Summer 2019 Internship Report

Gus Smith

November 18, 2019

Disclaimer: I have no formal background in program analysis!

Summer goal: Analyze deep learning workloads for "accelerability"

Realization: Most common workloads expressed in Relay are simply straight-line code. These can be analyzed statically.

My solution:

1. Build simple analysis framework in Relay

2. Write analyses in the framework that gather useful information

3. Meet with architects, iterate on analyses

(my definitions of) Static and Dynamic Analysis

- In my terminology, **static analysis ≈ dataflow analysis,** because these are straight-line programs
- **Dynamic analysis:** instrumenting code
- What other types of analysis are there out there?

Static Analysis Framework

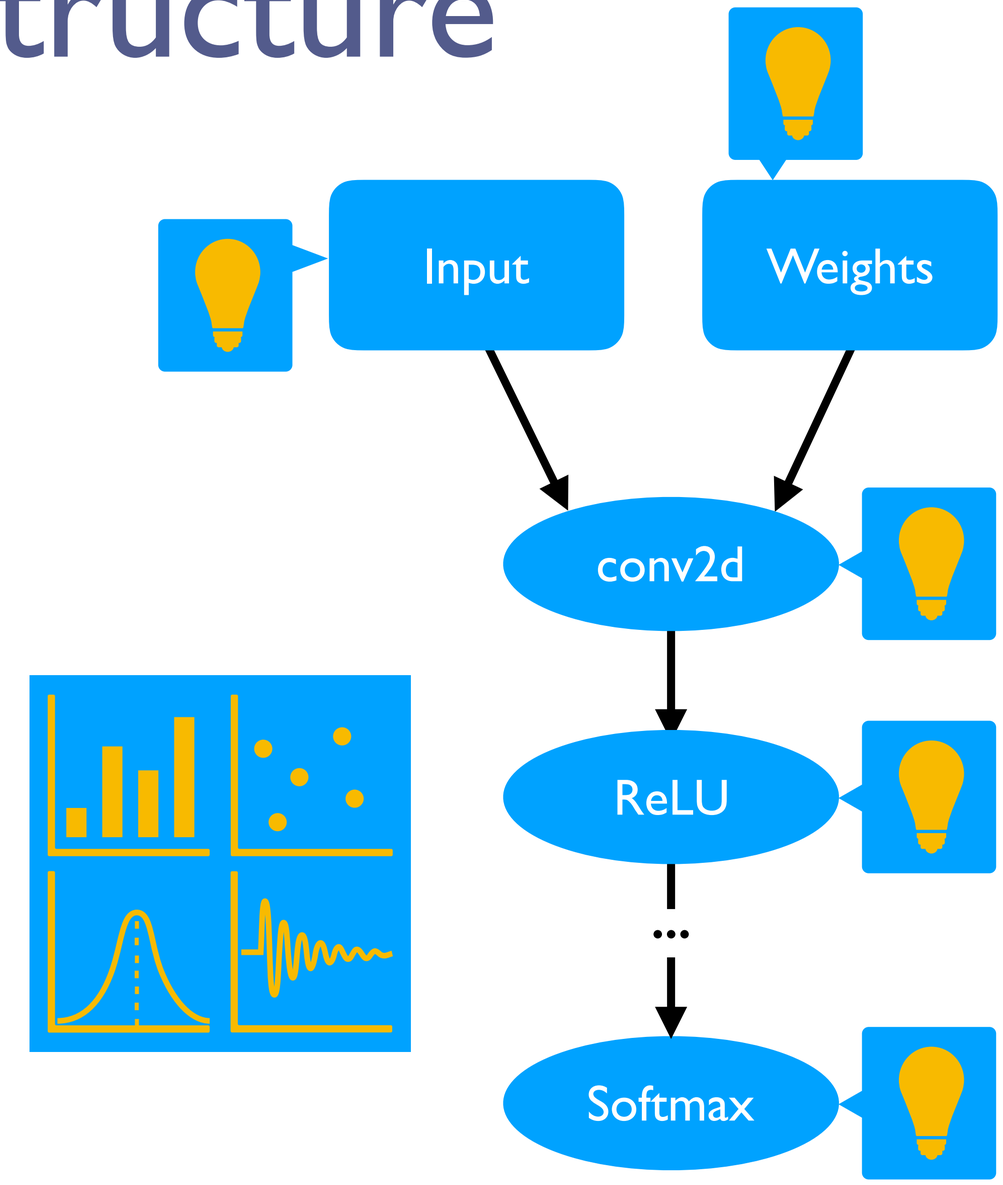
The analysis framework helps the user **design passes** which extract static information about a program, and helps the user **organize and export** that information.

Analysis Structure

The analysis framework imposes a general structure for analyses

- Two parts: **initial analysis phase** (visiting every program node and extracting data) and **summary phase** (visiting the extracted data and producing summary results)

- Essentially **map** and **reduce** phases



Background: ExprVisitor

The analysis framework is built off of Relay's ExprVisitor.

```
1  import tvm
2  from tvm import relay
3
4
5  class MyPass(relay.ExprVisitor):
6      def visit_call(self, call):
7          super().visit_call(call)
8          print(call.op)
9
10
11 MyPass().visit(relay.const(1) - (relay.var('x') * relay.var('y')))
```

```
/Users/gus/.pyenv/versions/3.7.4/bin/python tmp.py
v0.0.4
multiply
v0.0.4
subtract
```

AnalysisPass Class

- A thin wrapper over the ExprVisitor class
- Helper functions like _add_detail make it easy to attach analysis data to a node
- Passes can depend on data generated by previous passes, but dependencies are implicit right now

Demo

https://github.com/gussmith23/tvm/blob/analysis-framework-demo/demo.ipynb

Dynamic Analysis Experiments

We can instrument the program with counters in two ways:

- Using references—easy to implement, but not good Relay style
- "Functionally", where every value becomes a tuple of (value, counter)—better Relay style, but harder to implement

Note: Relay programs may not be dynamic enough to warrant dynamic analysis

Improvements/Future Directions

- Separate analysis description from analysis results
- Explicit dependencies between passes
- Make the framework more useful for gathering actual compiler passes
- Build dynamic analysis tools
  - Add passes to Relay which rewrite programs, passing around counters and other data-gathering things

Links

https://github.com/microsoft/Analysis-Framework-for-TVM

https://github.com/gussmith23/tvm/blob/analysis-framework-demo/demo.ipynb