

# Dive into Deep Learning for NLP

## 6. Deployment with TVM

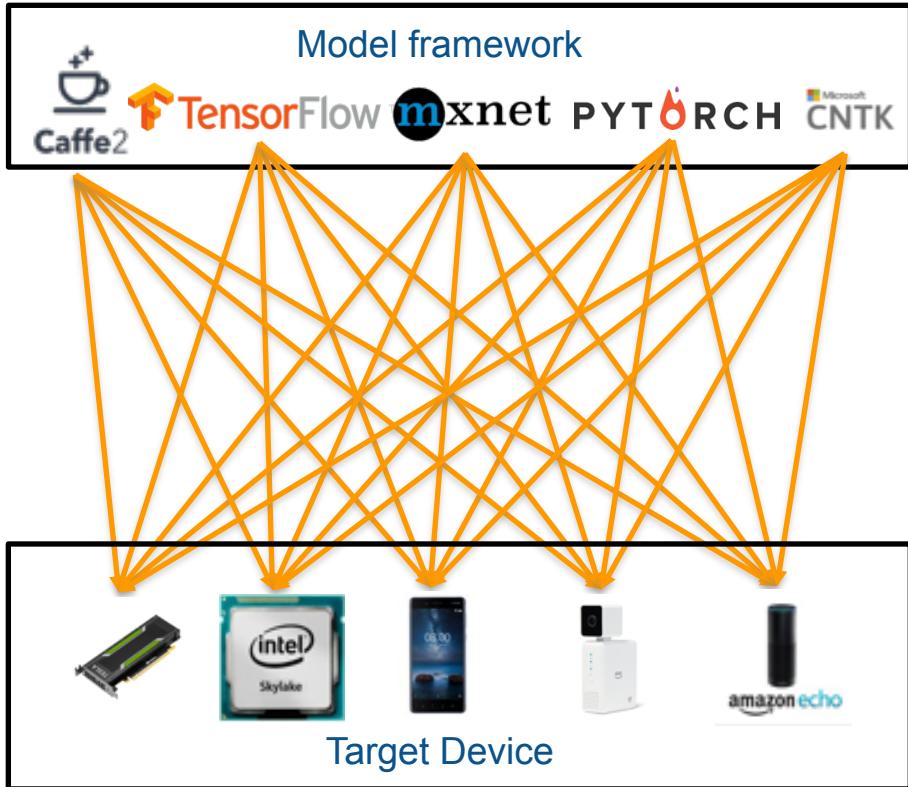
Haichen Shen

[gluon-nlp.mxnet.io](http://gluon-nlp.mxnet.io)

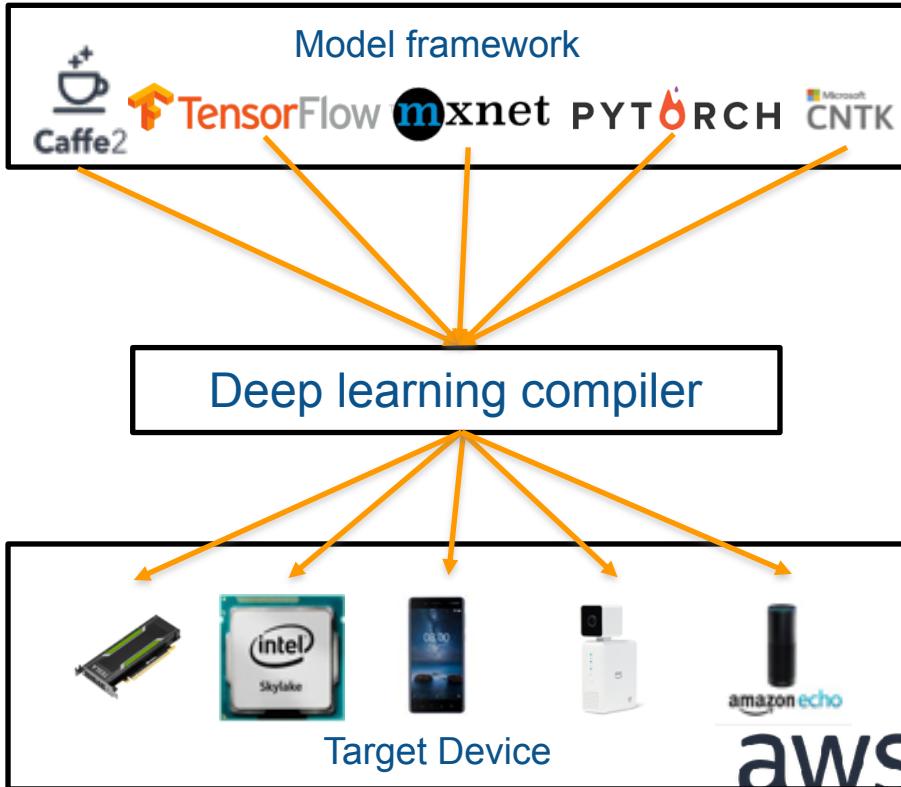
13:15-14:15	Natural Language Processing and Deep Learning Basics
14:15-14:25	Break
14:25-15:15	Context-free Representations with Word Embeddings
15:15-15:55	Machine Translation and Sequence Generation
15:55-16:35	Contextual Representations with BERT
16:35-16:45	Break
16:45-17:15	Model Deployment with TVM

# Deep learning inference is complicated

Before



After



# Models and hardware targets are far away!



Computation graph optimization

Tensor operation optimization

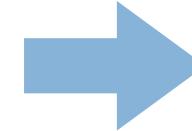
Machine code generation



# End-to-end deep learning compiler

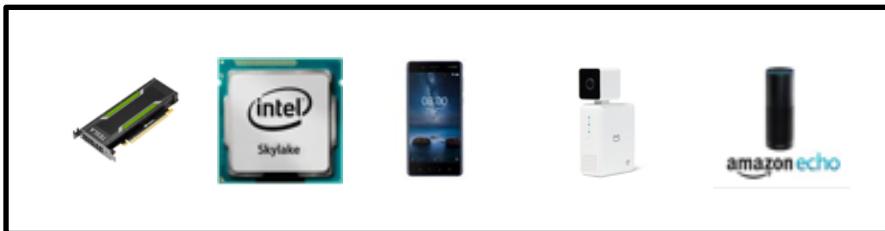


tvm



Amazon  
SageMaker Neo

end-to-end optimization



# TVM overview



Relay: High-Level Differentiable IR

Tensor Expression IR

LLVM, CUDA, Metal, ...



Edge  
FPGA

Cloud  
FPGA

ASIC

## Optimization

Graph optimization

Tensor optimization

AutoTVM

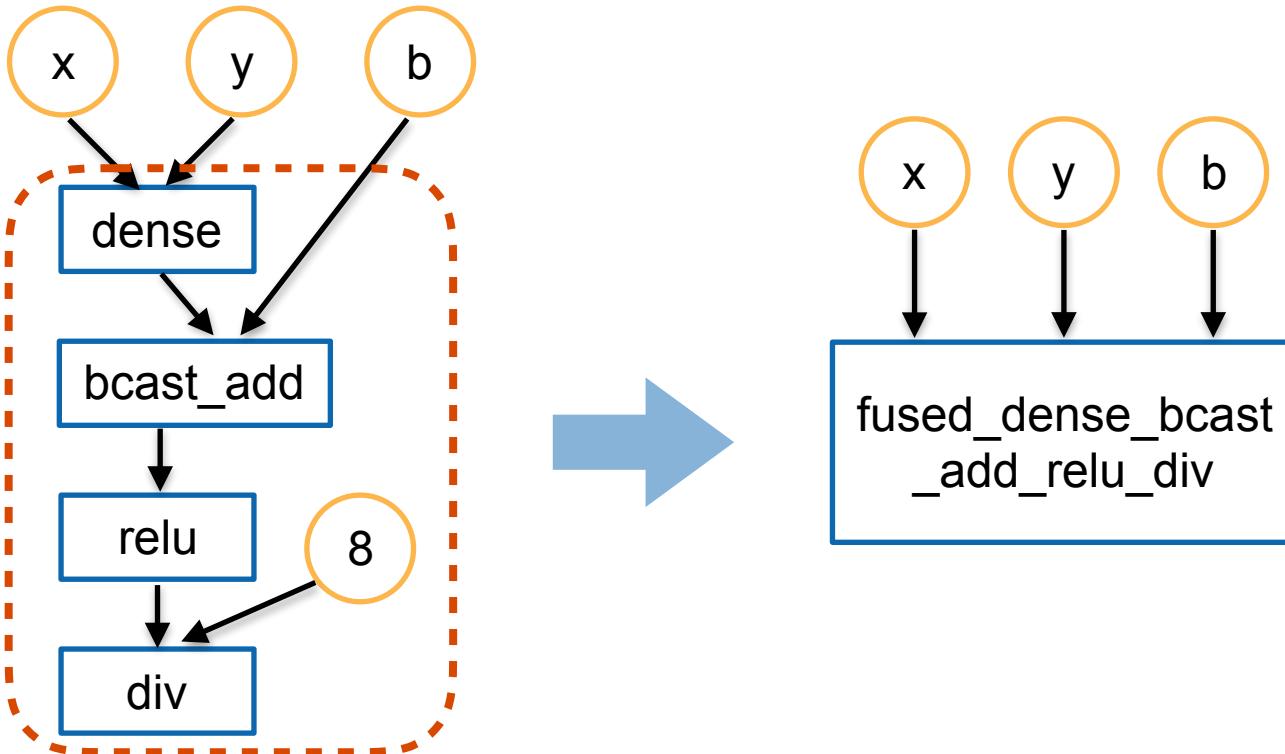
## Runtime

Graph runtime

VM

Interpreter

# Graph level optimization



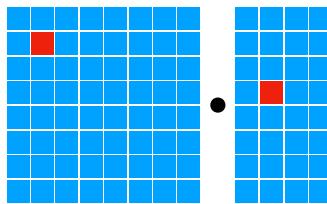
# Tensor level optimization

## Hardware

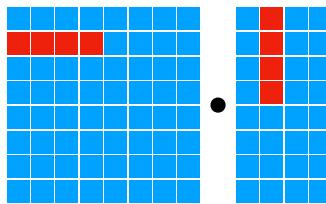


CPUs

## Compute Primitives



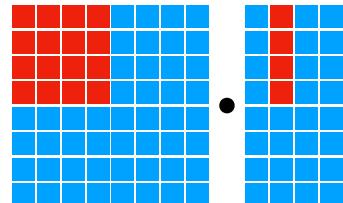
scalar



vector

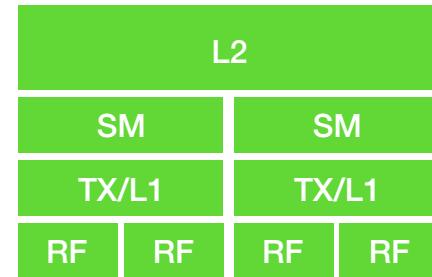
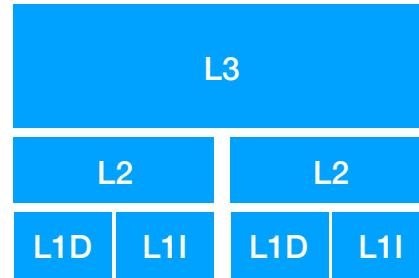


GPUs



tensor

## Memory Subsystem



## Matrix multiplication

```
C = tvm.compute((m, n),  
    lambda y, x: tvm.sum(A[k, y] * B[k, x], axis=k))
```

Billions of possible optimization choices

Vanilla Code

```
for y in range(1024):  
    for x in range(1024):  
        C[y][x] = 0  
        for k in range(1024):  
            C[y][x] += A[k][y] * B[k][x]
```

```
for xo in range(64):  
    for yo in range(64):  
        C[yo*16:yo*16+16][xo*16:xo*16+16] = 0  
    for ko in range(64):  
        for ki in range(16):  
            for yi in range(16):  
                for xi in range(16):  
                    C[yo*16+yi][xo*16+xi] +=  
                        A[ko*16+ki][yo*16+yi] * B[ko*16+ki][xo*16+xi]
```

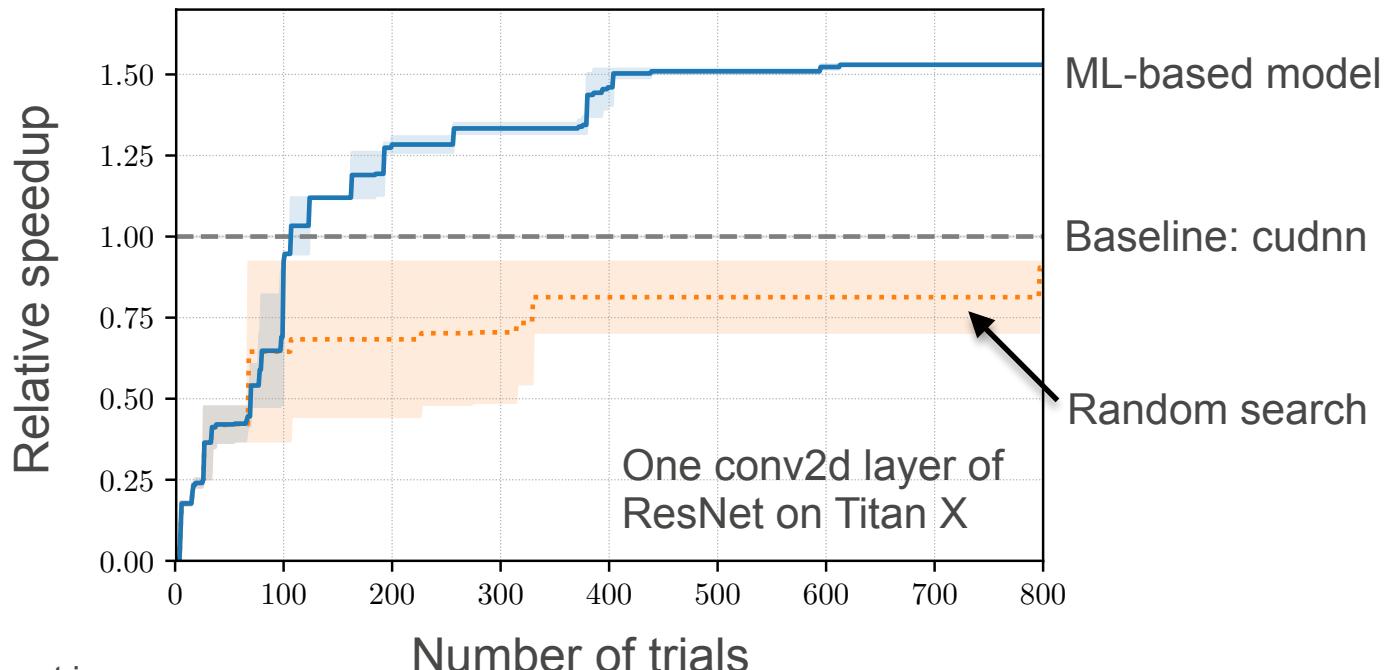


```
for yo in range(128):  
    for xo in range(128):  
        C[yo*8:yo*8+8][xo*8:xo*8+8] = 0  
    for ko in range(128):  
        for yi in range(8):  
            for xi in range(8):  
                for ki in range(8):  
                    C[yo*8+yi][xo*8+xi] +=  
                        A[ko*8+ki][yo*8+yi] * B[ko*8+ki][xo*8+xi]
```

```
for yo in range(64):  
    for xo in range(64):  
        C[yo*16:yo*16+16][xo*16:xo*16+16] = 0  
    for ko in range(64):  
        for yi in range(16):  
            for xi in range(16):  
                for ki in range(16):  
                    C[yo*16+yi][xo*16+xi] +=  
                        A[ko*16+ki][yo*16+yi] * B[ko*16+ki][xo*16+xi]
```

# AutoTVM: ML-based tensor optimization

- Use machine learning to learn hyper-parameter in the *schedule template*



# Using TVM for deployment

- Model coverage

Task	Model
Image classification	Resnet, VGG, Inception, MobileNet, DenseNet, etc.
Object detection	SSD, Yolo, Faster R-CNN
NLP	(bidirectional) LSTM/GRU, Transformer, Bert

- Device coverage

- Intel CPU, ARM CPU, Nvidia GPU, Mali GPU, Qualcomm, etc.

# BERT for QA performance on ARM and Intel CPU

Max seq length	Instance	TVM latency (ms)	MXNet latency (ms)	Speedup
128	a1.4x	251.59	1041.81	4.14
256	a1.4x	536.15	1322.22	4.14
128	c5.9x	59.82	37.66	0.63
256	c5.9x	88.52	68.24	0.77

# Work in progress

- More optimization for BERT on Intel CPU
- New runtime that supports control flow
- Dynamic shape support
- Quantization

# Thank you!

Contact: shaichen@, wangyida@, vinarm@

