Graph-Level Optimization

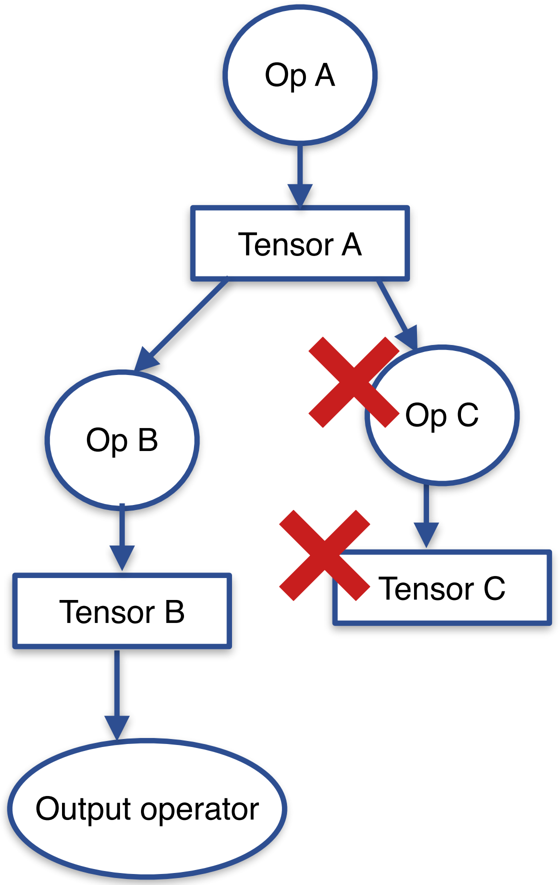
Yang-Ge Ma, May 2020

Outline

- Graph-Level Optimizations
  - Target Dependent
  - Target Independent
- Lab
  - Add Optimization Pass
    - Example: EliminateNopTranspose
  - Add Visualization Pass
  - Demo

Graph-Level Optimizations

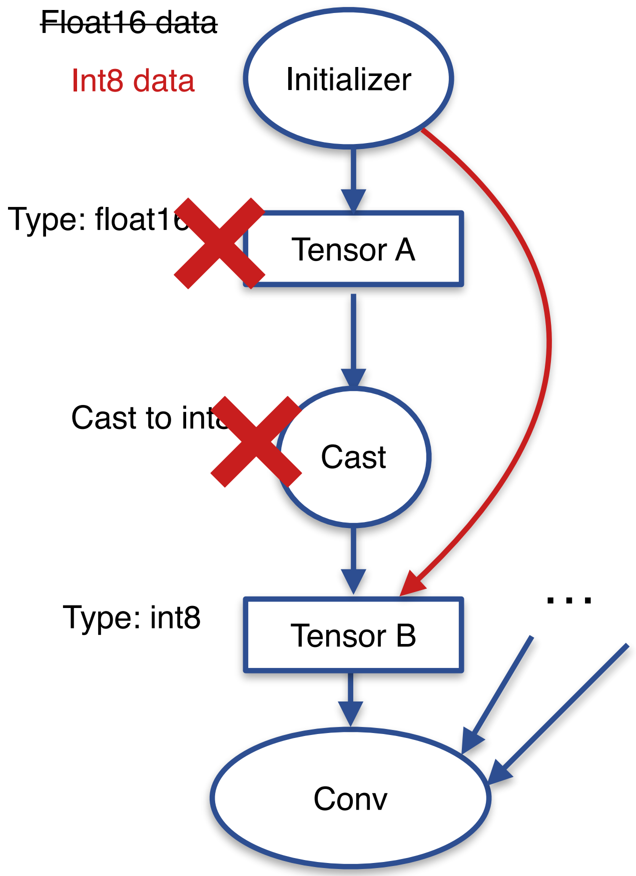
- Manipulation on ONNC IR graphs
- Simplifying IR graphs
  - Node removal
    - `DeadNodeElimination`: Remove nodes whose outputs are not used at all.
  - Layer fusion
    - `EliminateCast`: Fuse Cast operators to preceding initializers.
  - Operator refactoring
    - `ExpandBatchNormalization`: Convert BN to an Add and a Mul.



skymizer

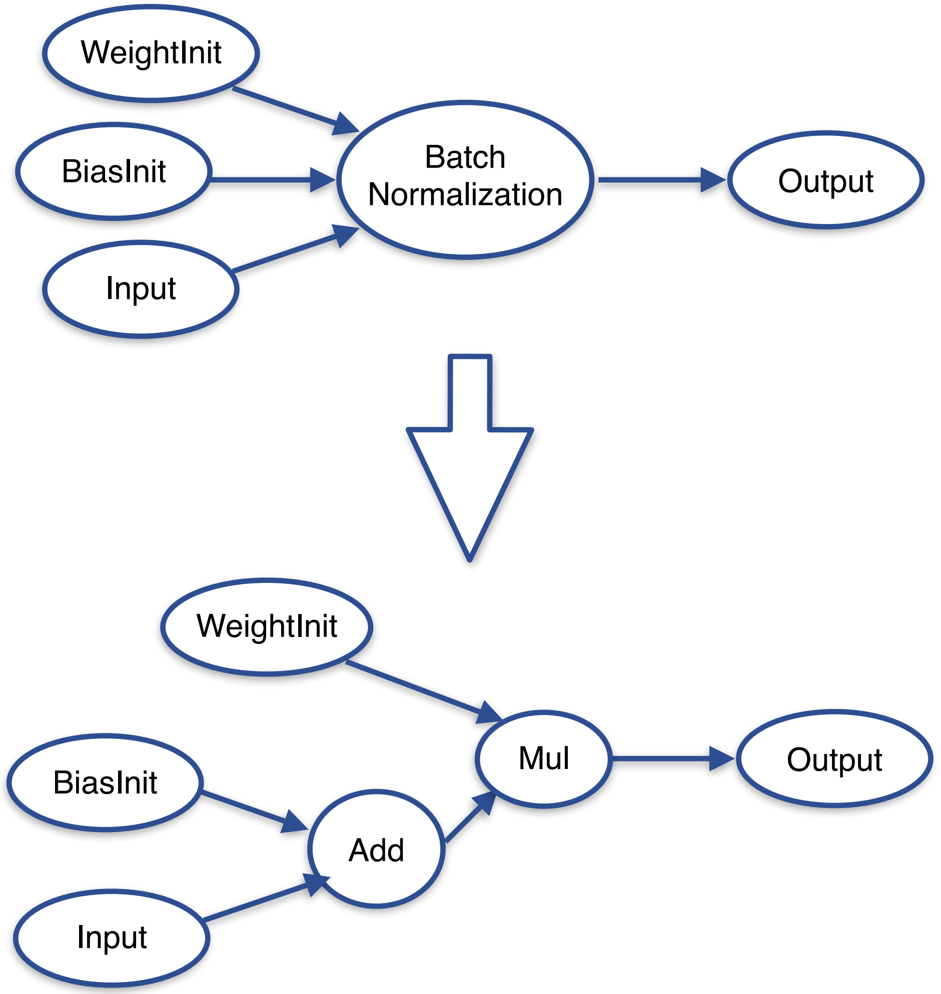
Graph-Level Optimizations

- Manipulation on ONNC IR graphs
- Simplifying IR graphs
  - Node removal
    - `DeadNodeElimination`: Remove nodes whose outputs are not used at all.
  - Layer fusion
    - `EliminateCast`: Fuse Cast operators to preceding initializers.
  - Operator refactoring
    - `ExpandBatchNormalization`: Convert BN to an Add and a Mul.



Graph-Level Optimizations

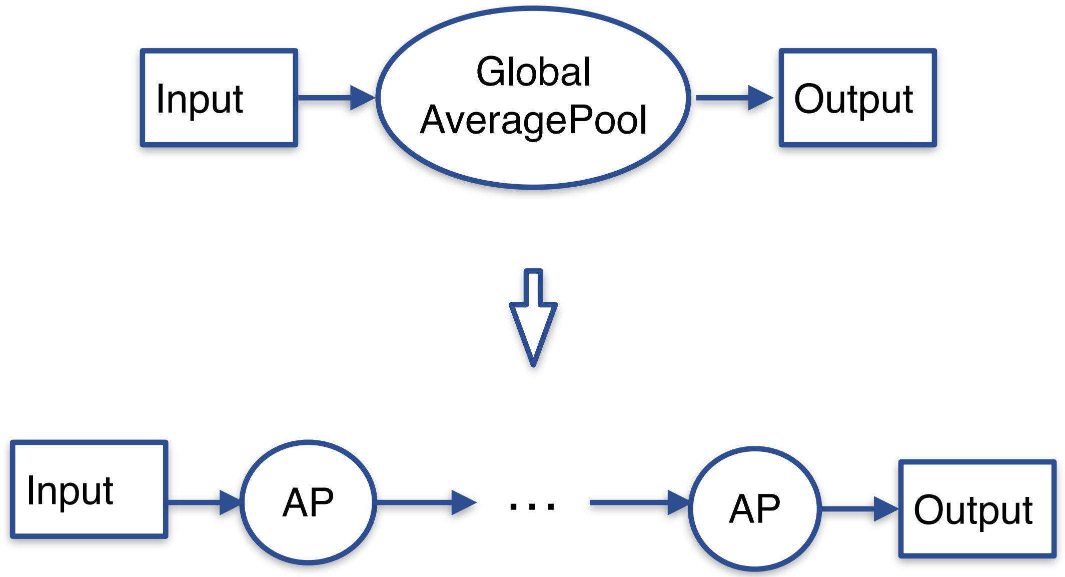
- Manipulation on ONNC IR graphs
- Simplifying IR graphs
  - Node removal
    - `DeadNodeElimination`: Remove nodes whose outputs are not used at all.
  - Layer fusion
    - `EliminateCast`: Fuse Cast operators to preceding initializers.
  - Operator refactoring
    - `ExpandBatchNormalization`: Convert BN to an Add and a Mul.



skymizer

Graph-Level Optimizations

- Legalization: Convert unsupported operators into supported operators
  - `DivideGlobalAPsIntoAPs`: NVDLA does not support GlobalAveragePool
- Meet hardware constraints
  - `SplitConvPass`: NVDLA's convolution buffer size is limited



skymizer

Graph-Level Optimizations

- Legalization: Convert unsupported operators into supported operators
  - `DivideGlobalAPsIntoAPs`: NVDLA does not support GlobalAveragePool
- Meet hardware constraints
  - `SplitConvPass`: NVDLA's convolution buffer size is limited

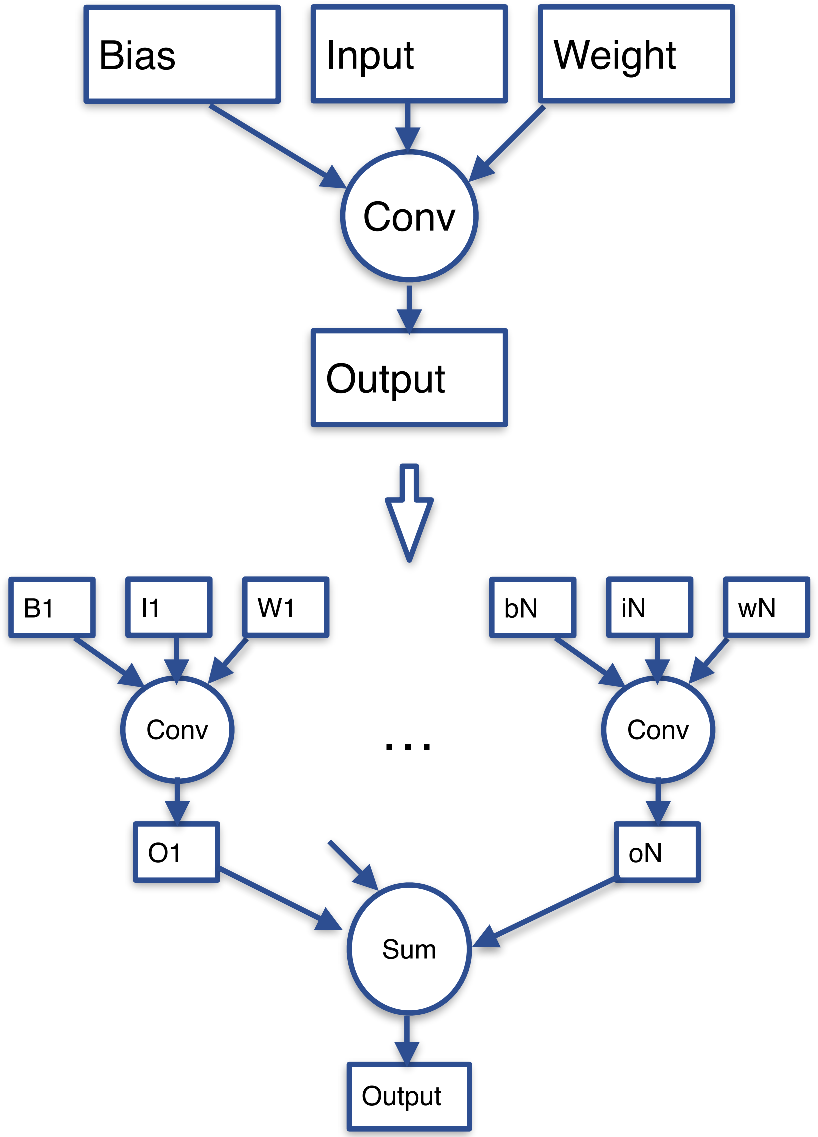


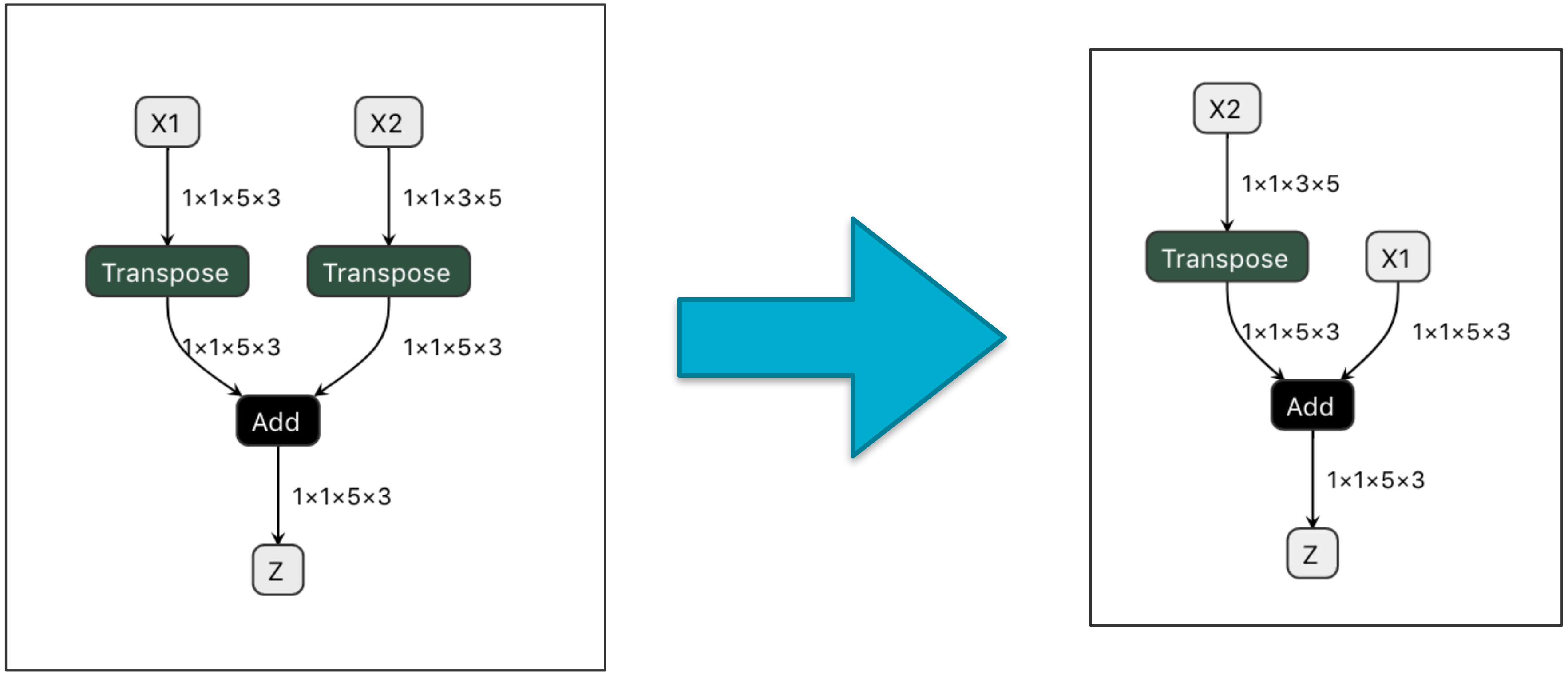
Table of All Currently Supported Optimizations

	Category	Functionality
`EliminateIdentity`	Node removal	Remove all Identity operators
`DeadNodeElimination`	Node removal	Remove nodes whose output are totally unused
`RemoveTrainingNodes`	Node removal	Remove all Dropout nodes yielded during training
`EliminateCast`	Layer fusion	Fuse Cast operators to preceding initializers
`PropagateConstWithDiffShape`	Layer fusion	Fuse general reshaping operators to preceding initializers
`ExpandBatchNormalization`	Operator refactoring	Replace any BatchNormalization operator with an Add and a Mul
`ExtractConstToInitializer`	Operator refactoring	Make outputs of Const operators initializers
`ReplaceGemmByConv`	Legalization	Replace Gemm operators, with a Conv and some other operators.
`DivideGlobalAPIntoAPs`	Legalization	Divide and conquer: break GlobalAPs into AveragePools of smaller windows recursively.
`SplitConvPass`	Meeting HW constraints	Split input into smaller pieces, do Conv for each of them and accumulate the results.

skymizer

Lab: Add an Optimization Pass

- EliminateNopTranspose: We demonstrate with an optimization that remove Transpose operator whose permutation is identity.



Workflow Overview

- Steps
  - Implementing a pass
    - Derive `CustomPass` class template
  - Registering the pass to the target backend implementation
    - Add it to the `PassManager`
  - Adding the files of the new pass to the building system

Derive Custom Pass

```
#include <onnc/Core/CustomPass.h>

namespace onnc {
class EliminateNopTranspose: public CustomPass<EliminateNopTranspose>
{
public:
  EliminateNopTranspose() = default;

  // Entry point of the pass
  ReturnType runOnModule(Module& pModule) override;
};
} // namespace of onnc
```

Implementation of runOnModule()

```
Pass::ReturnType EliminateNopTranspose::runOnModule(Module& pModule) {
  using namespace internal;
  Pass::ReturnType ret = Pass::kModuleNoChanged;

  // Iterate through all ComputeGraphs
  for (auto it = pModule.cgBegin(); it != pModule.cgEnd(); ++it) {
    ComputeGraph& cg = *it->value();

    /* Procedure for each compute graph
     * ...
     */
  }

  return ret;
}
```

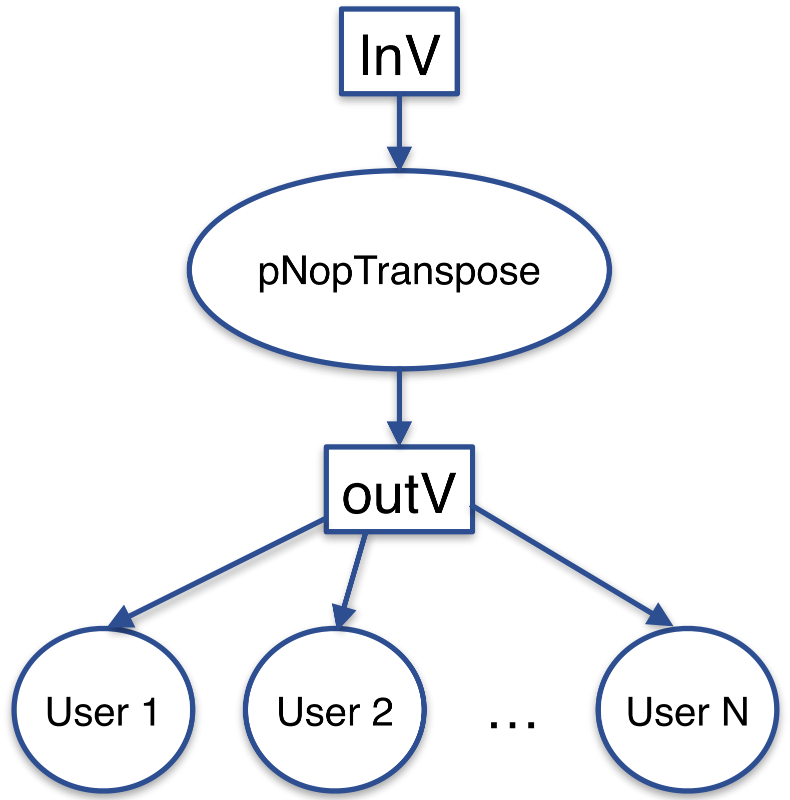
Implementation of runOnModule()

```
// In each iteration

std::vector<ComputeOperator*> removeList;

for (ComputeOperator& node : cg) {
  ret = Pass::kModuleChanged;
  if(isNopTranspose(&node))
   removeList.emplace_back(&node);
}

for (auto* pNopTranspose : removeList) {
  Value* inV  = pNopTranspose->getInput(0);
  Value* outV = pNopTranspose->getOutput(0);
  outV->replaceAllUsesWith(*inV);
  pNopTranspose->removeAllInputs();
  cg.erase(*pNopTranspose);
  cg.erase(*outV);
}
```



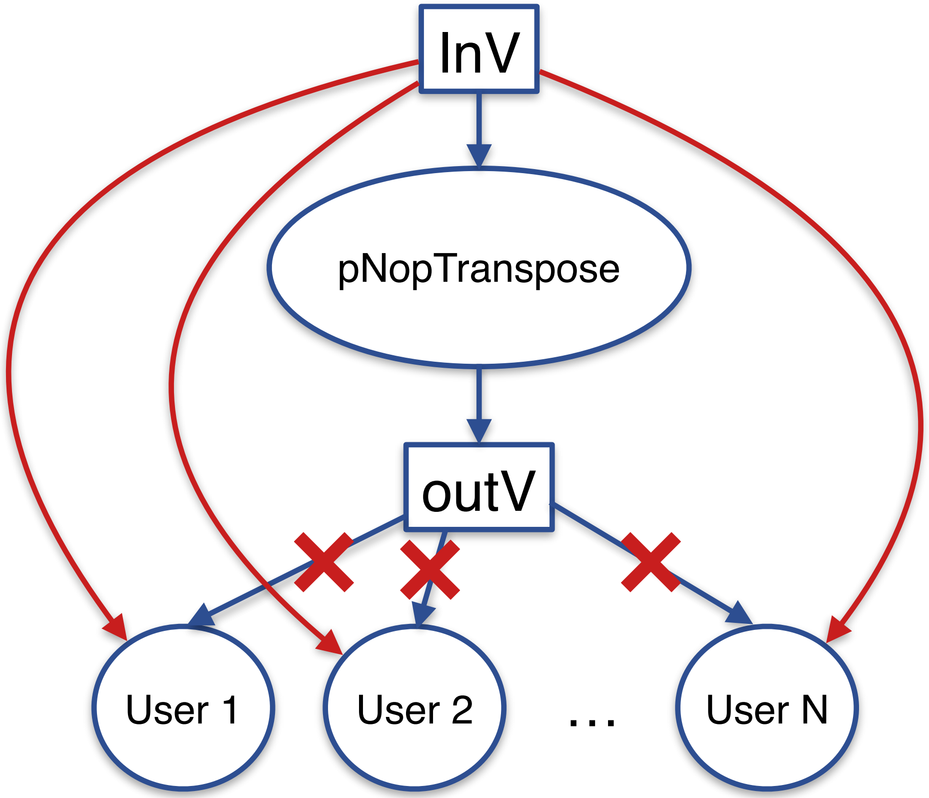
Implementation of runOnModule()

```
// In each iteration

std::vector<ComputeOperator*> removeList;

for (ComputeOperator& node : cg) {
  ret = Pass::kModuleChanged;
  if(isNopTranspose(&node))
   removeList.emplace_back(&node);
}

for (auto* pNopTranspose : removeList) {
  Value* inV  = pNopTranspose->getInput(0);
  Value* outV = pNopTranspose->getOutput(0);
  outV->replaceAllUsesWith(*inV);
  pNopTranspose->removeAllInputs();
  cg.erase(*pNopTranspose);
  cg.erase(*outV);
}
```



skymizer

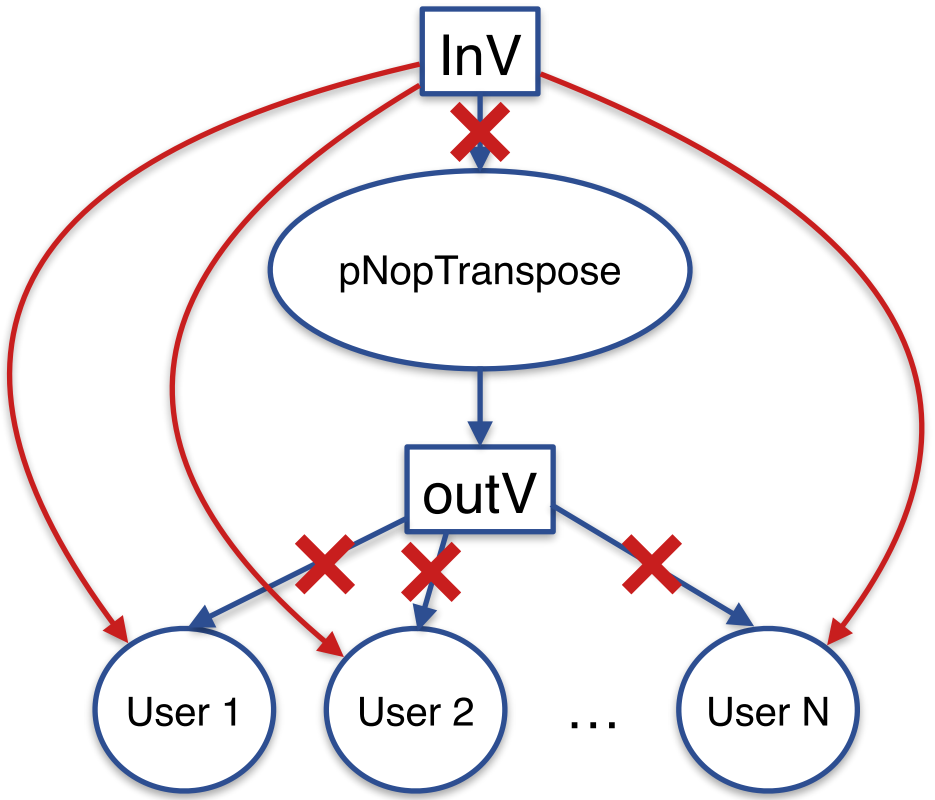
Implementation of runOnModule()

```
// In each iteration

std::vector<ComputeOperator*> removeList;

for (ComputeOperator& node : cg) {
  ret = Pass::kModuleChanged;
  if(isNopTranspose(&node))
   removeList.emplace_back(&node);
}

for (auto* pNopTranspose : removeList) {
  Value* inV  = pNopTranspose->getInput(0);
  Value* outV = pNopTranspose->getOutput(0);
  outV->replaceAllUsesWith(*inV);
  pNopTranspose->removeAllInputs();
  cg.erase(*pNopTranspose);
  cg.erase(*outV);
}
```



skymizer

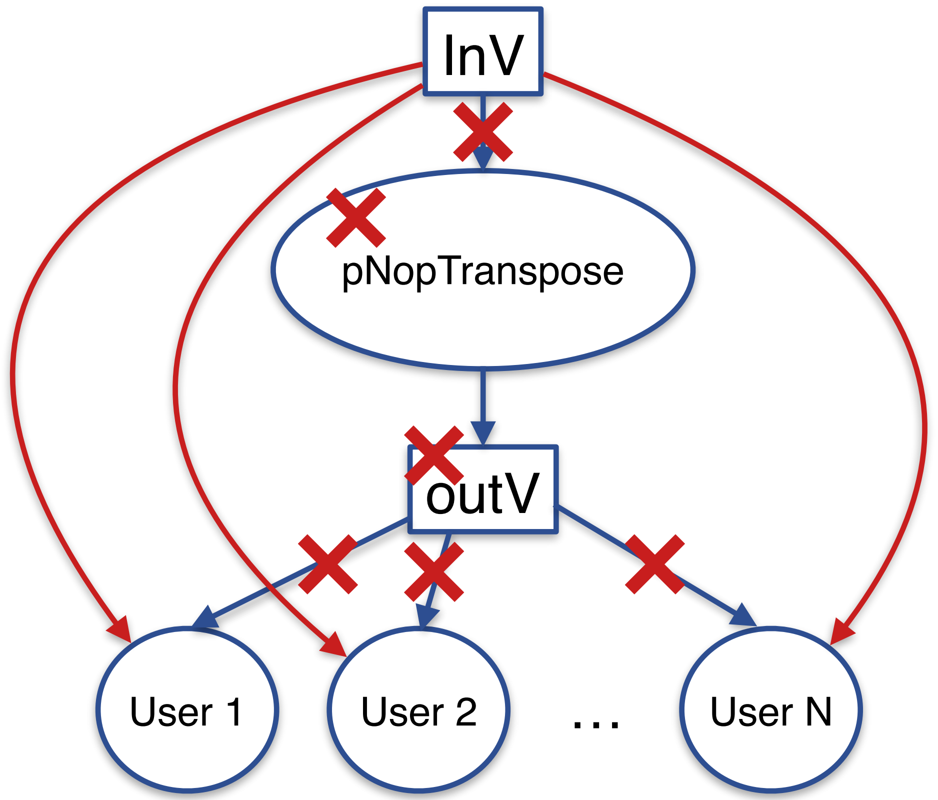
Implementation of runOnModule()

```
// In each iteration

std::vector<ComputeOperator*> removeList;

for (ComputeOperator& node : cg) {
  ret = Pass::kModuleChanged;
  if(isNopTranspose(&node))
   removeList.emplace_back(&node);
}

for (auto* pNopTranspose : removeList) {
  Value* inV  = pNopTranspose->getInput(0);
  Value* outV = pNopTranspose->getOutput(0);
  outV->replaceAllUsesWith(*inV);
  pNopTranspose->removeAllInputs();
  cg.erase(*pNopTranspose);
  cg.erase(*outV);
}
```



Register New Optimization Pass

- In your backend implementation (say NvDlaBackend.cpp):

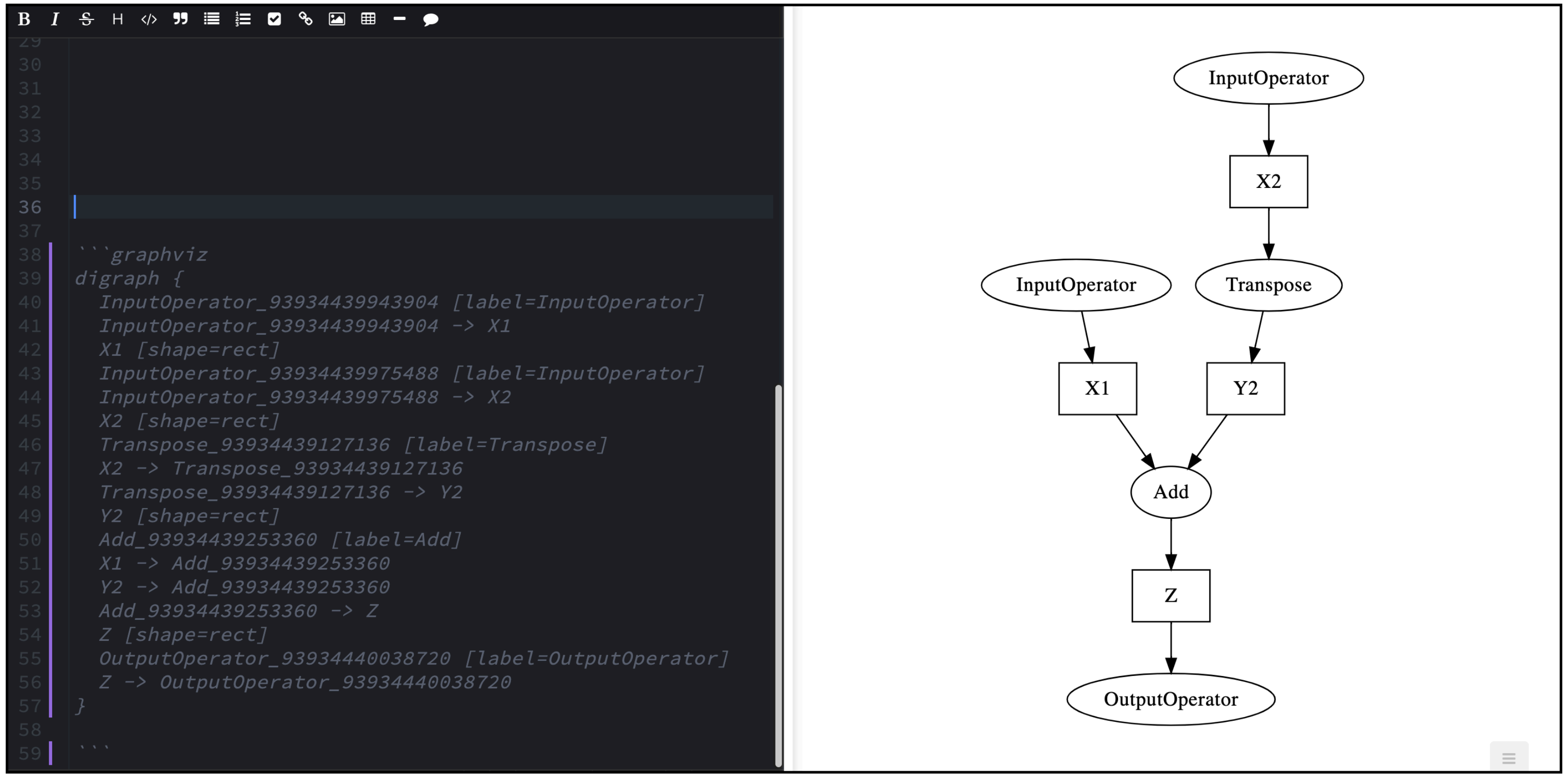
```
+ #include “EliminateNopTranspose.h”
  #include <onnc/Transforms/Optimizations/EliminateIdentity.h>
  #include <onnc/Transforms/Optimizations/PropagateConstWithDiffShape.h>
  NvDlaBackend::addOnncIrOptimization
    (PassManager& passManager, OptimizationOptions& options) {
    // ..
+   passManager.add<EliminateNopTranspose>();
    // ..
  }
```

- In CMakeList.txt where you put your Pass implementation:

```
 add_libonnc_src(
     GraphvizONNCIRPass.cpp
+    EliminateNopTranspose.cpp
 )
```

Lab: Add Visualization Pass

- Graphviz: A graph visualization language supported in markdown
- Implement a pass that dumps the graph in Graphviz format to see the change after optimization.



~~~
```graphviz
digraph {
  InputOperator_93934439943904 [label=InputOperator]
  InputOperator_93934439943904 -> X1
  X1 [shape=rect]
  InputOperator_93934439975488 [label=InputOperator]
  InputOperator_93934439975488 -> X2
  X2 [shape=rect]
  Transpose_93934439127136 [label=Transpose]
  X2 -> Transpose_93934439127136
  Transpose_93934439127136 -> Y2
  Y2 [shape=rect]
  Add_93934439253360 [label=Add]
  X1 -> Add_93934439253360
  Y2 -> Add_93934439253360
  Add_93934439253360 -> Z
  Z [shape=rect]
  OutputOperator_93934440038720 [label=OutputOperator]
  Z -> OutputOperator_93934440038720
}

```
~~~

skymizer

Boost deep learning accelerator
with compiler technology

Skymizer Taiwan Inc.

CONTACT US

E-mail sales@skymizer.com **Tel** +886 2 8797 8337

HQ 12F-2, No.408, Ruiguang Rd., Neihu Dist., Taipei City 11492, Taiwan
BR Center of Innovative Incubator, National Tsing Hua University, Hsinchu Taiwan

https://skymizer.com