT}(G) = \min(M((\text{conv}1, \text{cuDNN})) + M(\text{conv}2, \text{ReLU}, \text{TRT}) + M((\text{conv}3 + \text{add}), \text{TVM}) + M((\text{dense}), \text{TVM}), \text{TRT}) + M((\text{conv}3 + \text{add}), \text{TVM}) + M((\text{conv}2 + \text{relu}), \text{cuDNN}) + M((\text{dense}), \text{TRT}), \ldots) \]

Graph-level Placement Optimizer (Sec 4.3) – Fine-tune placements with evolutionary search

Optimized Backend Placement
**Overview**

**Computation Graph (G)**
- Conv2 → ReLU
- Conv3 → Add → Dense

**Backend Pattern Abstraction (Sec 3)**
1) Op pattern
   - `conv = is_op("nn.conv2d")(*, *)`
   - `fused = conv.has_attr("OpPattern": K_ELEM)`
   - `add_pattern(backend= 'cudnn', pattern= fused)`
2) Op pattern rule
   - `tvm_pattern_rule` is a func that checks if the
     # pattern rule can be applied on the input IR
     `add_pattern_rule(backend= 'tvm', rule= tvm_pattern_rule)

**Backend Pattern Generators (Sec 3)**
- Built-in Pattern Rules
- User-defined Pattern Rules

**Op-level Placement Optimizer (Sec 4.2)** – Optimize backend placements with DP
- `C_{OPT}(G) = min(M((conv1,cuD)) + M((conv2,relu), TRT) + M((conv3 + add), TVM) + M((dense), TVM), M((conv1,TRT)) + M((conv3 + add), TVM) + M((conv2 + relu), cuD) + M((dense), TRT), ...)`

**Op-level optimized placement**
- Graph Cost
- Measurer (M)
- Single Op
- Op Cost

**Graph-level Placement Optimizer (Sec 4.3)** – Fine-tune placements with evolutionary search
- Seed
- Pick best
- Op-level optimized placement

**Optimized Backend Placement**
End-to-end Evaluation: NVIDIA V100, Intel Xeon

- Stable performance across different networks and hardwares

1.39x Speedup

1.40x Speedup
Optimized Backend Placement

- **Collage** leverages unique strength of each backend
- Collage maps same type of operators to different backends based on the performance landscape
- **Collage** employs diverse operator fusion patterns
References

Arxiv Paper: https://arxiv.org/abs/2111.00655
Code: https://github.com/cmu-catalyst/collage