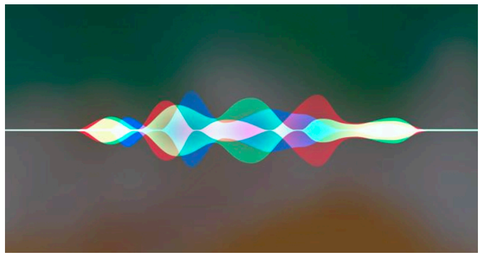
TVM: An End to End Automated Deep Learning Compiler

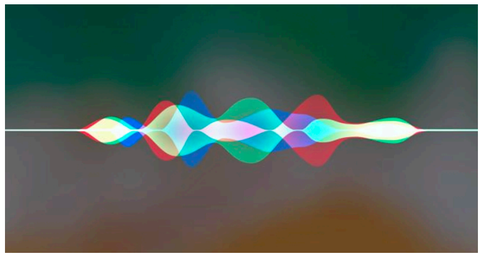
Tianqi Chen

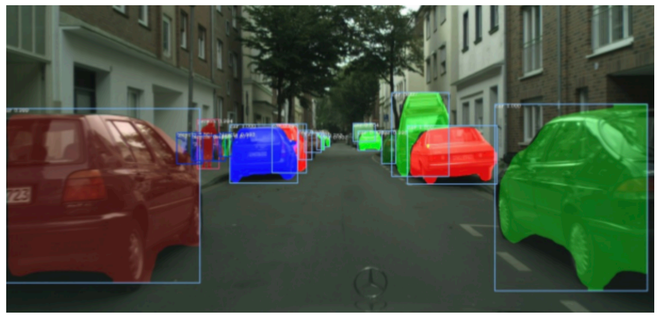
Machine Learning is Everywhere

Machine Learning is Everywhere

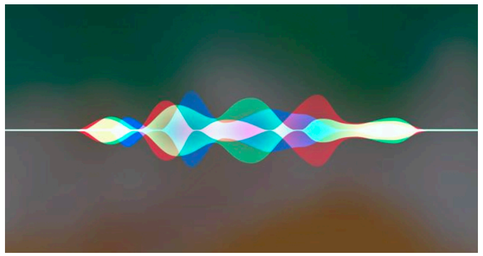


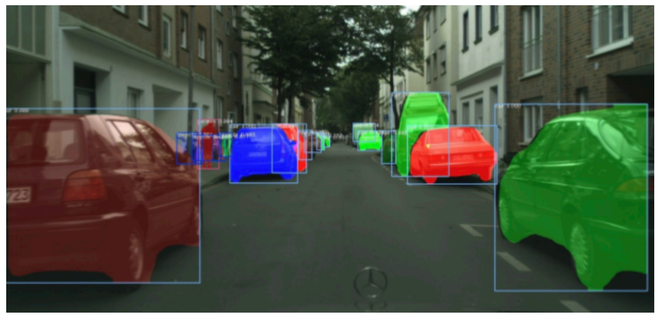
Machine Learning is Everywhere

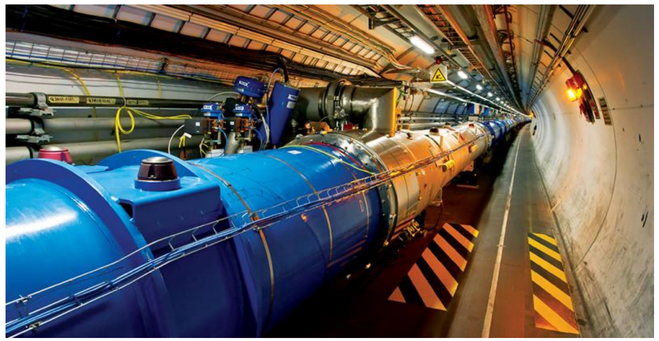


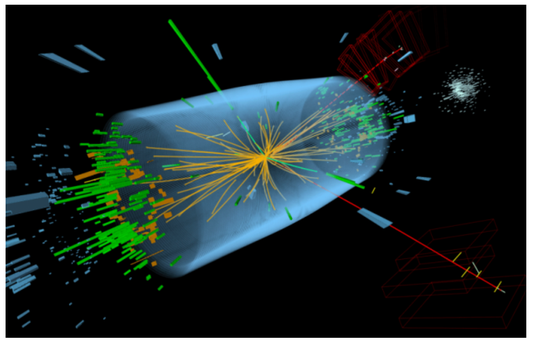


Machine Learning is Everywhere









Higgs challenge

Abbreviated History of Machine Learning

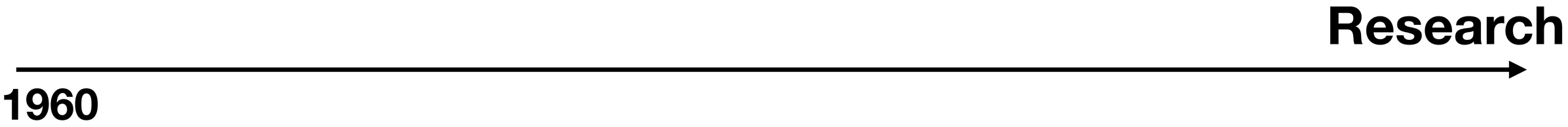
Based on personal view

Abbreviated History of Machine Learning

1960

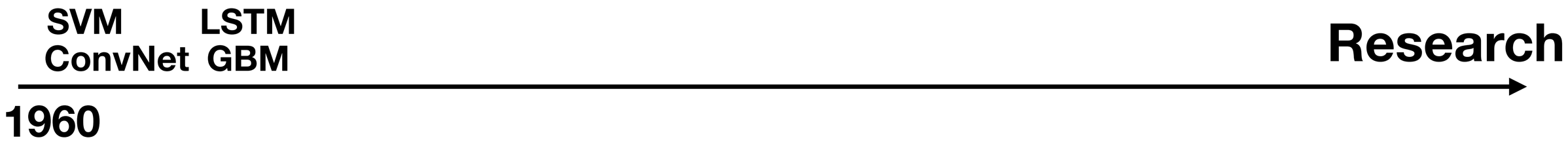
Based on personal view

Abbreviated History of Machine Learning



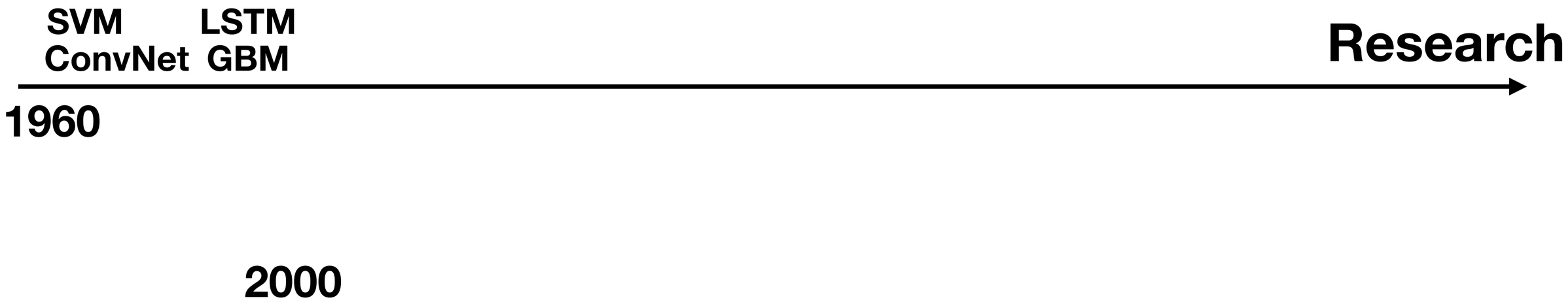
Based on personal view

Abbreviated History of Machine Learning



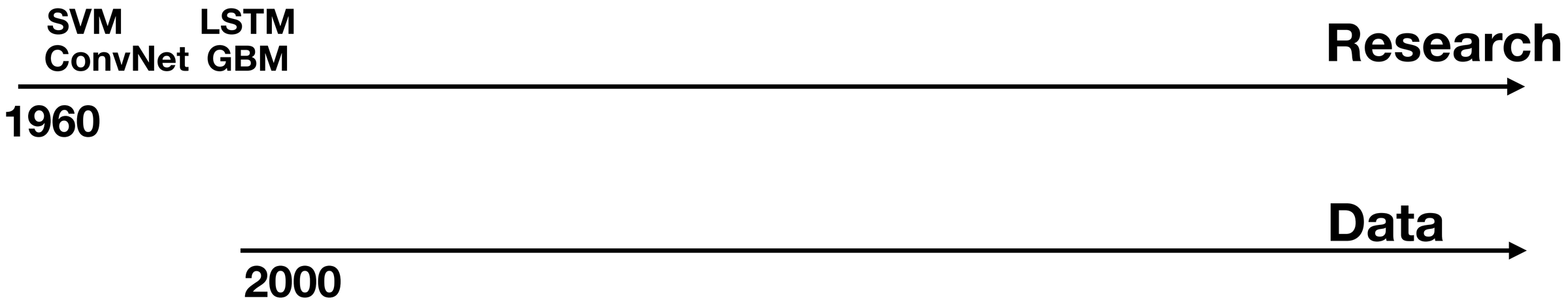
Based on personal view

Abbreviated History of Machine Learning



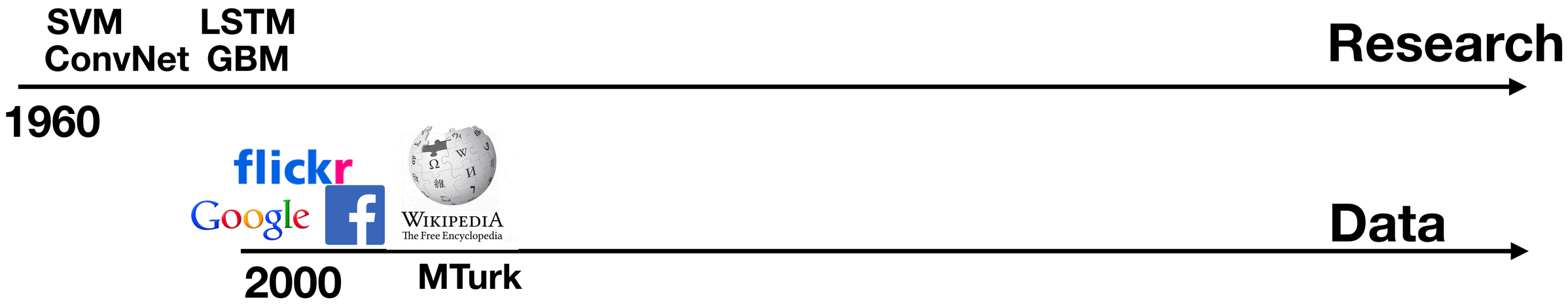
Based on personal view

Abbreviated History of Machine Learning



Based on personal view

Abbreviated History of Machine Learning



Based on personal view

Abbreviated History of Machine Learning

Based on personal view

Abbreviated History of Machine Learning

Based on personal view

Abbreviated History of Machine Learning

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

1960

Data

2000

MTurk

IMAGENET

Hardware

2006

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

1960

kaggle

Data

2000

MTurk

IMAGENET

Public
Cloud

Hardware

2006

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

1960

kaggle

Data

2000

MTurk

IMAGENET

Public
Cloud

Hardware

2006

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

1960

kaggle

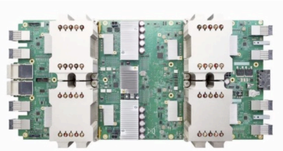
Data

2000

MTurk

IMAGENET

Public
Cloud



Hardware

2006

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

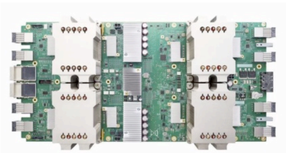
1960

Data

2000

MTurk

Public
Cloud



Hardware

2006

2010

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

1960

kaggle

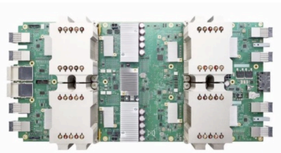
Data

2000

MTurk

IMAGENET

Public
Cloud



Hardware

2006

ML
Systems

2010

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

1960

kaggle

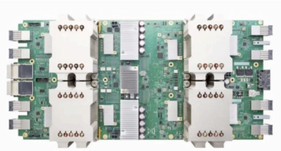
Data

2000

MTurk

IMAGENET

Public
Cloud



Hardware

2006

ML
Systems

2010

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

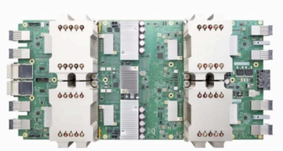
1960

kaggle

Data

2000 **MTurk** IMAGENET

Public
Cloud



Hardware

2006

ML
Systems

GraphLab

APACHE Spark™

2010

scikit learn

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

1960

kaggle

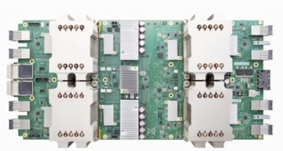
Data

2000

MTurk

IMAGENET

Public
Cloud



Hardware

2006

ML
Systems

2010

Caffe

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

1960

kaggle

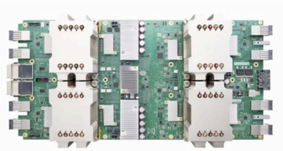
Data

2000

MTurk

IMAGENET

Public
Cloud



Hardware

2006

ML
Systems

2010

Caffe

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

1960

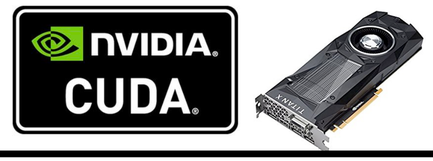
Data

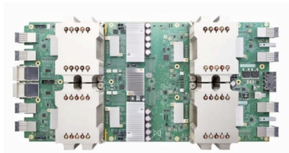
2000

MTurk

IMAGENET

Public Cloud





Hardware

2006

ML Systems

2010

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

1960

kaggle

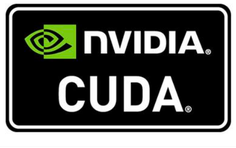
Data

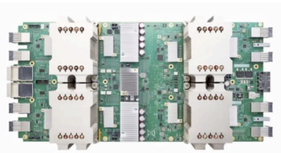
2000

MTurk

IMAGENET

Public
Cloud





Hardware

2006

ML
Systems

2010

Caffe

Based on personal view

Abbreviated History of Machine Learning

SVM
ConvNet
LSTM
GBM

Research

1960

kaggle

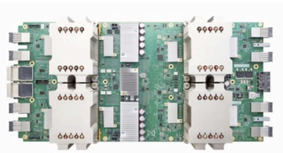
Data

2000

MTurk

IMAGENET

Public
Cloud



Hardware

2006

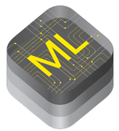
ML
Systems

2010

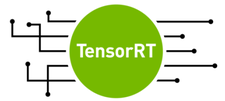
Based on personal view

Current Deep Learning Systems em

Frameworks and Inference engines





DL Compilers


Kenrel Libraries

CuDNN NNPack MKL-DNN

Hand optimized

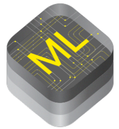
Hardware



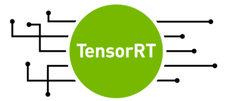


Current Deep Learning Systems em

Frameworks and Inference engines



DL Compilers


Kenrel Libraries

CuDNN NNPack MKL-DNN

Hand optimized

Hardware



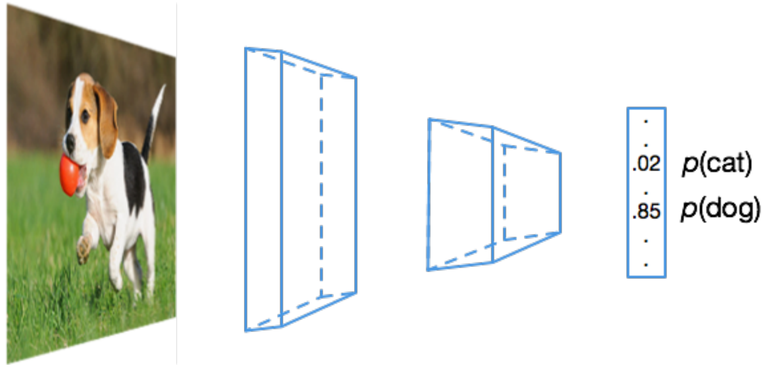


tvm

Open source, automated end-to-end optimization framework for deep learning.

Problem: Deep Learning Deployment

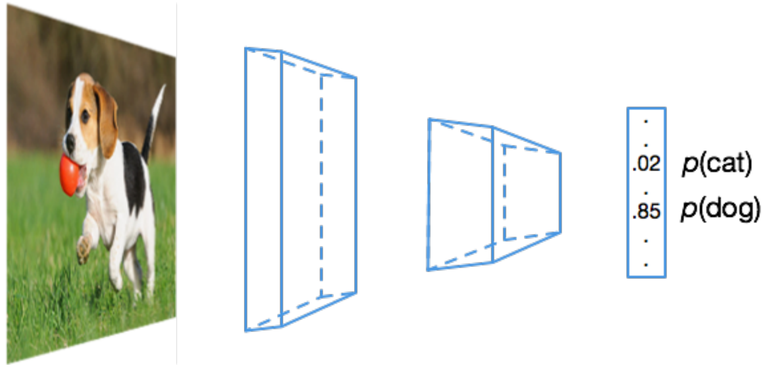
Model



Hardware Backends

Problem: Deep Learning Deployment

Model

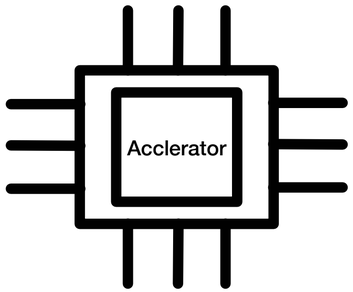


Deploy

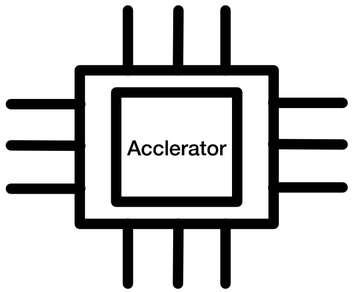
Hardware Backends

Original Motivating Problem

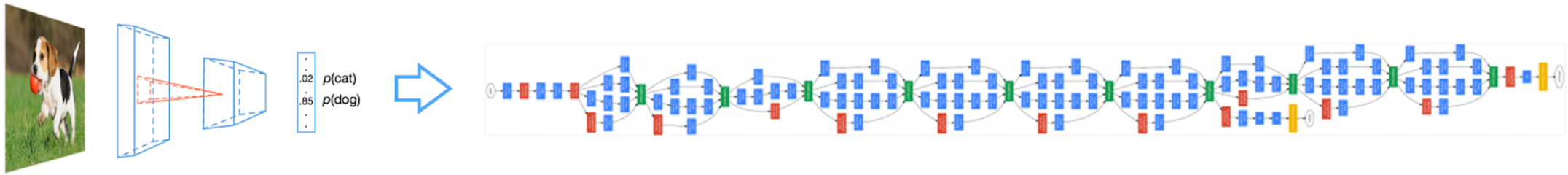
Original Motivating Problem



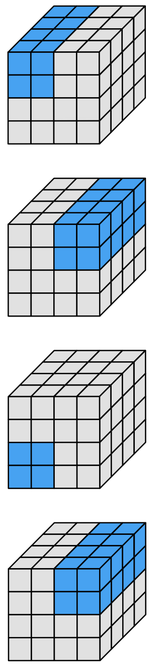
Original Motivating Problem



Original Motivating Problem




```
// Pseudo-code for convolution program for the VIA accelerator
// Virtual Thread 0
0x00: LOAD(PARAM[ 0-71])                                              // LD@TID0
0x01: LOAD(ACTIV[ 0-24])                                              // LD@TID0
0x02: LOAD(LDBUF[ 0-31])                                              // LD@TID0
0x03: PUSH(LD->EX)                                                    // LD@TID0
0x04: POP (LD->EX)                                                    // EX@TID0
0x05: EXE (ACTIV[ 0-24],PARAM[ 0-71],LDBUF[ 0-31],STBUF[ 0- 7]) // EX@TID0
0x06: PUSH(EX->LD)                                                    // EX@TID0
0x07: PUSH(EX->ST)                                                    // EX@TID0
0x08: POP (EX->ST)                                                    // ST@TID0
0x09: STOR(STBUF[ 0- 7])                                              // ST@TID0
0x0A: PUSH(ST->EX)                                                    // ST@TID0
// Virtual Thread 1
0x0B: LOAD(ACTIV[25-50])                                              // LD@TID1
0x0C: LOAD(LDBUF[32-63])                                              // LD@TID1
0x0D: PUSH(LD->EX)                                                    // LD@TID1
0x0E: POP (LD->EX)                                                    // EX@TID1
0x0F: EXE (ACTIV[25-50],PARAM[ 0-71],LDBUF[32-63],STBUF[32-39]) // EX@TID1
0x10: PUSH(EX->LD)                                                    // EX@TID1
0x11: PUSH(EX->ST)                                                    // EX@TID1
0x12: POP (EX->ST)                                                    // ST@TID1
0x13: STOR(STBUF[32-39])                                              // ST@TID1
0x14: PUSH(ST->EX)                                                    // ST@TID1
// Virtual Thread 2
0x15: POP (EX->LD)                                                    // LD@TID2
0x16: LOAD(PARAM[ 0-71])                                              // LD@TID2
0x17: LOAD(ACTIV[ 0-24])                                              // LD@TID2
0x18: LOAD(LDBUF[ 0-31])                                              // LD@TID2
0x19: PUSH(LD->EX)                                                    // LD@TID2
0x1A: POP (LD->EX)                                                    // EX@TID2
0x1B: POP (ST->EX)                                                    // EX@TID2
0x1C: EXE (ACTIV[ 0-24],PARAM[ 0-71],LDBUF[ 0-31],STBUF[ 0- 7]) // EX@TID2
0x1D: PUSH(EX->ST)                                                    // EX@TID2
0x1E: POP (EX->ST)                                                    // ST@TID2
0x1F: STOR(STBUF[ 0- 7])                                              // ST@TID2
// Virtual Thread 3
0x20: POP (EX->LD)                                                    // LD@TID3
0x21: LOAD(ACTIV[25-50])                                              // LD@TID3
0x22: LOAD(LDBUF[32-63])                                              // LD@TID3
0x23: PUSH(LD->EX)                                                    // LD@TID3
0x24: POP (LD->EX)                                                    // EX@TID3
0x25: POP (ST->EX)                                                    // EX@TID2
0x26: EXE (ACTIV[25-50],PARAM[ 0-71],LDBUF[32-63],STBUF[32-39]) // EX@TID3
0x27: PUSH(EX->ST)                                                    // EX@TID3
0x28: POP (EX->ST)                                                    // ST@TID3
0x29: STOR(STBUF[32-39])                                              // ST@TID3
```



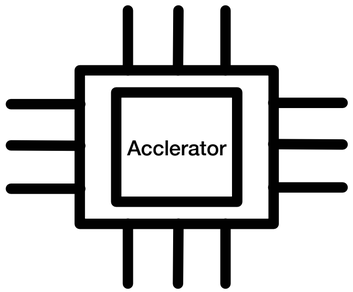
(a) Blocked convolution program with multiple thread contexts

```
// Convolution access pattern dictated by micro-coded program.
// Each register index is derived as a 2-D affine function.
// e.g. idx_rf = a_rf y+b_rf x+c_rf^0, where c_rf^0 is specified by
//      micro op 0 fields.
for y in [0…i)
  for x in [0…j)
    rf[idx_rf^0] += GEVM(act[idx_act^0], par[idx_par^0])
    rf[idx_rf^1] += GEVM(act[idx_act^1], par[idx_par^1])
    …
    rf[idx_rf^n] += GEVM(act[idx_act^n], par[idx_par^n])
```

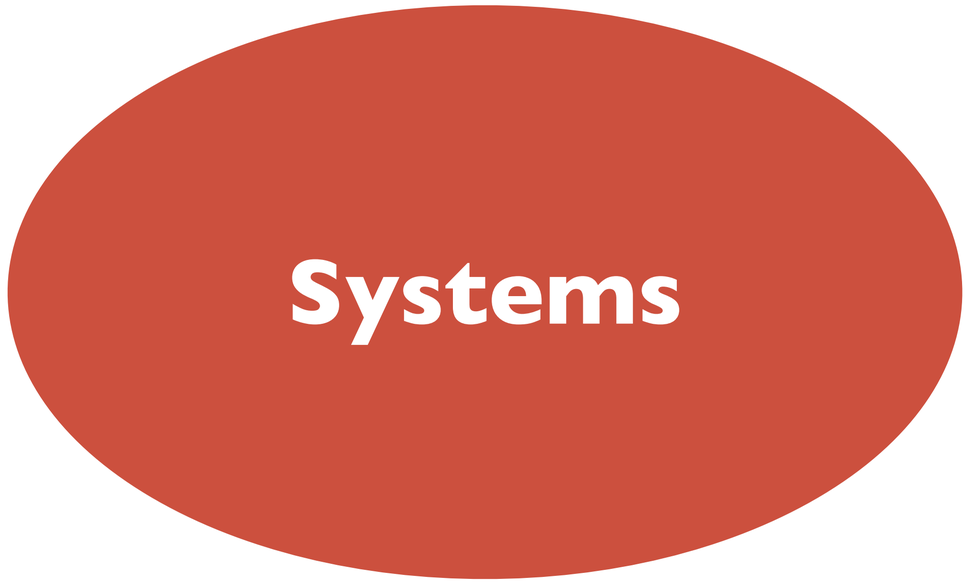
(b) Convolution micro-coded program

```
// Max-pool, batch normalization and activation function
// access pattern dictated by micro-coded program.
// Each register index is derived as a 2D affine function.
// e.g. idx_dst = a_dst y+b_dst x+c_dst^0, where c_dst^0 is specified by
//      micro op 0 fields.
for y in [0…i)
  for x in [0…j)
    // max pooling
    rf[idx_dst^0] = MAX(rf[idx_dst^0], rf[idx_src^0])
    rf[idx_dst^1] = MAX(rf[idx_dst^1], rf[idx_src^1])
    …
    // batch norm
    rf[idx_dst^m] = MUL(rf[idx_dst^m], rf[idx_src^m])
    rf[idx_dst^m+1] = ADD(rf[idx_dst^m+1], rf[idx_src^m+1])
    rf[idx_dst^m+2] = MUL(rf[idx_dst^m+2], rf[idx_src^m+2])
    rf[idx_dst^m+3] = ADD(rf[idx_dst^m+3], rf[idx_src^m+3])
    …
    // activation
    rf[idx_dst^n-1] = RELU(rf[idx_dst^n-1], rf[idx_src^n-1])
    rf[idx_dst^n] = RELU(rf[idx_dst^n], rf[idx_src^n])
```

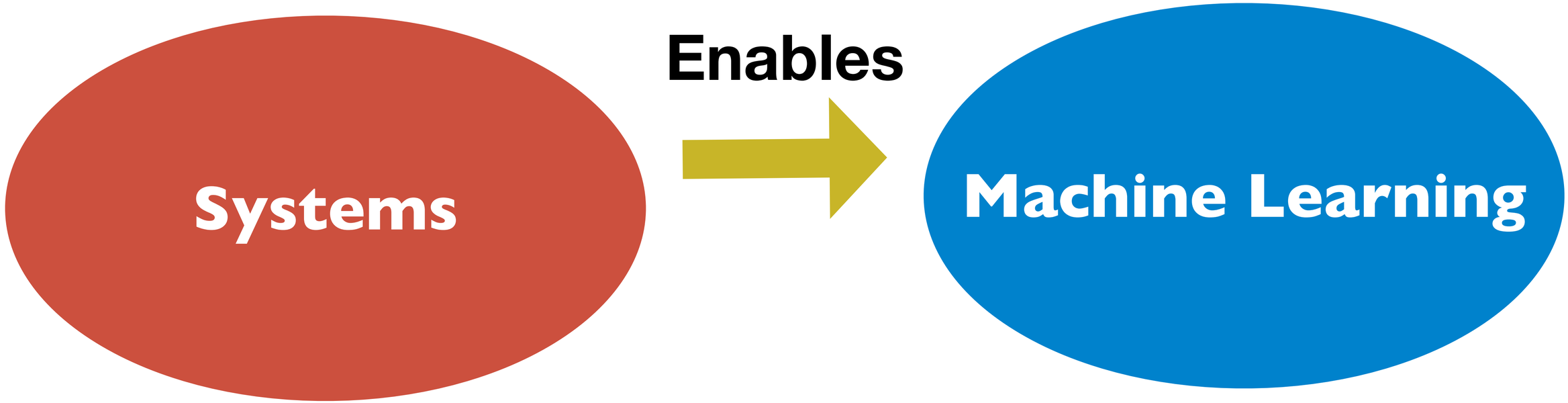
(c) Max pool, batch norm and activation micro-coded program



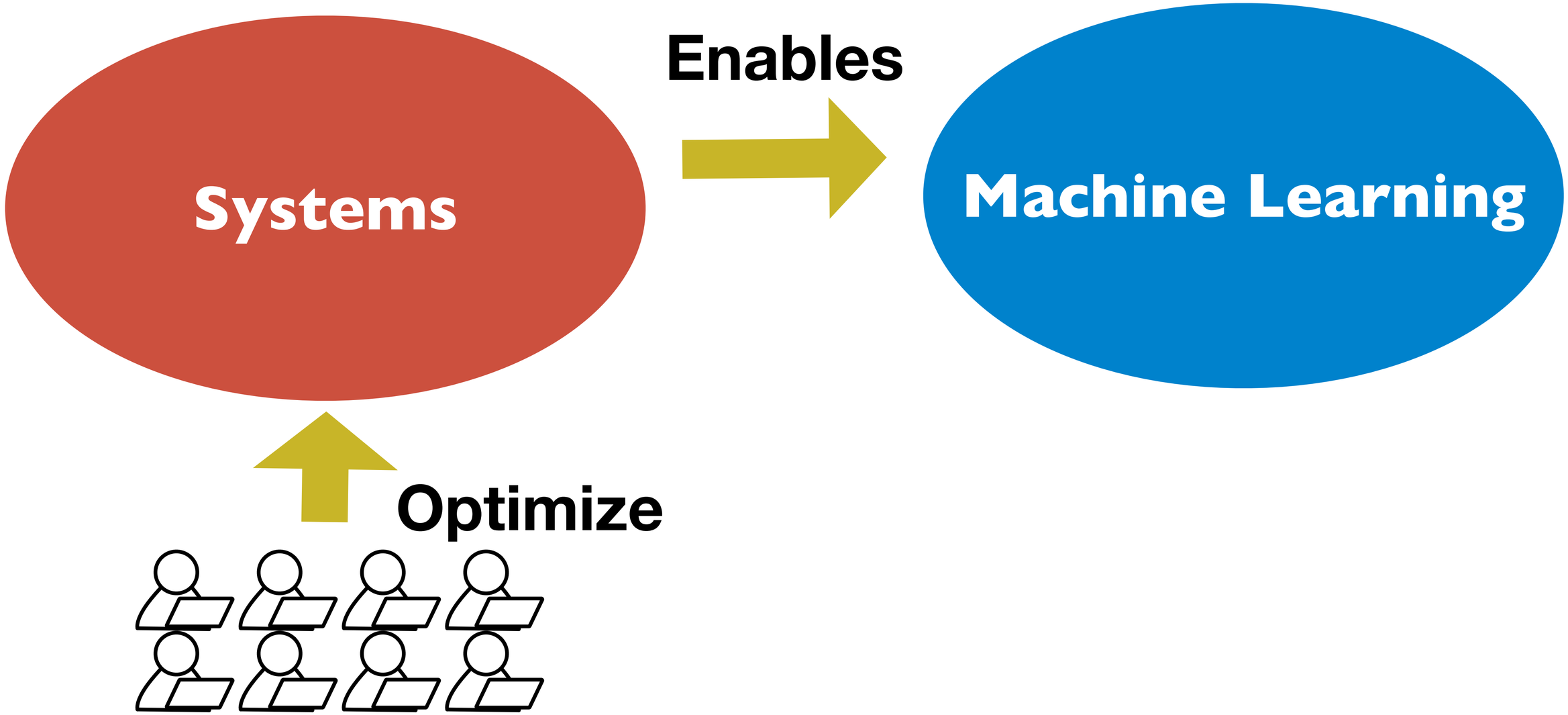
Current Learning Systems



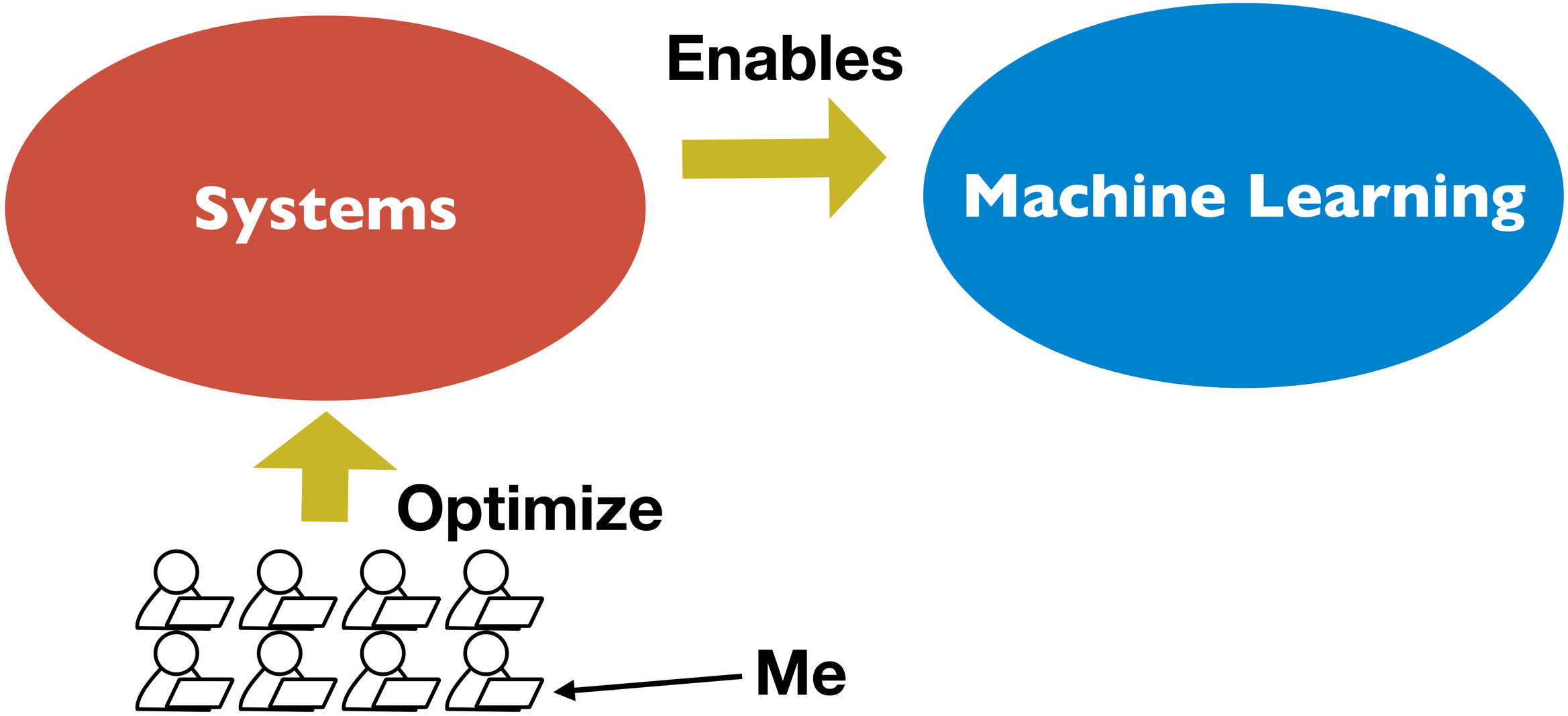
Current Learning Systems



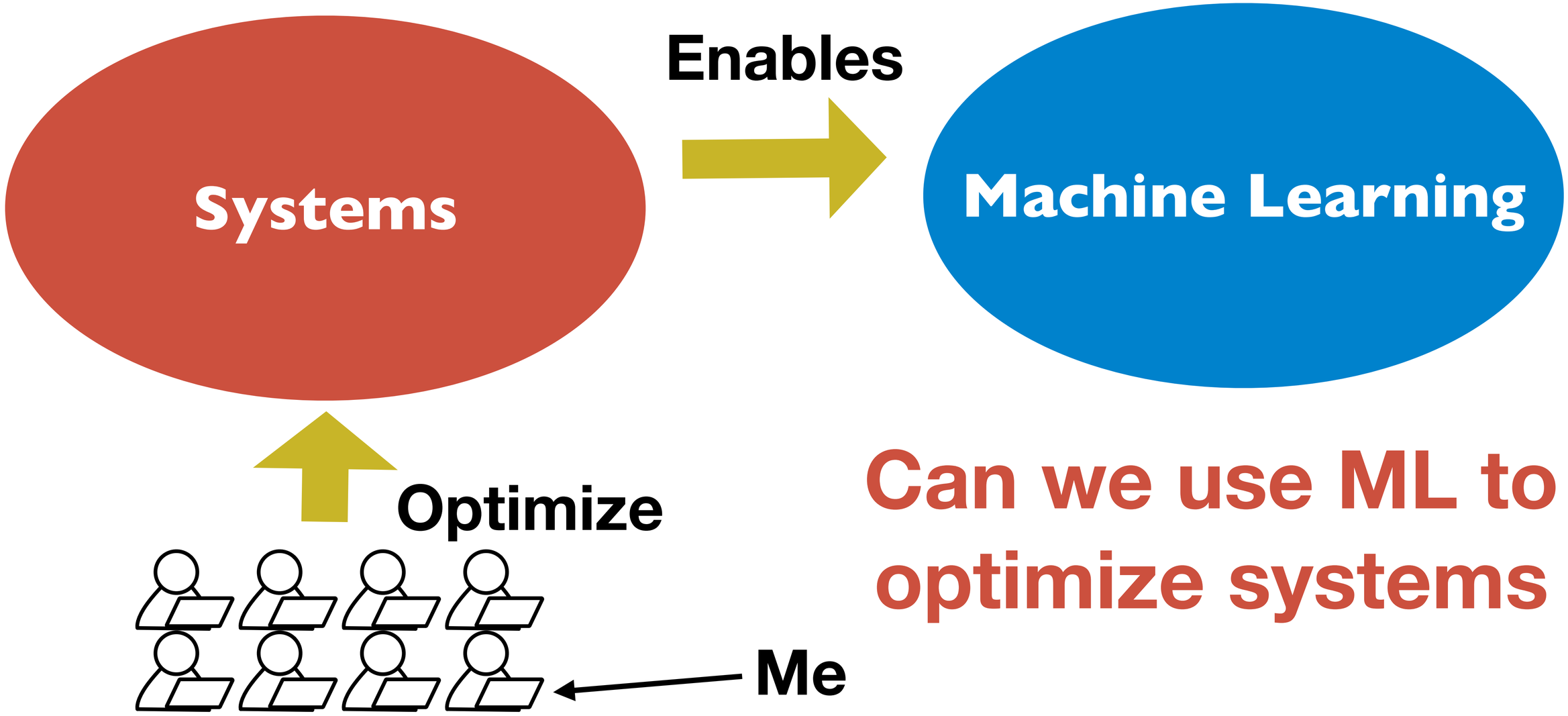
Current Learning Systems



Current Learning Systems

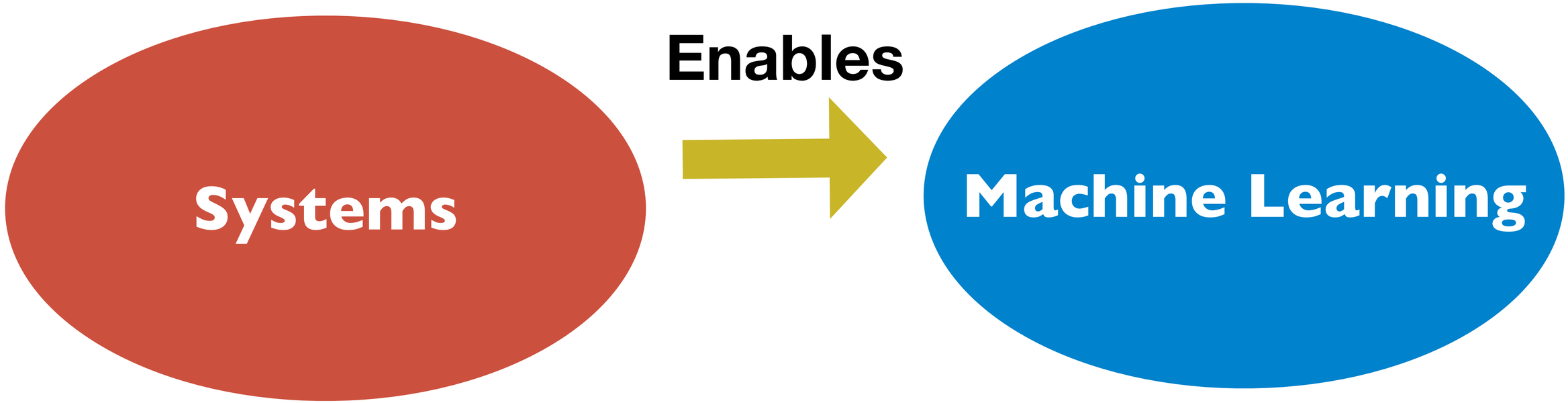


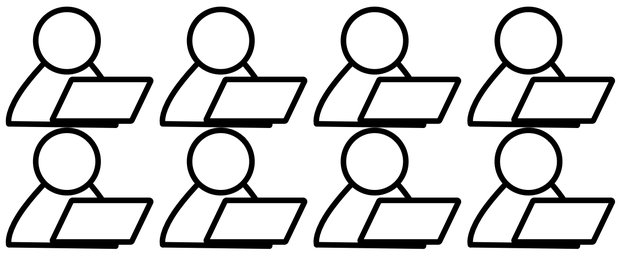
Current Learning Systems



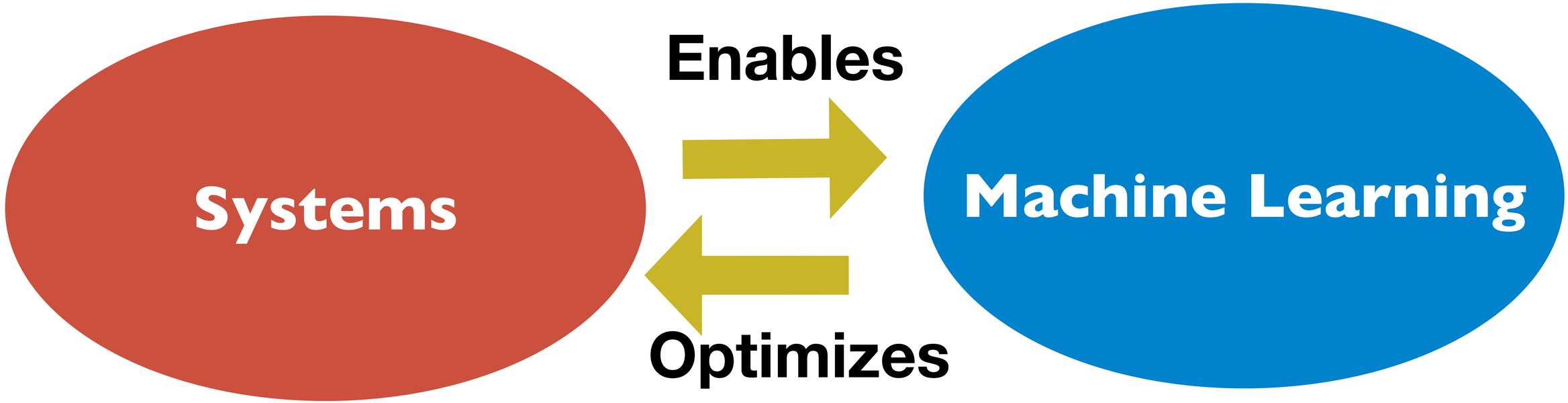
Can we use ML to optimize systems

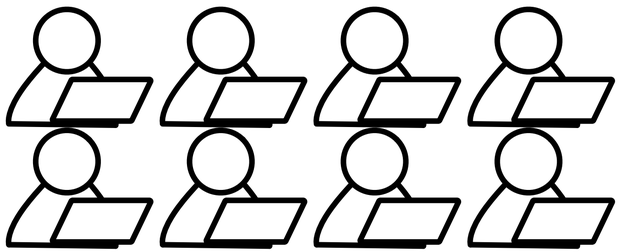
Learning-based Learning Systems



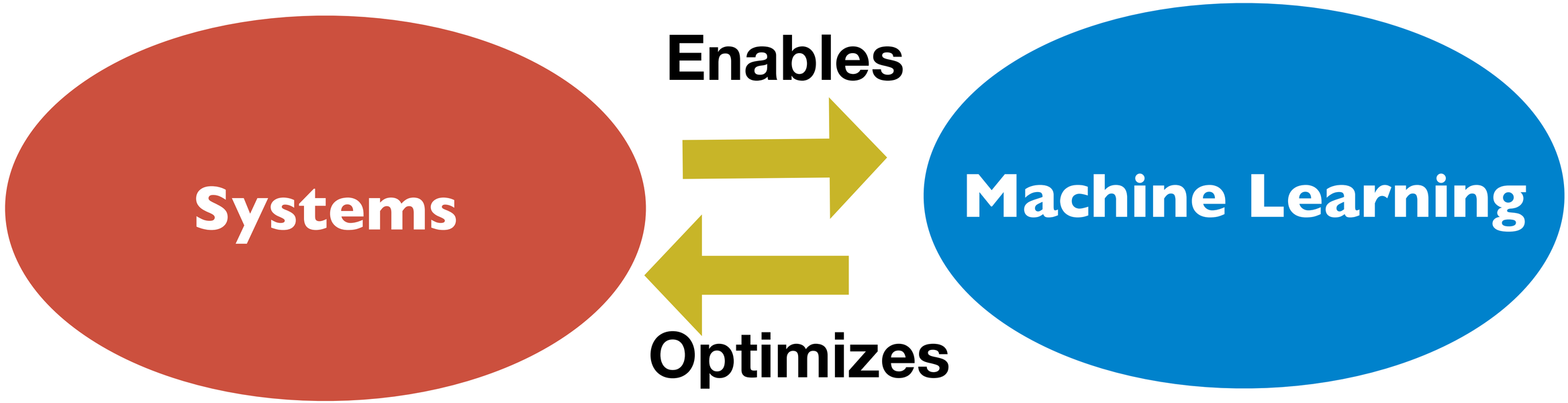


Learning-based Learning Systems





Learning-based Learning Systems



TVM: Learning-based Learning System

Why do we need machine learning for systems

How to build intelligent systems with learning

End to end learning-based learning system stack

TVM: Learning-based Learning System

Why do we need machine learning for systems

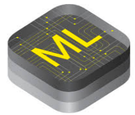
How to build intelligent systems with learning

End to end learning-based learning system stack

Deploy Deep Learning Everywhere

Hardware

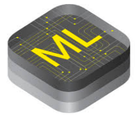
Deploy Deep Learning Everywhere

Frameworks      

Hardware 

Deploy Deep Learning Everywhere

Frameworks

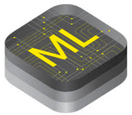





Hardware


Deploy Deep Learning Everywhere

Frameworks

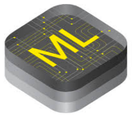


Hardware



Deploy Deep Learning Everywhere

Frameworks

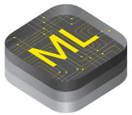


Hardware




Deploy Deep Learning Everywhere

Frameworks

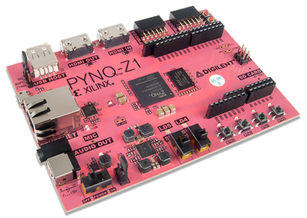





Hardware

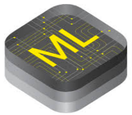






Deploy Deep Learning Everywhere

Frameworks

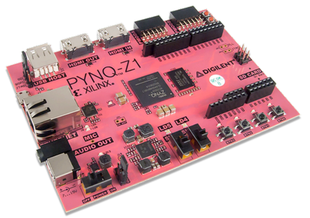



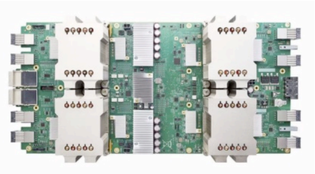


Hardware



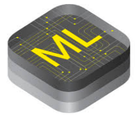






Deploy Deep Learning Everywhere

Frameworks

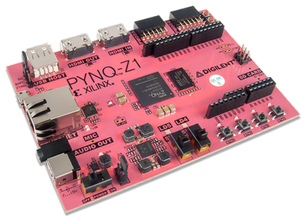


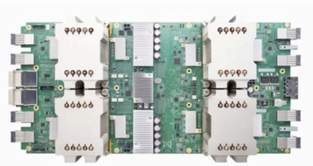
Huge gap between model/frameworks and hardware backends

Hardware



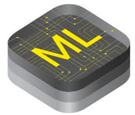






Existing Deep Learning Frameworks

Frameworks

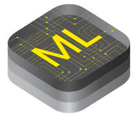





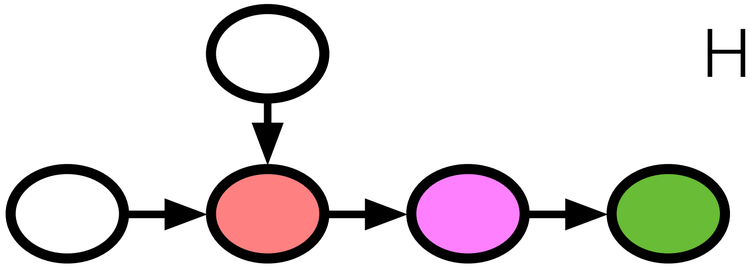
Hardware

Existing Deep Learning Frameworks

Frameworks



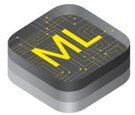


High-level data flow graph

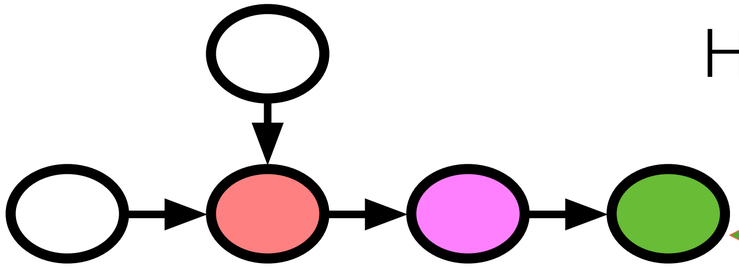
Hardware

Existing Deep Learning Frameworks

Frameworks





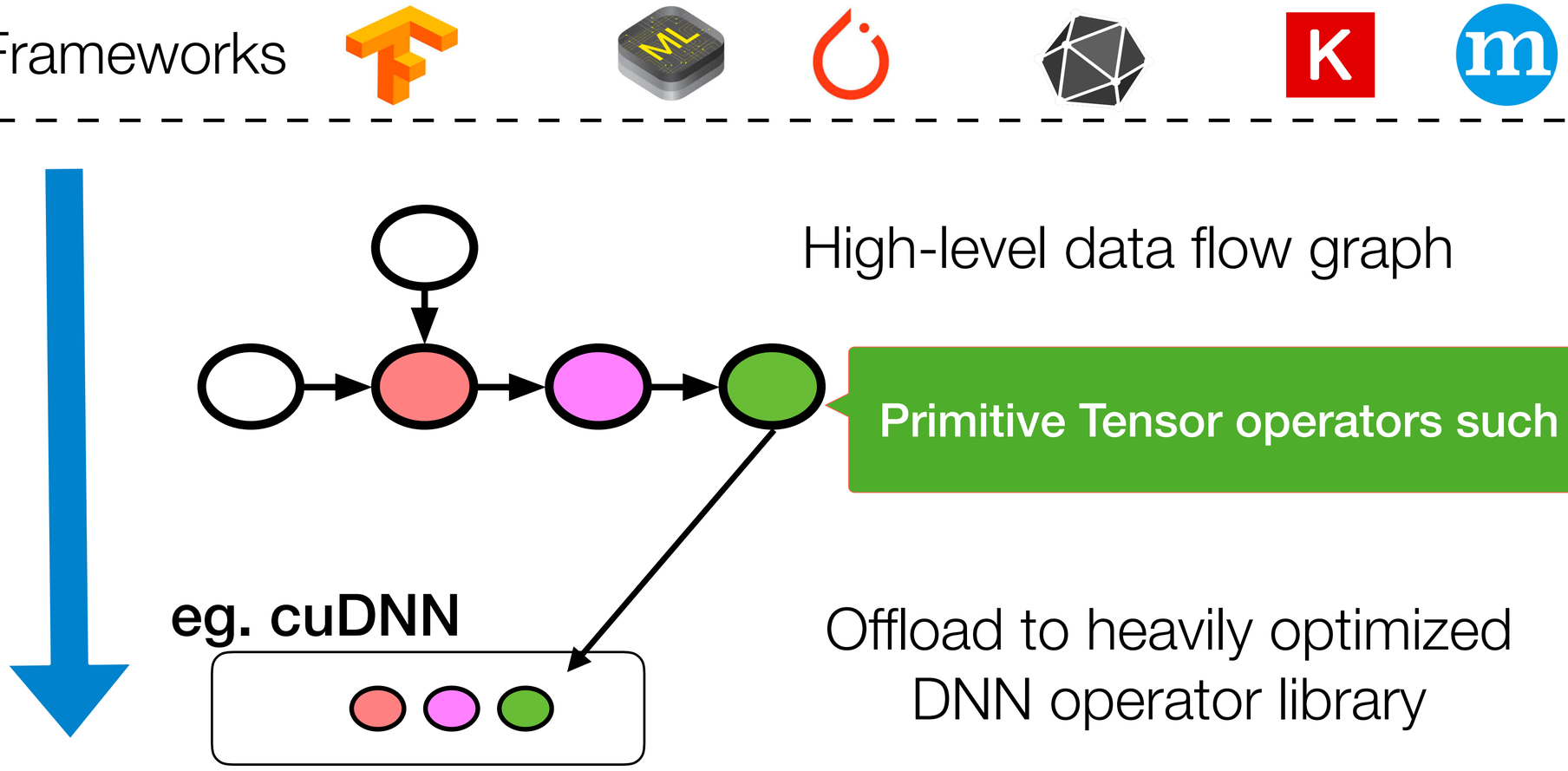
High-level data flow graph

Primitive Tensor operators such as Conv2D

Hardware

Existing Deep Learning Frameworks

Frameworks

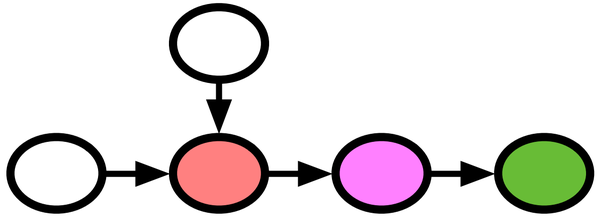


Hardware

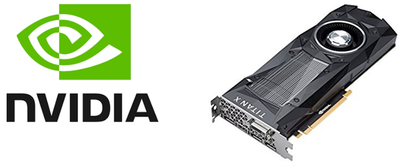
NVIDIA

Limitations of Existing Approach

Frameworks

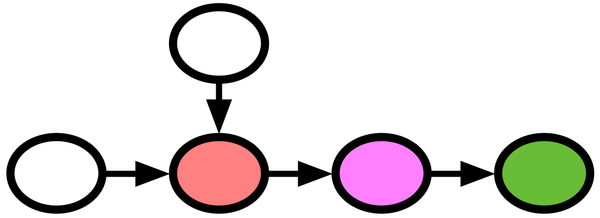


cuDNN

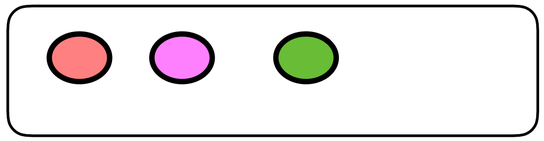


Limitations of Existing Approach

Frameworks



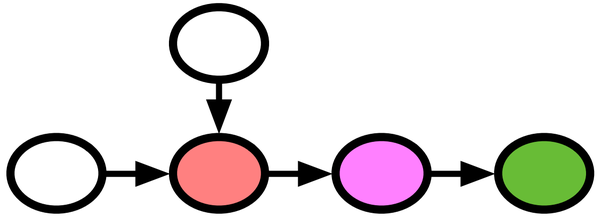
cuDNN



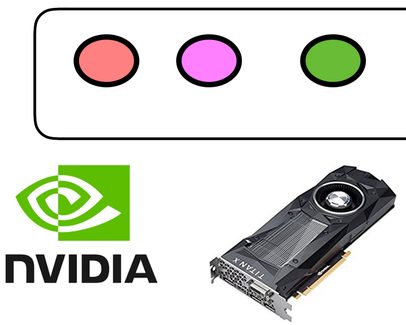
NVIDIA

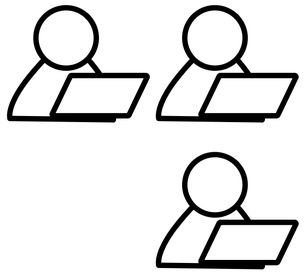
Limitations of Existing Approach

Frameworks



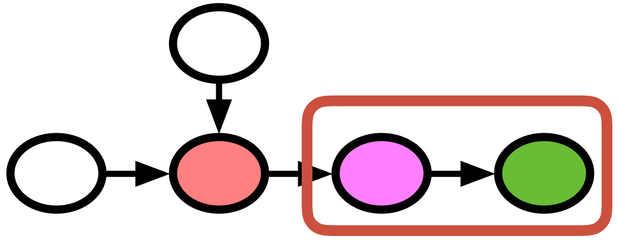
cuDNN



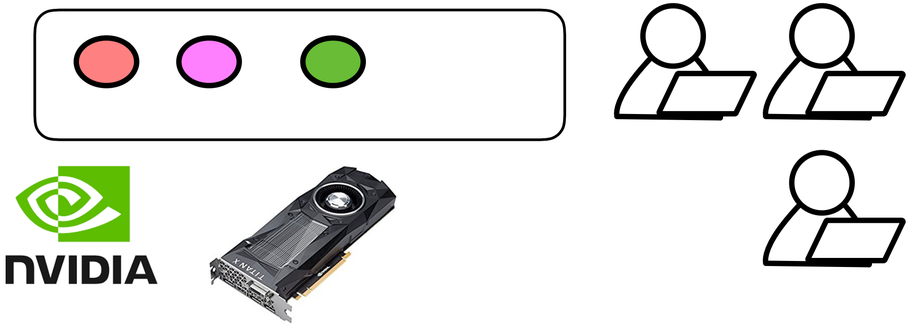


Limitations of Existing Approach

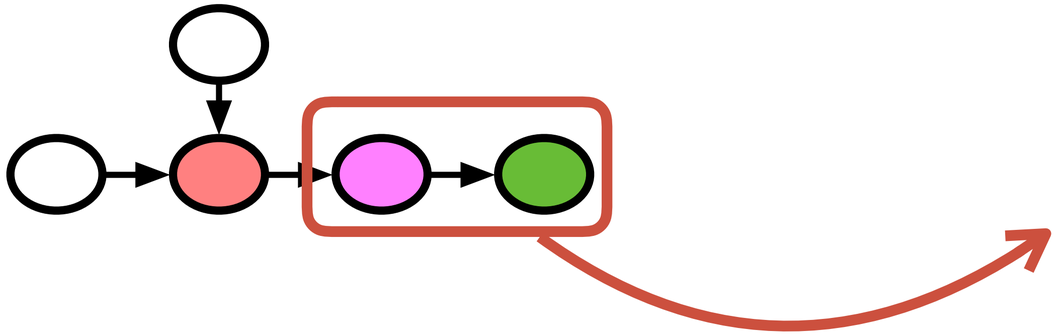
Frameworks



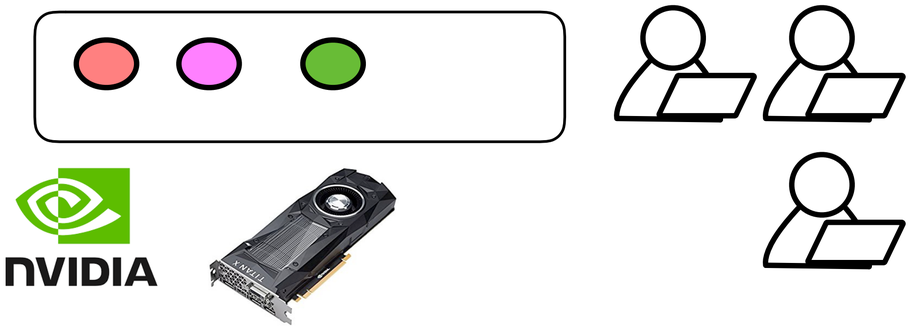
cuDNN



Limitations of Existing Approach

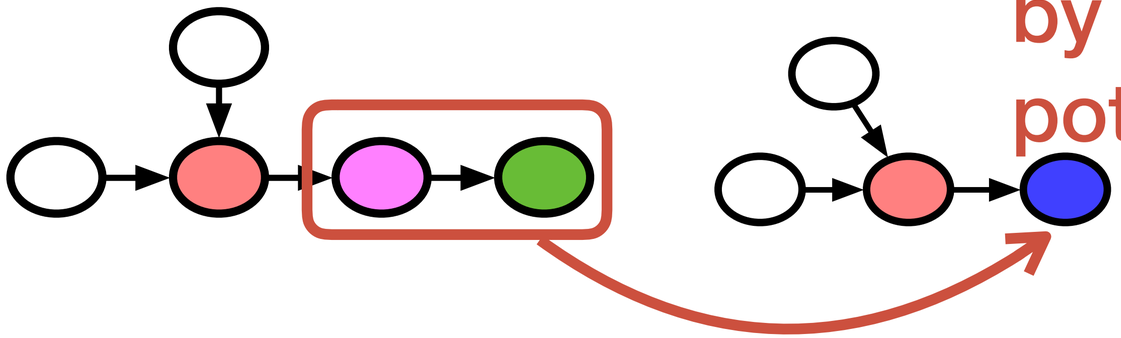


cuDNN



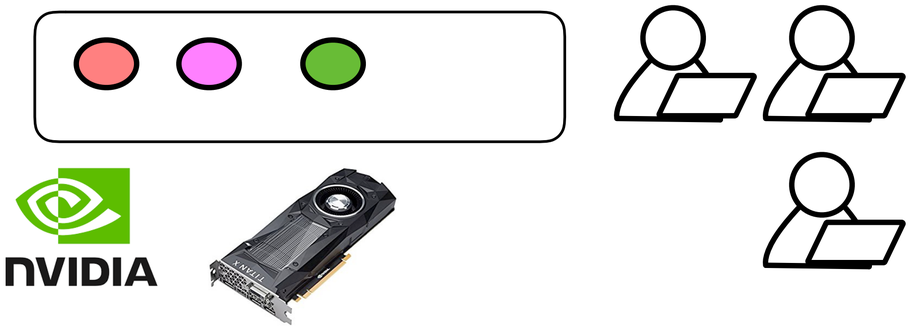
Limitations of Existing Approach

Frameworks



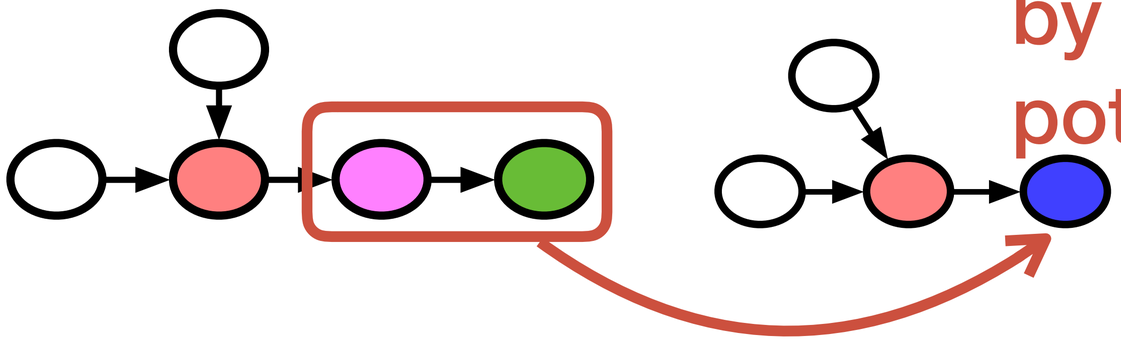
New operator introduced
by operator fusion optimization
potential benefit: 1.5x speedup

cuDNN



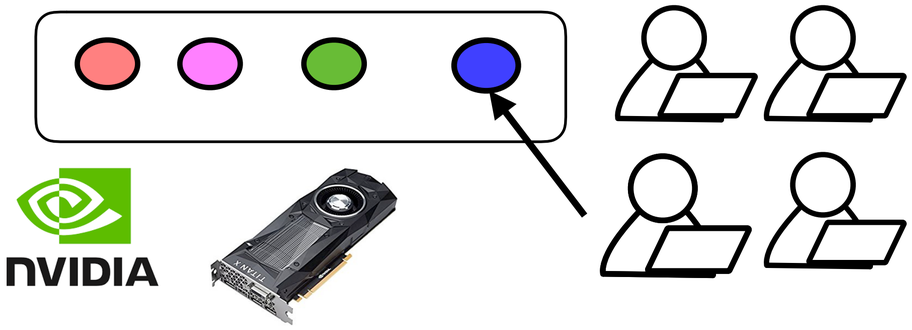
Limitations of Existing Approach

Frameworks



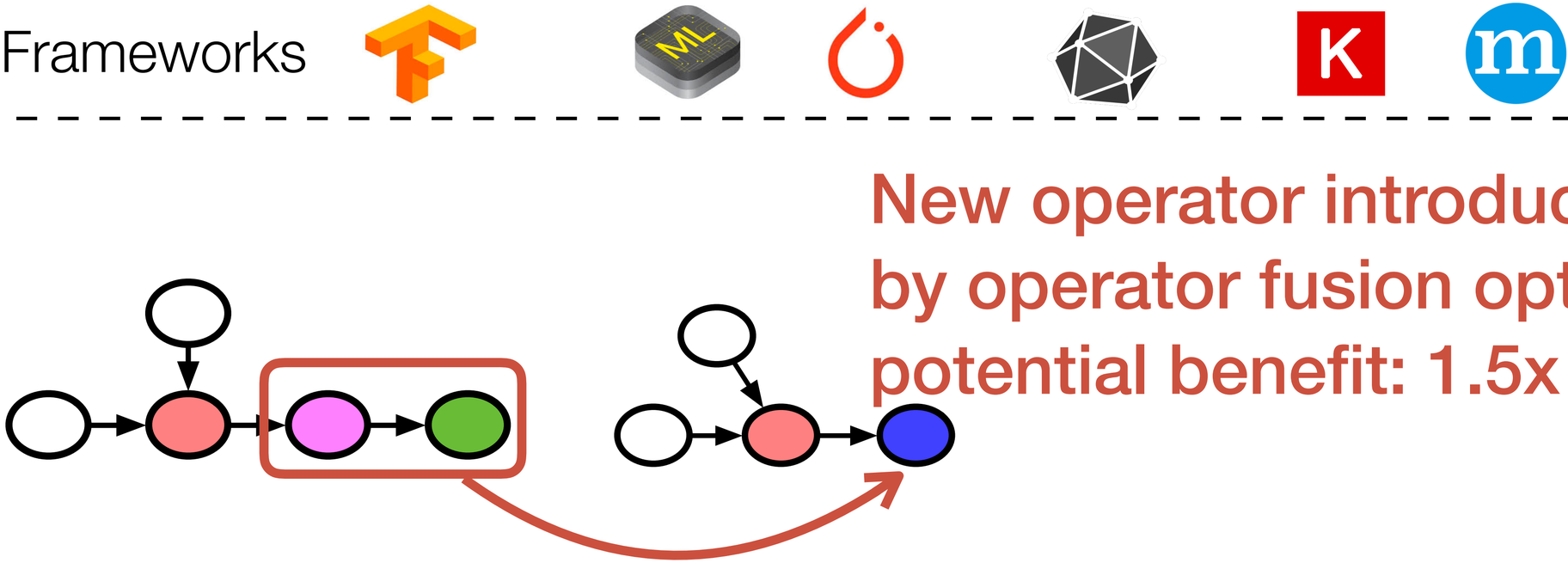
New operator introduced
by operator fusion optimization
potential benefit: 1.5x speedup

cuDNN



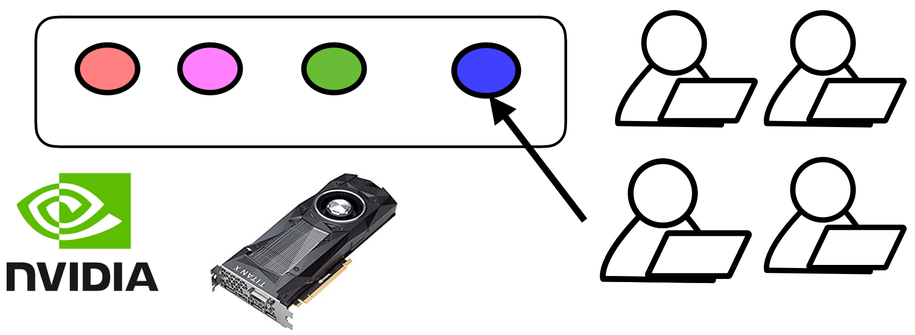
Limitations of Existing Approach

Frameworks



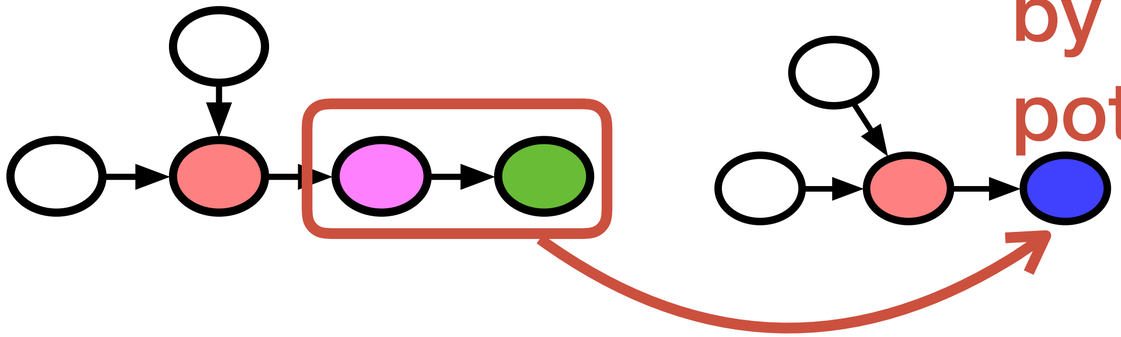
New operator introduced
by operator fusion optimization
potential benefit: 1.5x speedup

cuDNN



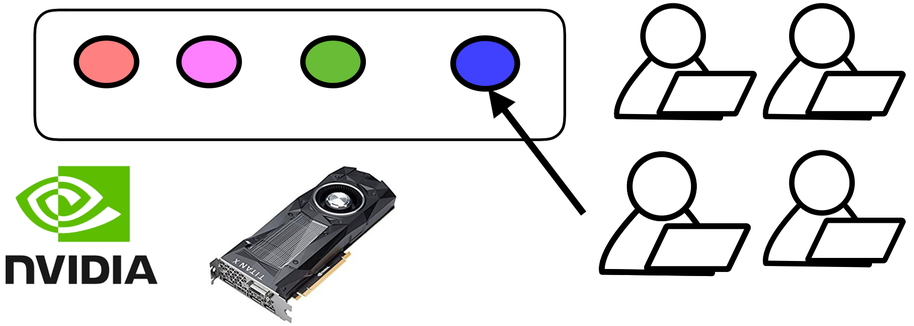
Limitations of Existing Approach

Frameworks



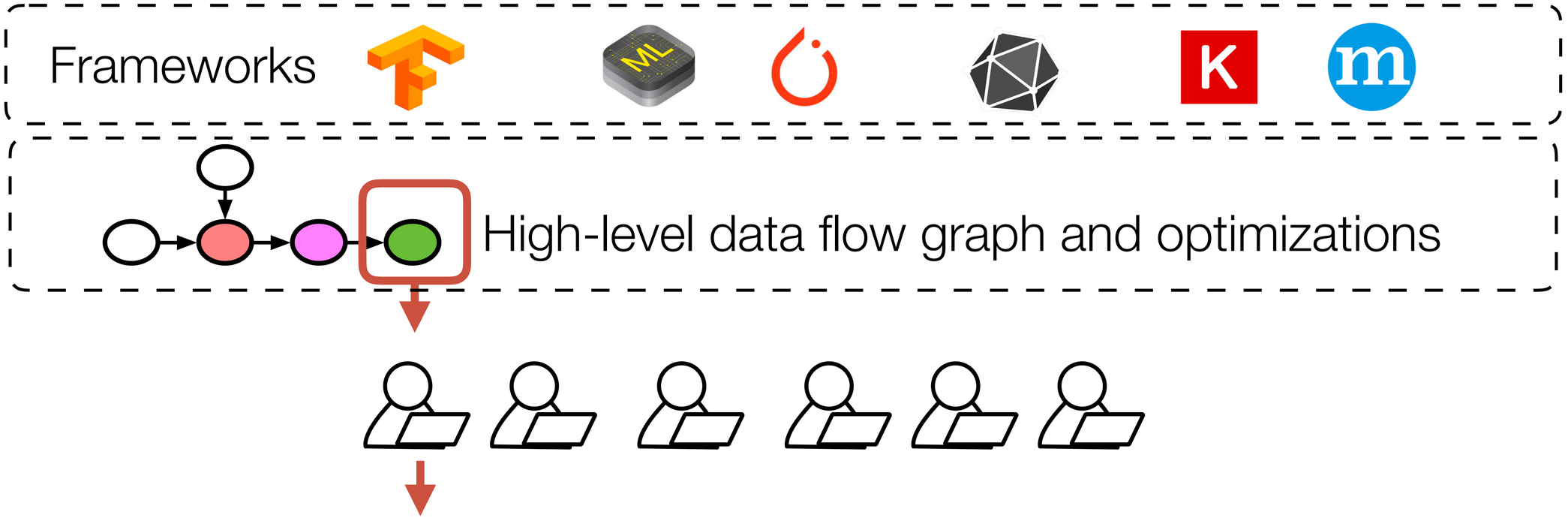
New operator introduced
by operator fusion optimization
potential benefit: 1.5x speedup

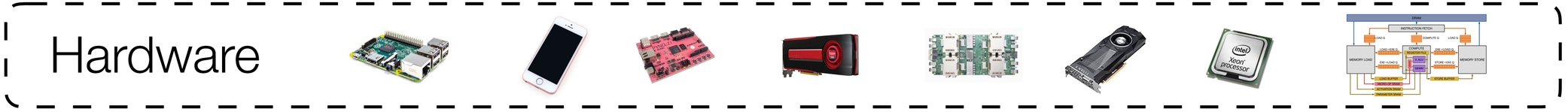
cuDNN



Engineering intensive

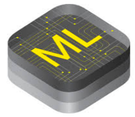
Learning-based Learning System



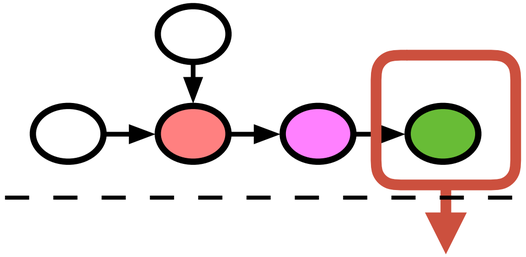


Learning-based Learning System

Frameworks





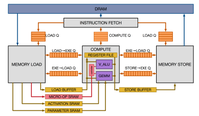
High-level data flow graph and optimizations

Hardware

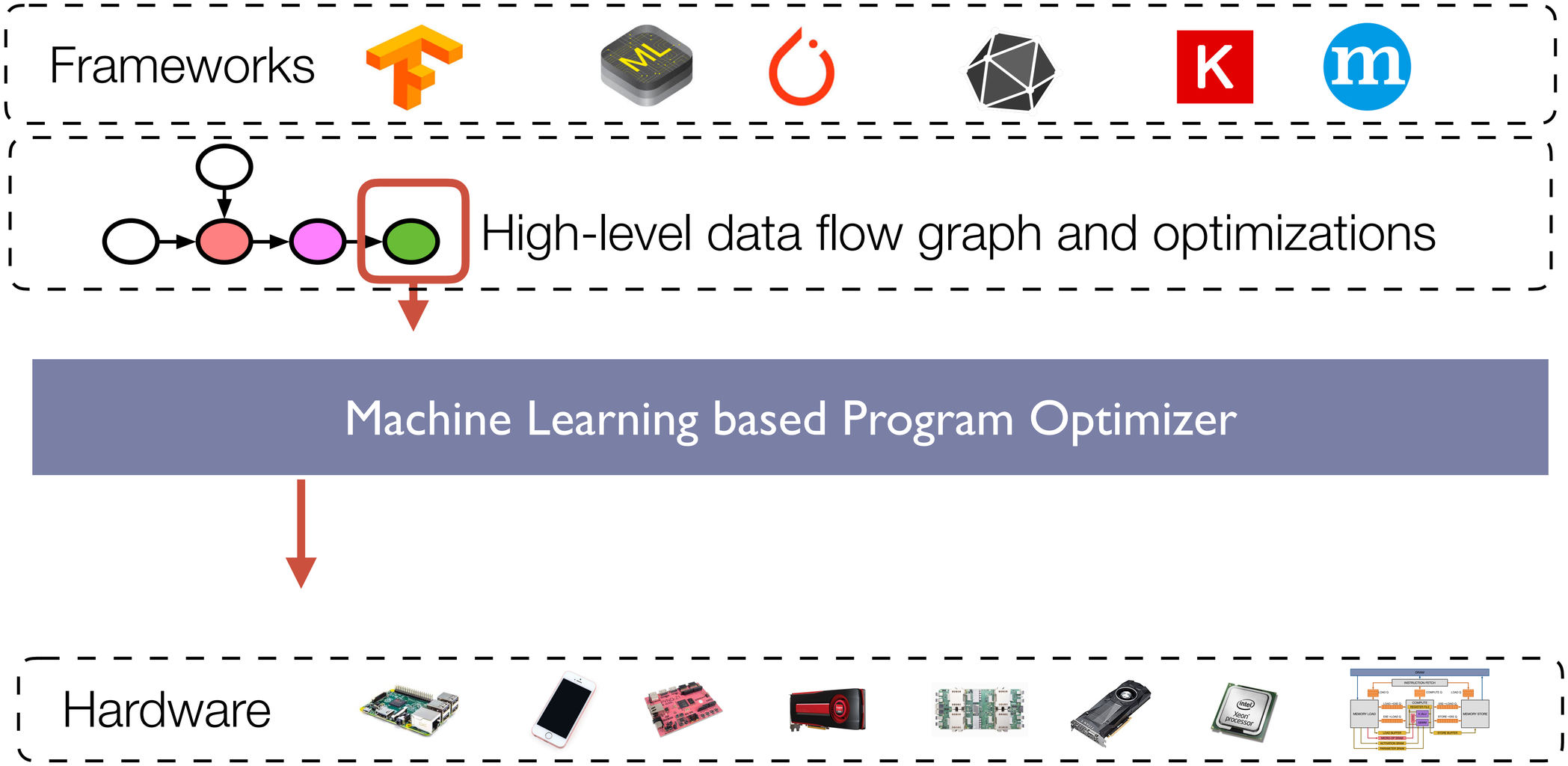




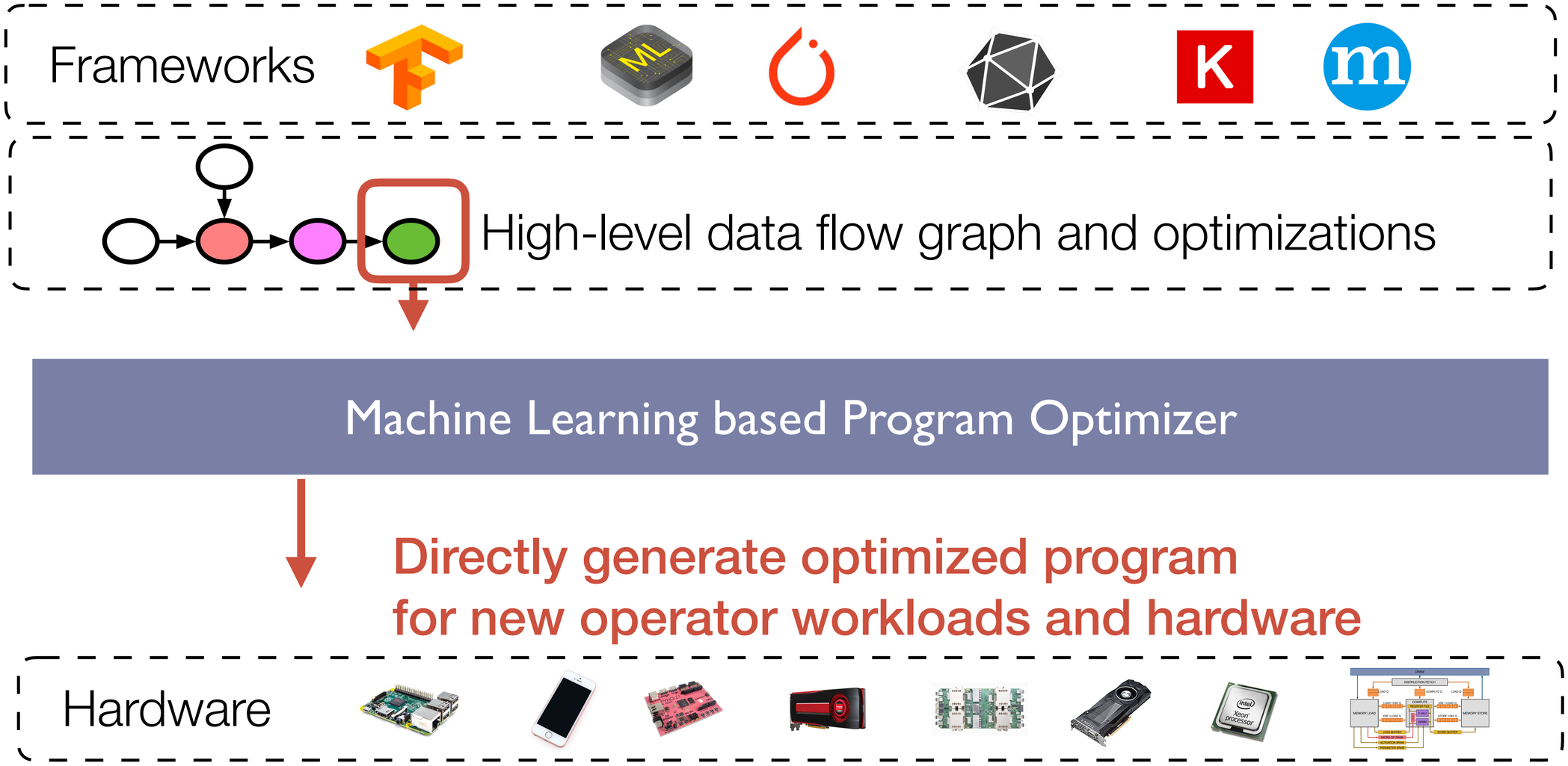




Learning-based Learning System



Learning-based Learning System



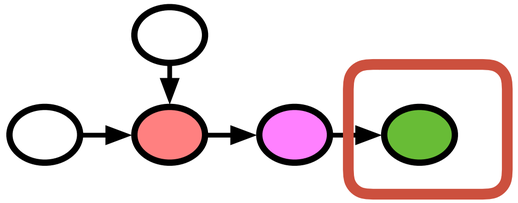
TVM: Learning-based Learning System

Why do we need machine learning for systems

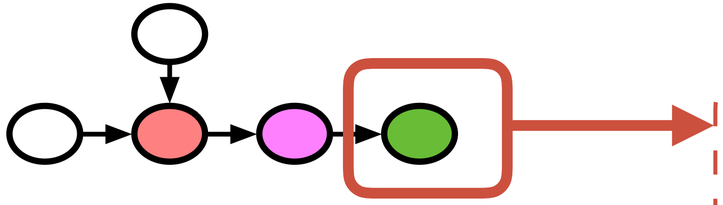
How to build intelligent systems with learning

End to end learning-based learning system stack

Problem Setting



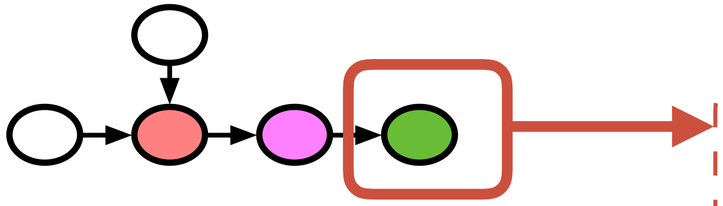
Problem Setting



Tensor Expression (Specification)

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

Problem Setting

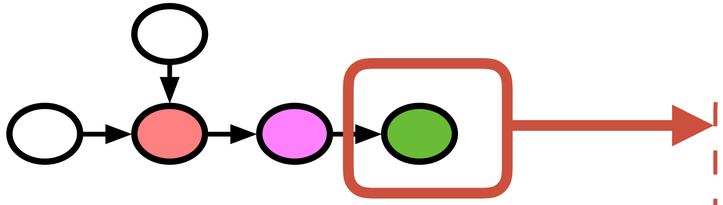


Tensor Expression (Specification)

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

Search Space of Possible Program Optimizations

Problem Setting



Tensor Expression (Specification)

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

Search Space of Possible Program Optimizations

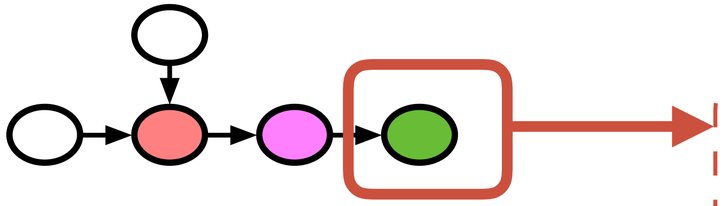
Low-level Program Variants

```
inp_buffer AL[8][8], BL[8][8]
acc_buffer CL[8][8]
for yo in range(128):
  for xo in range(128):
    vdla.fill_zero(CL)
    for ko in range(128):
      vdla.dma_copy2d(AL, A[ko*8:ko*8+8][yo*8:yo*8+8])
      vdla.dma_copy2d(BL, B[ko*8:ko*8+8][xo*8:xo*8+8])
      vdla.fused_gemm8x8_add(CL, AL, BL)
    vdla.dma_copy2d(C[yo*8:yo*8+8,xo*8:xo*8+8], CL)
```

```
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 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]
```

Example Instance in a Search Space

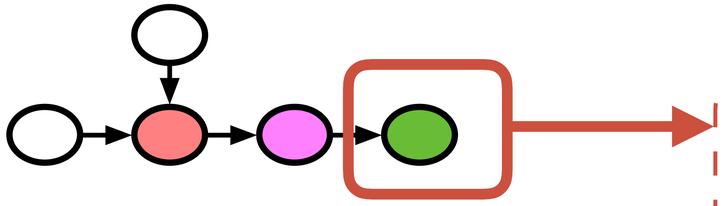


Tensor Expression (Specification)

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

Search Space of Possible Program Optimizations

Example Instance in a Search Space



Tensor Expression (Specification)

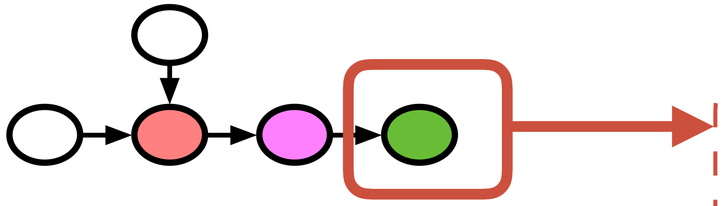
```
C = tvm.compute((m, n),
    lambda y, x: tvm.sum(A[k, y] * B[k, x], axis=k))
```

Search Space of Possible Program Optimizations

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]
```

Example Instance in a Search Space



Tensor Expression (Specification)

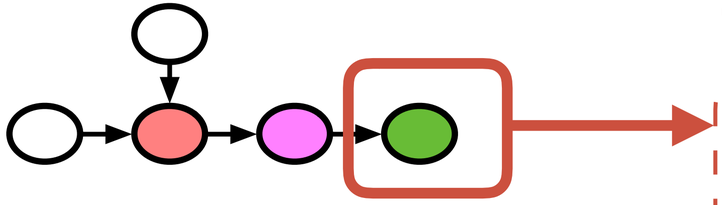
```
C = tvm.compute((m, n),
    lambda y, x: tvm.sum(A[k, y] * B[k, x], axis=k))
```

Search Space of Possible Program Optimizations

Loop Tiling for Locality

```
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]
```

Example Instance in a Search Space



Tensor Expression (Specification)

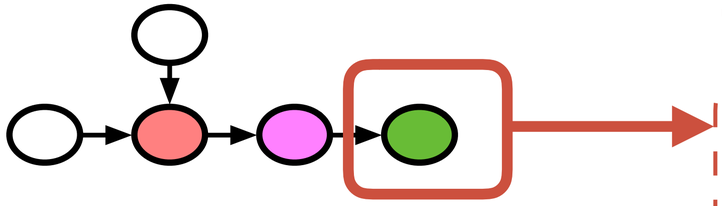
```
C = tvm.compute((m, n),
    lambda y, x: tvm.sum(A[k, y] * B[k, x], axis=k))
```

Search Space of Possible Program Optimizations

Map to Accelerators

```
inp_buffer AL[8][8], BL[8][8]
acc_buffer CL[8][8]
for yo in range(128):
  for xo in range(128):
    vdla.fill_zero(CL)
    for ko in range(128):
      vdla.dma_copy2d(AL, A[ko*8:ko*8+8][yo*8:yo*8+8])
      vdla.dma_copy2d(BL, B[ko*8:ko*8+8][xo*8:xo*8+8])
      vdla.fused_gemm8x8_add(CL, AL, BL)
    vdla.dma_copy2d(C[yo*8:yo*8+8,xo*8:xo*8+8], CL)
```

Optimization Choices in a Search Space



Tensor Expression (Specification)

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

Loop Transformations

Thread Bindings

Cache Locality

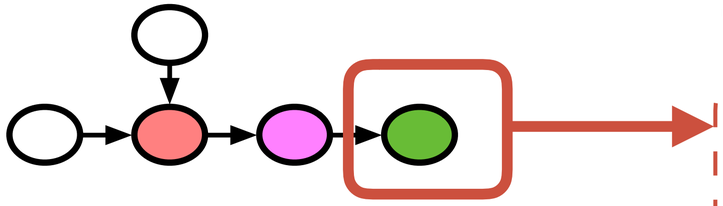
Thread Cooperation

Tensorization

Latency Hiding

More details about the search space
in the second half of the talk

Optimization Choices in a Search Space



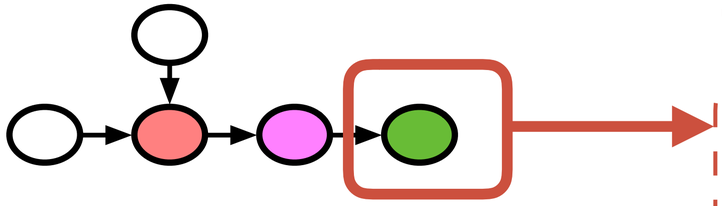
Tensor Expression (Specification)

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

- Loop Transformations
- Thread Bindings
- Cache Locality
- Thread Cooperation
- Tensorization
- Latency Hiding

More details about the search space in the second half of the talk

Optimization Choices in a Search Space



Tensor Expression (Specification)

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

Billions of possible optimization choices

- Loop Transformations
- Thread Bindings
- Cache Locality
- Thread Cooperation
- Tensorization
- Latency Hiding

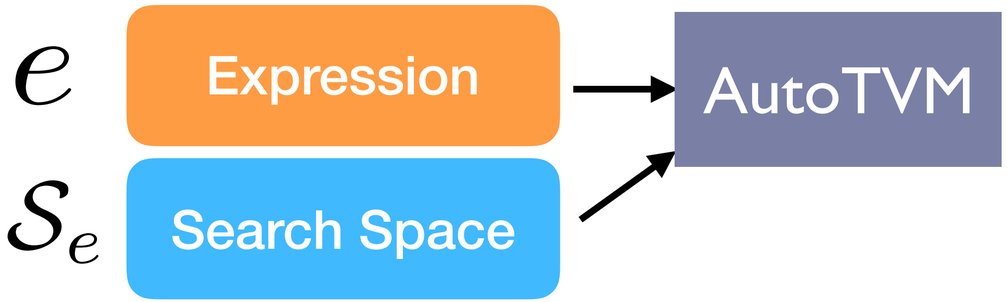
More details about the search space in the second half of the talk

Problem Formalization

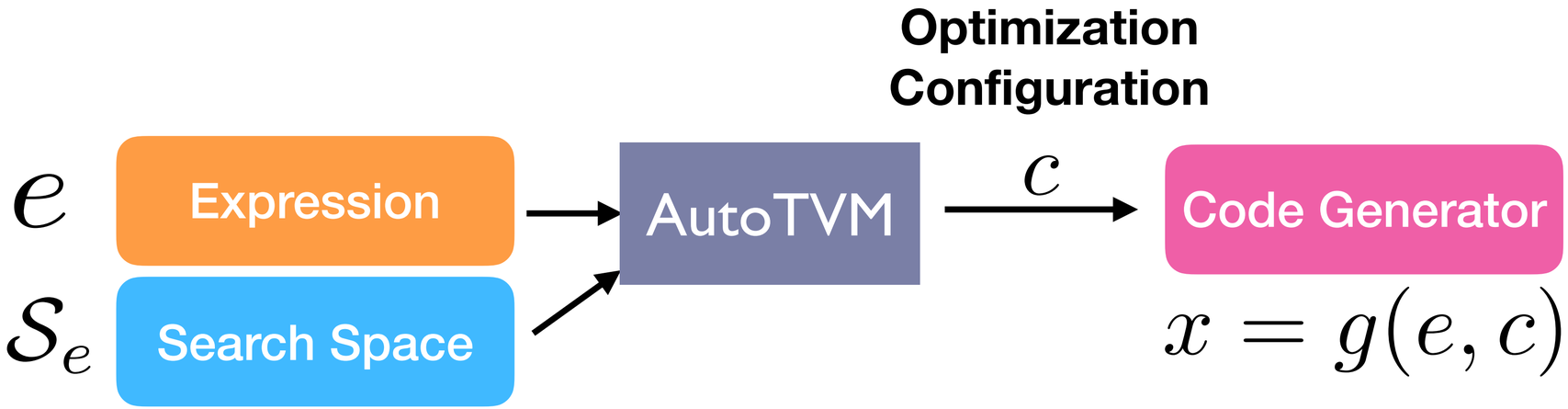
e Expression

$\mathcal{S}_e$ Search Space

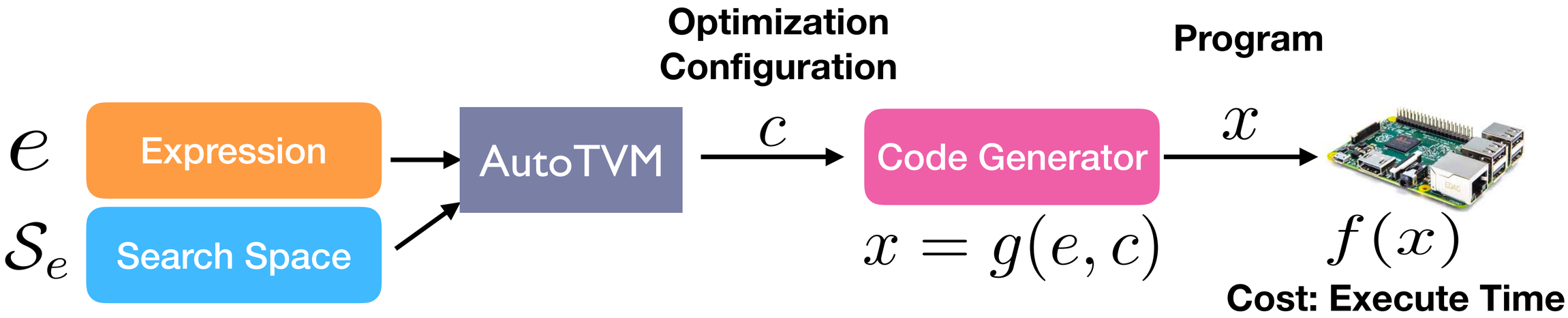
Problem Formalization



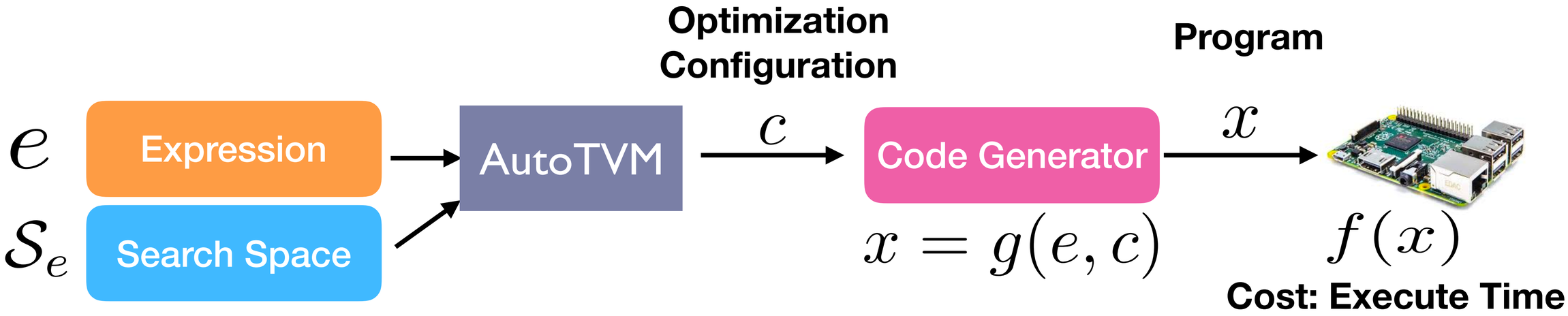
Problem Formalization



Problem Formalization



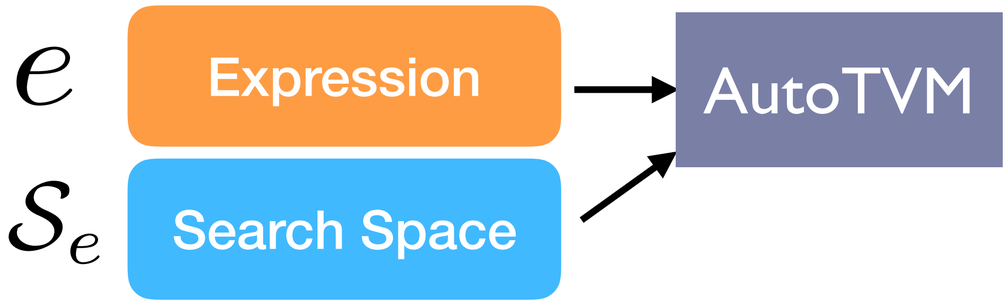
Problem Formalization



Objective $argmin_{c \in S_e} f(g(e, c))$

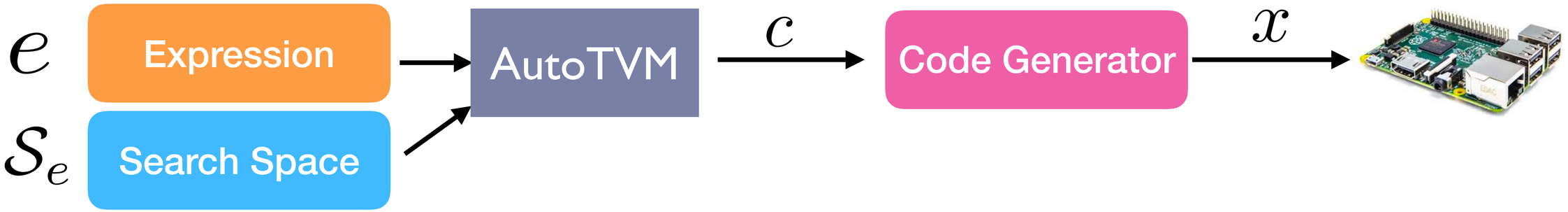
Black-box Optimization

Try each configuration c until we find a good one



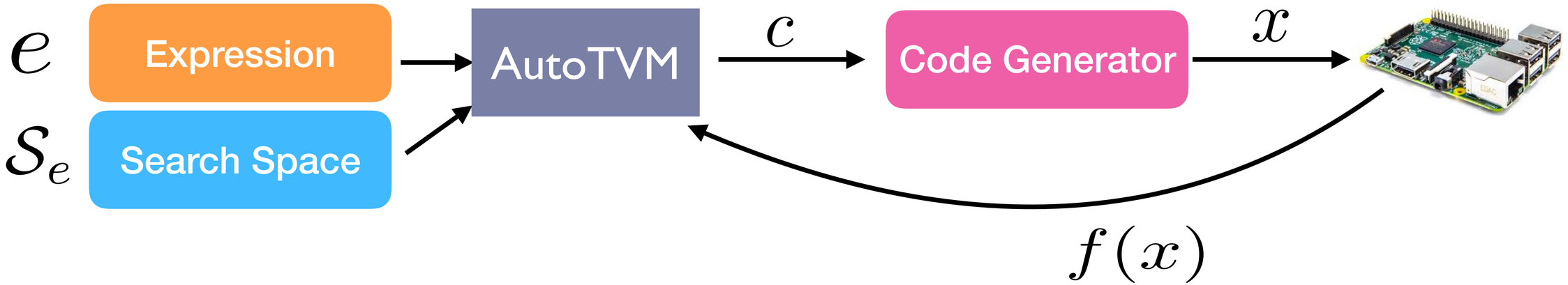
Black-box Optimization

Try each configuration c until we find a good one



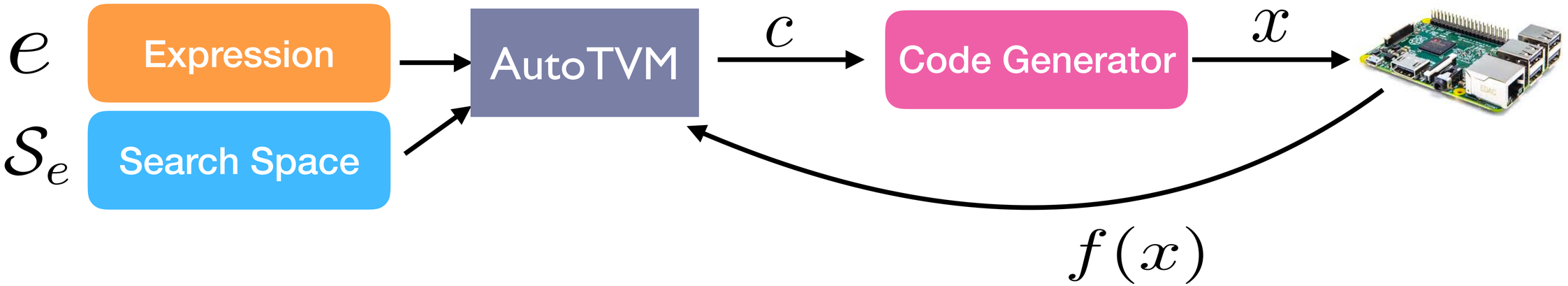
Black-box Optimization

Try each configuration c until we find a good one



Black-box Optimization

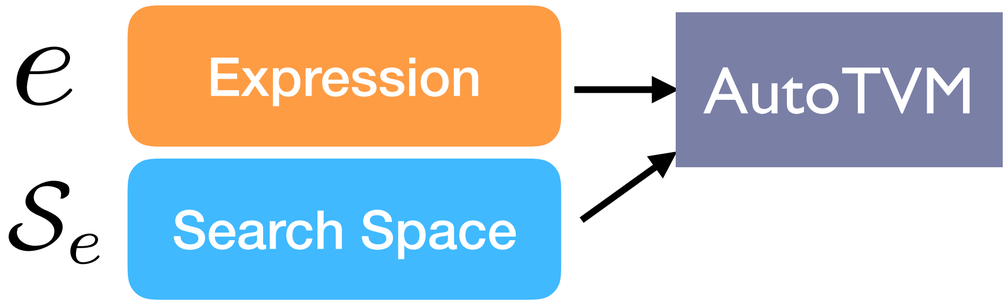
Try each configuration c until we find a good one



Challenge: lots of experimental trials, each trial costs ~1 second

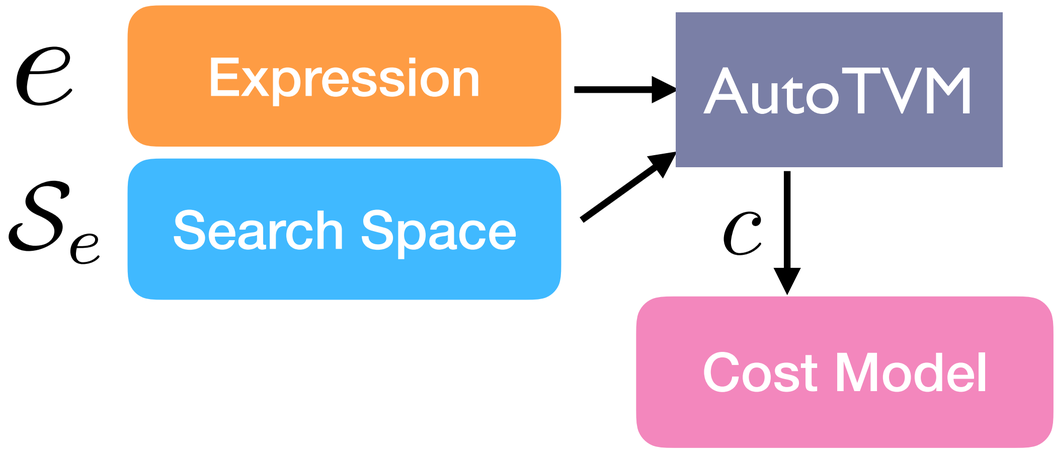
Cost-model Driven Approach

Use cost model to pick configuration



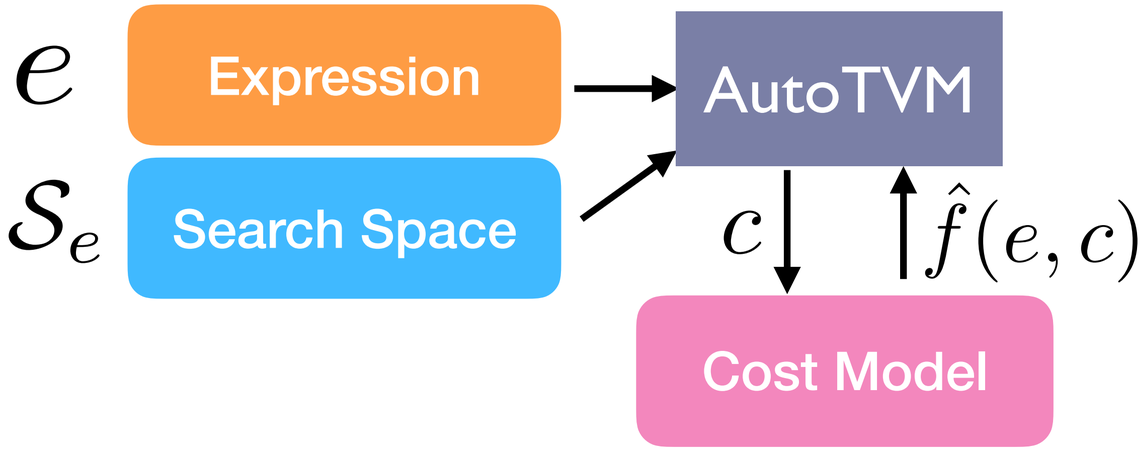
Cost-model Driven Approach

Use cost model to pick configuration



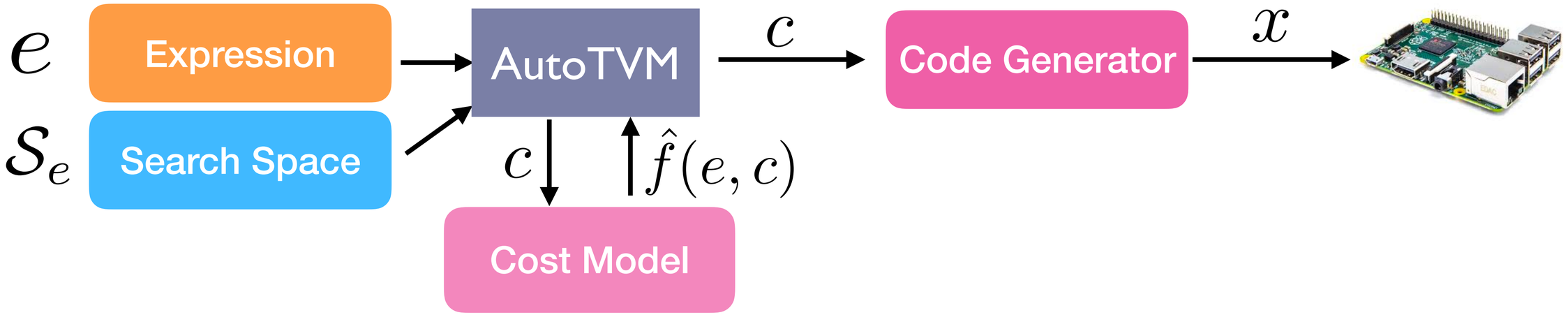
Cost-model Driven Approach

Use cost model to pick configuration



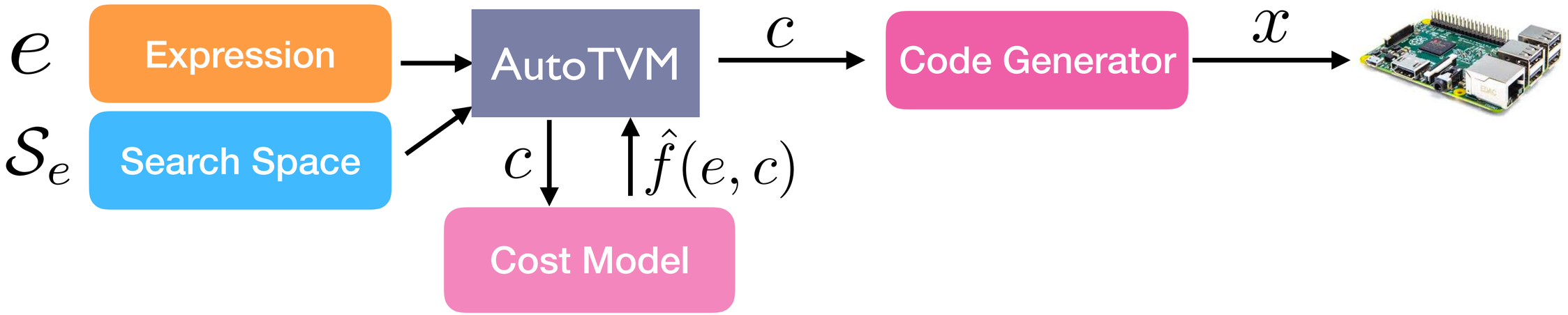
Cost-model Driven Approach

Use cost model to pick configuration



Cost-model Driven Approach

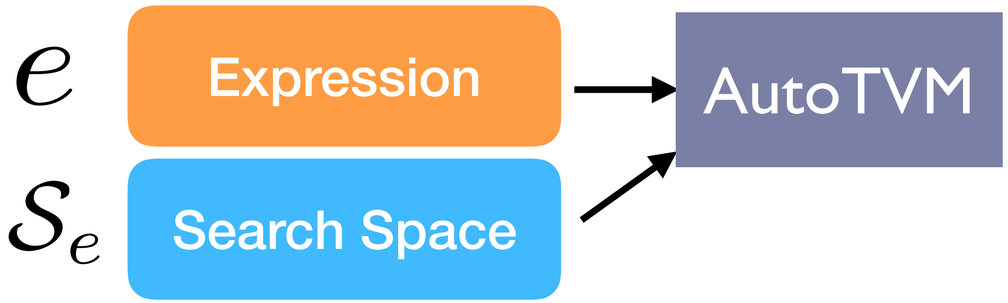
Use cost model to pick configuration



Challenge: Need reliable cost model per hardware

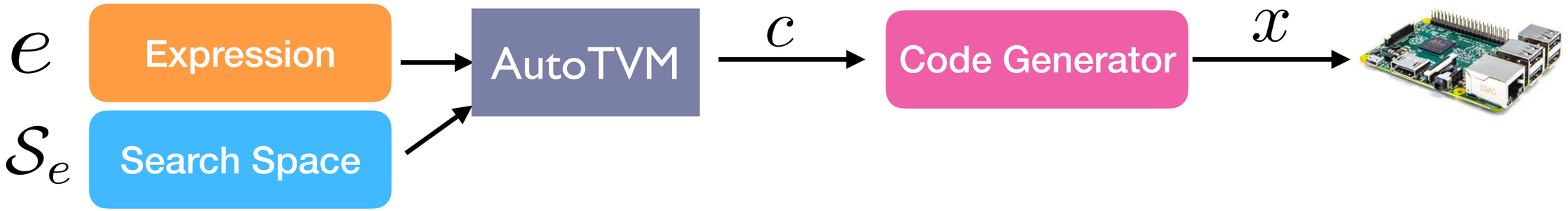
Statistical Cost Model

Use machine learning to learn a statistical cost model



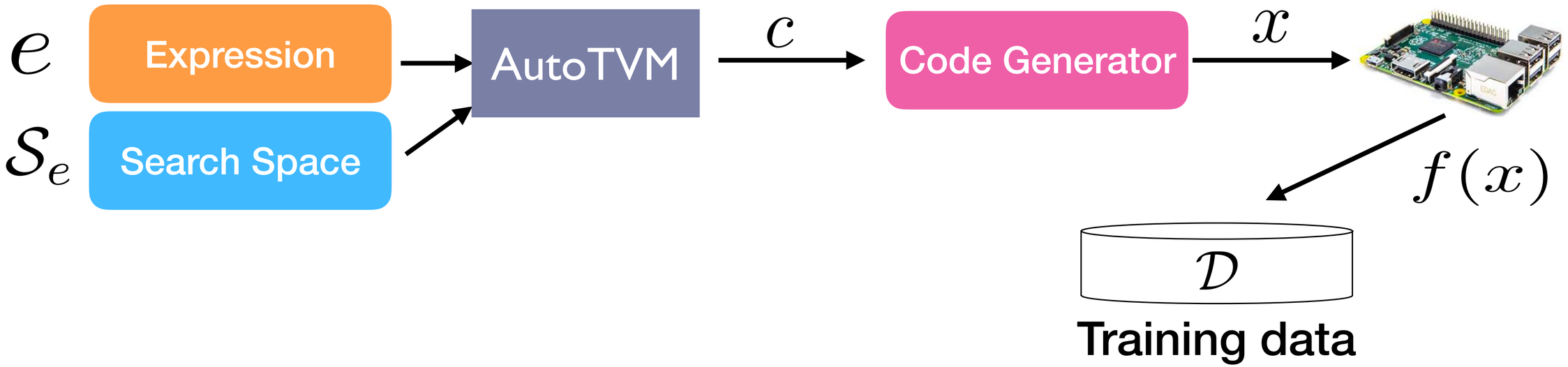
Statistical Cost Model

Use machine learning to learn a statistical cost model



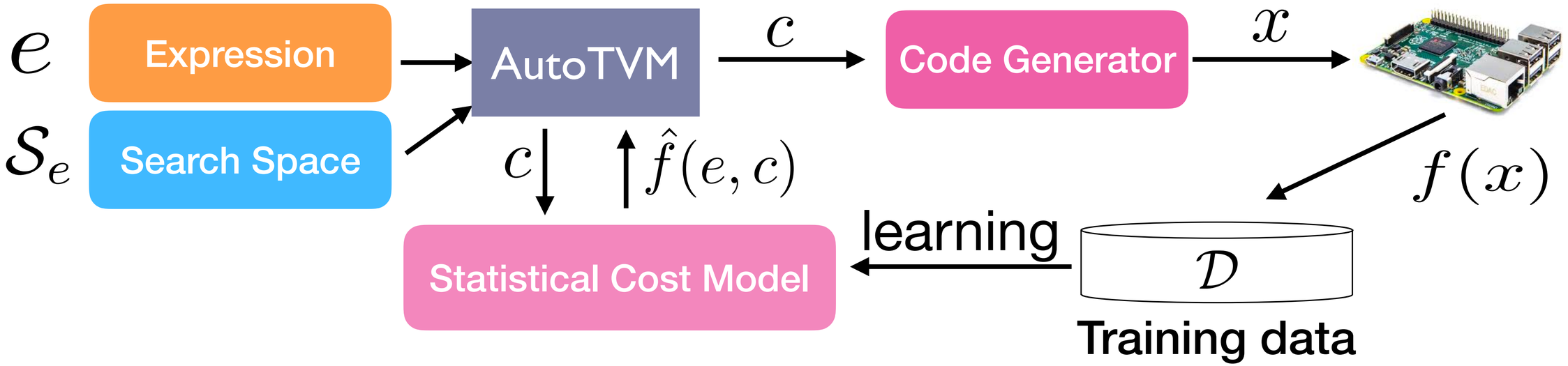
Statistical Cost Model

Use machine learning to learn a statistical cost model



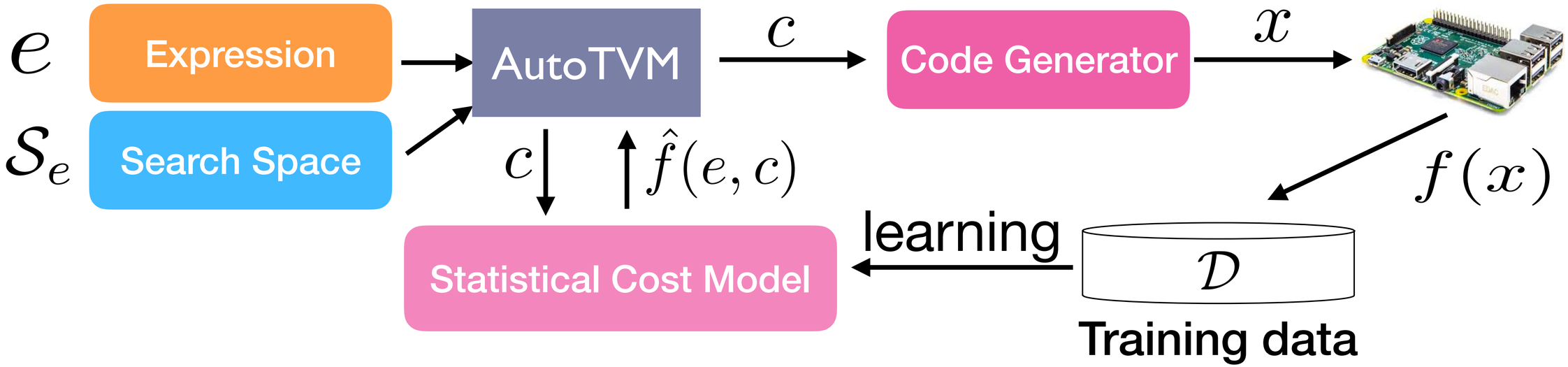
Statistical Cost Model

Use machine learning to learn a statistical cost model



Statistical Cost Model

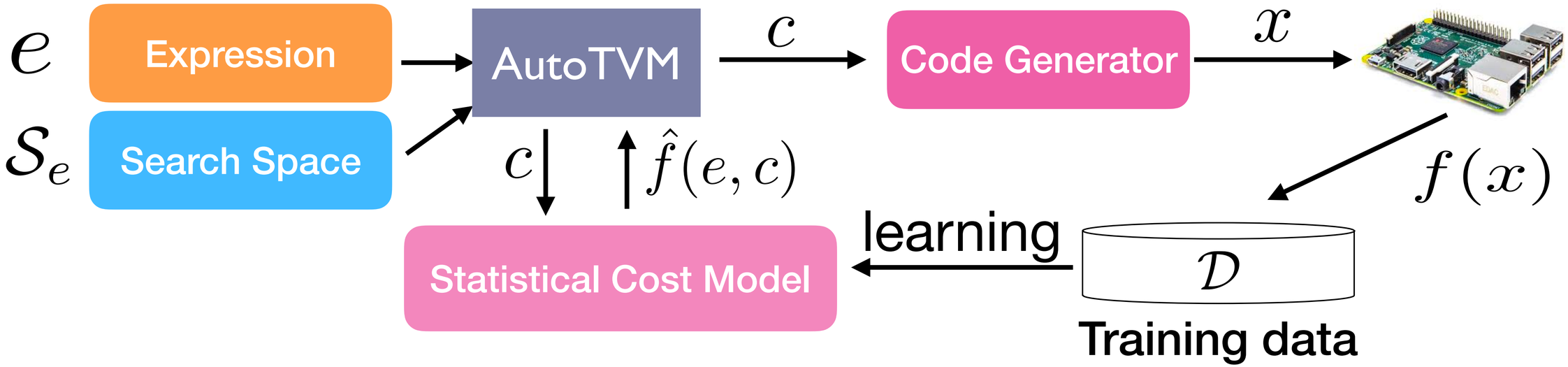
Use machine learning to learn a statistical cost model



Benefit: Automatically adapt to hardware type

Statistical Cost Model

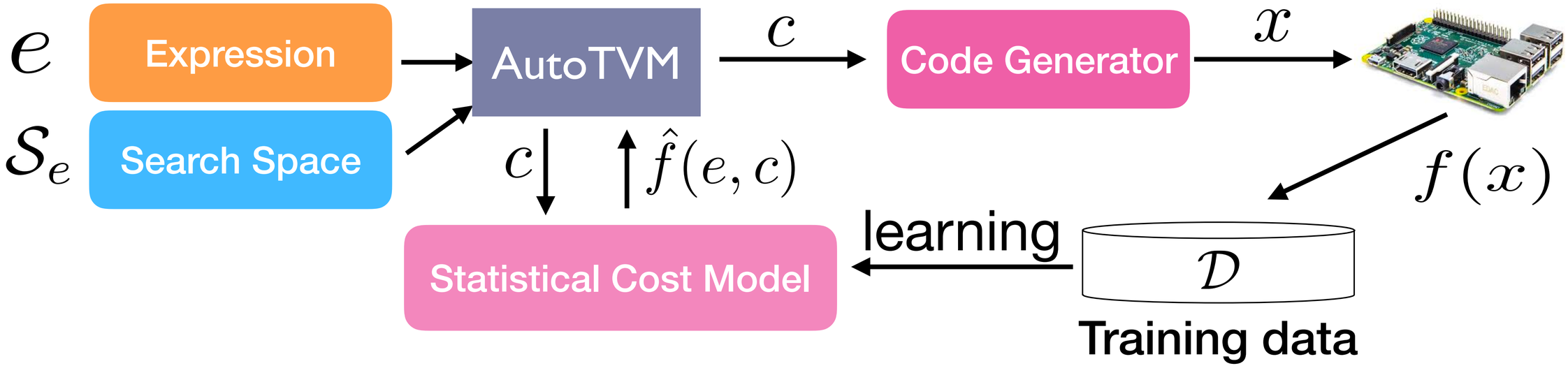
Use machine learning to learn a statistical cost model



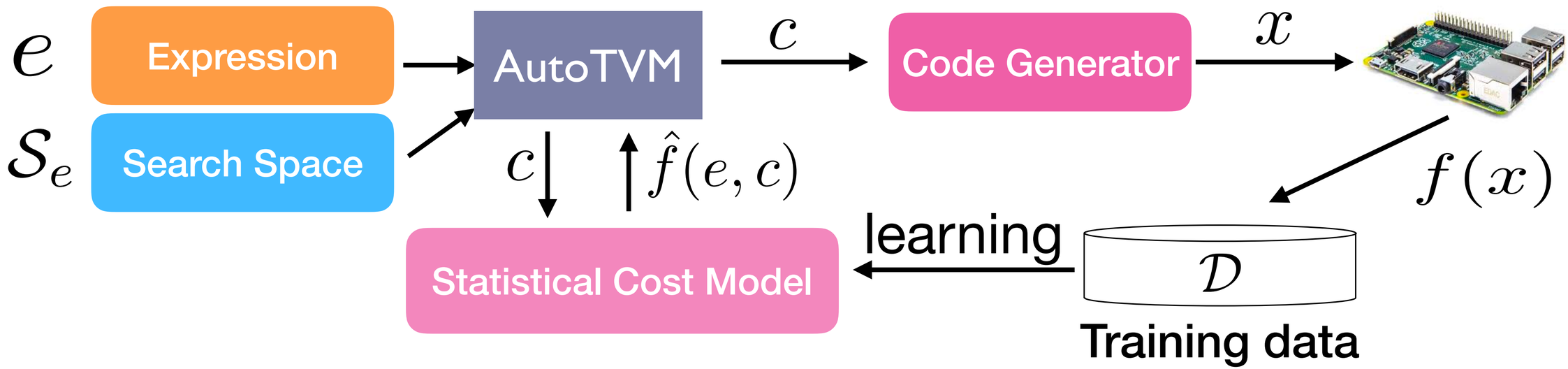
Benefit: Automatically adapt to hardware type

Challenge: How to design the cost model

Unique Problem Characteristics

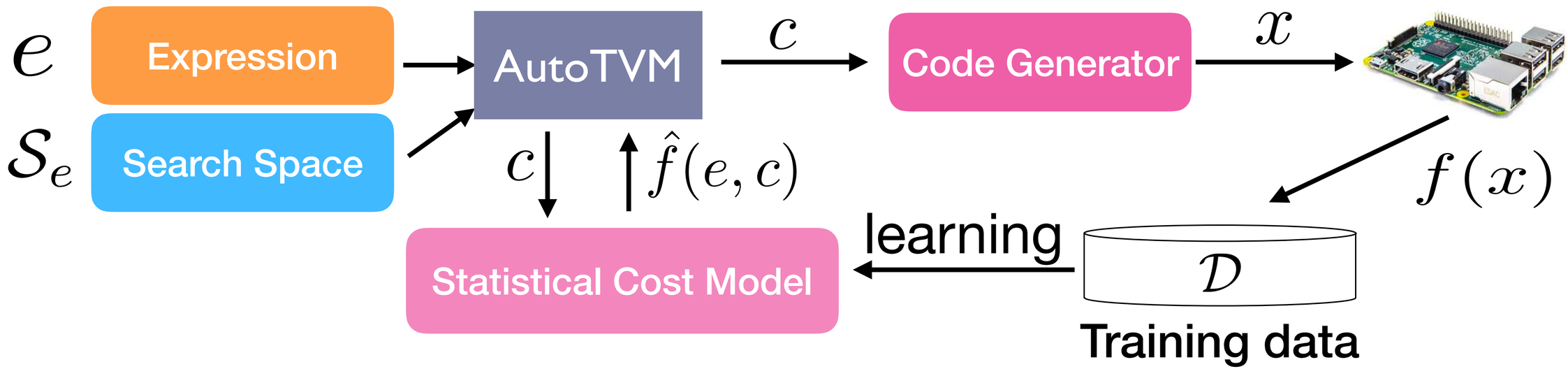


Unique Problem Characteristics



Relatively low experiment cost

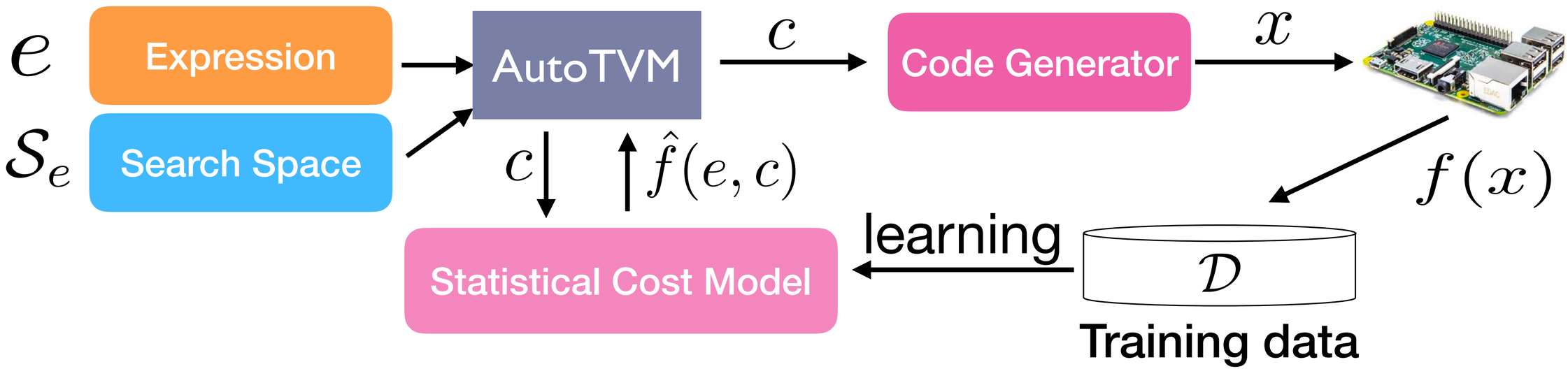
Unique Problem Characteristics



Relatively low experiment cost

Program-aware modeling

Unique Problem Characteristics

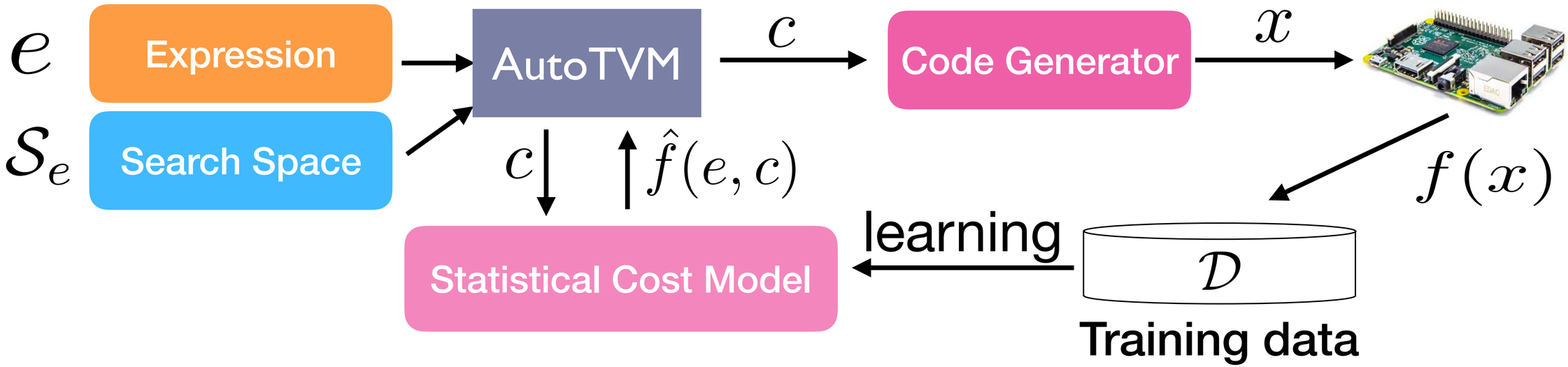


Relatively low experiment cost

Program-aware modeling

Large number of similar tasks

Unique Problem Characteristics



Relatively low experiment cost

Program-aware modeling

Large number of similar tasks

Vanilla Cost Modeling

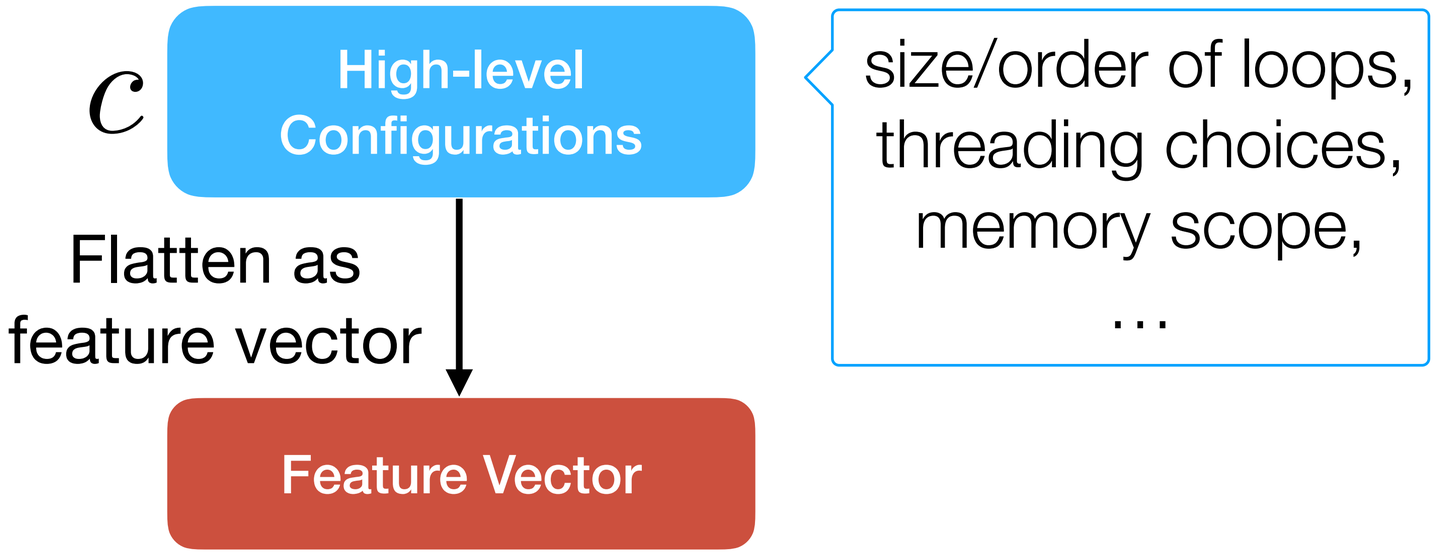
c **High-level Configurations**

Vanilla Cost Modeling

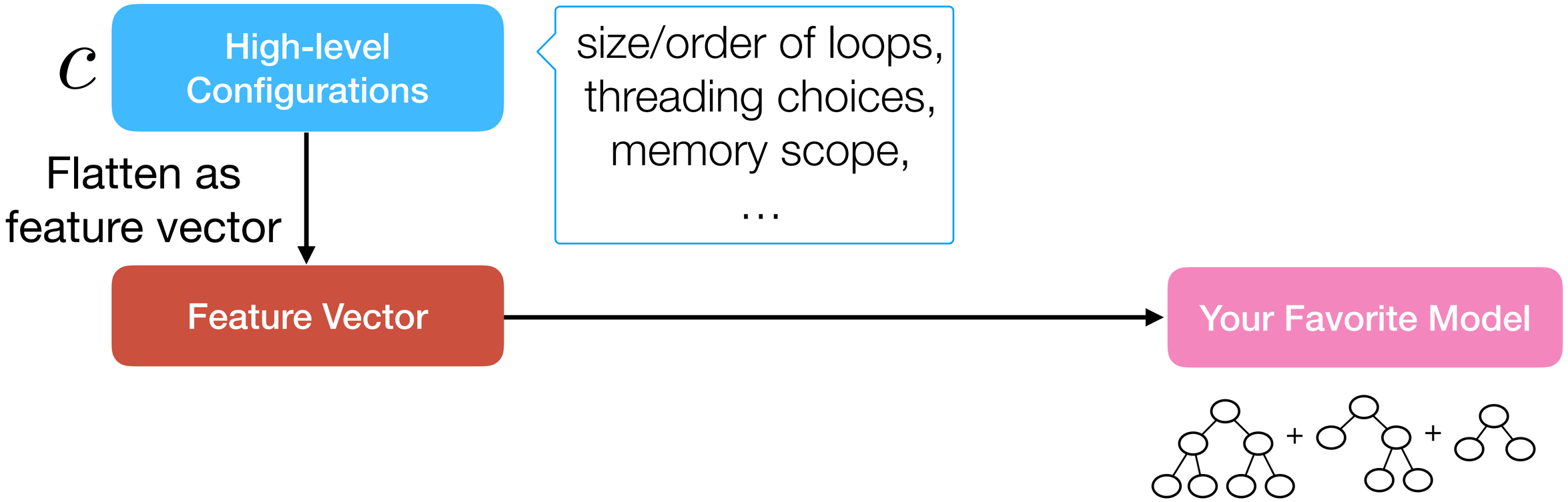
c **High-level Configurations**

size/order of loops,
threading choices,
memory scope,
…

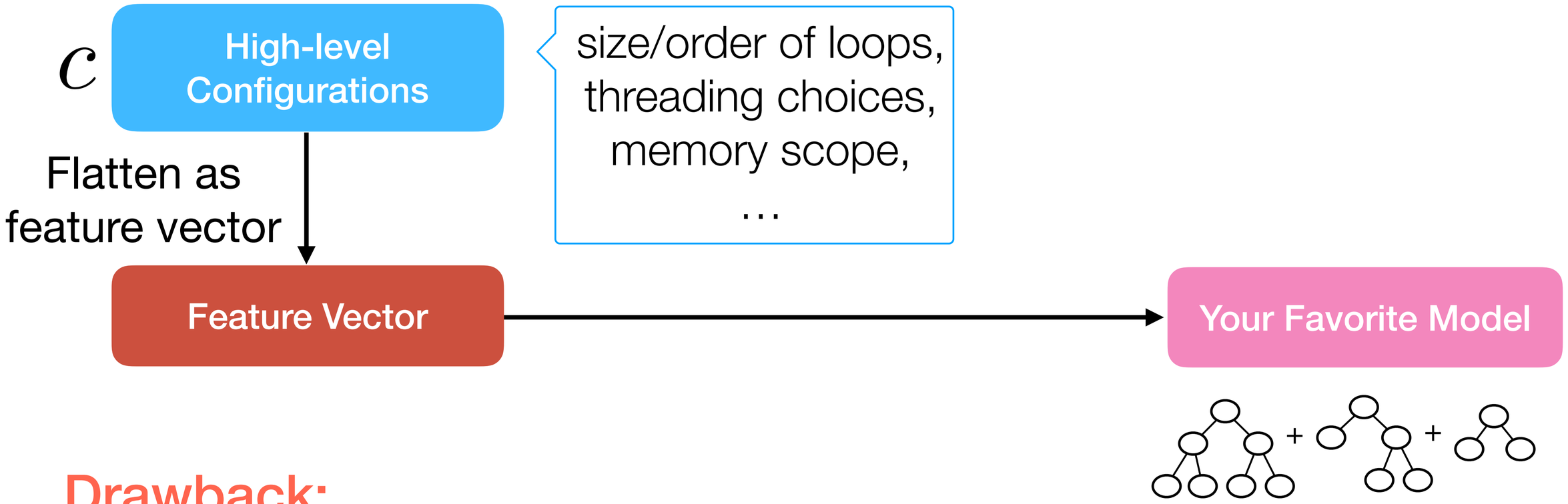
Vanilla Cost Modeling



Vanilla Cost Modeling



Vanilla Cost Modeling



Drawback:

Ignores domain knowledge

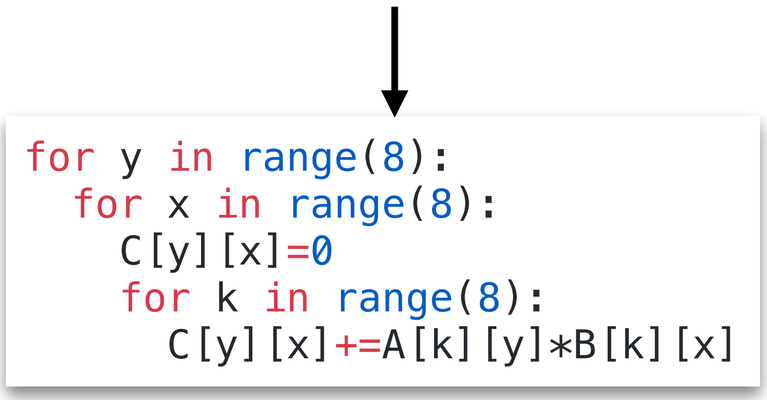
Set of configurations can differ per task (task dependent)

Program-aware Modeling: Tree-based Approach

c **High-level Configurations**

Program-aware Modeling: Tree-based Approach

c **High-level Configurations**

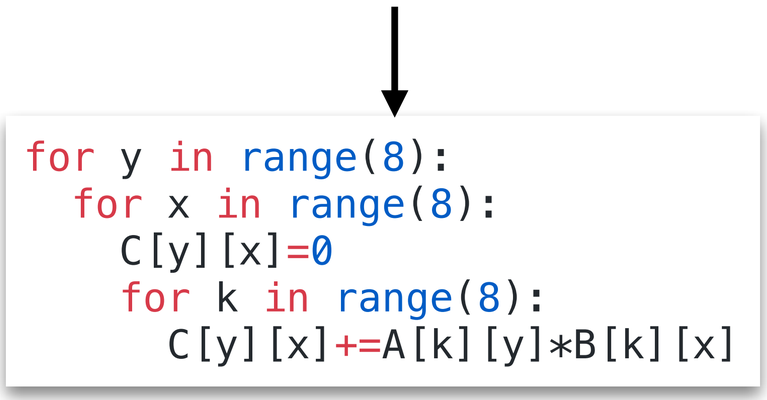


```
for y in range(8):
  for x in range(8):
    C[y][x]=0
    for k in range(8):
      C[y][x]+=A[k][y]*B[k][x]
```

Low-level
Abstract Syntax Tree (AST)

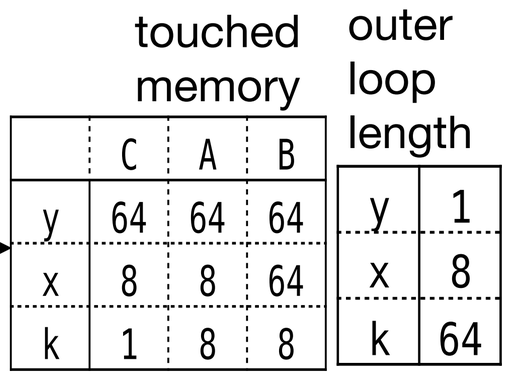
Program-aware Modeling: Tree-based Approach

c **High-level Configurations**



```
for y in range(8):
  for x in range(8):
    C[y][x]=0
    for k in range(8):
      C[y][x]+=A[k][y]*B[k][x]
```

Low-level
Abstract Syntax Tree (AST)



	C	A	B
y	64	64	64
x	8	8	64
k	1	8	8

touched memory

y	1
x	8
k	64

outer loop length

Statistical features
of AST

Program-aware Modeling: Tree-based Approach

c **High-level Configurations**

```
for y in range(8):
  for x in range(8):
    C[y][x]=0
    for k in range(8):
      C[y][x]+=A[k][y]*B[k][x]
```

Low-level
Abstract Syntax Tree (AST)

touched memory

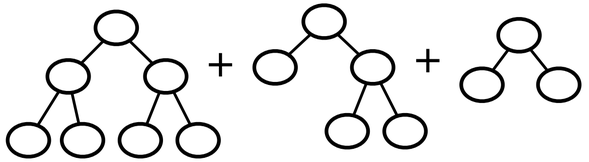
	C	A	B
y	64	64	64
x	8	8	64
k	1	8	8

outer loop length

y	1
x	8
k	64

Statistical features of AST

Your Favorite Model



Program-aware Modeling: Tree-based Approach

c **High-level Configurations**

```
for y in range(8):
  for x in range(8):
    C[y][x]=0
    for k in range(8):
      C[y][x]+=A[k][y]*B[k][x]
```

Low-level Abstract Syntax Tree (AST)

touched memory

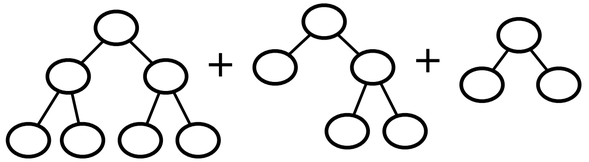
	C	A	B
y	64	64	64
x	8	8	64
k	1	8	8

outer loop length

y	1
x	8
k	64

Statistical features of AST

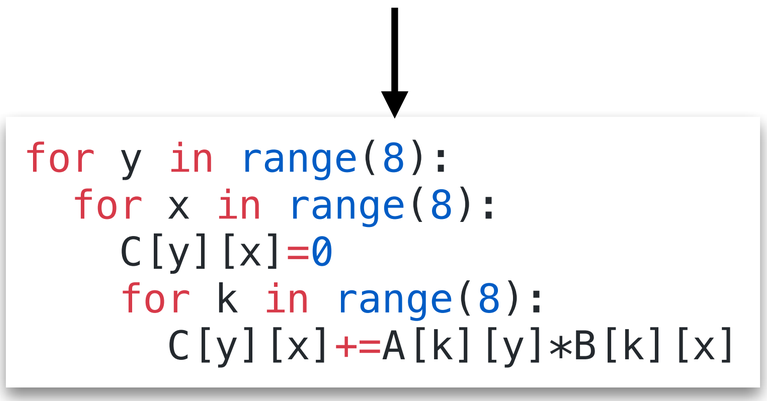
Your Favorite Model



Benefit: low-level AST is a common representation (task invariant)

Program-aware Modeling: Neural Approach

c **High-level Configurations**



```
for y in range(8):
  for x in range(8):
    C[y][x]=0
    for k in range(8):
      C[y][x]+=A[k][y]*B[k][x]
```

Low-level
Abstract Syntax Tree (AST)

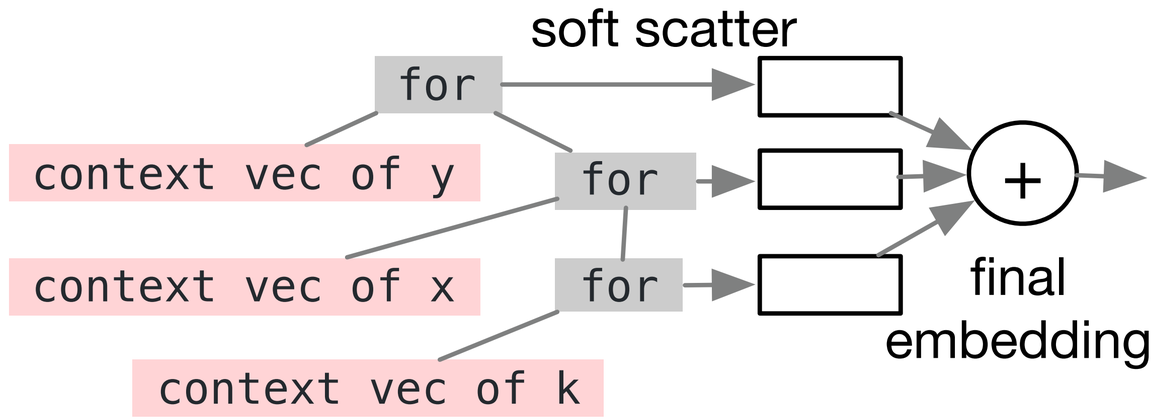
Program-aware Modeling: Neural Approach

c **High-level Configurations**

```
for y in range(8):
  for x in range(8):
    C[y][x]=0
    for k in range(8):
      C[y][x]+=A[k][y]*B[k][x]
```

Low-level Abstract Syntax Tree (AST)

TreeGRU on AST



Comparisons of Models

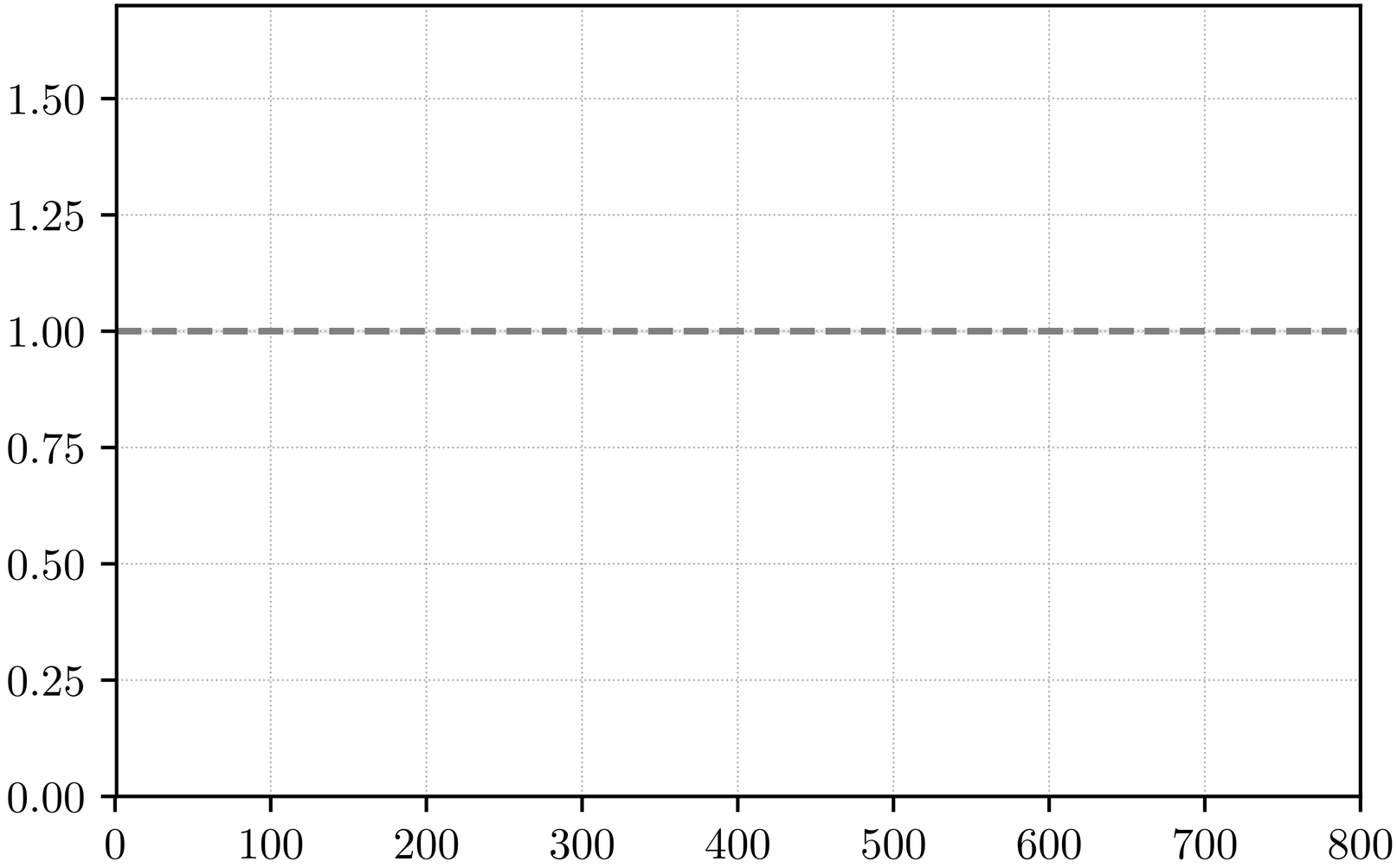
	Task Invariant	Time Cost	Predictive Accuracy
Vanilla Model	No	Low	Medium
Tree-based Model	Yes	Low	Good
Neural Model	Yes	High	Good

Comparisons of Models

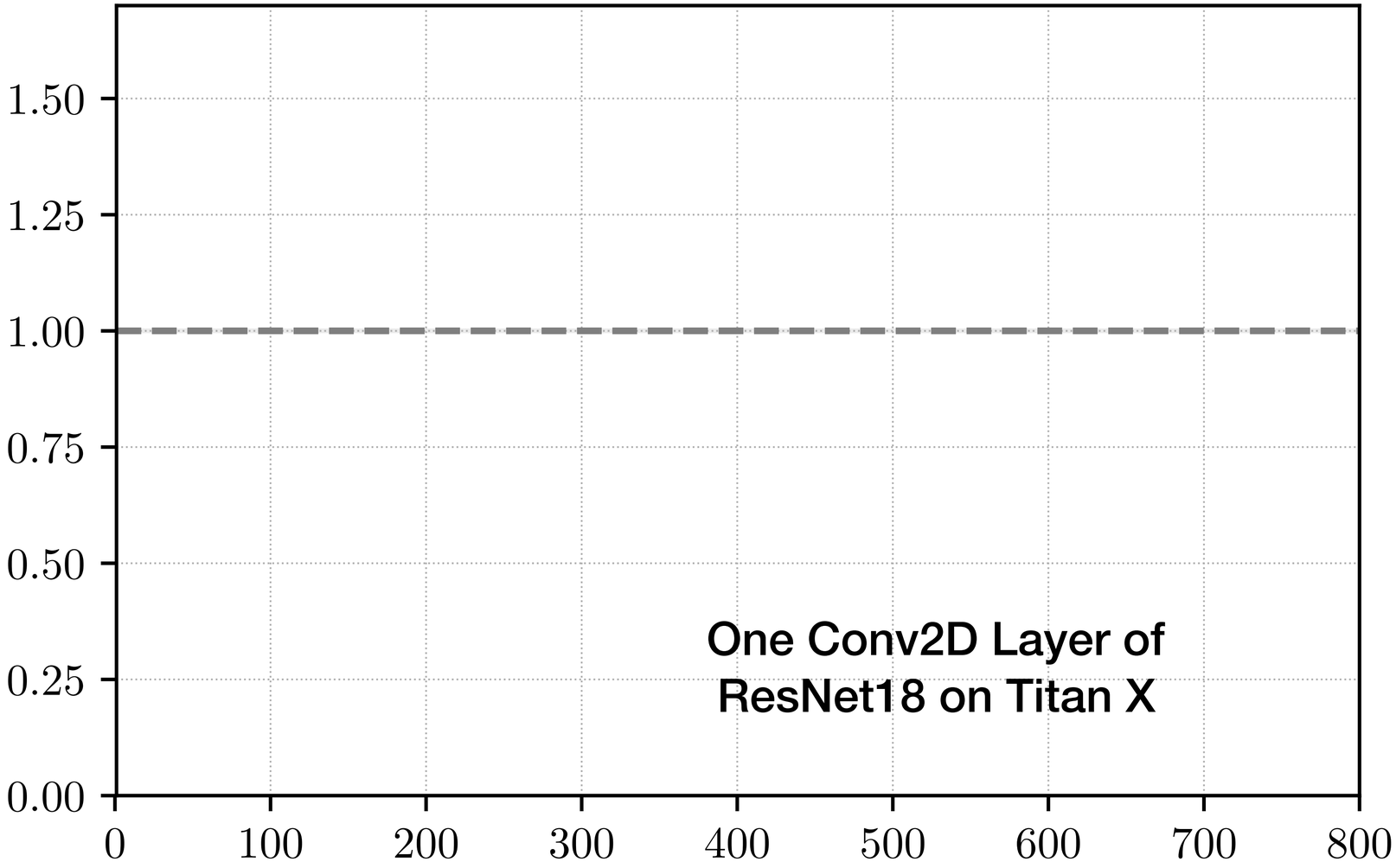
	Task Invariant	Time Cost	Predictive Accuracy
Vanilla Model	No	Low	Medium
Tree-based Model	Yes	Low	Good
Neural Model	Yes	High	Good

Choose tree-based model by default

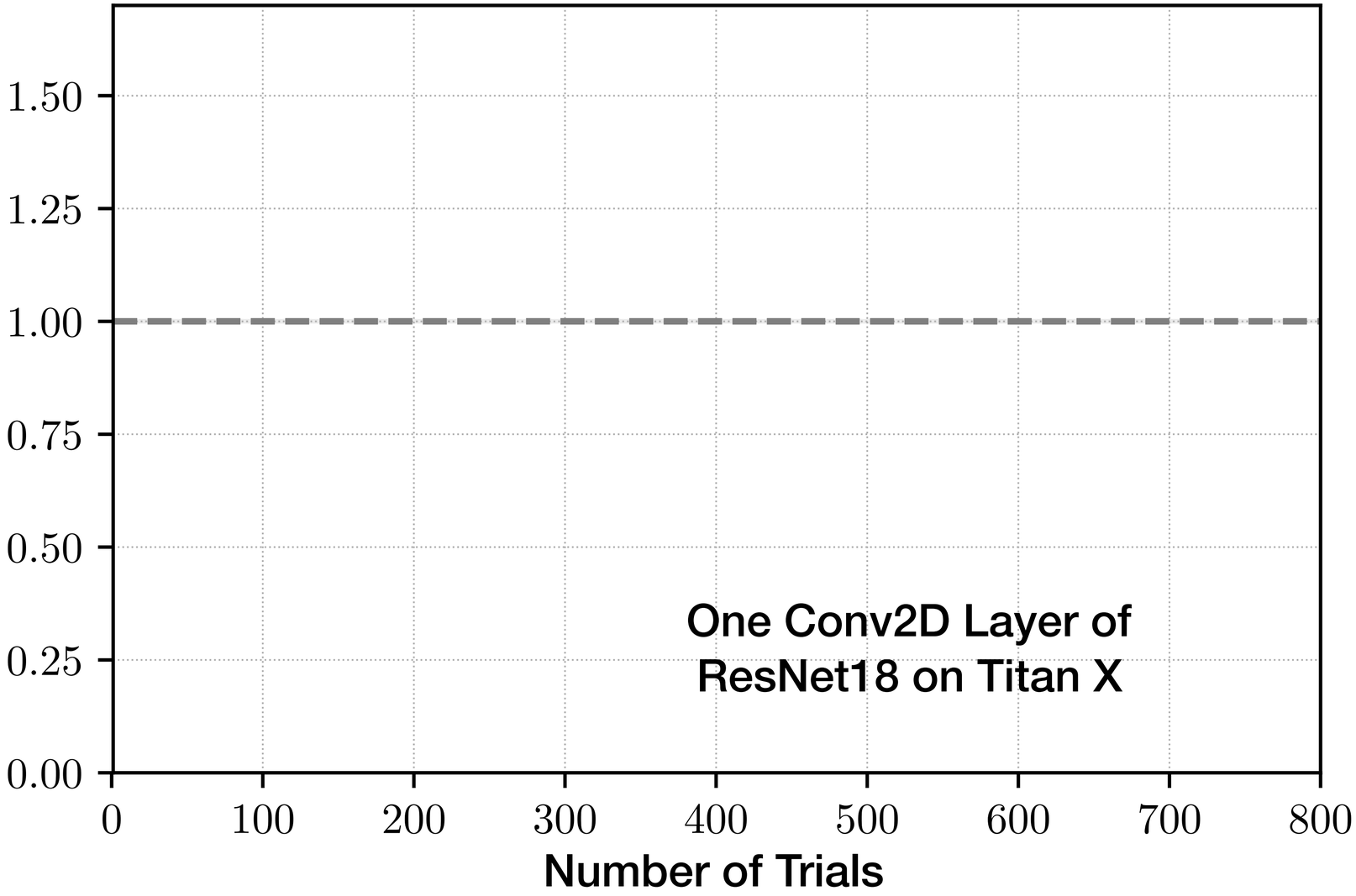
Effectiveness of ML based Model



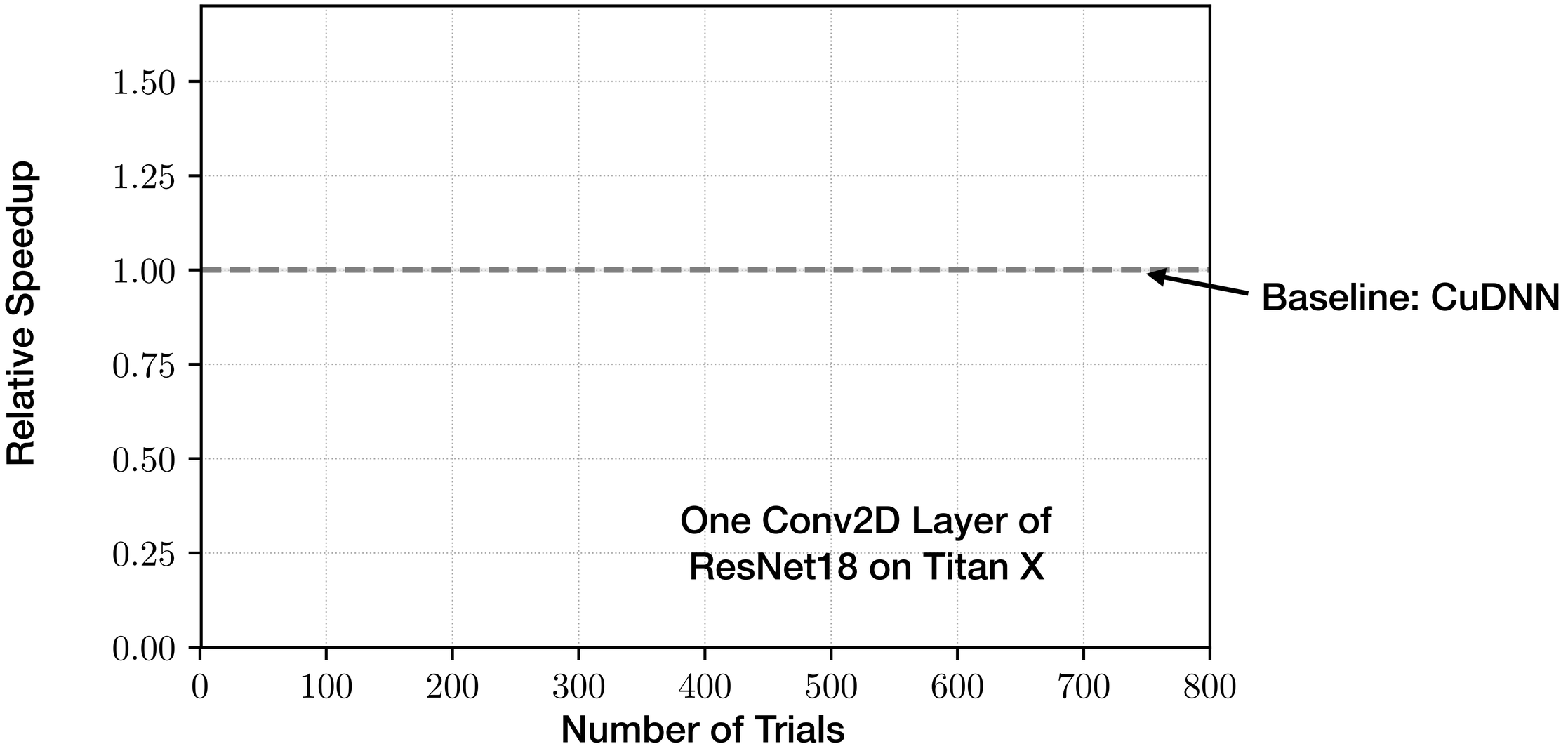
Effectiveness of ML based Model



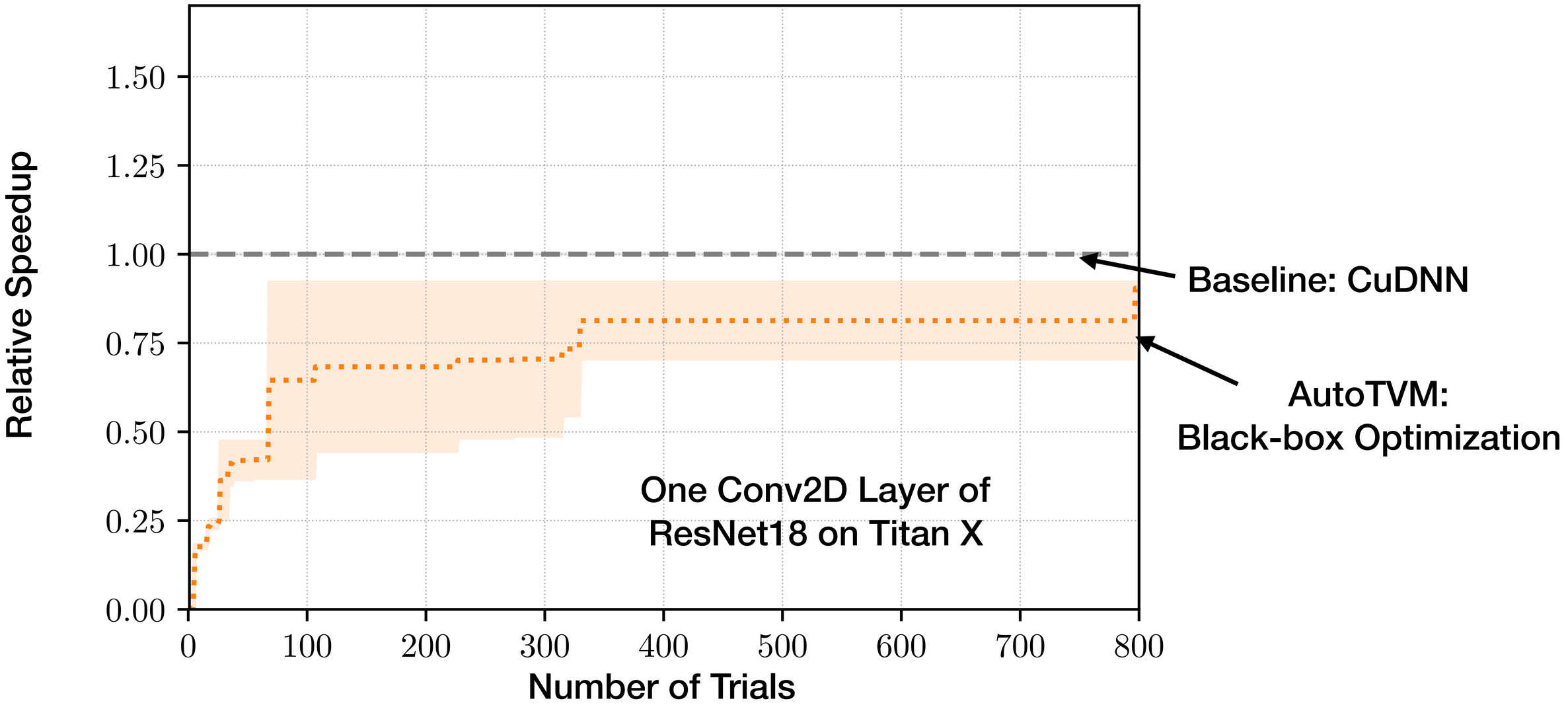
Effectiveness of ML based Model



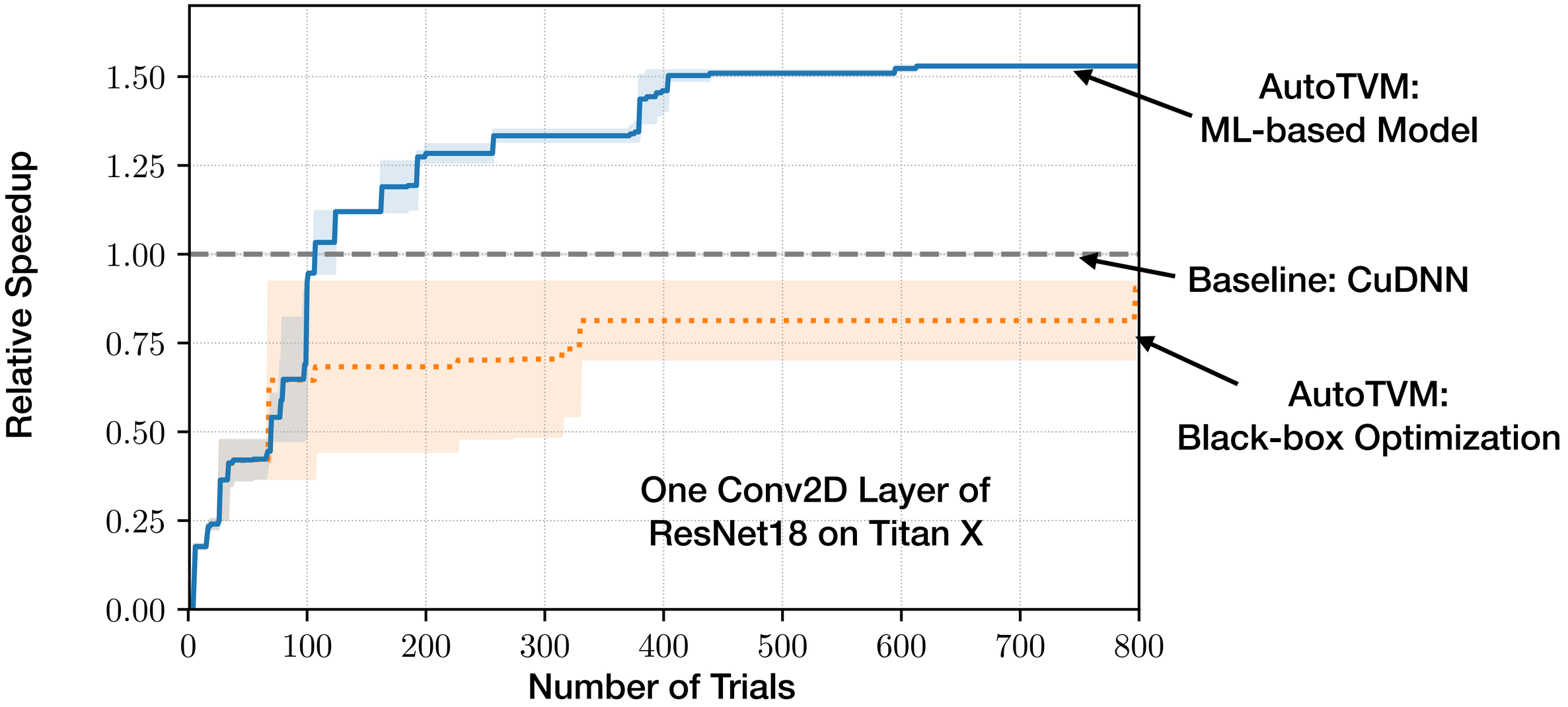
Effectiveness of ML based Model



Effectiveness of ML based Model



Effectiveness of ML based Model



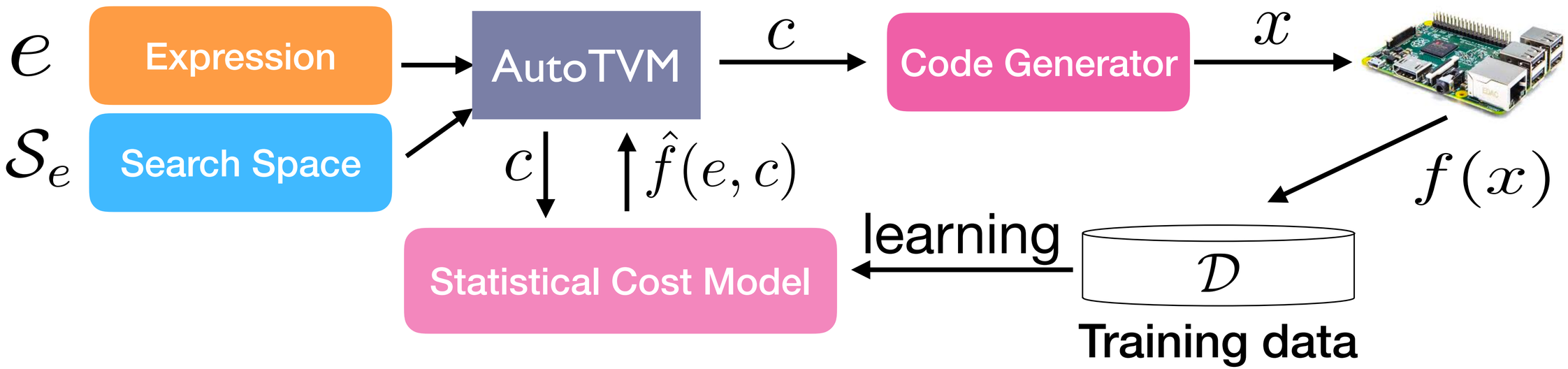
Comparisons of Models

	Task Invariant	Time Cost	Predictive Accuracy
Vanilla Model	No	Low	Medium
Tree-based Model	Yes	Low	Good
Neural Model	Yes	High	Good

Comparisons of Models

	Task Invariant	Time Cost	Predictive Accuracy
Vanilla Model	No	Low	Medium
Tree-based Model	Yes	Low	Good
Neural Model	Yes	High	Good

Unique Problem Characteristics

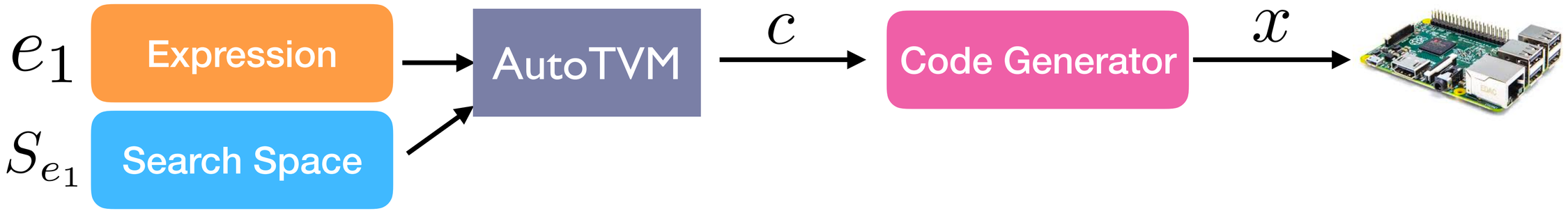


Relatively low experiment cost

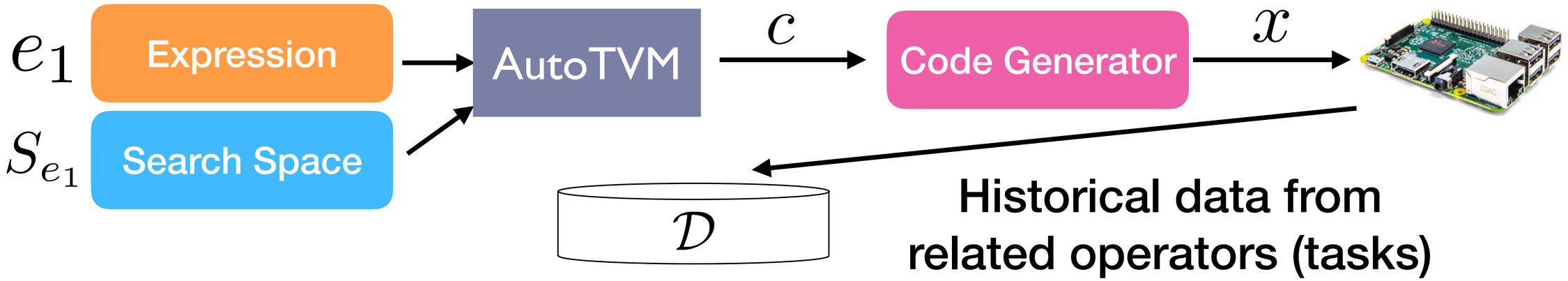
Program-aware modeling

Large number of similar tasks

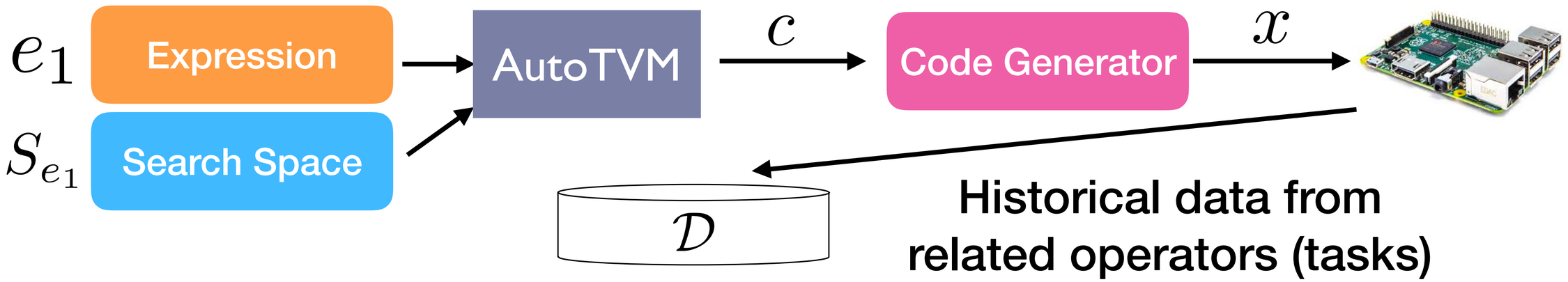
Transferable Cost Model



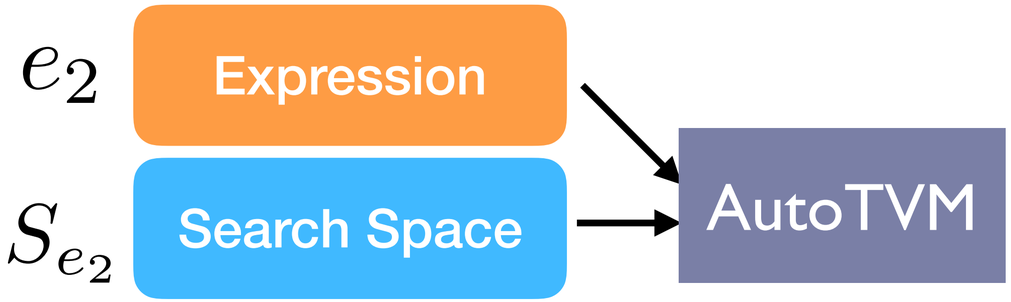
Transferable Cost Model



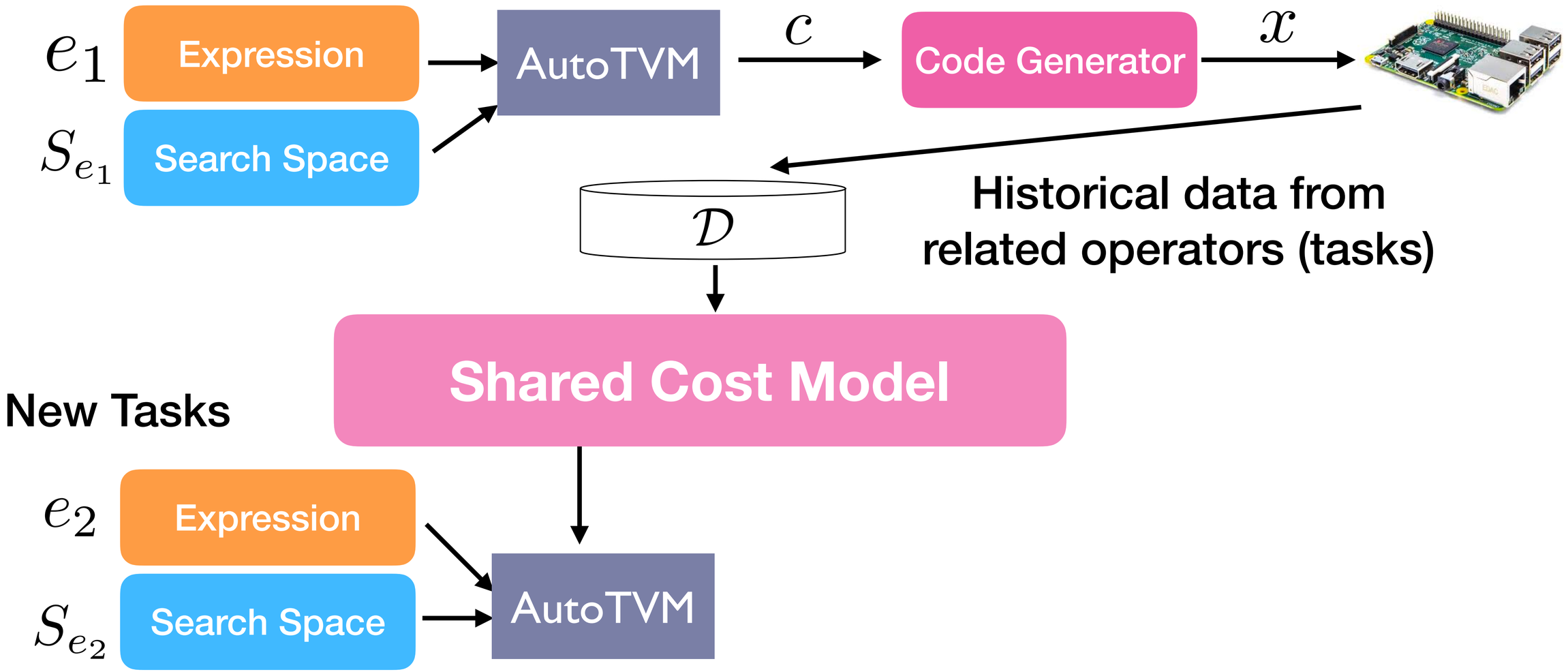
Transferable Cost Model



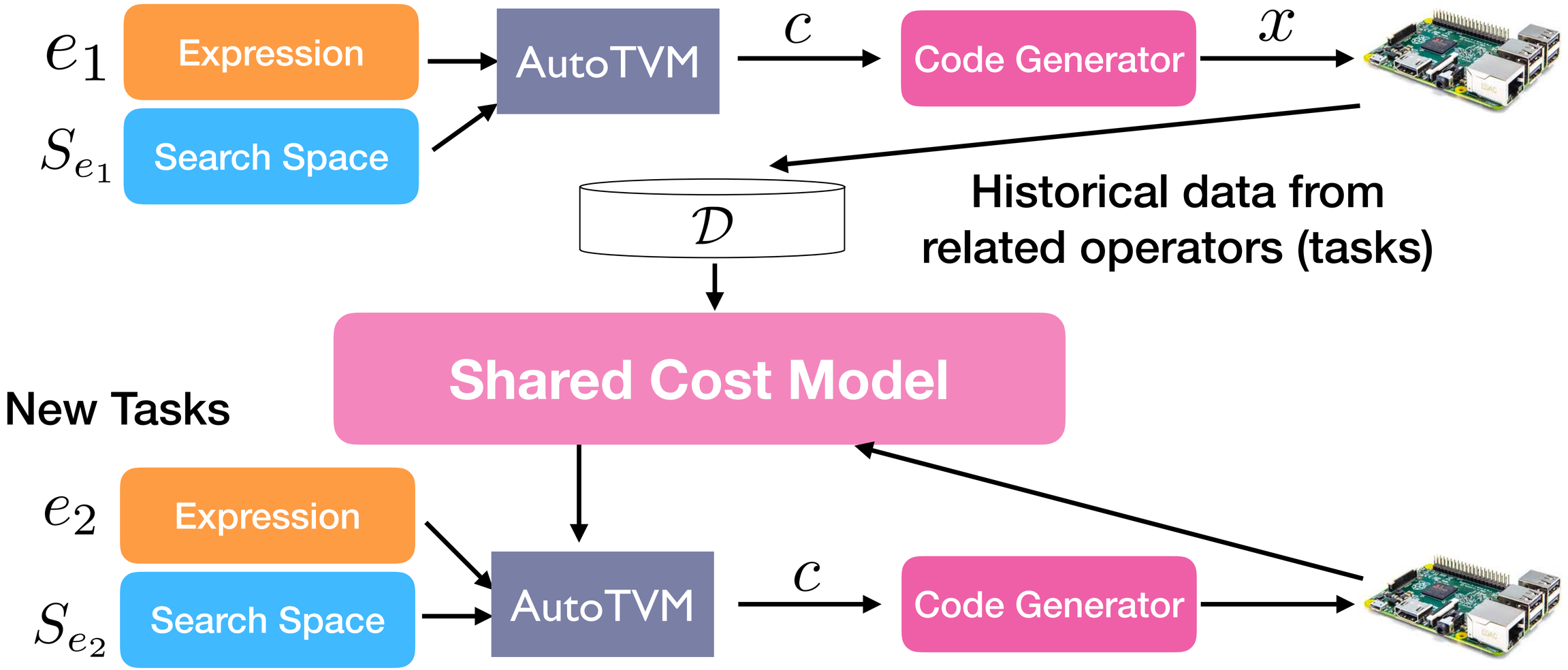
New Tasks



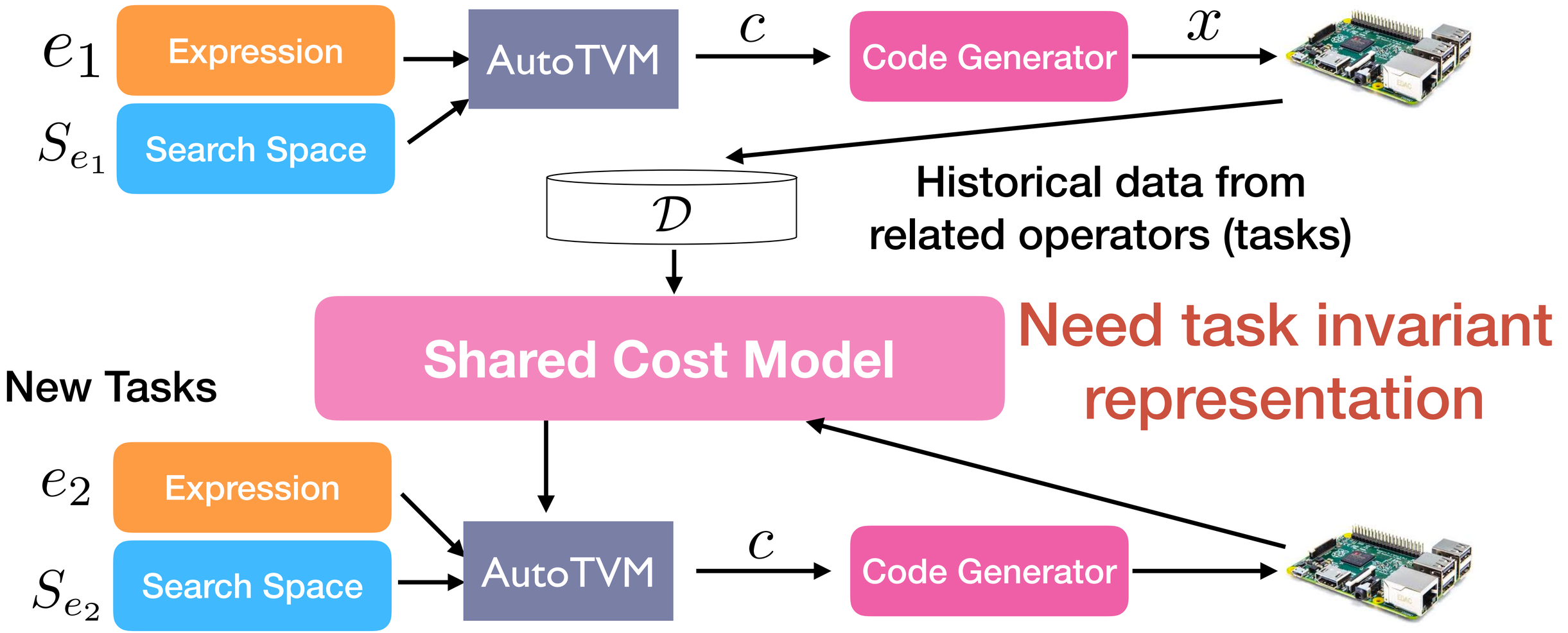
Transferable Cost Model



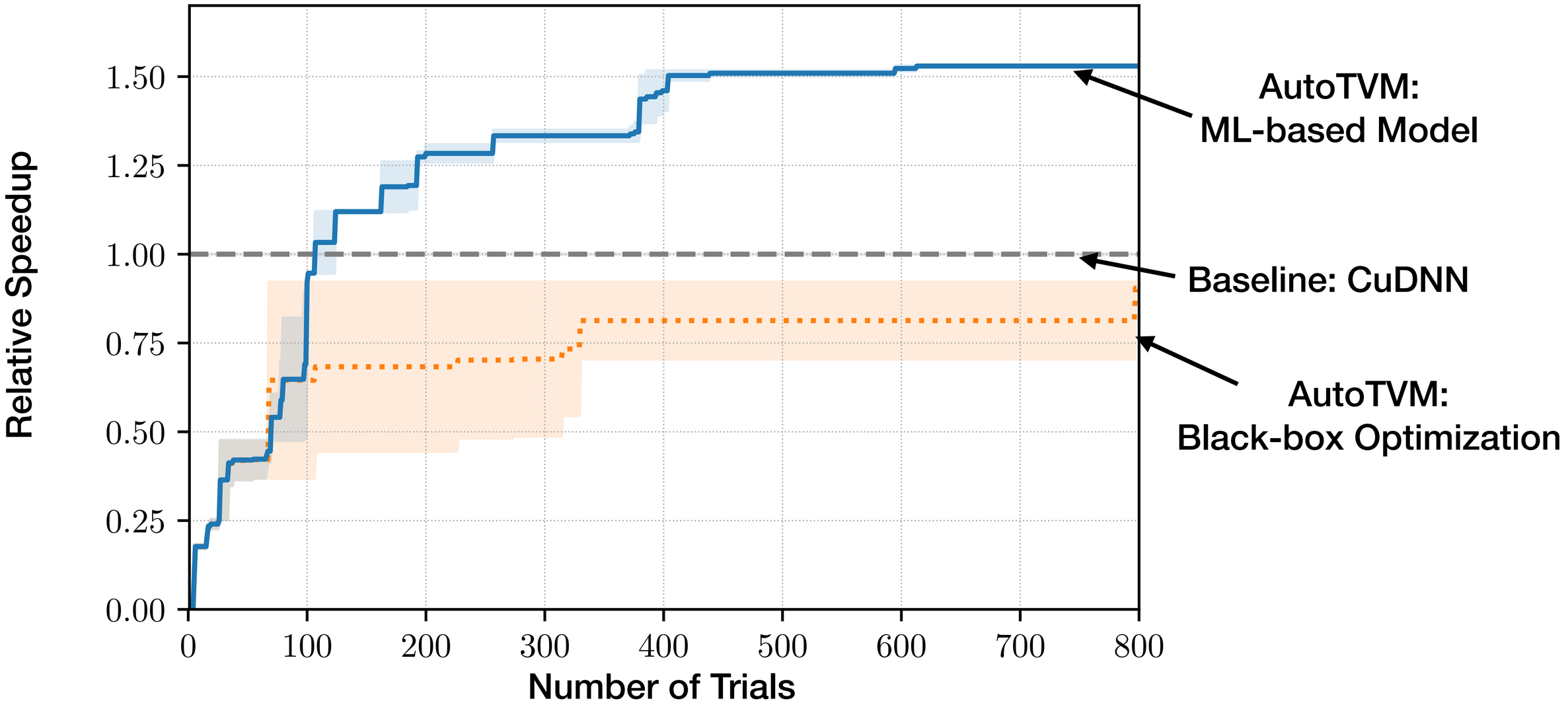
Transferable Cost Model



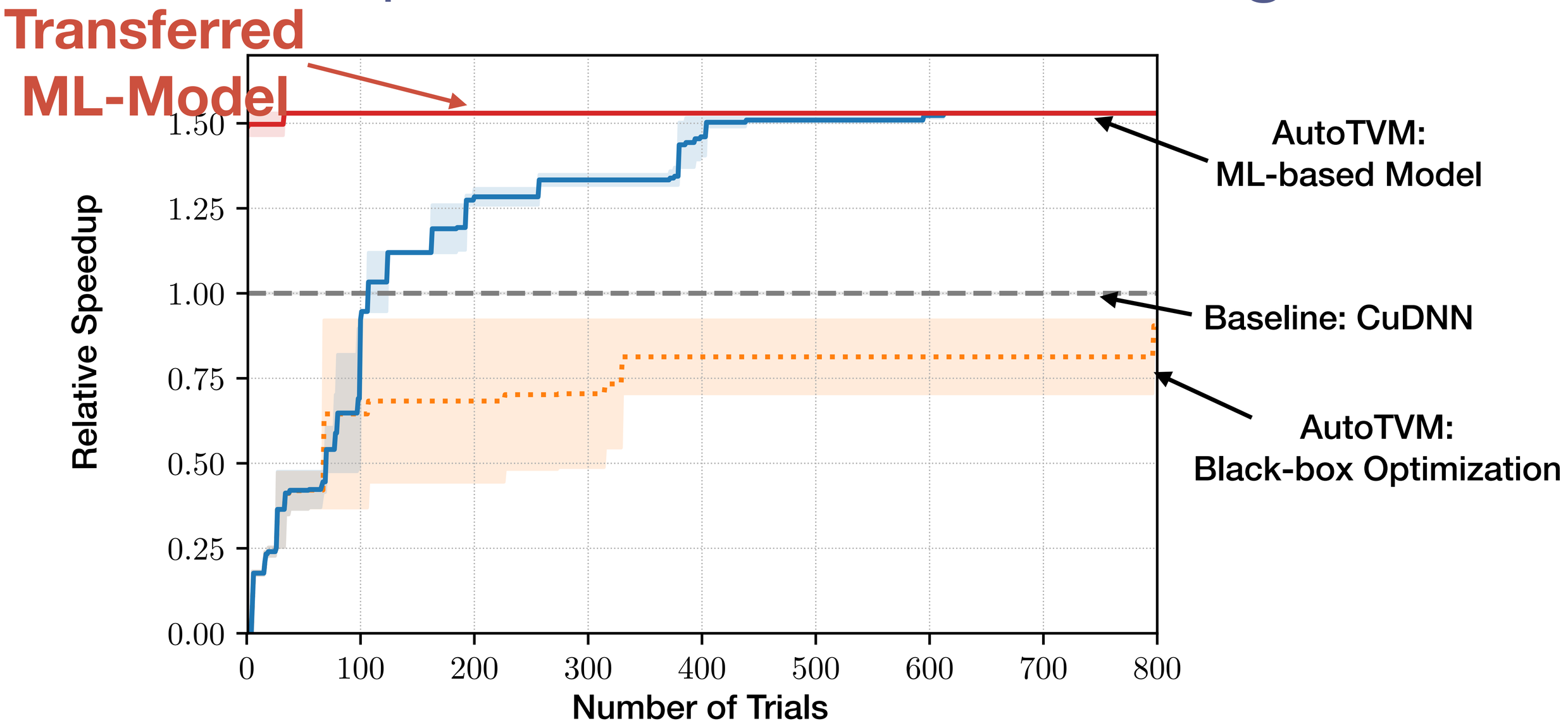
Transferable Cost Model



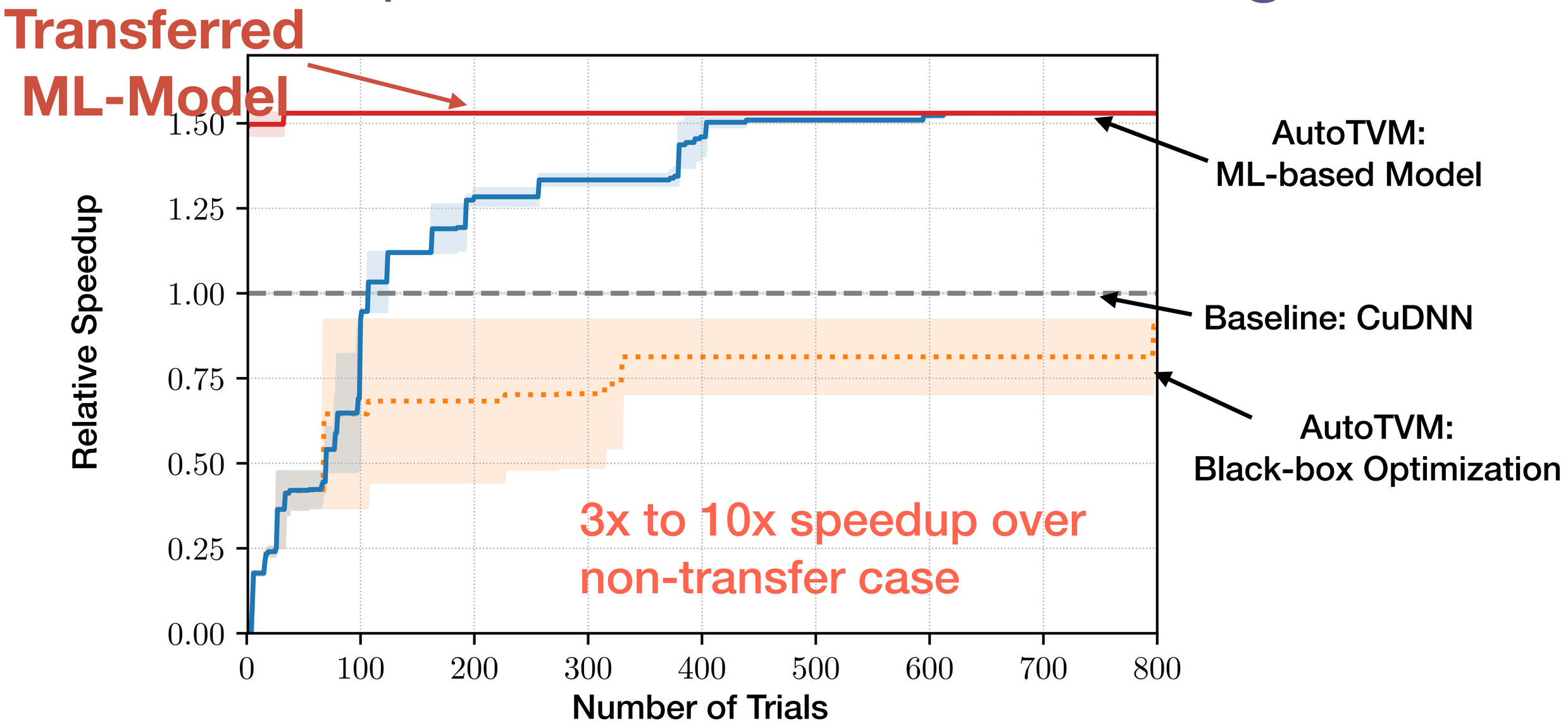
Impact of Transfer Learning



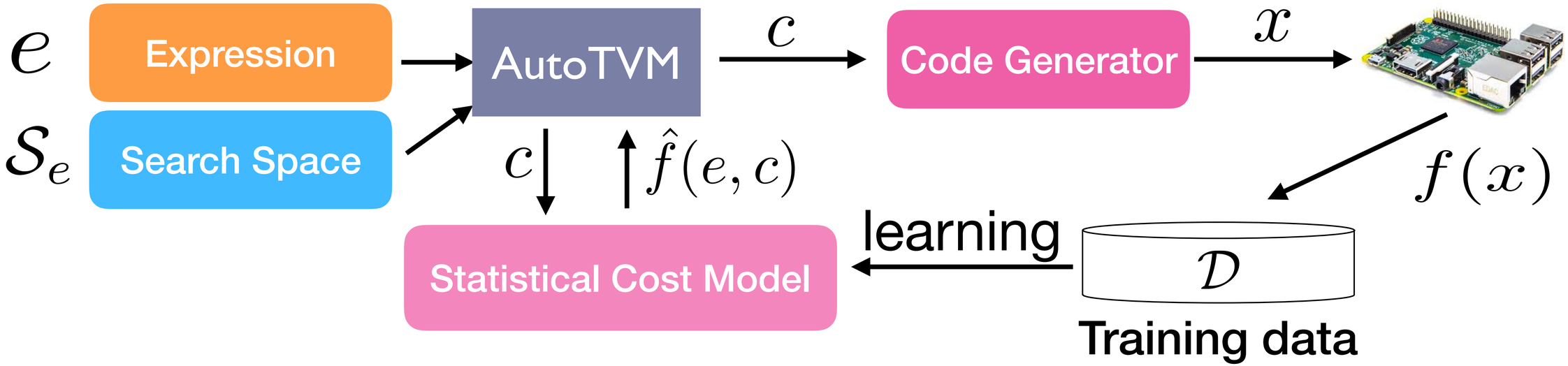
Impact of Transfer Learning



Impact of Transfer Learning



Learning to Optimize Tensor Programs



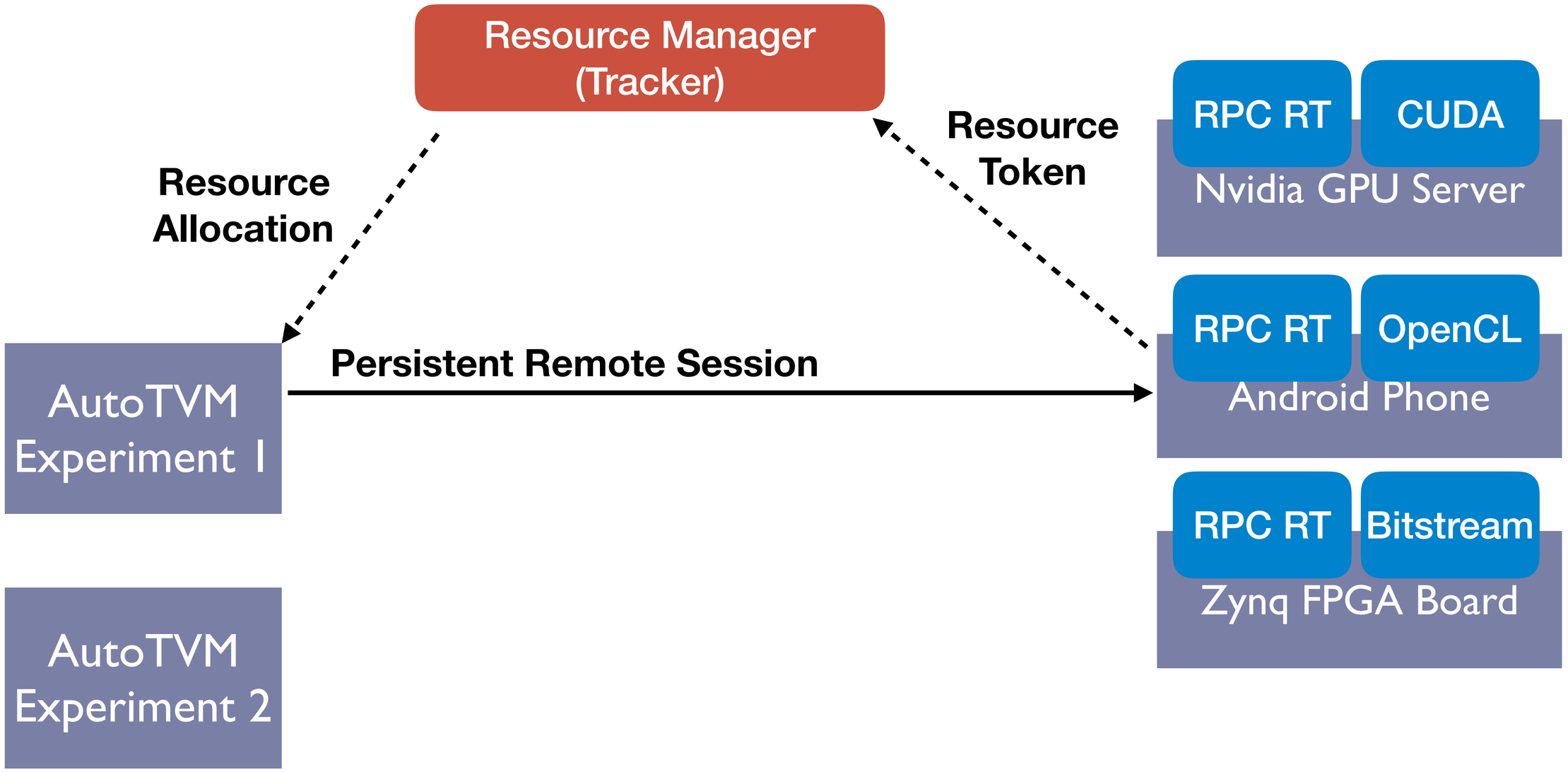
Relatively low experiment cost

Program-aware modeling

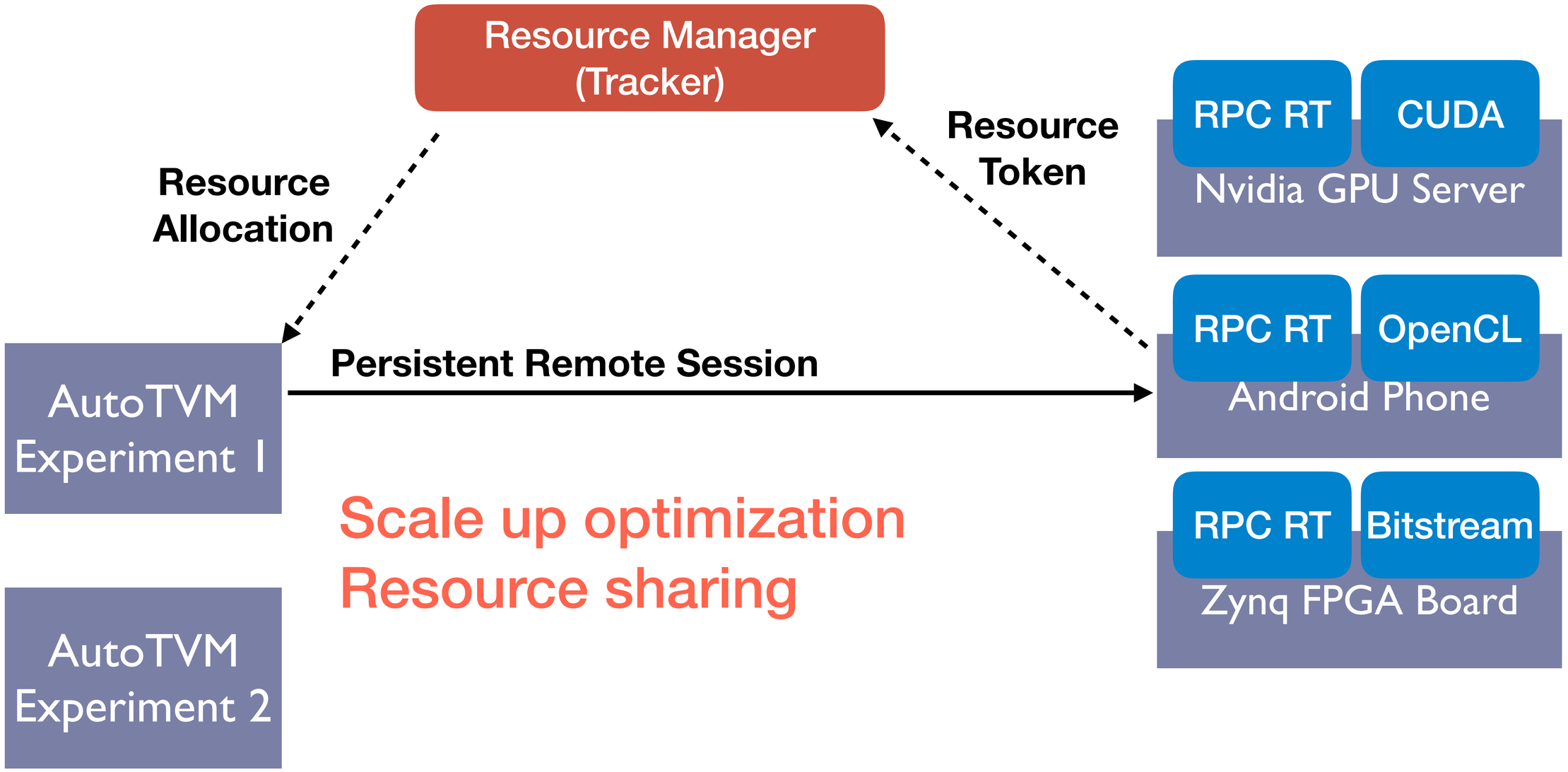
Large number of similar tasks

Learning to Optimize Tensor Programs. **Chen et al. NeurIPS 18**

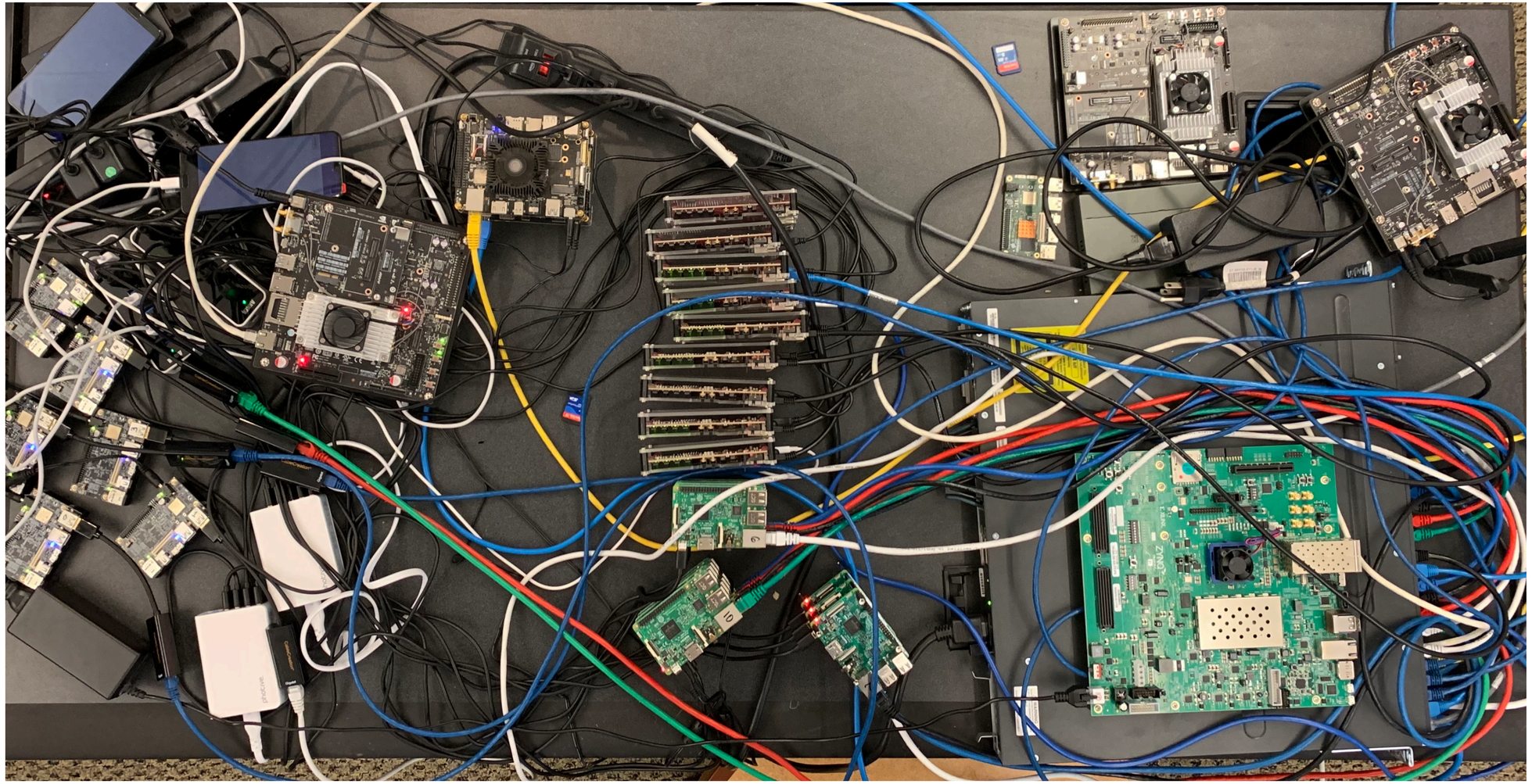
Device Fleet: Distributed Test Bed for AutoTVM



Device Fleet: Distributed Test Bed for AutoTVM



Device Fleet in Action



TVM: Learning-based Learning System

Why do we need machine learning for systems

How to build intelligent systems with learning

End to end learning-based learning system stack

TVM: End to End Deep Learning Compiler

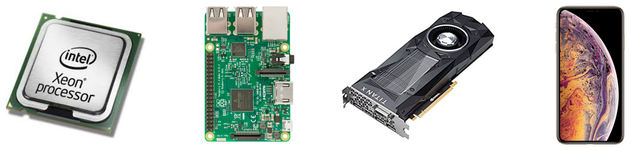
AutoTVM

TVM: End to End Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression and Optimization Search Space

LLVM, CUDA, Metal

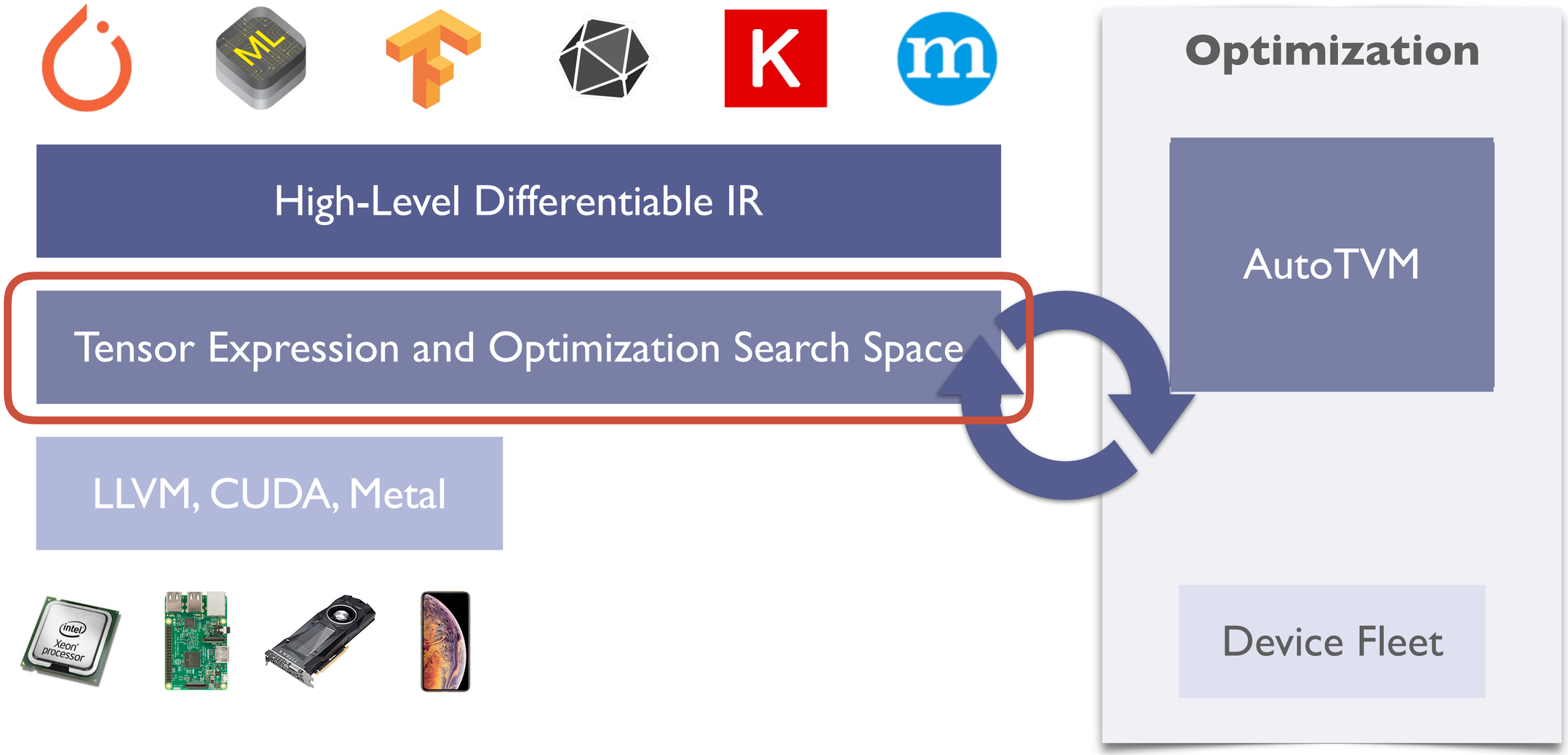


Optimization

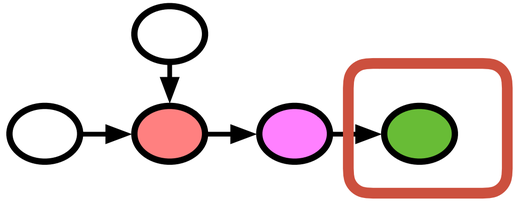
AutoTVM

Device Fleet

TVM: End to End Deep Learning Compiler

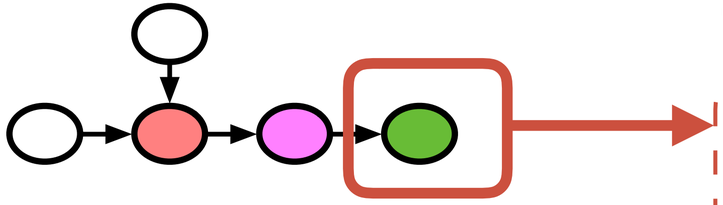


Tensor Expression and Optimization Search Space



Based on Halide's compute/schedule separation

Tensor Expression and Optimization Search Space



Tensor Expression (Specification)

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

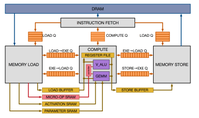
Based on Halide's compute/schedule separation

Hardware

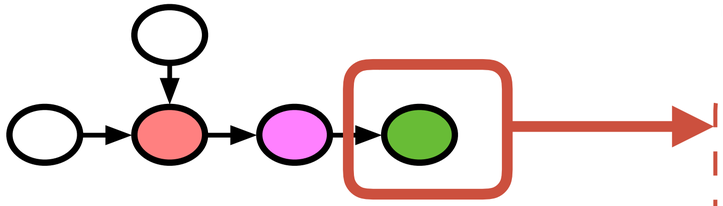






Tensor Expression and Optimization Search Space



Tensor Expression (Specification)

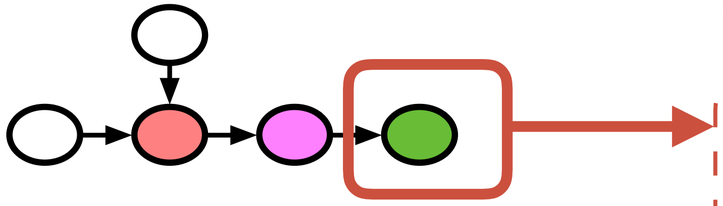
```
C = tvm.compute((m, n),
    lambda y, x: tvm.sum(A[k, y] * B[k, x], axis=k))
```

Search space:
Possible mappings from the expression to valid hardware programs

Based on Halide's compute/schedule separation

Hardware

Tensor Expression and Optimization Search Space



Tensor Expression (Specification)

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

Search space:
Possible mappings from the expression to valid hardware programs

Based on Halide's compute/schedule separation

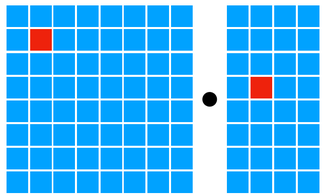
What is the search space

Hardware

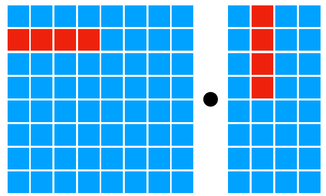
Search Space for CPUs

CPUs

Compute Primitives



scalar



vector

Memory Subsystem

implicitly managed

Loop Transformations

Cache Locality

Vectorization

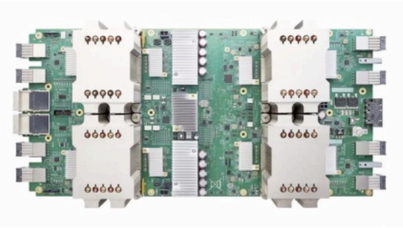
Reuse primitives from prior work:
Halide, Loopy

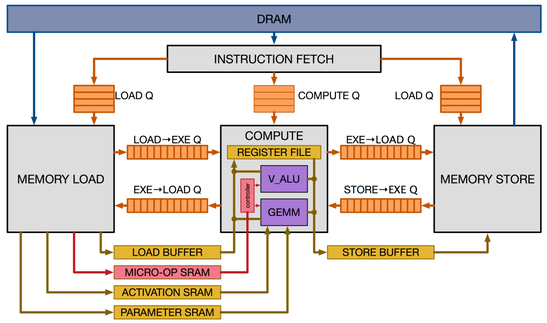
Hardware-aware Search Space

CPUs

GPUs

TPU-like specialized Accelerators

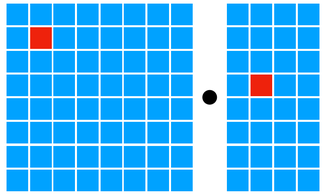




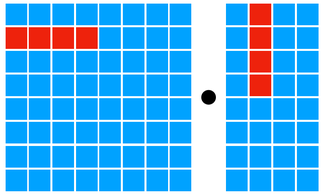
Search Space for GPUs

GPUs

Compute Primitives



scalar



vector

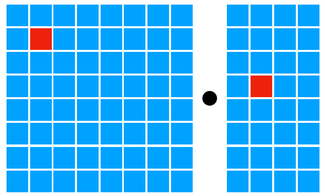
Memory Subsystem

mixed

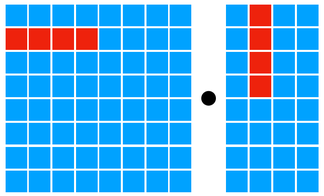
Search Space for GPUs

GPUs

Compute Primitives

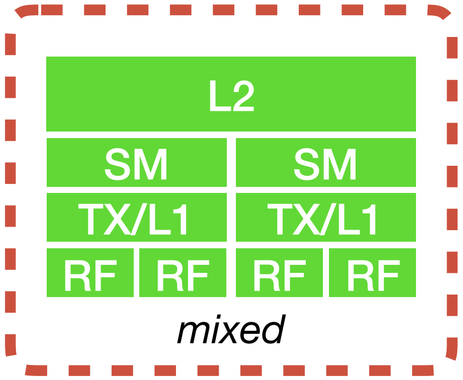


scalar



vector

Memory Subsystem

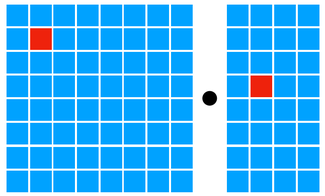


Shared memory among compute cores

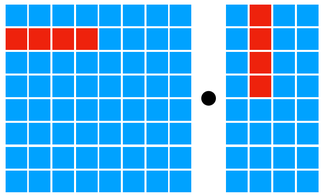
Search Space for GPUs

GPUs

Compute Primitives

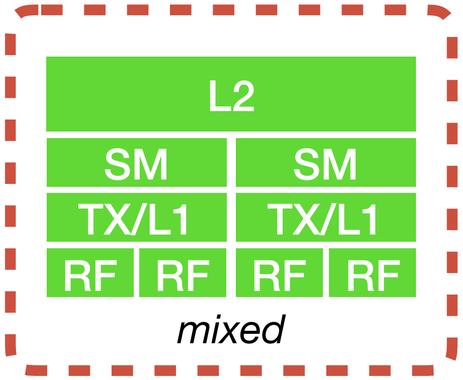


scalar



vector

Memory Subsystem



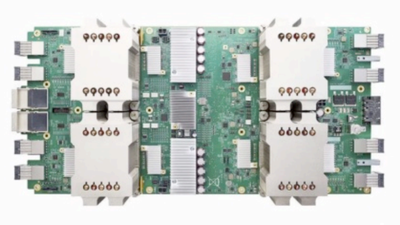
Shared memory among compute cores

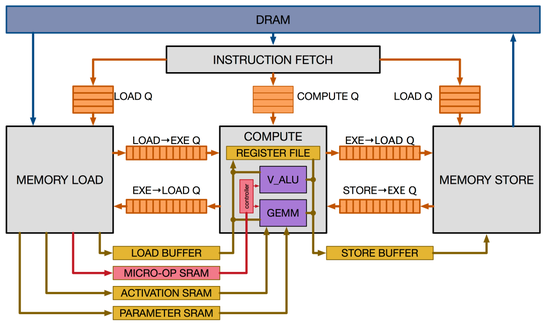
Use of Shared Memory

Thread Cooperation

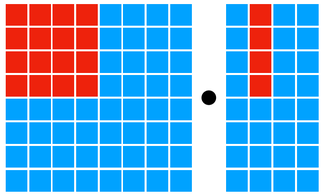
Search Space for TPU-like Specialized Accelerators

TPUs

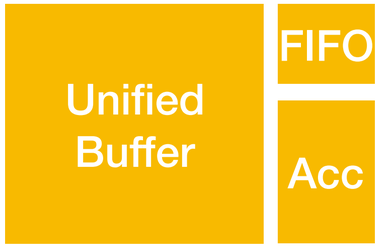




Tensor Compute Primitives

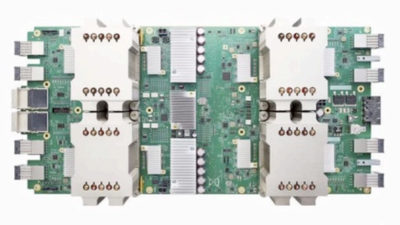


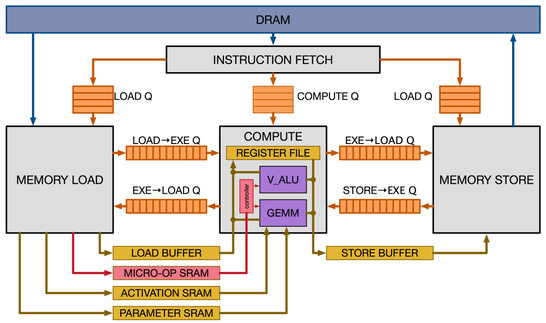
Explicitly Managed Memory Subsystem



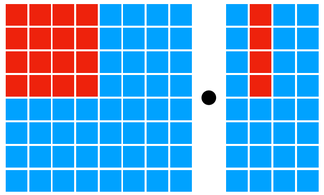
Search Space for TPU-like Specialized Accelerators

TPUs

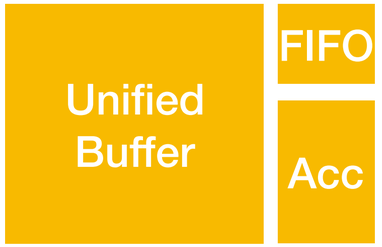




Tensor Compute Primitives



Explicitly Managed Memory Subsystem

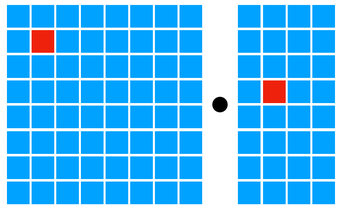


Tensorization Challenge

Compute primitives

Tensorization Challenge

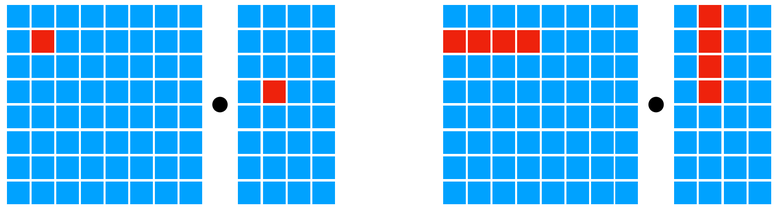
Compute primitives



scalar

Tensorization Challenge

Compute primitives

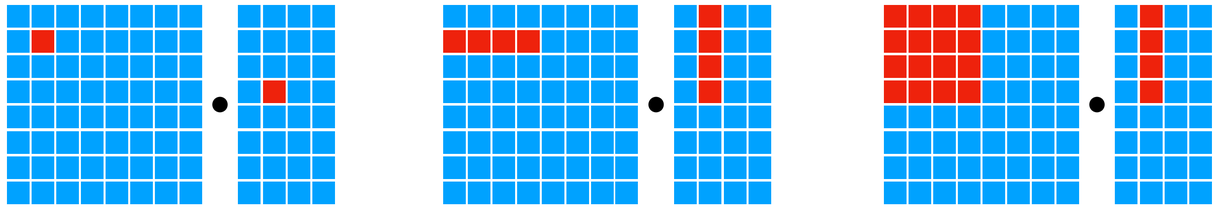


scalar

vector

Tensorization Challenge

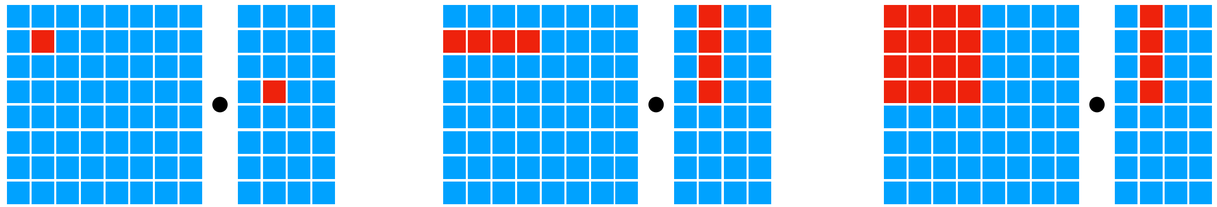
Compute primitives



scalar *vector* *tensor*

Tensorization Challenge

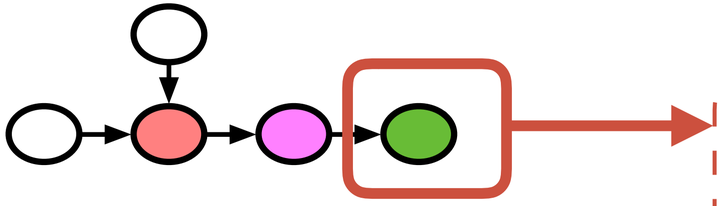
Compute primitives



scalar *vector* *tensor*

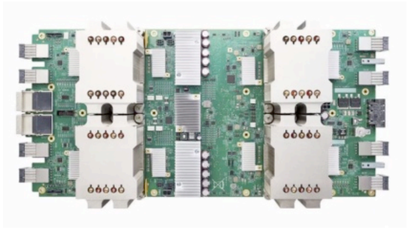
Challenge: Build systems to support emerging tensor instructions

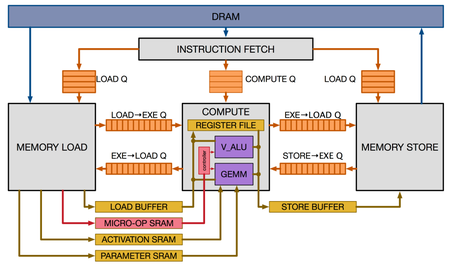
Tensorization Challenge



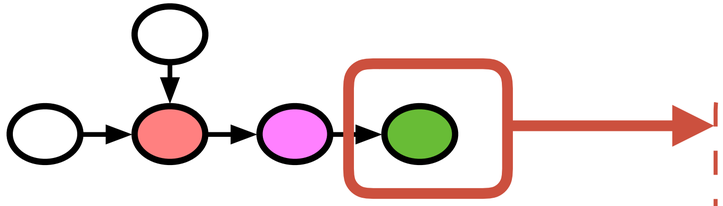
Computation Specification (Tensor Expression)

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



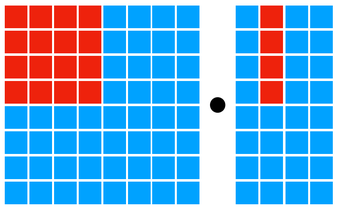


Tensorization Challenge



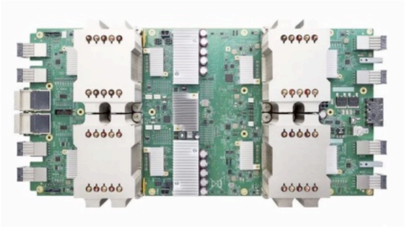
Computation Specification (Tensor Expression)

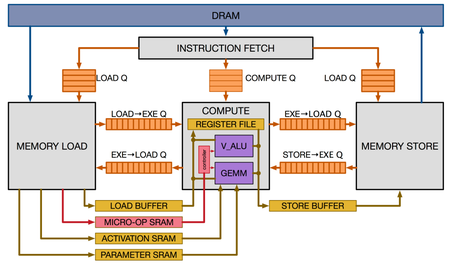
```
C = tvm.compute((m, n),
    lambda y, x: tvm.sum(A[k, y] * B[k, x], axis=k))
```



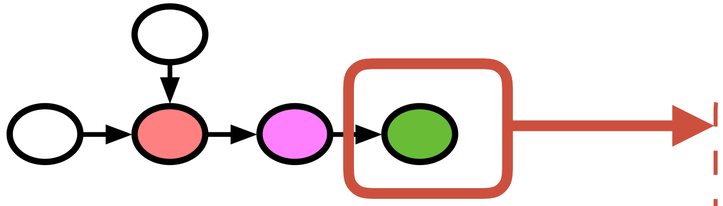
```
A = tvm.placeholder((8, 8))
B = tvm.placeholder((8,))
k = tvm.reduce_axis((0, 8))
C = tvm.compute((8, 8),
    lambda y, x: tvm.sum(A[k, y] * B[k], axis=k))
```

HW Interface Specification by Tensor Expression



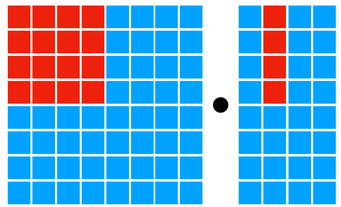


Tensorization Challenge



Computation Specification (Tensor Expression)

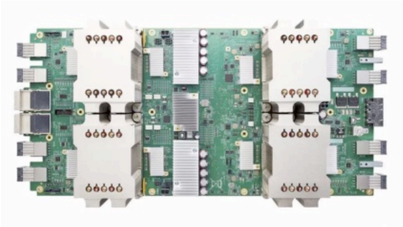
```
C = tvm.compute((m, n),
    lambda y, x: tvm.sum(A[k, y] * B[k, x], axis=k))
```

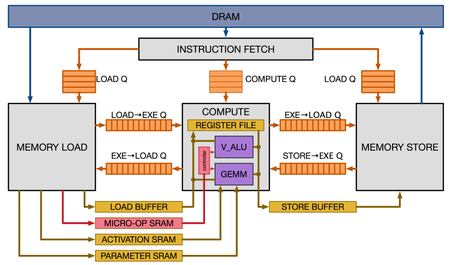


```
A = tvm.placeholder((8, 8))
B = tvm.placeholder((8,))
k = tvm.reduce_axis((0, 8))
C = tvm.compute((8, 8),
    lambda y, x: tvm.sum(A[k, y] * B[k], axis=k))
```

Tensorization

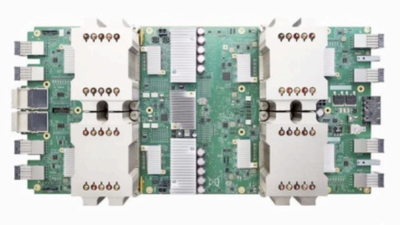
HW Interface Specification by Tensor Expression

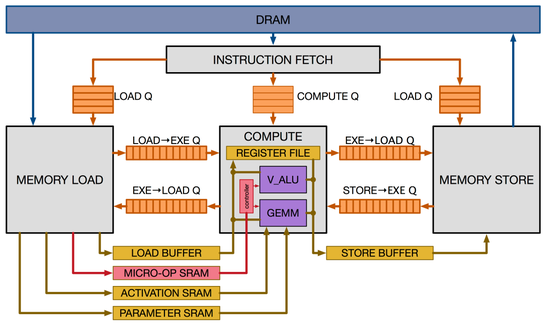




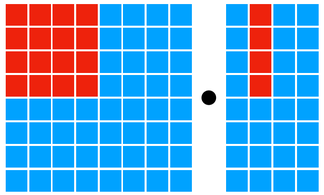
Search Space for TPU-like Specialized Accelerators

TPUs

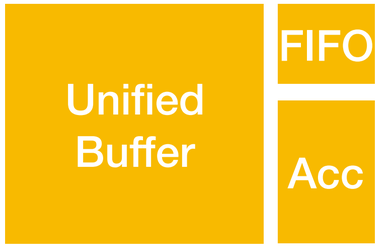




Tensor Compute Primitives

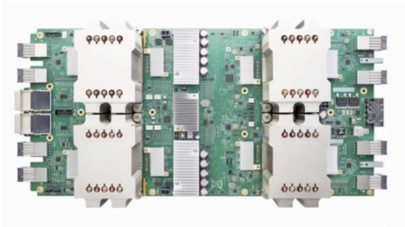


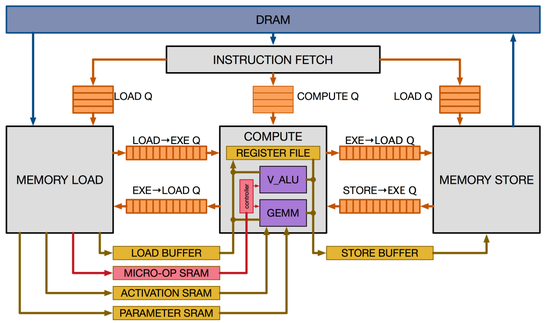
Explicitly Managed Memory Subsystem



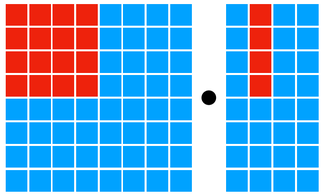
Search Space for TPU-like Specialized Accelerators

TPUs

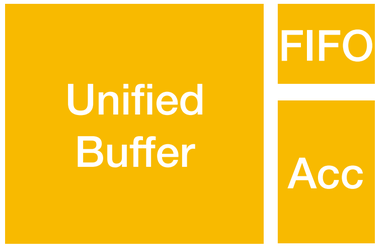




Tensor Compute Primitives



Explicitly Managed Memory Subsystem

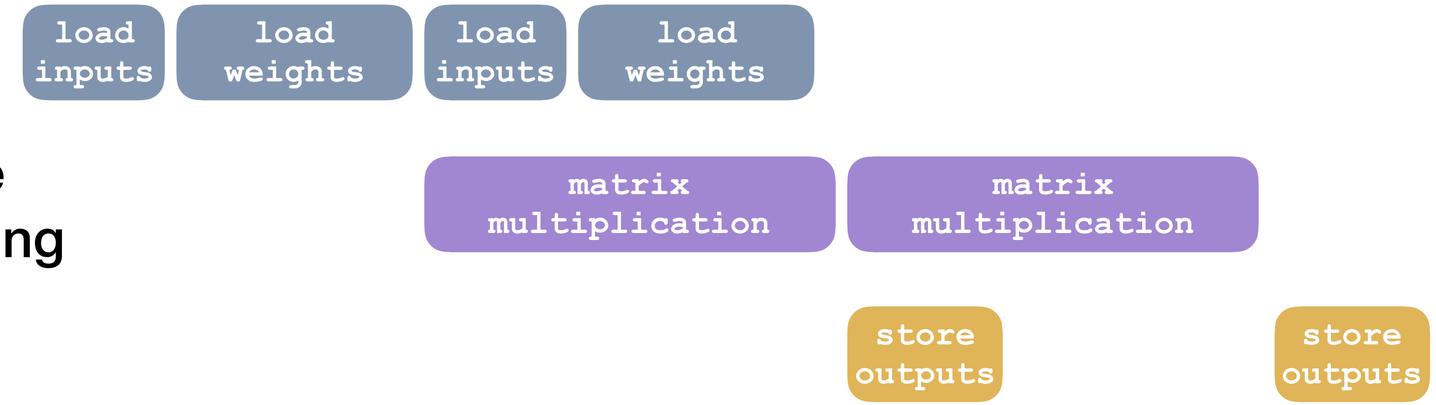


Software Support for Latency Hiding

Software Support for Latency Hiding

Single Module
No Task-Pipelining

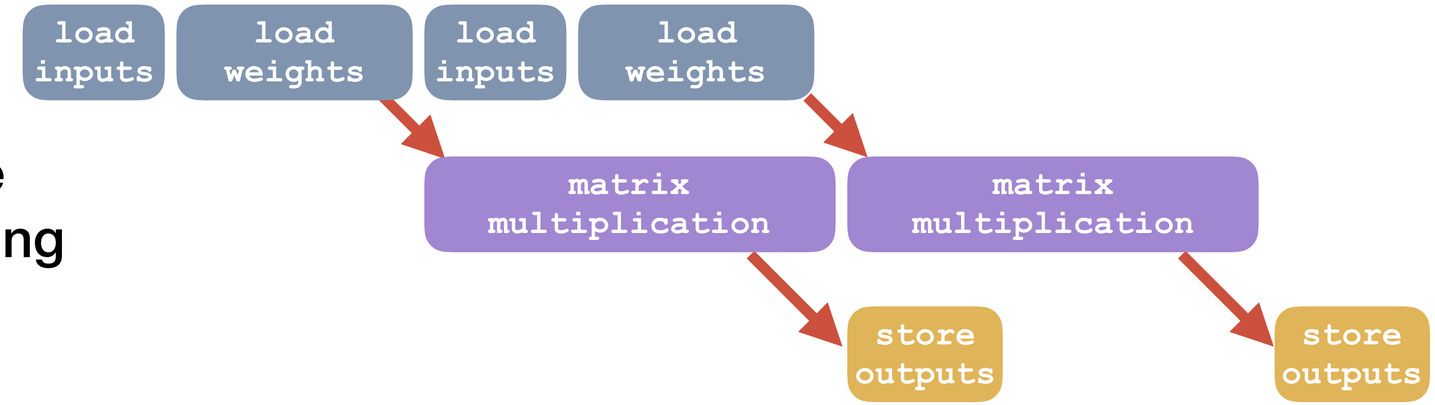
Multiple-Module
Task-Level Pipelining



Software Support for Latency Hiding

Single Module
No Task-Pipelining

Multiple-Module
Task-Level Pipelining

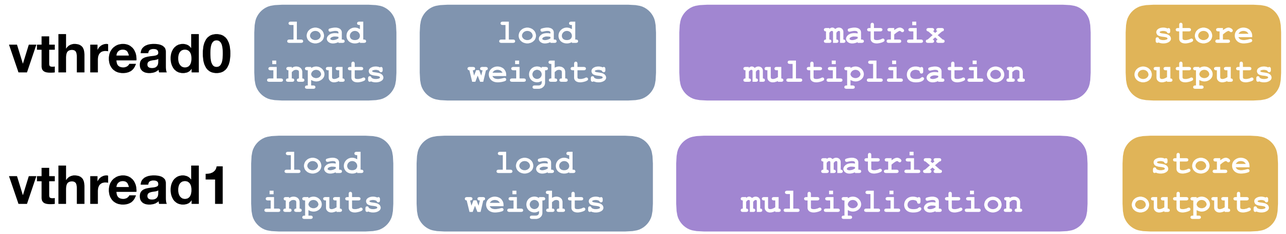


Explicit dependencies
managed by software to hide memory latency

Software Support for Latency Hiding

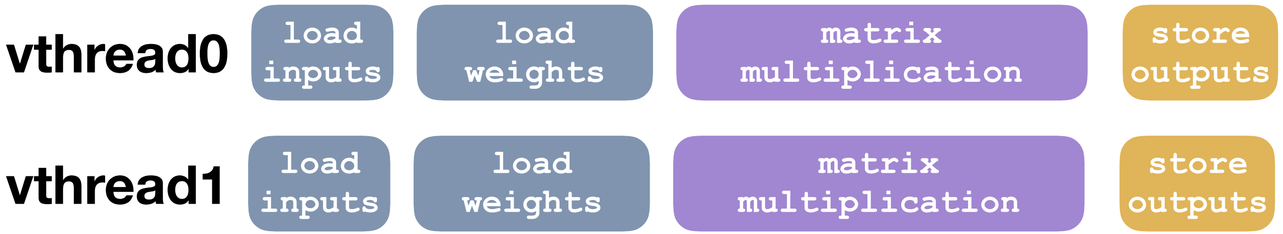
Software Support for Latency Hiding

Multi-threaded Program

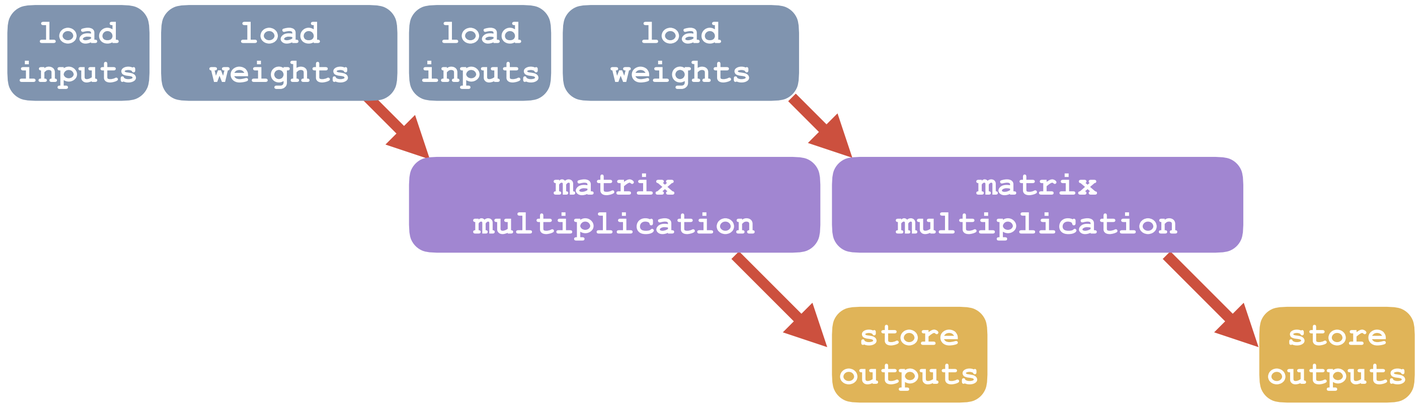


Software Support for Latency Hiding

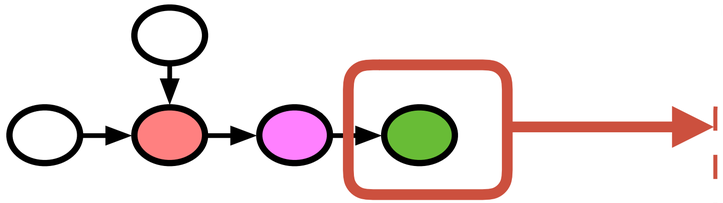
Multi-threaded Program



Program with Task-level Pipeline Instructions



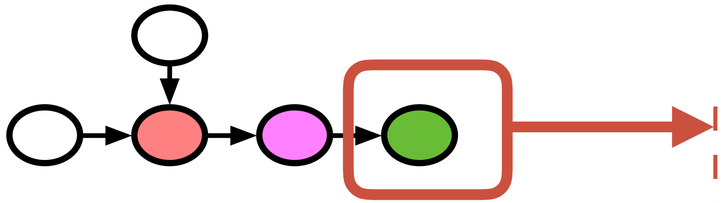
Summary: Hardware-aware Search Space



Tensor Expression Language

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

Summary: Hardware-aware Search Space



Tensor Expression Language

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

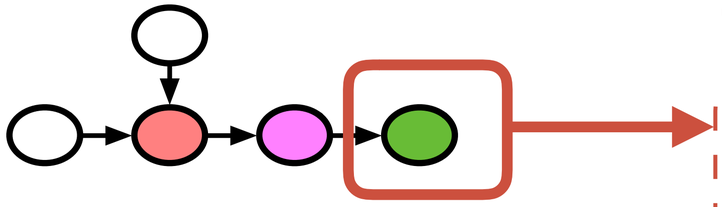
Primitives in prior work:
Halide, Loopy

Loop Transformations

Thread Bindings

Cache Locality

Summary: Hardware-aware Search Space



Tensor Expression Language

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

Primitives in prior work:
Halide, Loopy

Loop Transformations

Thread Bindings

Cache Locality

New primitives for GPUs, and enable TPU-like Accelerators

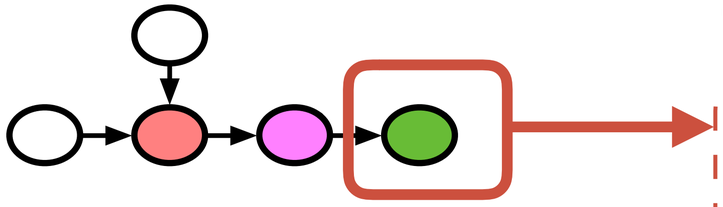
Thread Cooperation

Tensorization

Latency Hiding

...

Summary: Hardware-aware Search Space



Tensor Expression Language

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

Primitives in prior work:
Halide, Loopy

Loop Transformations

Thread Bindings

Cache Locality

New primitives for GPUs, and enable TPU-like Accelerators

Thread Cooperation

Tensorization

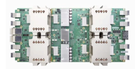
Latency Hiding

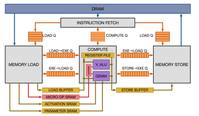
...

Hardware







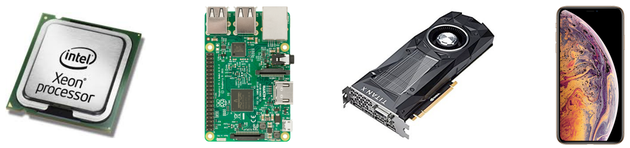



TVM: End to End Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression and Optimization Search Space

LLVM, CUDA, Metal

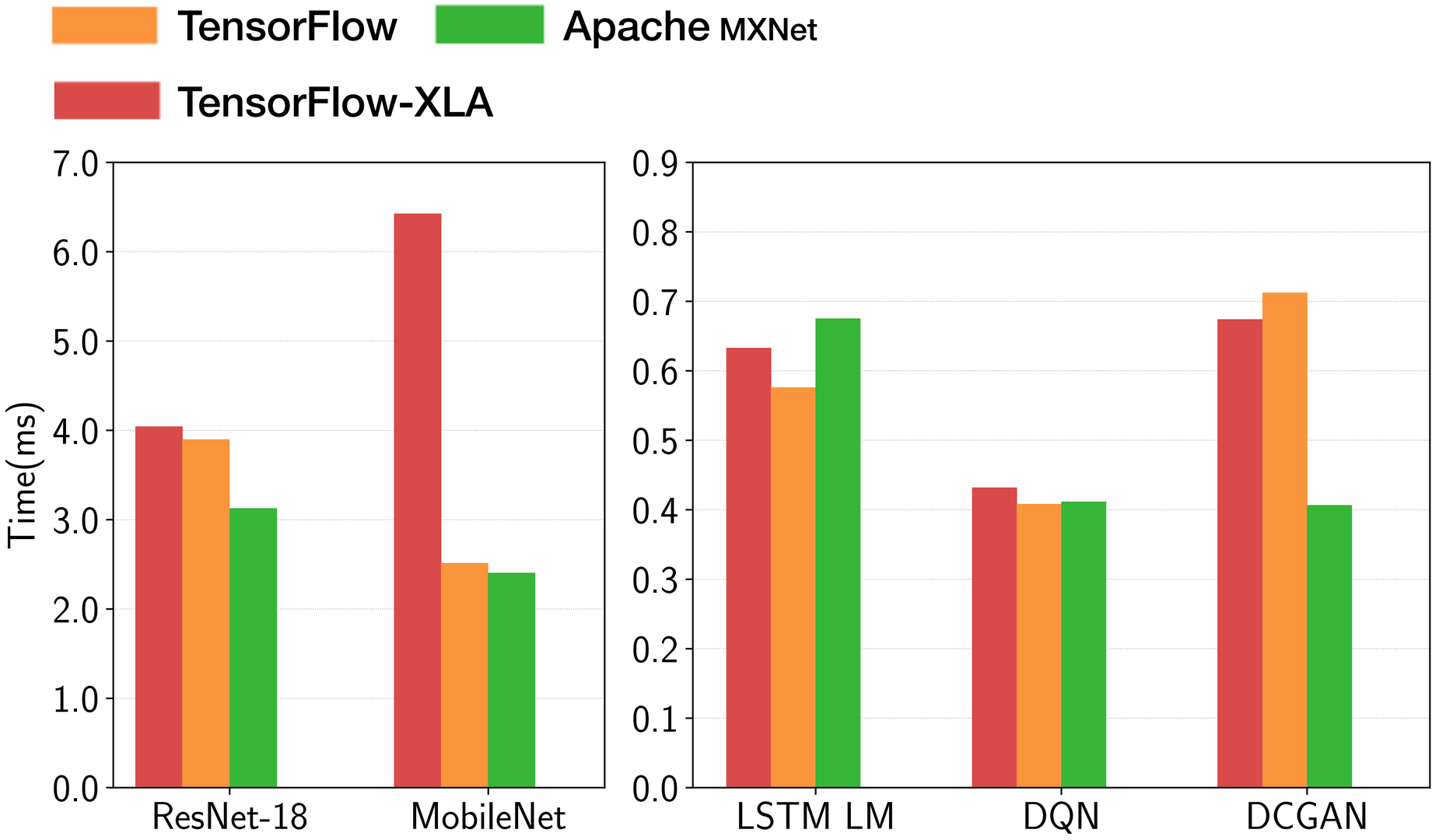


Optimization

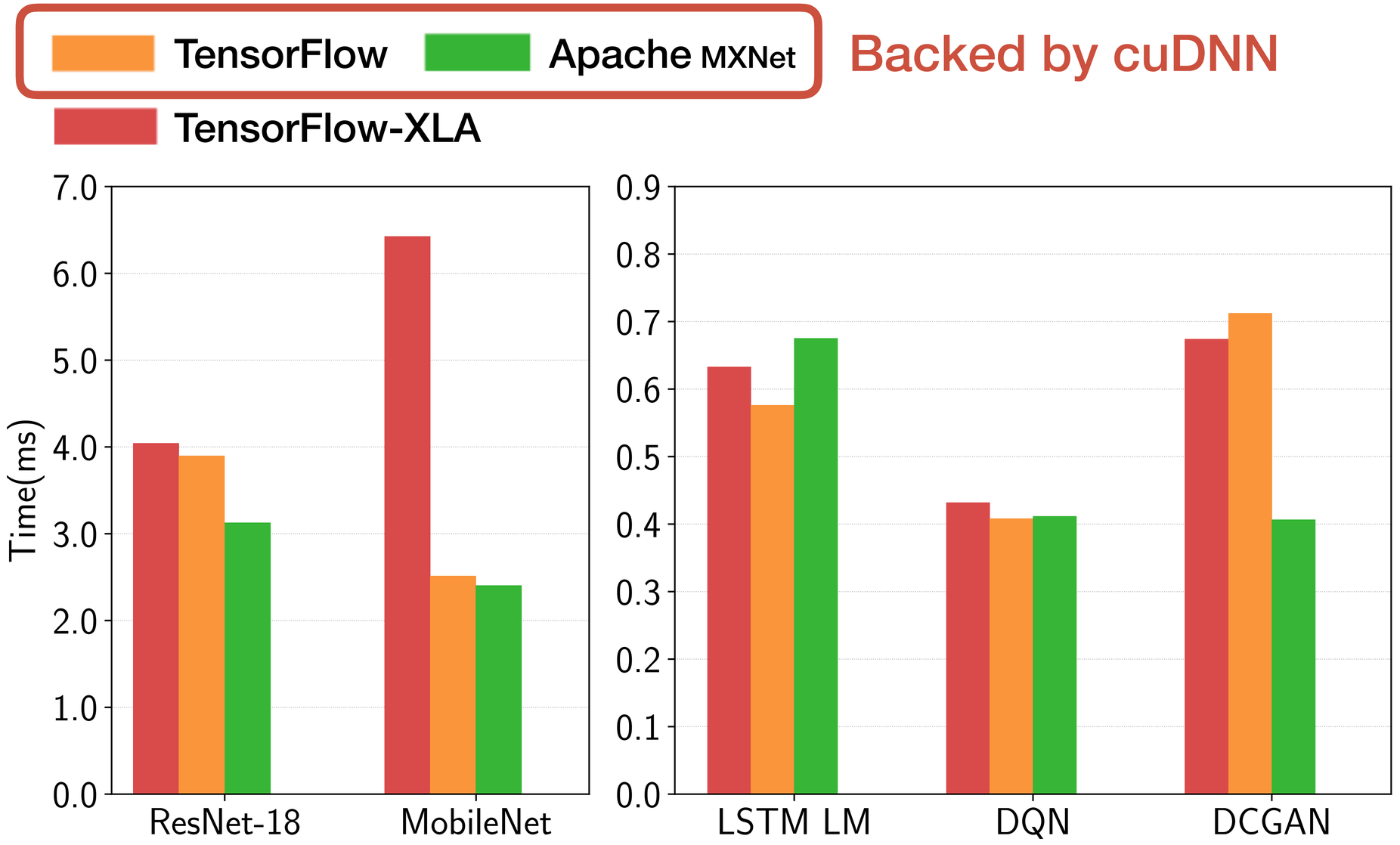
AutoTVM

Device Fleet

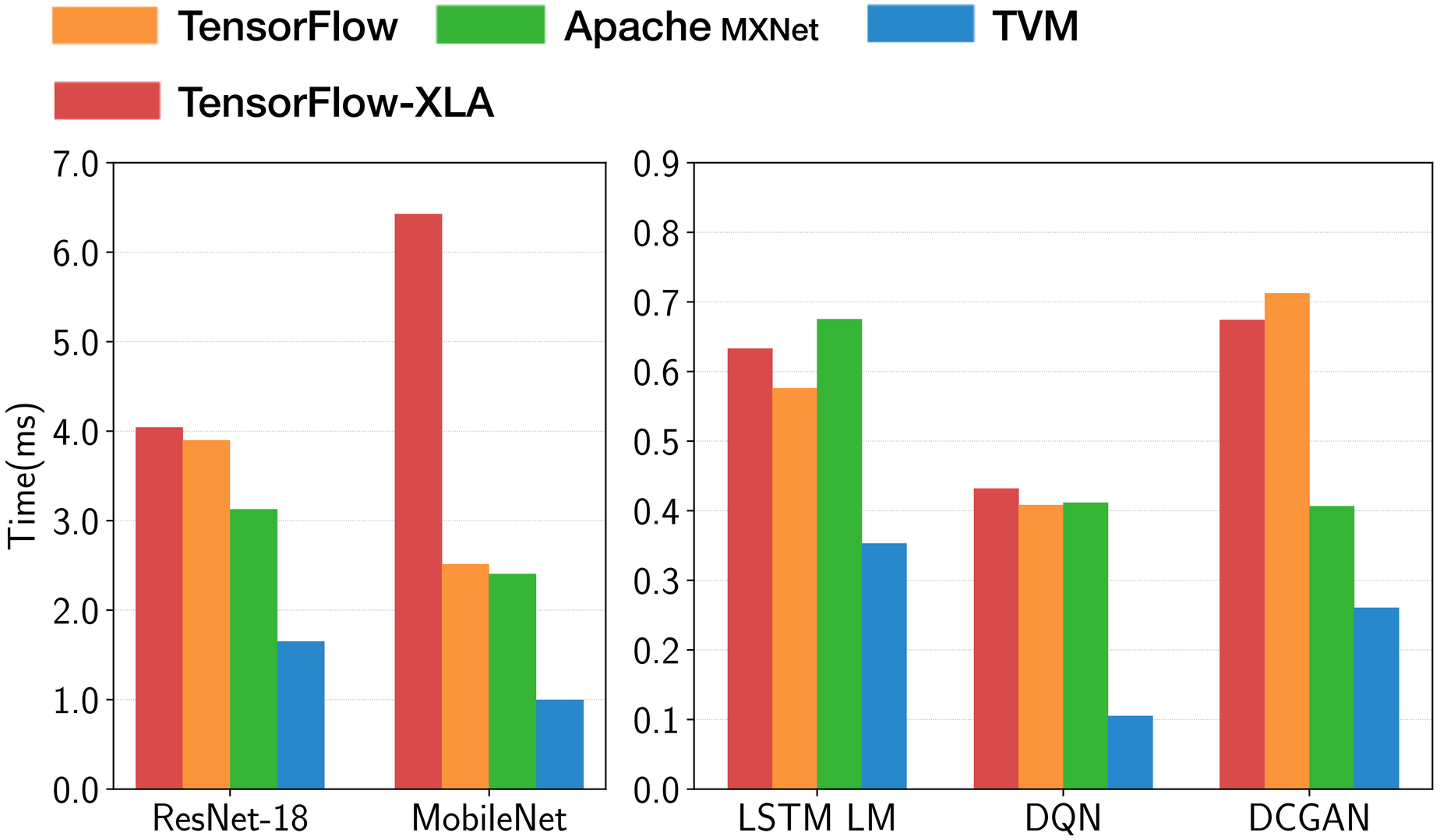
End to End Inference Performance (Nvidia Titan X)



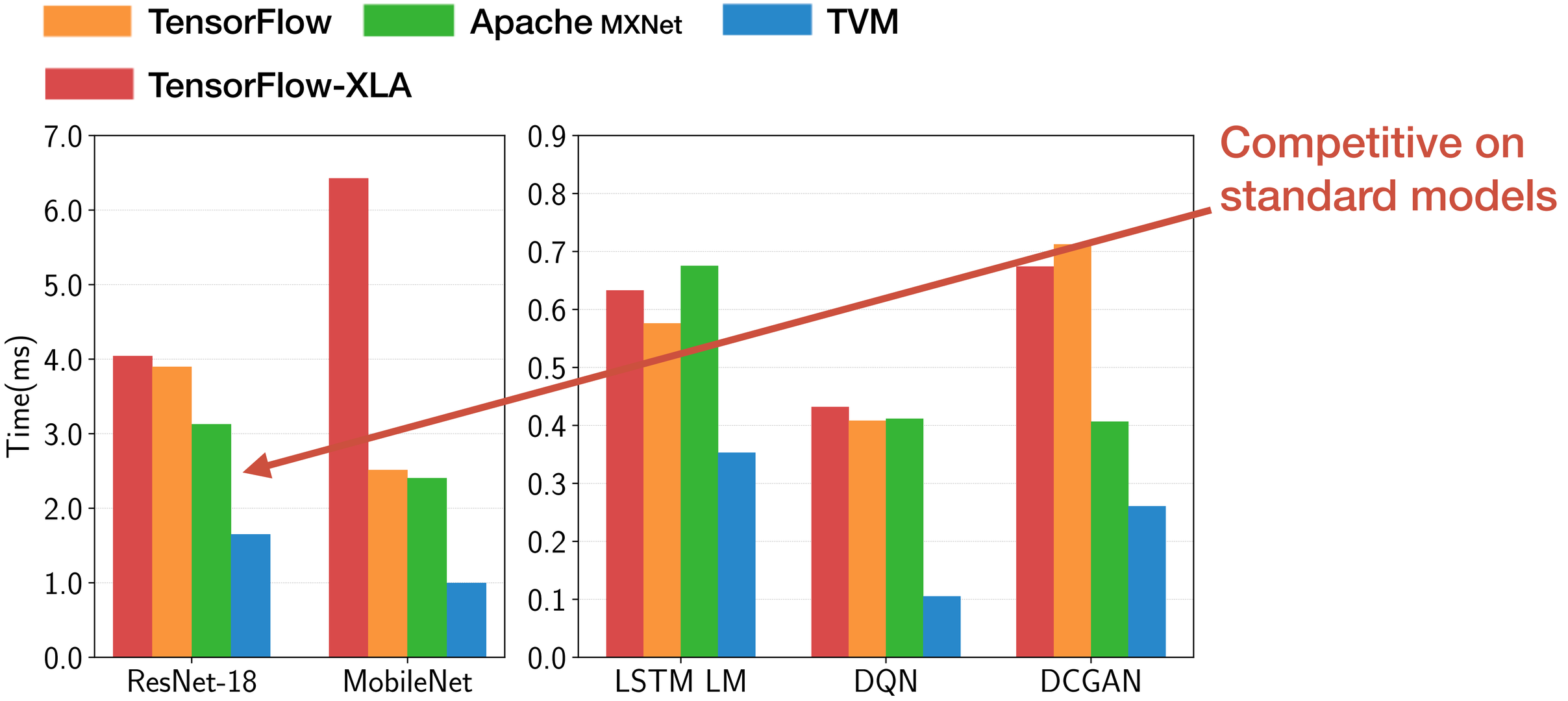
End to End Inference Performance (Nvidia Titan X)



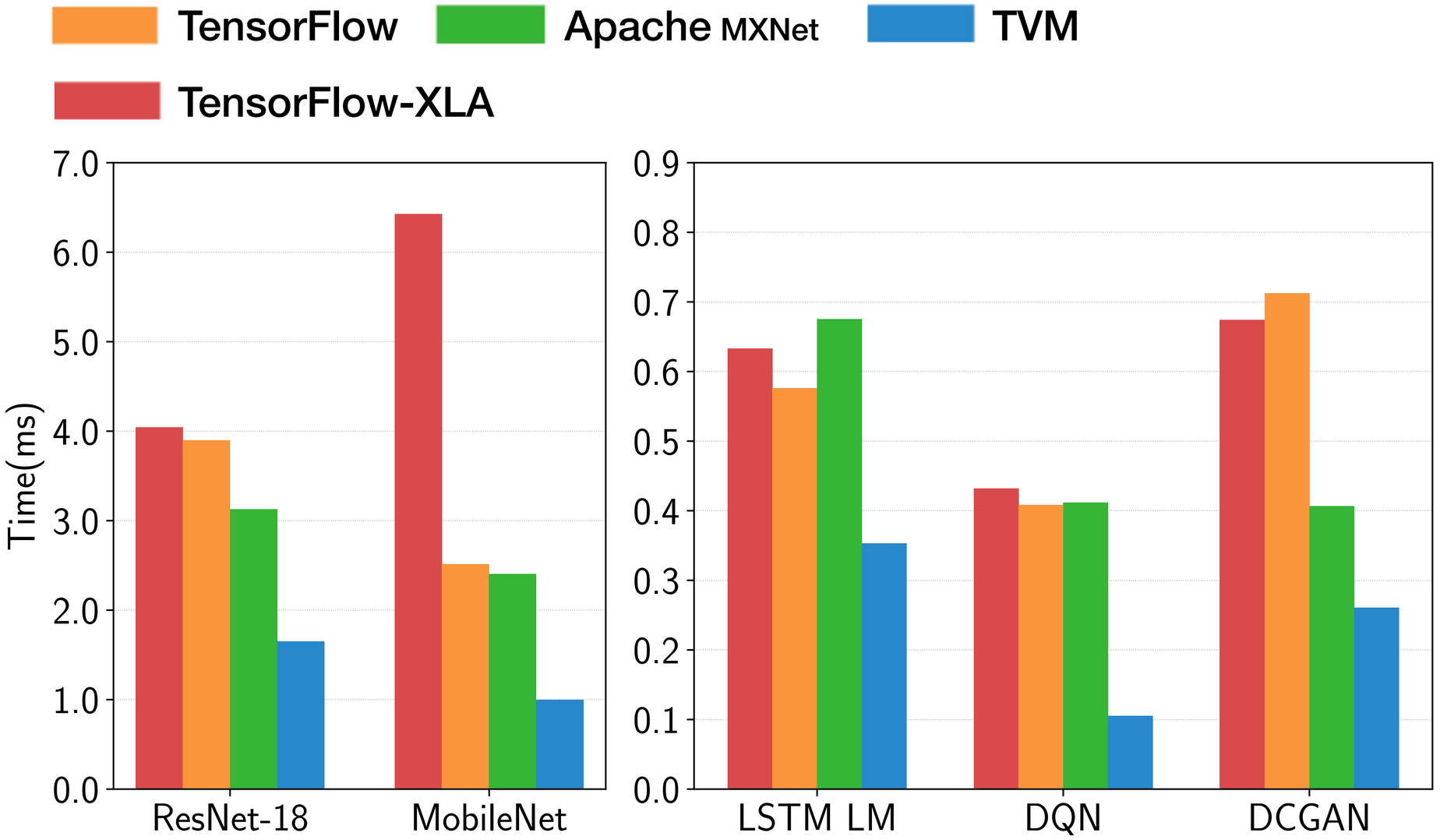
End to End Inference Performance (Nvidia Titan X)



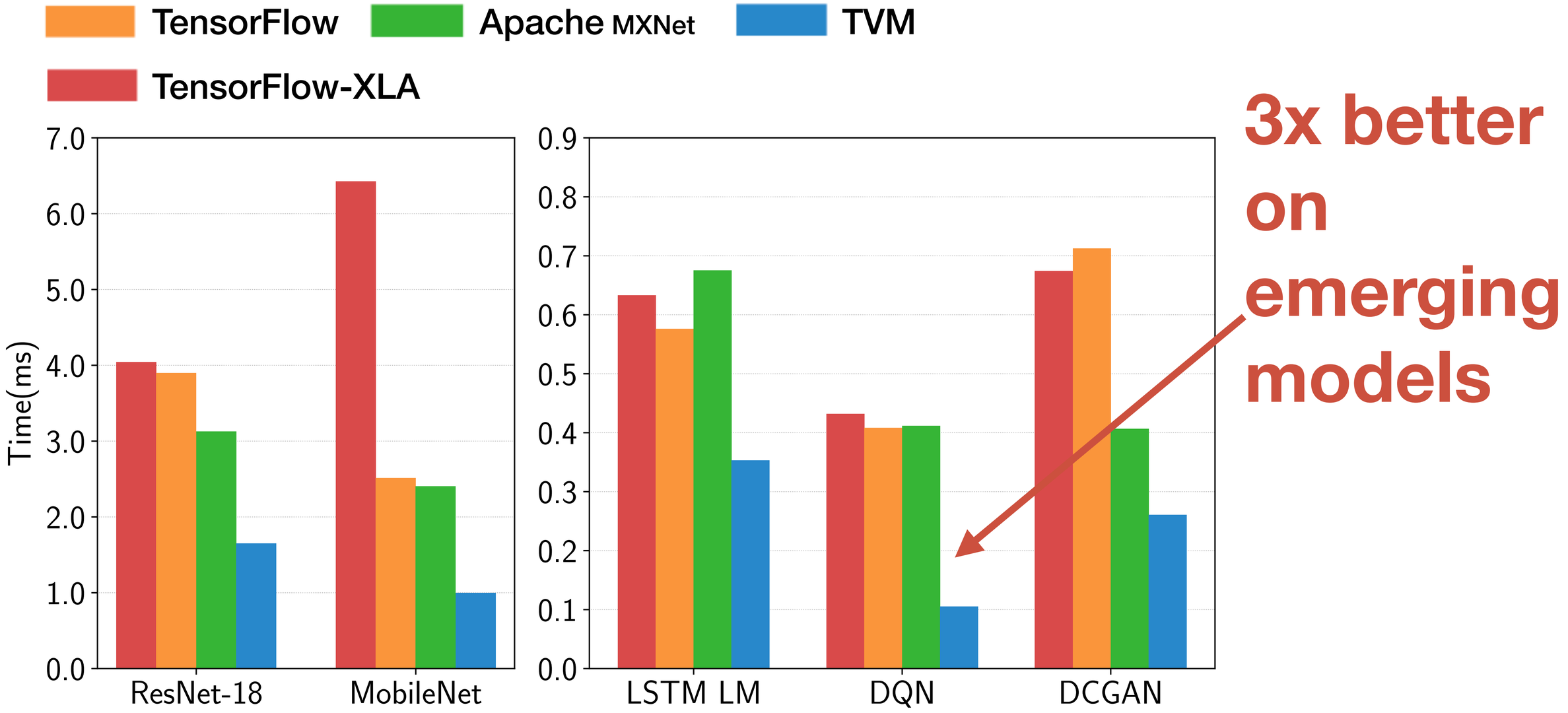
End to End Inference Performance (Nvidia Titan X)



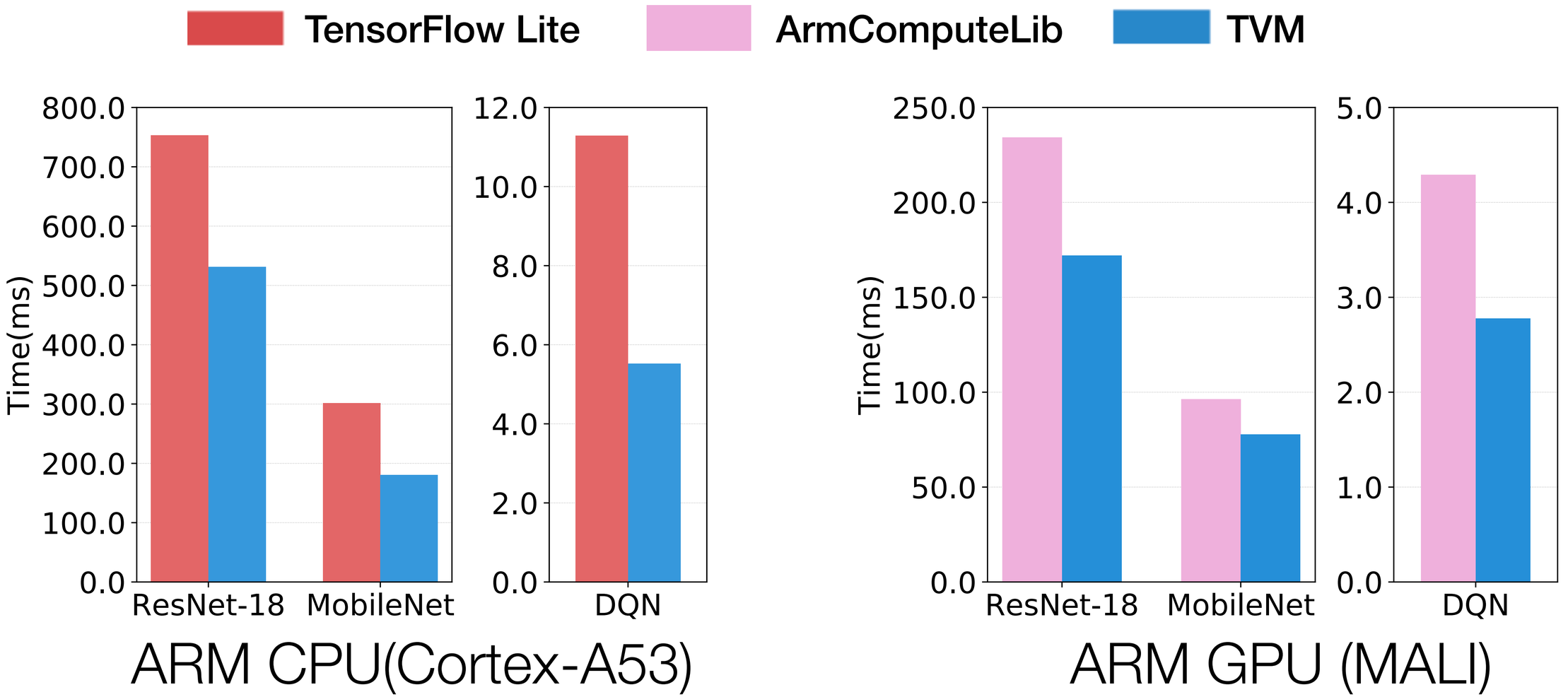
End to End Inference Performance (Nvidia Titan X)



End to End Inference Performance (Nvidia Titan X)

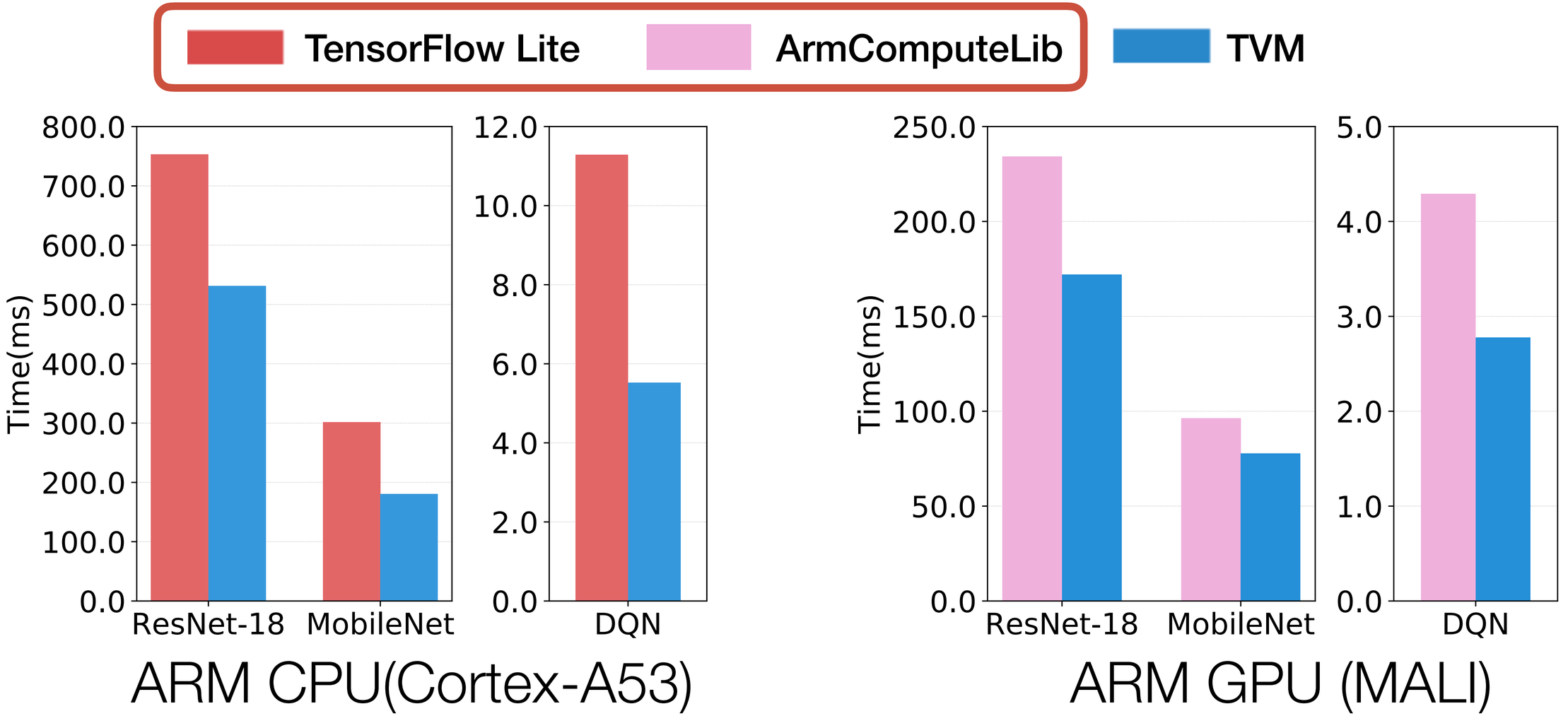


Portable Performance Across Hardware Platforms



Portable Performance Across Hardware Platforms

Special frameworks for the particular hardware platform



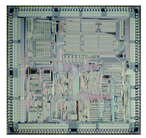
TVM: End to End Deep Learning Compiler

What about Accelerator Support?

VTA: Open & Flexible Deep Learning Accelerator

Current TVM Stack





Moreau, Chen, et al. work in progress

VTA: Open & Flexible Deep Learning Accelerator

Current TVM Stack

VTA MicroArchitecture

Moreau, Chen, et al. work in progress

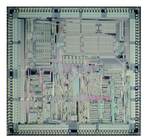
VTA: Open & Flexible Deep Learning Accelerator

Current TVM Stack

VTA Hardware/Software Interface (ISA)

VTA MicroArchitecture





Moreau, Chen, et al. work in progress

VTA: Open & Flexible Deep Learning Accelerator

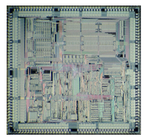
Current TVM Stack

VTA Runtime & JIT Compiler

VTA Hardware/Software Interface (ISA)

VTA MicroArchitecture





Moreau, Chen, et al. work in progress

VTA: Open & Flexible Deep Learning Accelerator

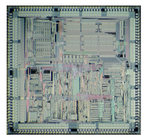
Current TVM Stack

VTA Runtime & JIT Compiler

VTA Hardware/Software Interface (ISA)

VTA MicroArchitecture

VTA Simulator



Moreau, Chen, et al. work in progress

VTA: Open & Flexible Deep Learning Accelerator

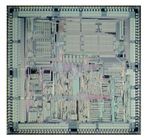
Current TVM Stack

VTA Runtime & JIT Compiler

VTA Hardware/Software Interface (ISA)

VTA MicroArchitecture

VTA Simulator



Moreau, Chen, et al. work in progress

- Runtime JIT compile accelerator micro code
- Support heterogenous devices, 10x better than CPU on the same board.
- Move hardware complexity to software

VTA: Open & Flexible Deep Learning Accelerator

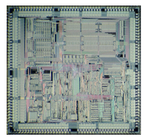
Current TVM Stack

VTA Runtime & JIT Compiler

VTA Hardware/Software Interface (ISA)

VTA MicroArchitecture

VTA Simulator



Moreau, Chen, et al. work in progress

- Runtime JIT compile accelerator micro code
- Support heterogenous devices, 10x better than CPU on the same board.
- Move hardware complexity to software

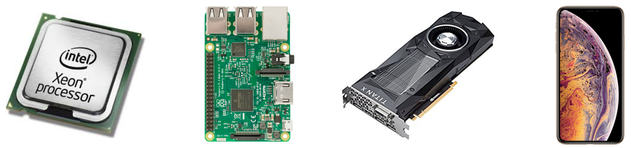
compiler, driver, hardware design full stack open source

TVM: End to End Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression and Optimization Search Space

LLVM, CUDA, Metal



Optimization

AutoTVM

Device Fleet

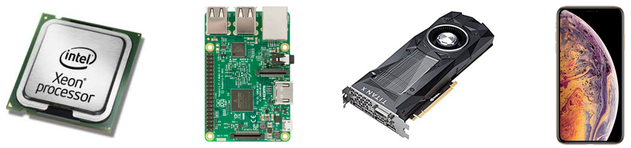
TVM: End to End Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression and Optimization Search Space

LLVM, CUDA, Metal

VTA



Edge FPGA

Cloud FPGA

ASIC

Optimization

AutoTVM

Device Fleet

Recent Developments

Community Highlights

More **Dynamism**

Tiny machine learning

Better core **Infra**

More Specialized **Accelerator Support**

Need for More Dynamism

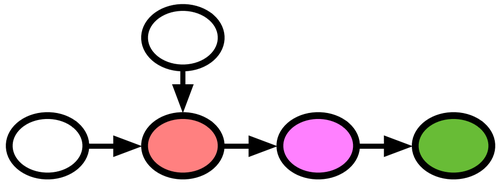
Model

Data

Need for More Dynamism

Model

static
computacional graph

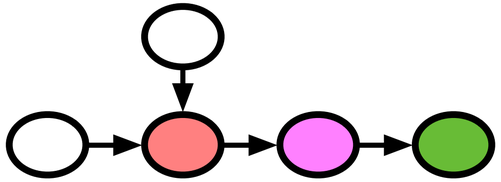


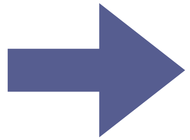
Data

Need for More Dynamism

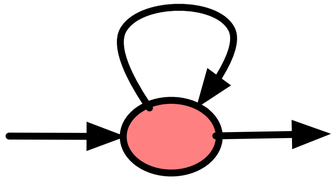
Model

static
computacional graph





program with
loops and recursions

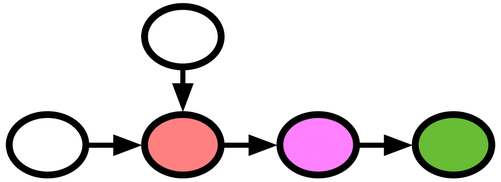


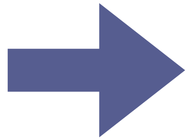
Data

Need for More Dynamism

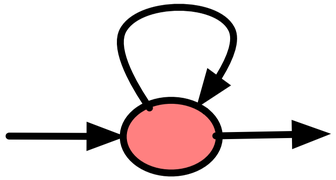
Model

static
computacional graph





program with
loops and recursions



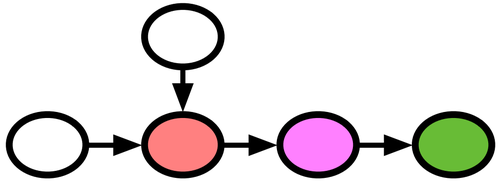
Data

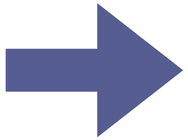
single tensor
with known shape

Need for More Dynamism

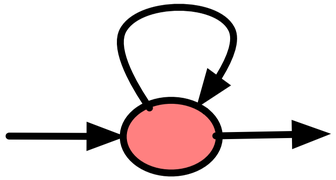
Model

static
computacional graph



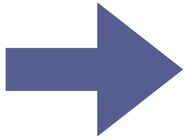


program with
loops and recursions

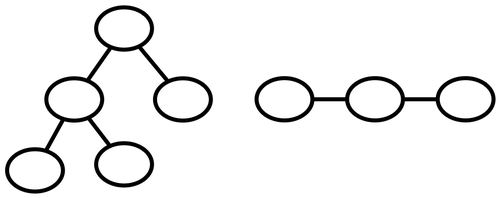


Data

single tensor
with known shape

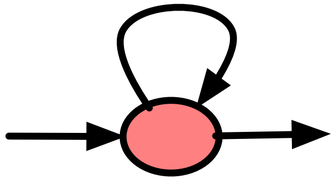


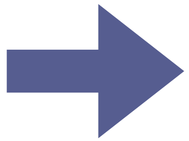
sequence, trees,
nested data structure



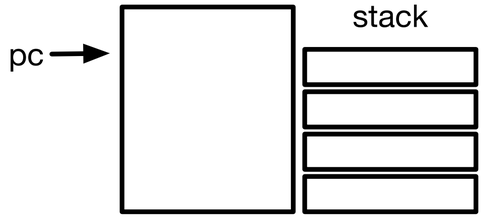
Relay Virtual Machine

source program





VM bytecode and runtime



Dynamic shape workloads

More runtime objects: Arrays, Tuples, Trees, ADTs

Minimum runtime for dynamic models

Credit: Jared Roesch, Haichen Shen et.al

Community Highlights

More **Dynamism**

Tiny machine learning

Better core **Infra**

More Specialized **Accelerator Support**

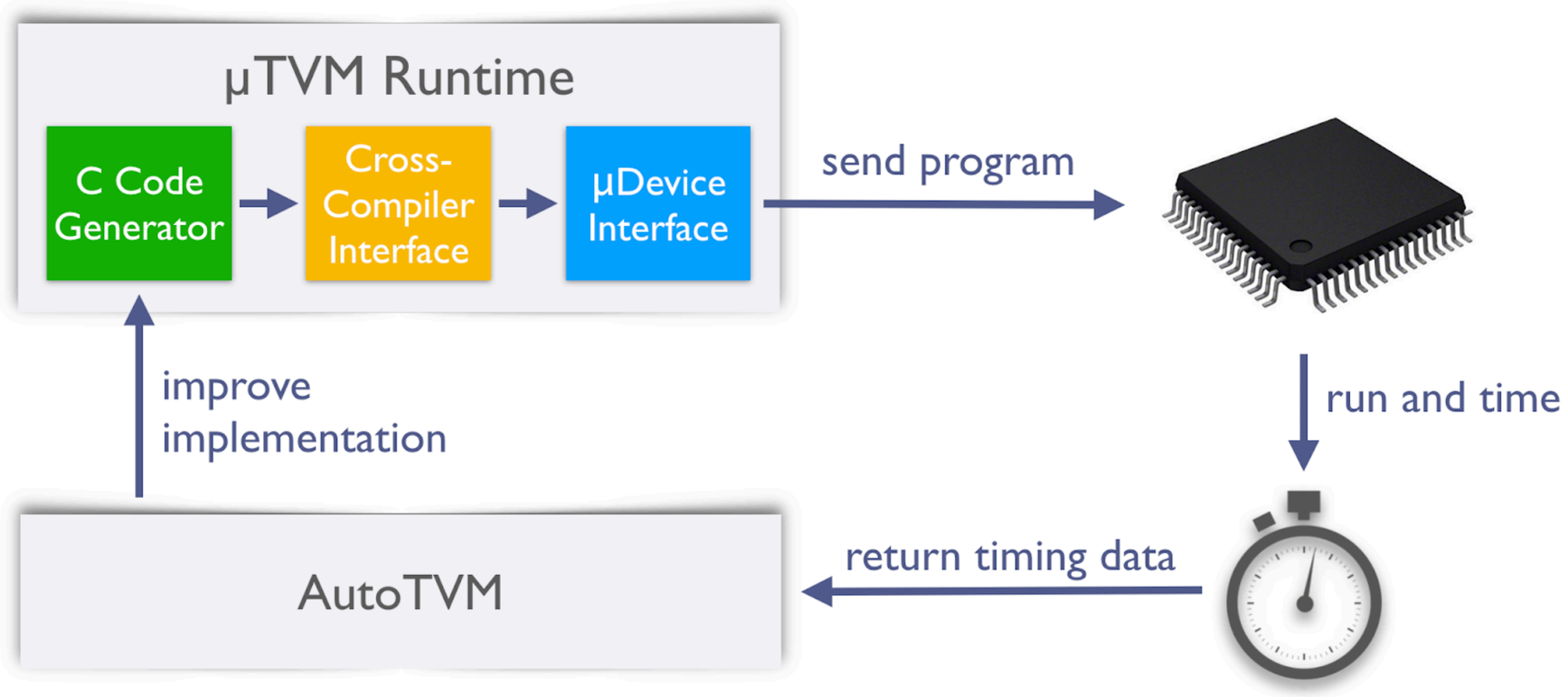
Machine Learning is Getting into Tiny Devices

Challenges: limited resources, OS support

uTVM: TVM on bare-metal Devices

Support bare-metal J-TAG devices, **no OS is needed**

ARM Cortex-M
RISC-V



Credit: Logan Weber et al

Community Highlights

More **Dynamism**

Tiny machine learning

Better core **Infra**

More Specialized **Accelerator Support**

Core Infrastructure

New integer simplification and analysis

Unified runtime object protocol

Core Infrastructure

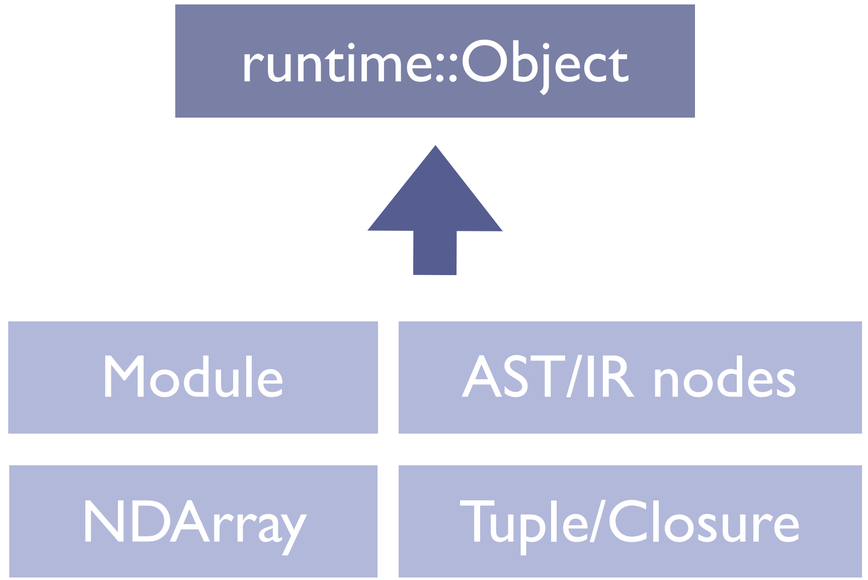
New integer simplification and analysis

Unified runtime object protocol

Core Infrastructure

New integer simplification and analysis

Unified runtime object protocol



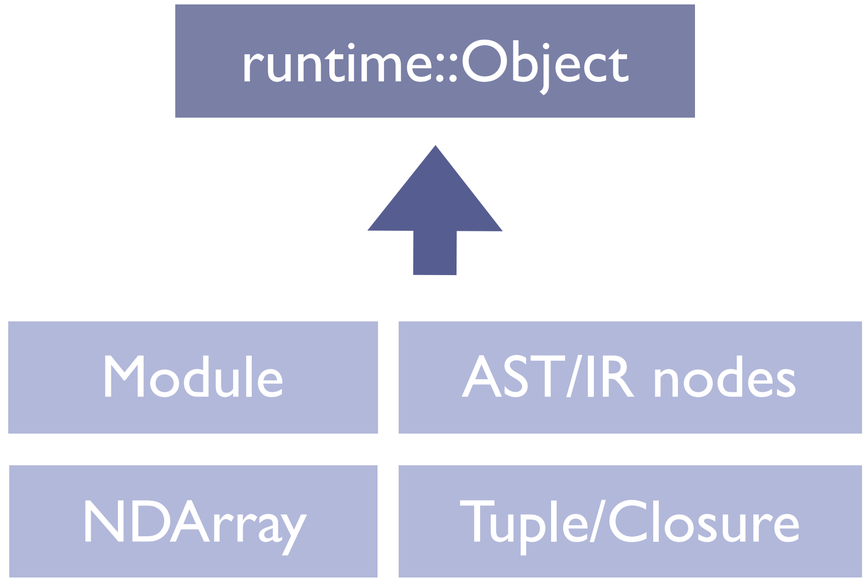
Core Infrastructure

New integer simplification and analysis

Unified runtime object protocol

Easy to add new objects (trees, graphs)

Cross language support



Community Highlights

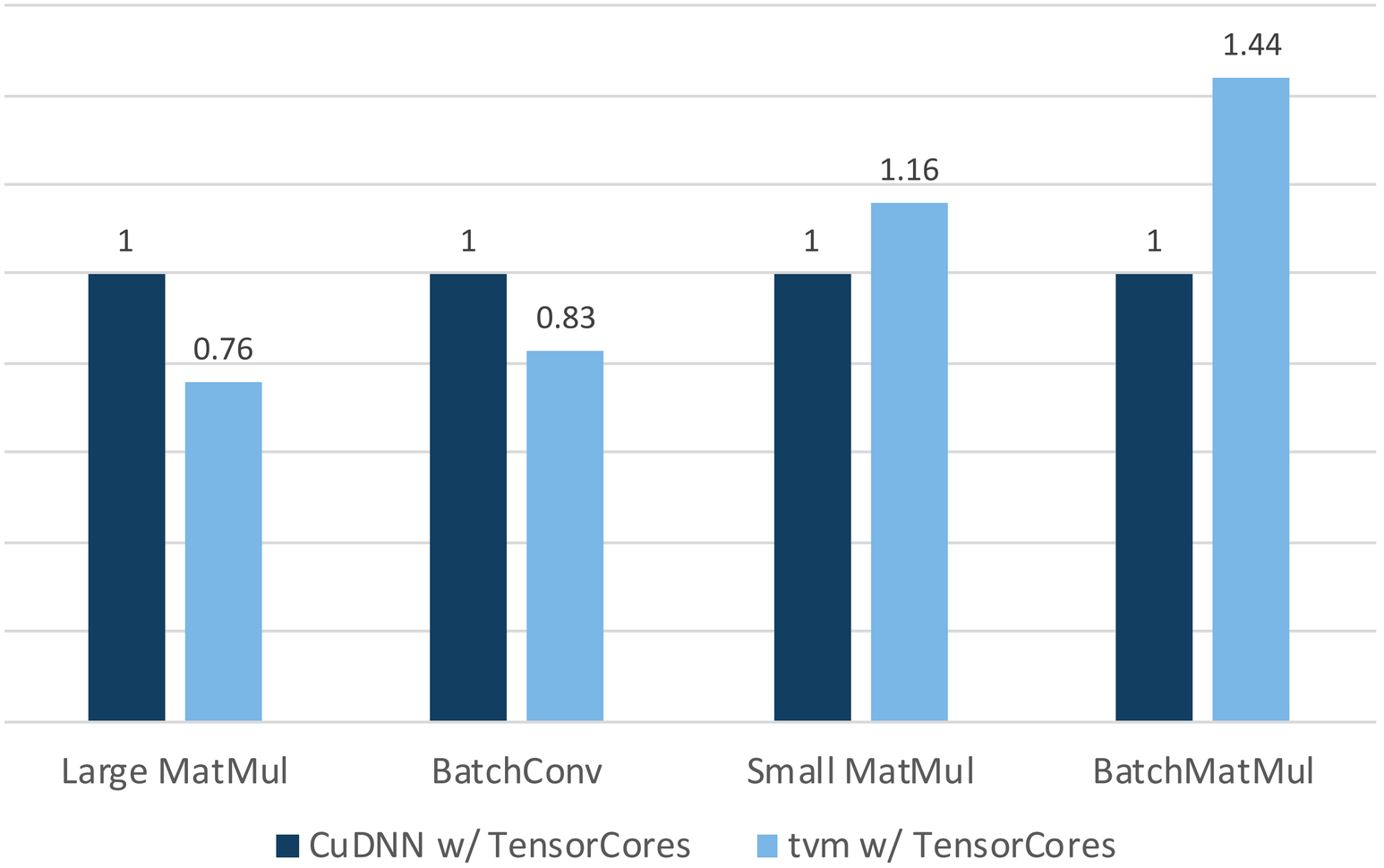
More **Dynamism**

Tiny machine learning

Better core **Infra**

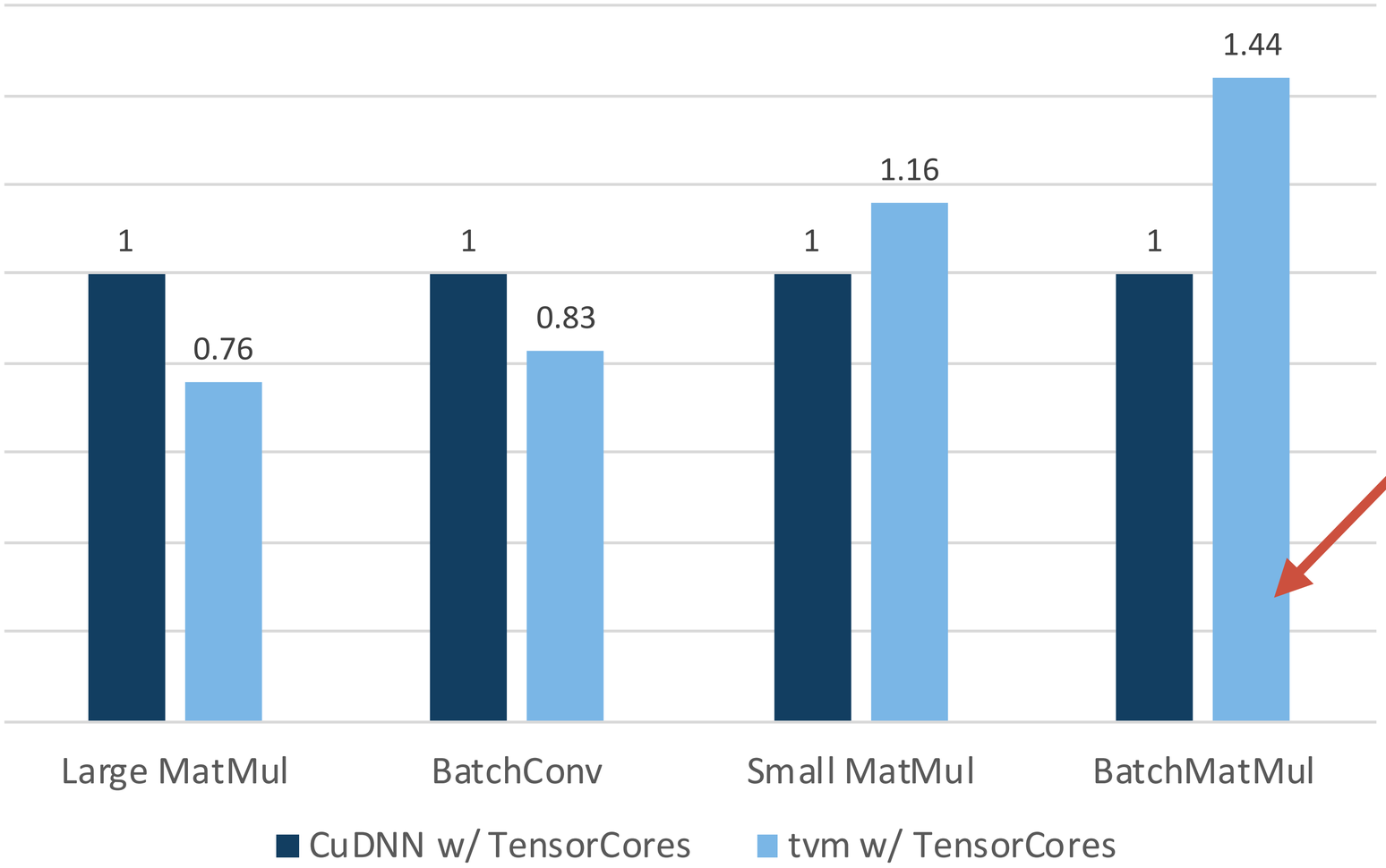
More Specialized **Accelerator Support**

TVM for TensorCore



Credit: Siyuan Feng

TVM for TensorCore



1.4x better on emerging workloads Transformer related workloads

Credit: Siyuan Feng

TSIM: Support for Future Hardware

Current TVM Stack

New NPU Runtime

TSIM Driver

Credit: Luis Vega, Thierry Moreau

TSIM: Support for Future Hardware

Current TVM Stack

New NPU Runtime

TSIM Driver

TSIM Binary

New Hardware Design in Verilog

Verilator

Credit: Luis Vega, Thierry Moreau

TSIM: Support for Future Hardware

Current TVM Stack

New NPU Runtime

TSIM Driver

TSIM Binary

New Hardware Design in Verilog

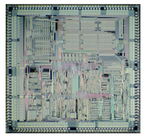
Verilator

Credit: Luis Vega, Thierry Moreau

Unified Runtime For Heterogeneous Devices

Device Drivers

NPU Driver



CUDA Driver

External Runtimes

TensorRT

Unified Runtime For Heterogeneous Devices

tvm::runtime::Module

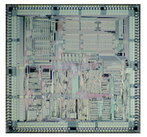
Runtime Module Interface

GetFunction(string) -> tvm::runtime::PackedFunc

SaveToBinary/LoadFromBinary

Device Drivers

NPU Driver



CUDA Driver

External Runtimes

TensorRT

Unified Runtime For Heterogeneous Devices

tvm::runtime::Module

Runtime Module Interface

GetFunction(string) -> tvm::runtime::PackedFunc

SaveToBinary/LoadFromBinary

NPUModule

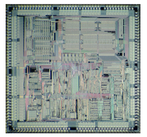
CUDAModule

TFModule

NPU Driver

CUDA Driver

Device Drivers



External Runtimes

TensorRT

Unified Runtime For Heterogeneous Devices

Runtime Module Interface

GetFunction(string) -> tvm::runtime::PackedFunc

SaveToBinary/LoadFromBinary

tvm::runtime::Module

Subclasses

NPUModule

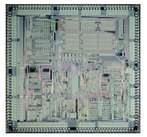
CUDAModule

TFModule

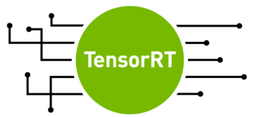
NPU Driver

CUDA Driver

Device Drivers



External Runtimes



Unified Runtime Benefit

Unified library packaging

```
mod.export_library("mylib.so")
```

Free API (Py/Java/Go)

```
lib = tvm.module.load("mylib.so")
func = lib["npufunction0"]
func(a, b)
```

Automatic RPC Support

```
remote = tvm.rpc.connect(board_url, port)
remote.upload("mylib.so")
remote_mod = remote.load_module(“mylib.so")
func = remote_mod[“npufunction0"]
func(remote_a, remote_b)
```

Community

Open Source Community

Incubated as Apache TVM. Independent governance, allowing competitors to collaborate.

Open Source Community

Incubated as Apache TVM. Independent governance, allowing competitors to collaborate.

Open Source Code

Open Development

Open Governance

Open Source Community

Incubated as Apache TVM. Independent governance, allowing competitors to collaborate.

Open Source Community

Incubated as Apache TVM. Independent governance, allowing competitors to collaborate.

Growing Developer Community

22 committers, 47 reviewers, 295 contributors

Open Source Community

Incubated as Apache TVM. Independent governance, allowing competitors to collaborate.

Growing Developer Community

22 committers, 47 reviewers, 295 contributors

~70% growth since TVM Conf 2018

Open Source Community

Incubated as Apache TVM. Independent governance, allowing competitors to collaborate.

Growing Developer Community

22 committers, 47 reviewers, 295 contributors

~70% growth since TVM Conf 2018

Monthly Statistics

~50 authors, ~140 PRs, ~1000 discuss forum posts

More on TVM Community Conference

TensorCore

Driving

Quantization

Binary Neural Networks

ARM

Sparse

Dynamic Workloads

Automation

DSPs

RISCV

…

Videos and slides available at http://tvmconf.org/

Open Source Community

Open source: ~280 contributors from UW, Berkeley, Cornell, UCLA, Amazon, Huawei, NTT, Facebook, Microsoft, Qualcomm, Alibaba, Intel, …

Open Source Community

Open source: ~280 contributors from UW, Berkeley, Cornell, UCLA, Amazon, Huawei, NTT, Facebook, Microsoft, Qualcomm, Alibaba, Intel, …

Used in production

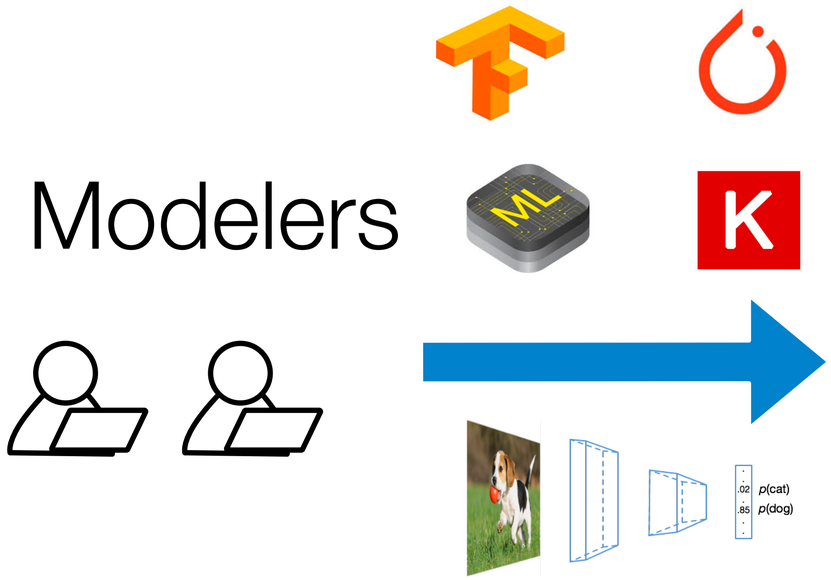
Open Source Community

Open source: ~280 contributors from UW, Berkeley, Cornell, UCLA, Amazon, Huawei, NTT, Facebook, Microsoft, Qualcomm, Alibaba, Intel, …

Used in production

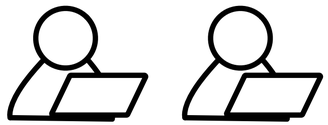
Incubated as Apache TVM recently. Independent governance, allowing competitors to collaborate.

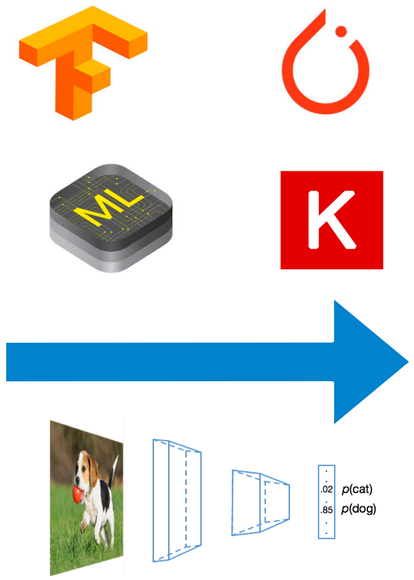
ML Deployment Flow with Automated Compiler



ML Deployment Flow with Automated Compiler

Modelers



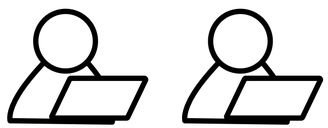


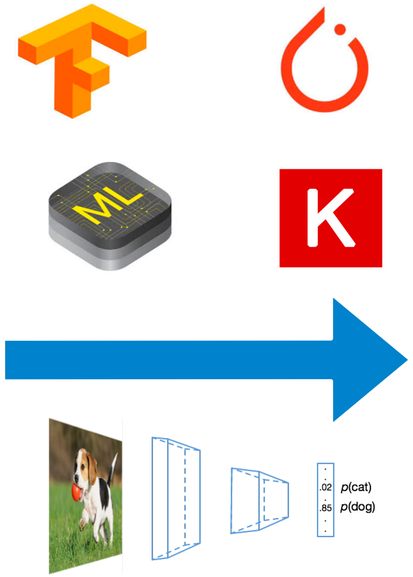
tvm

ML Compilation

ML Deployment Flow with Automated Compiler

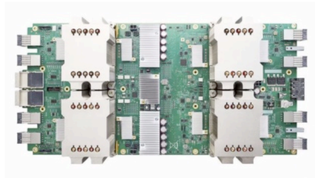
Modelers





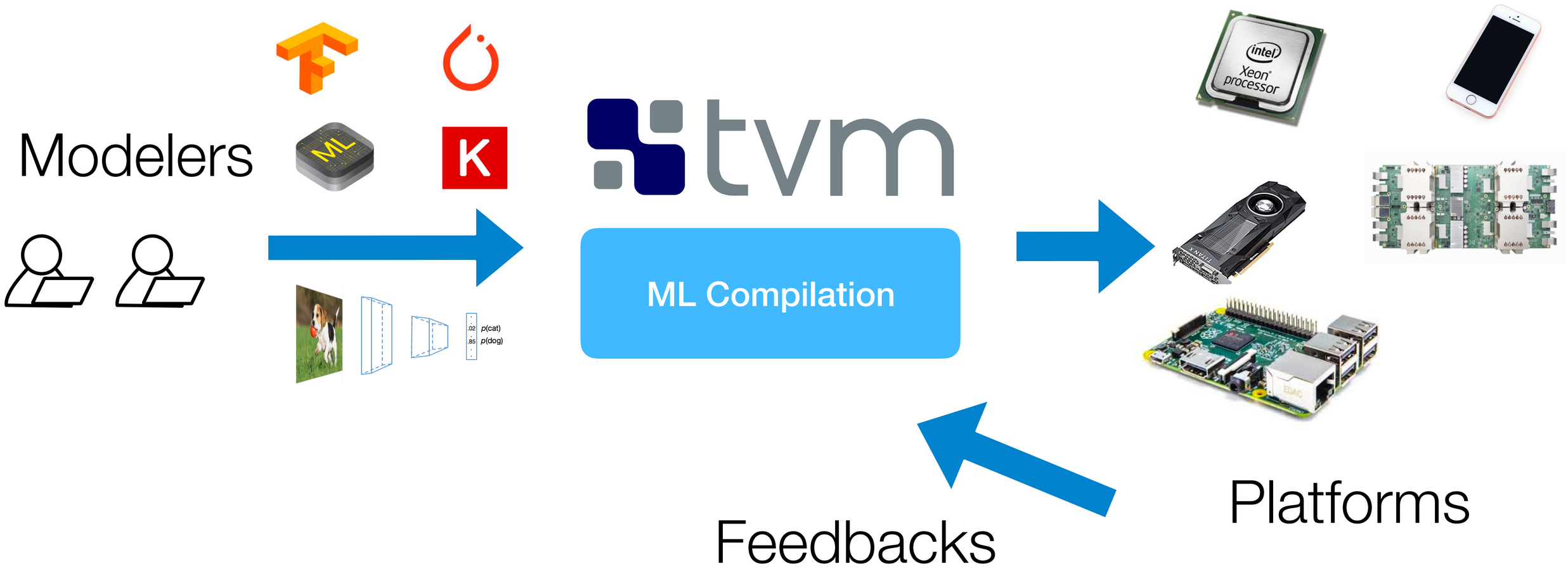
tvm

ML Compilation

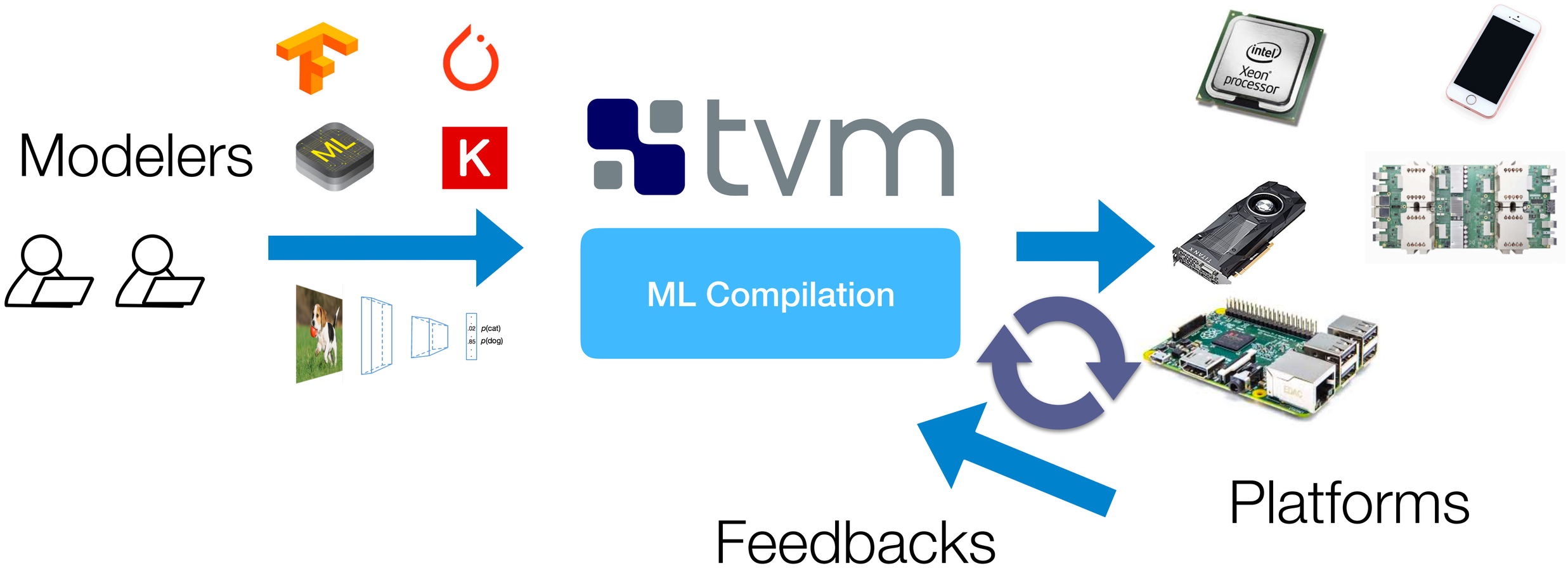


Platforms

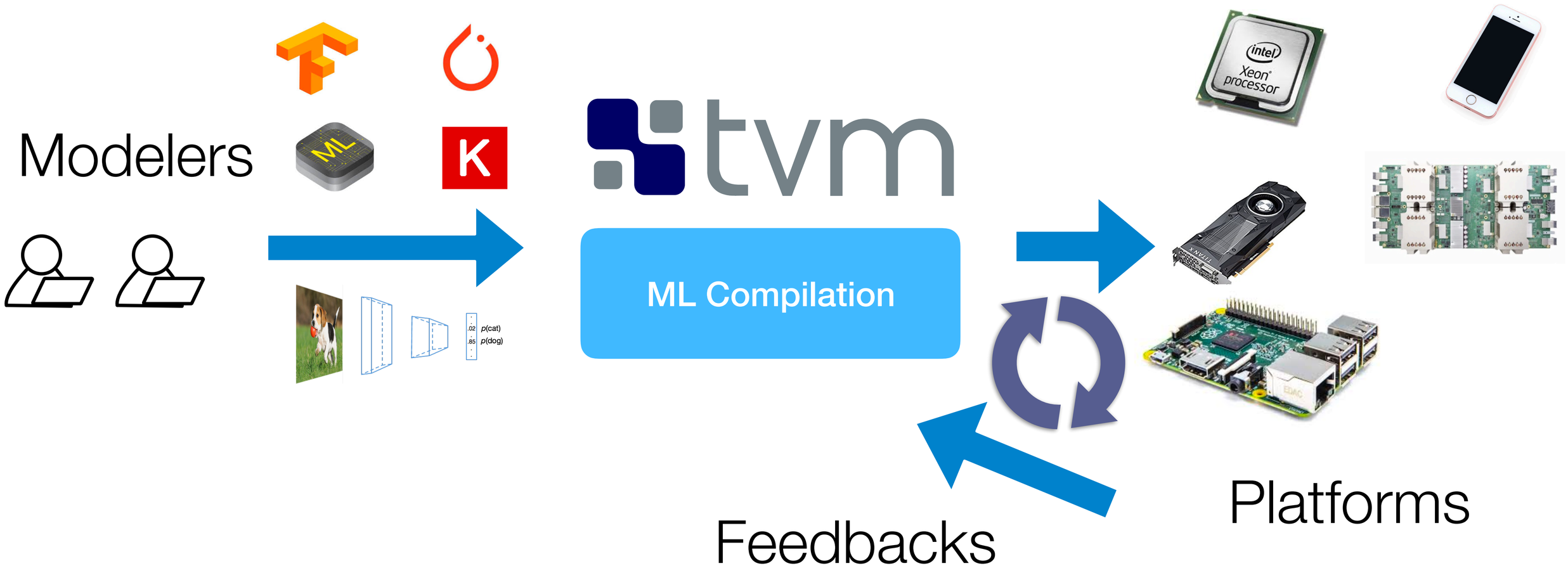
ML Deployment Flow with Automated Compiler



ML Deployment Flow with Automated Compiler



ML Deployment Flow with Automated Compiler



Full Stack Learning-based Learning System

High-Level Differentiable IR

Tensor Expression IR

LLVM, CUDA | VTA

Edge FPGA | Cloud FPGA | ASIC

Full Stack Learning-based Learning System

High-Level Differentiable IR

Tensor Expression IR

LLVM, CUDA | VTA

Edge FPGA | Cloud FPGA | ASIC

ML Models

System Optimizations

Hardware Design

Full-stack co-design

ML as the first-class citizen