Accelerating NNEF Framework on OpenCL Devices Using clDNN

Meng-Shiun Yu, Tai-Liang Chen, and Jenq-Kuen Lee

Department of Computer Science,

National Tsing Hua University, Hsinchu, Taiwan

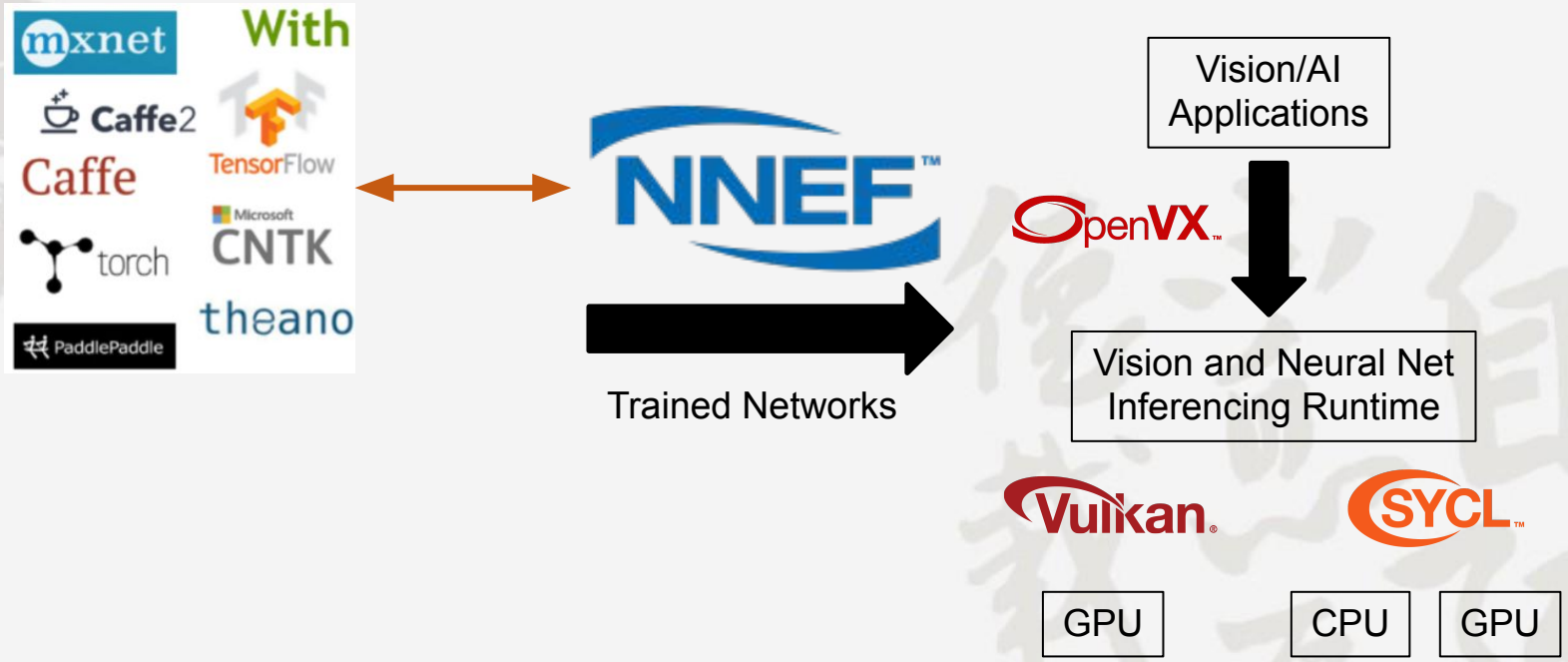
{msyu, tlchen}@pllab.cs.nthu.edu.tw, jklee@cs.nthu.edu.tw

Agenda

- Overview
- Design of Software Stack
- Experiments Results

Background

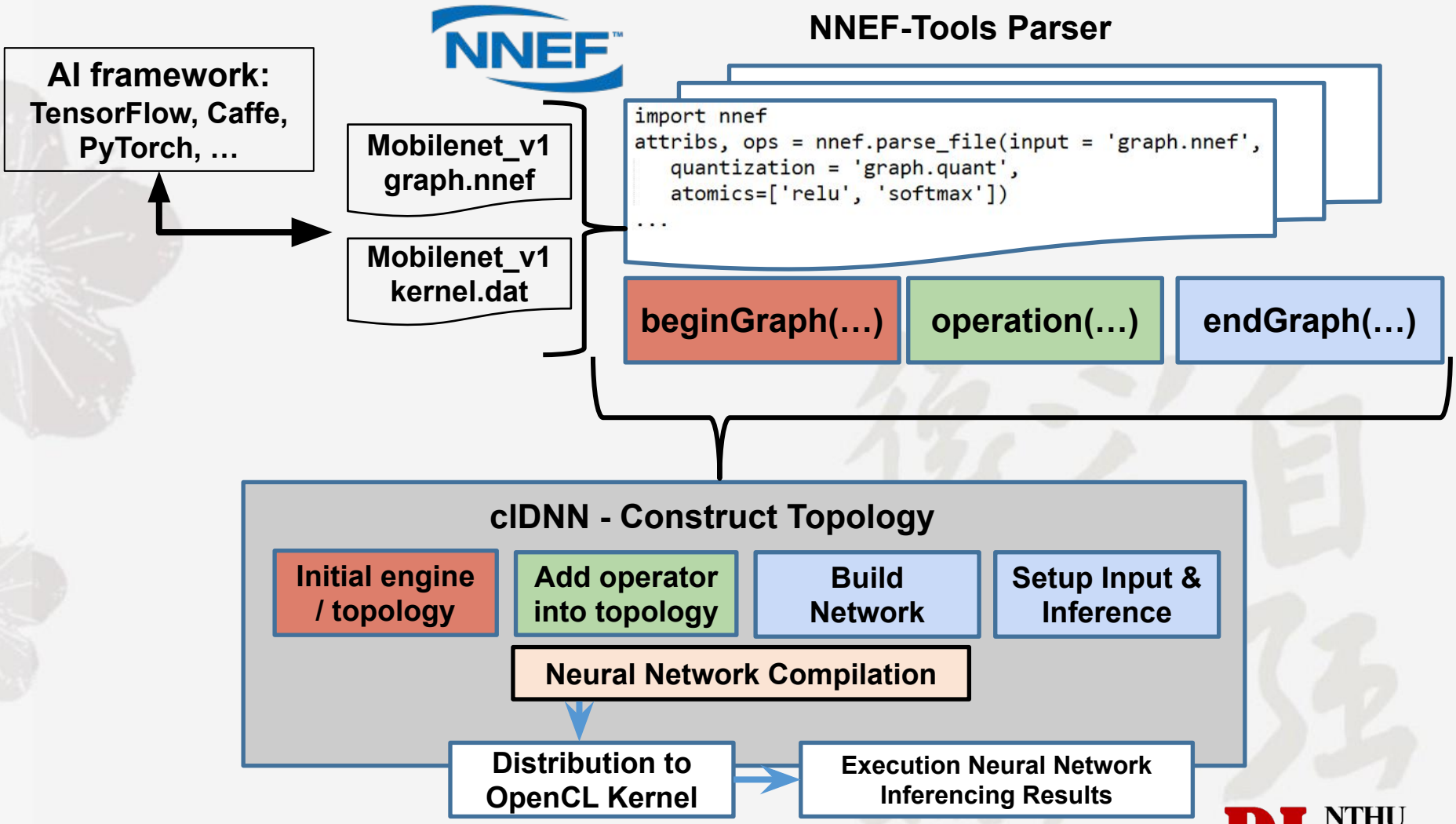
- NNEF - Neural Network Exchange Format
  An intermediate representation of open specification and the well-defined



Overview


Training frameworks
mxnet
Caffe2
TensorFlow
Caffe
Microsoft
CNTK
torch
theano
PaddlePaddle
NNEF Converter
NNEF
NNEF Translator
clDNN
Intel HD Graphics

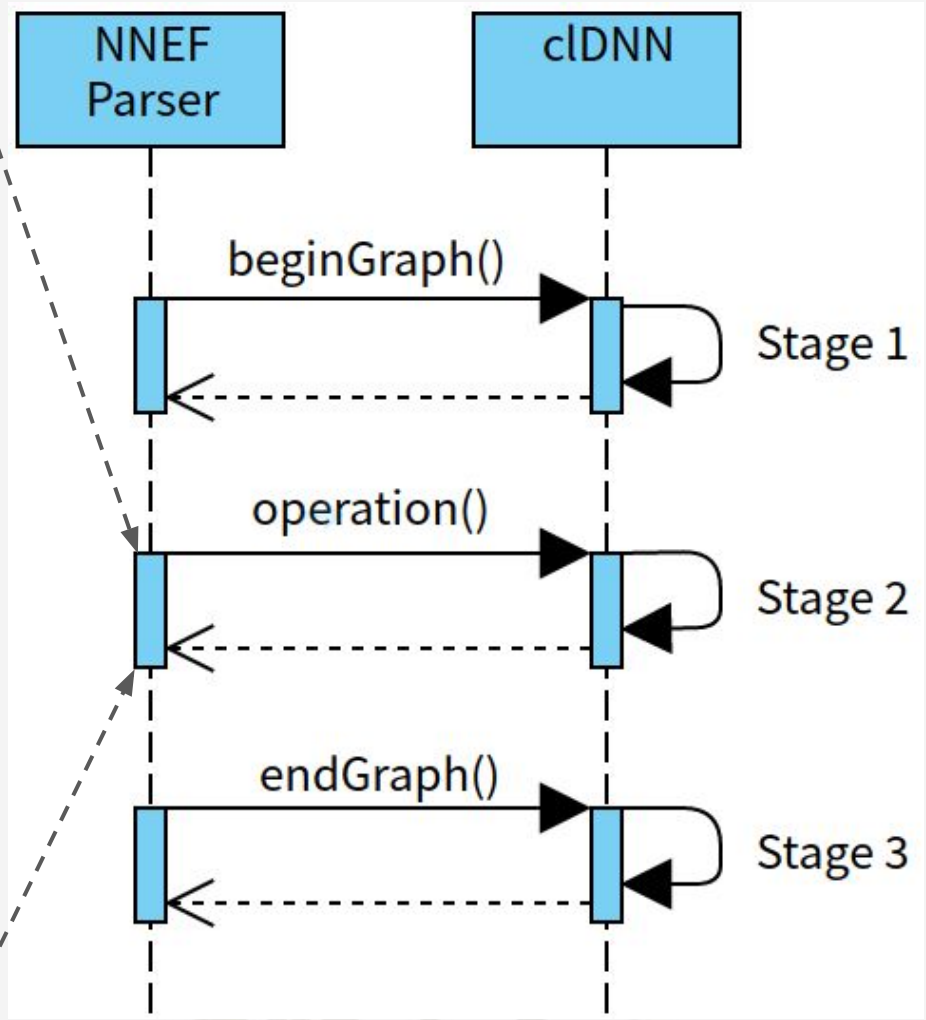
The Flow for NNEF Enabled in clDNN with OpenCL



The Flow for NNEF Enabled in clDNN with OpenCL

MobileNet Body Architecture

Layer	Type / Stride	Filter Shape	Input Size
1	Conv / s2	3 x 3 x 3 x 32	224 x 224 x 3
2	Conv dw / s1	3 x 3 x 32 dw	112 x 112 x 32
3	Conv / s1	1 x 1 x 32 x 64	112 x 112 x 32
4	Conv dw / s2	3 x 3 x 64 dw	112 x 112 x 64
5	Conv / s1	1 x 1 x 64 x 128	56 x 56 x 64
6	Conv dw / s1	3 x 3 x 128 dw	56 x 56 x 128
7	Conv / s1	1 x 1 x 128 x 128	56 x 56 x 128
8	Conv dw / s2	3 x 3 x 128 dw	56 x 56 x 128
9	Conv / s1	1 x 1 x 128 x 256	28 x 28 x 128
10	Conv dw / s1	3 x 3 x 256 dw	28 x 28 x 256
11	Conv / s1	1 x 1 x 256 x 256	28 x 28 x 256
12	Conv dw / s2	3 x 3 x 256 dw	28 x 28 x 256
13	Conv / s1	1 x 1 x 256 x 512	14 x 14 x 256
14 ~ 23	Conv dw / s1	3 x 3 x 512 dw	14 x 14 x 512
	Conv / s1	1 x 1 x 512 x 512	14 x 14 x 512
24	Conv dw / s2	3 x 3 x 512 dw	14 x 14 x 512
25	Conv / s1	1 x 1 x 512 x 1024	7 x 7 x 512
26	Conv dw / s2	3 x 3 x 1024 dw	7 x 7 x 1024
27	Conv / s1	1 x 1 x 512 x 1024	7 x 7 x 1024
28	Avg Pool / s1	Pool 7 x 7	7 x 7 x 1024
29	FC / s1	1024 x 1000	1 x 1 x 1024
30	Softmax / s1	Classifier	1 x 1 x 1000



NNEF Interpreter

```
void cldnn_add_operation(cldnn::engine &engine, cldnn::topology &topology, Operation operation)
{
        auto id = operation.outputs.get(0).identifier();
        static map<string, Operation> op_dict;
        op_dict[id] = operation;
        /* input node */
        if ("external" == operation.name) {
                add_input_node(engine, topology, operation);
        } else if ("variable" == operation.name) {
                add_data_node(engine, topology, operation);
        } else if ("conv" == operation.name) {
                add_op_conv(engine, topology, operation, op_dict);
        } else if ("add" == operation.name) {
                add_op_add(engine, topology, operation);
        }
        …
        else {
        std::cout << "unsupported op: " << operation.name << std::endl;
        }
}
```

NNEF Interpreter

```
static void add_op_conv(cldnn::engine &engine, cldnn::topology topology, Operation &operation,
                        map<string, Operation> op_dict, struct op_shape &shape_info)
{
        string output = operation.outputs.get(0).identifier();
        string input = operation.inputs.get(0).identifier();
        string weight = operation.inputs.get(1).identifier();
        auto stride_shape = operation.attribs.get("stride").
        …
        vector<int> dia_v{dia_h, dia_w};
        tensor dia_ts(dia_v);
        vector<int> stride{1,1,stride_h, stride_w};
        tensor stride_ts(stride);
        vector<int> pad_v{0, 0, padding_h, padding_w};
        tensor pad_ts(pad_v);
        ...
        auto conv_op =  convolution(name, input, {weight}, {bias_name}, stride_ts, pad_ts, dia_ts,
false, 1.0, last_pad_ts);

        topology.add(conv_op);
}
```

NNEF Interpreter

```
void cldnn_execute(cldnn::engine& engine, cldnn::topology& topology) {
        vector<float> ftensor;
        load_image(input_img, ftensor);

        network network(engine, topology);
        layout in_layout(data_types::f32, format::bfyx, {1,3,224,224});
        memory input_mem = memory::allocate(engine, in_layout);
        set_values(input_mem, move(ftensor));

        network.set_input_data("input", input_mem);
        auto outputs = network.execute();
        auto output_ptr = outputs.at("output").get_memory().pointer<float>();
        ...
}
```

Experiments Environments

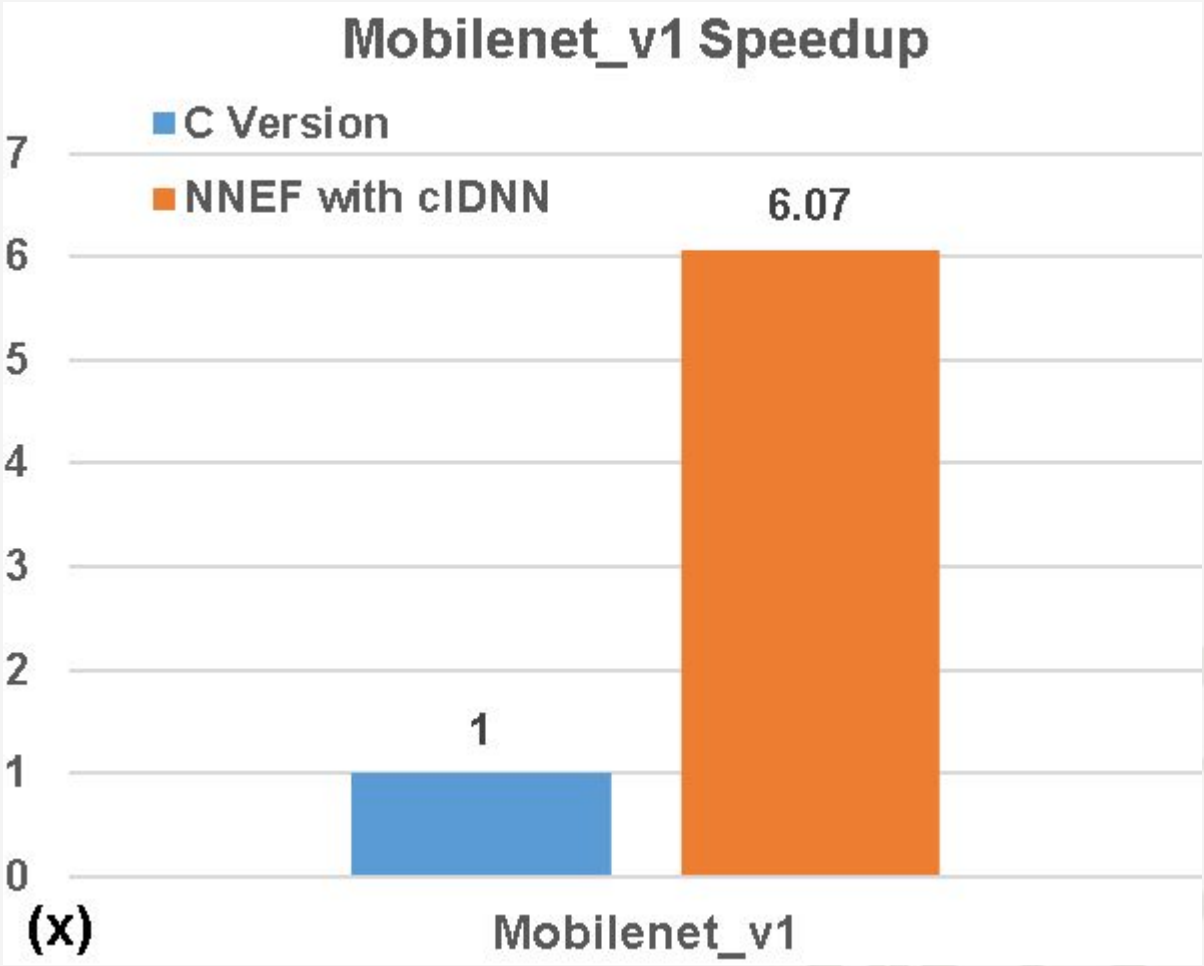
Hardware:

- Intel Core i7-7700 CPU 3.60GHz
- HD Graphics 630 graphics card

Software:

- clDNN 2019 R2
- OpenCL 2.1
- NNEF parser v1.0

Experimental Results



Conclusion

- We proposed a translator that accelerated NNEF on OpenCL devices via clDNN.
- The experimental results shown that we improved the execution efficiency about six times