UNIVERSITY OF CAMBRIDGE

Neural Architecture Search as Program Transformation Exploration

Jack Turner, Elliot J. Crowley, Michael F.P. O'Boyle

Presented by Thomas Yuan

Background

Goal: Improve performance of DNNs

Two main, distinct approaches

- Program Transformation (Compilers)
  - Hardware specific optimizations
- Neural Architecture Search
  - Replace components with computationally cheaper methods

Problems

- Inaccurate choice of program transformations for a powerful architecture
- NAS limited to pre-designed list of convolutional alternatives

Overview

Combine both approaches!

Example:

- Program transformation: Loop interleaving
- NAS technique: bottlenecking

```
for co in [0,CO-1]:
    for ci in [0,CI-1]:
        ...
```

→

```
for co in [0,CO/B-1]:
    for ci in [0,CI-1]:
        ...
```

→

```
for i in [0,CI-1]:
    for co in [0,CO/B-1]:
        ...
```

→

```
for i in [0,CI/B-1]:
    for co in [0,CO/B-1]:
        ...
```

Overview

Concerns:

Legality

- NAS methods change architecture
    - do not guarantee transformation safety
- Need a way to measure new “transformation safety”
    - Fisher Potential

Models and Implementation

Polyhedral Model

- Describes program transformations
- Domain
  - Collection of statement instances
- Set of accesses
  - Mapping of iteration space to memory
- Schedule
  - Assigns timestamps
- Legality of transformation
  - If data dependence -> relative ordering must be preserved

Example

Algorithm 1 Naive implementation of 1×1 tensor convolution.

```
for (co=0; co<Co; co++)
  for (oh=0; oh<OH; oh++)
    for (ow=0; ow<OW; ow++)
S1    O[c_o][h][w] = 0.;
      for (ci=0; ci<Ci; ci++)
S2      O[co][oh][ow] +=
          W[co][1][1] *
          I[ci][oh][ow];
```

We can also describe the schedule as follows:

$$T_{S1}(c_o, h, w) = (c_o, h, w)$$
$$T_{S2}(c_o, h, w, c_i) = (c_o, h, w, c_i)$$

Loop Interchange: $T_{S1}(c_o, h, w) = (c_o, w, h)$

Legality: $\forall i, j, S1, S2, D \quad i \rightarrow j \in d_{S1,S2} \rightarrow T(i) \leq T(j)$

Models and Implementation

Bottlenecking:

- Reduce number of filters from C_O to C_O/B

$$T_S(c_o, J') = (c'_o, J') \mid c'_o < C_o/B$$

Grouping

- Split C_I input channels into G groups
- Each group independently convolved
- C_O/G output channels -> concatenated

$$T_S(c_o, c_i, J'') = (g, c_o/G, c_i/G, J')$$

Depthwise Convolution

- Special case of grouping when C_O = C_I = G

Models and Implementation

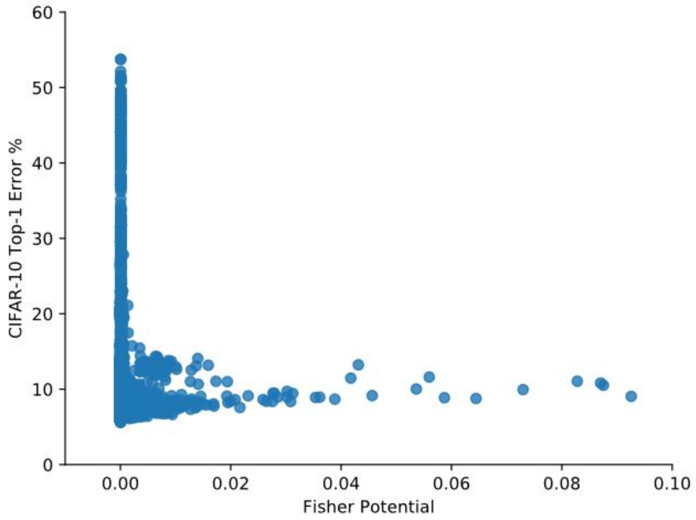
Optimization	Description
	Program Transformations
`reorder`	Interchange nested loops
`tile`	Cache and register blocking
`unroll`	Loop unrolling
`prefetch`	Memory coalescing between threads
`split`	Divide iteration into multiple axes
`fuse`	Combine two axes into one
	Neural Architecture Transformations
`bottleneck`	Reduce domain by factor B
`group`	Slice and offset two loops by factor G
	Mapping to GPU
`blockIdx`	Block-wise parallelism
`threadIdx`	Threads within blocks
`vthread`	Striding thread access

Models and Implementation

Fisher Potential

- Total information that each loop nest (layer) contains about class labels under a simplifying assumption of conditional independence.
- Or, how much each layer would affect the loss if deleted

$$\Delta_c = \frac{1}{2N} \sum_{n}^{N} \left(- \sum_{i}^{W} \sum_{j}^{H} A_{nij} g_{nij} \right)^2 .$$



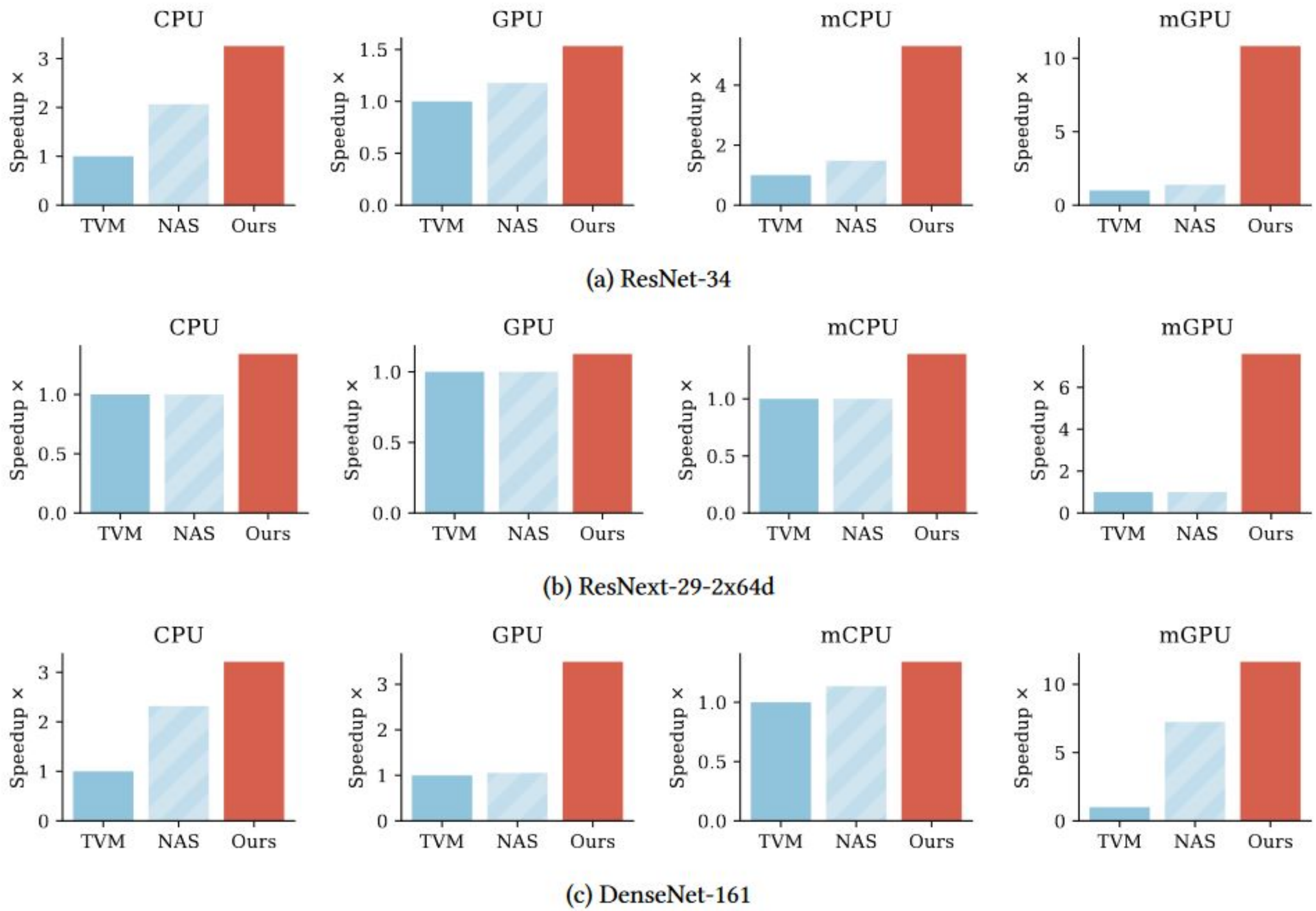
Models and Implementation

Search over 1000 configurations

Check which candidates satisfy Fisher Potential test and select best performing one

Compared to TVM & NAS (applying NAS then using TVM to compile)

Results

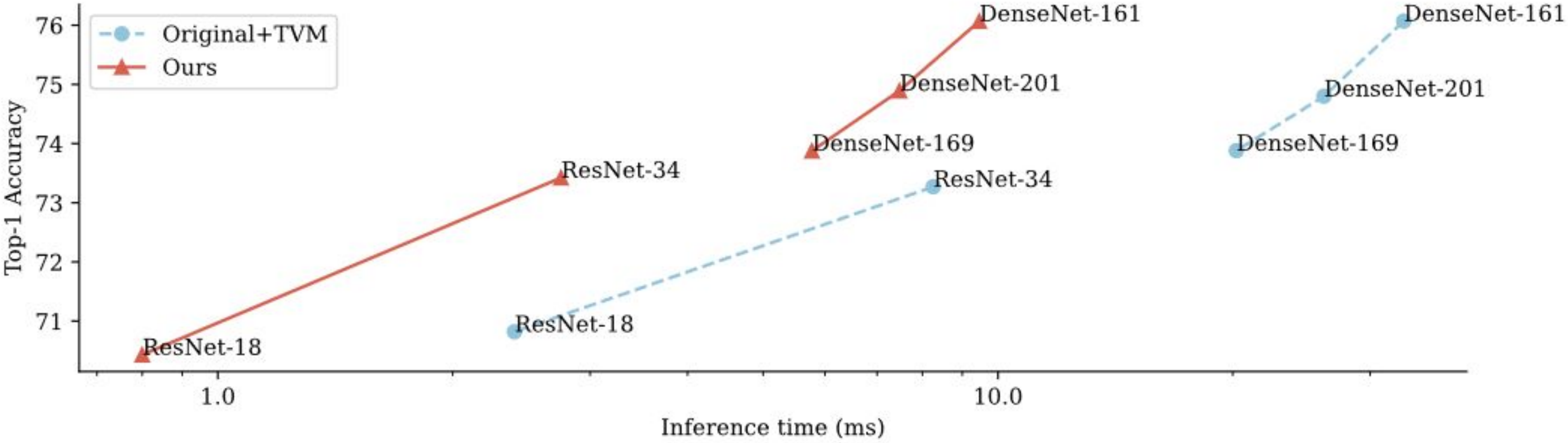


(a) ResNet-34

(b) ResNext-29-2x64d

(c) DenseNet-161

Results



Original+TVM
Ours
DenseNet-161
DenseNet-201
DenseNet-169
ResNet-34
ResNet-18
DenseNet-161
DenseNet-201
DenseNet-169
ResNet-34
ResNet-18
76
75
74
73
72
71
Top-1 Accuracy
1.0
10.0
Inference time (ms)

Results

3 Sequence of Operations Stood Out

1. [split → interchange → group → interchange → fuse]
   a. Group kernels over spatial domain

2. [unroll → group → interchange]
   a. Output channels unrolled by factor 16, then grouped by G = 2

3. [split → group → interchange → group]
   a. Splitting up iteration domain by applying different levels of grouping

Results

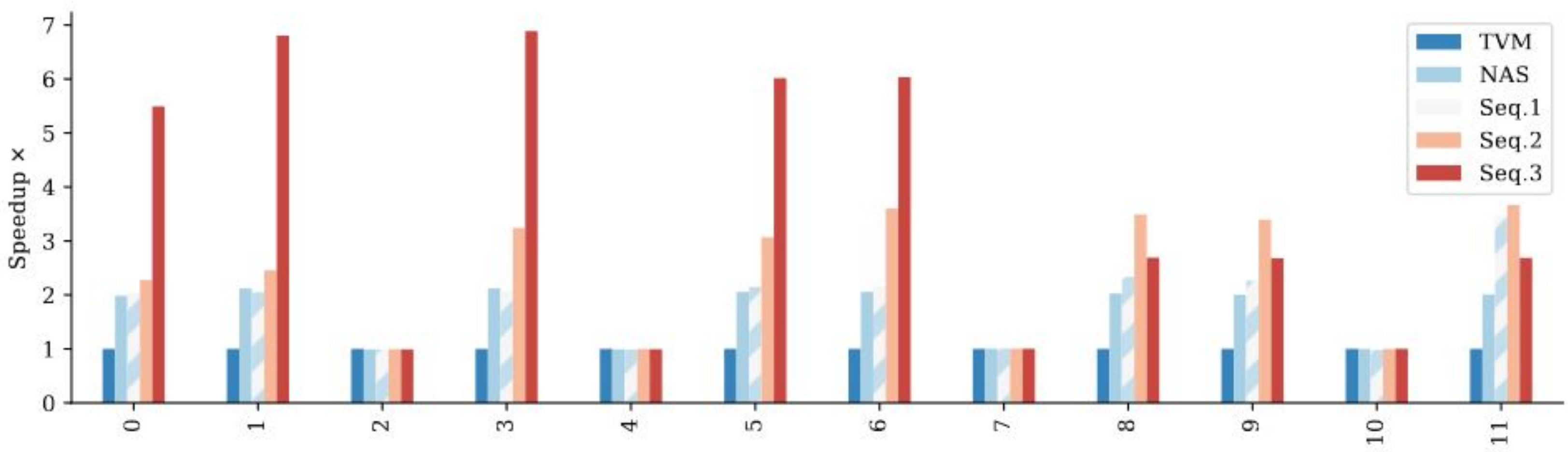
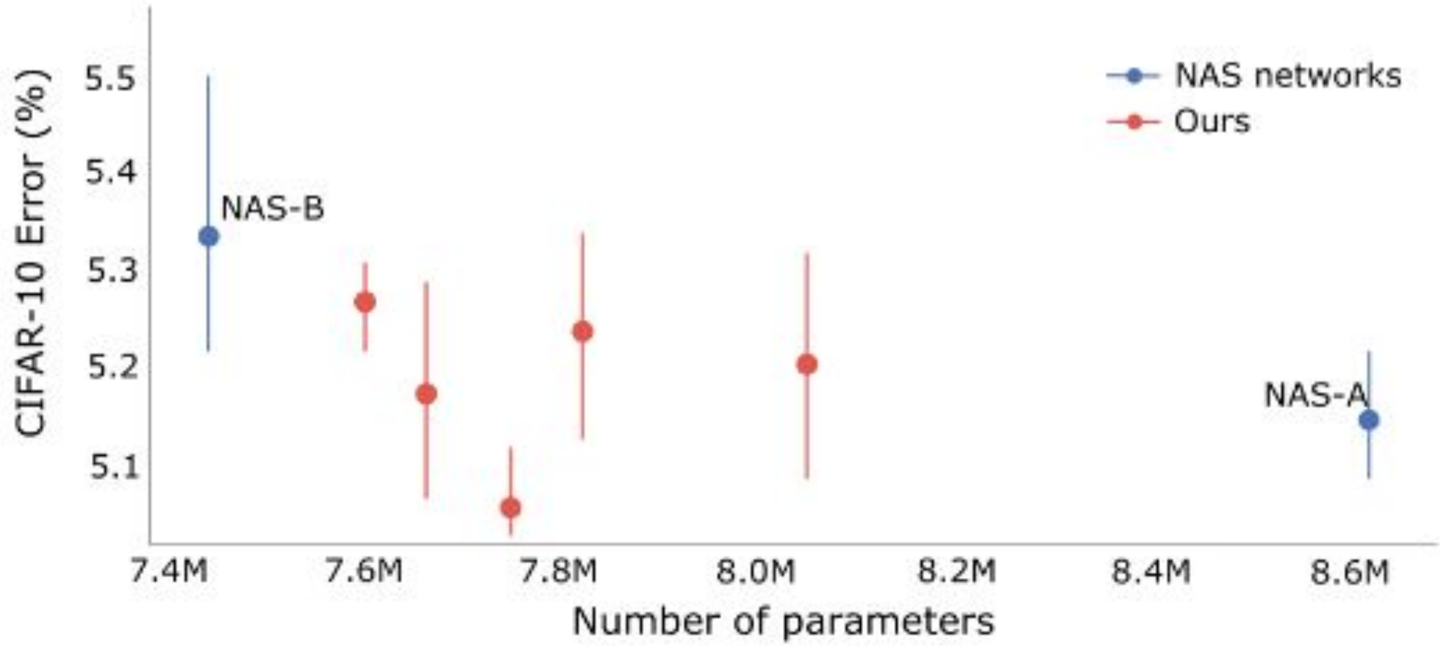


Figure 6: Exploring different sequences of transformations for an individual layer of ResNet-34 on the Intel Core i7 CPU. NAS is the result of applying grouping with factor 2 first, then compiling with TVM. The other three sequences are interleaved transformations produced by our method.

Results


CIFAR-10 Error (%)
5.5
5.4
5.3
5.2
5.1
NAS-B
NAS-A
NAS networks
Ours
7.4M
7.6M
7.8M
8.0M
8.2M
8.4M
8.6M
Number of parameters

Critiques and Concerns

Experiments: Compare best performance? Average performance?

Scalability and deployability

- Retraining models when deployed
- Distributed training?

Skeptical about Fisher Potential

- Could have benefited from more data

Search process too naive

Usefulness of bottlenecking

Limited NAS techniques that can be applied in program transformation

Questions?