Machine Learning is Everywhere
Machine Learning is Everywhere
Machine Learning is Everywhere
Machine Learning is Everywhere
Abbreviated History of Machine Learning

Based on personal view
Abbreviated History of Machine Learning

1960

Based on personal view
Abbreviated History of Machine Learning

1960

Research

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

1960
- SVM
- ConvNet
- GBM

2000
- MTurk
- kaggle
- IMAGENET

Based on personal view
Abbreviated History of Machine Learning

1960

1960

2000

2006

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

Public Cloud

2006

Hardware

Based on personal view
Abbreviated History of Machine Learning

1960

SVM
ConvNet
GBM

LSTM

Research

1960

flickr
Google
Facebook
Wikipedia
Netflix
kaggle

Public
Cloud

Data

2000

MTurk

IMAGENET

Hardware

2006

NVIDIA
CUDA

Based on personal view
Abbreviated History of Machine Learning

1960

- SVM
- ConvNet
- LSTM
- GBM

2000

- Data
  - flickr
  - Google
  - Wikipedia
  - Netflix
  - kaggle
  - MTurk
  - ImageNet

- Hardware
  - Public Cloud
  - NVIDIA CUDA

2006

Based on personal view
Abbreviated History of Machine Learning

1960
- SVM
- ConvNet
- LSTM
- GBM

2000
- MTurk
- flickr
- Google
- Wikipedia
- Netflix
- kaggle
- Public Cloud
- IMAGENET

2006
- NVIDIA CUDA
- Hardware

2010
- ML Systems

Based on personal view
Abbreviated History of Machine Learning

1960

SVM
ConvNet
LSTM
GBM

Research

2000

Data

flickr
Google
Wikipedia
Netflix
kaggle
MTurk

Public Cloud

Hardware

2006

IMAGENET

2010

ML Systems

Based on personal view
Based on personal view
Abbreviated History of Machine Learning

1960
- **Research**
  - SVM
  - ConvNet
  - LSTM
  - GBM

2000
- **Data**
  - flickr
  - Google
  - Wikipedia
  - Netflix
  - kaggle
  - IMAGENET

- **Hardware**
  - MTurk

2006
- **Public Cloud**
  - NVIDIA CUDA

2010
- **ML Systems**
  - GraphLab
  - Apache Spark

Based on personal view
Abbreviated History of Machine Learning

1960
- SVM
- ConvNet
- LSTM
- GBM

2000
- MTurk
- Wikipedia
- IMAGENET
- Public Cloud
- NVIDIA CUDA

2006
- GraphLab
- Spark
- XGBoost

2010
- Data
- Hardware
- ML Systems
- Research

Based on personal view
### Abbreviated History of Machine Learning

<table>
<thead>
<tr>
<th>Year</th>
<th>Research</th>
<th>Data</th>
<th>Hardware</th>
<th>ML Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>1960</td>
<td>SVM, ConvNet, LSTM, GBM</td>
<td>flickr, Google, Wikipedia, Netflix, kaggle, MTurk, IMAGENET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td>Public Cloud, NVIDIA, CUDA, IMAGENET</td>
<td></td>
<td>GraphLab, Apache Spark, dmic, XGBoost</td>
</tr>
<tr>
<td>2006</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td></td>
<td></td>
<td>MXNet</td>
</tr>
</tbody>
</table>

Based on personal view
Abbreviated History of Machine Learning

1960
- SVM
- ConvNet
- GBM
- LSTM
- Research

2000
- flickr
- Google
- Wikipedia
- Netflix
- Kaggle
- MTurk
- IMAGENET
- Data

2006
- Public Cloud
- NVIDIA CUDA
- Hardware

2010
- GraphLab
- Apache Spark
- XGBoost
- TensorFlow
- ML Systems

Based on personal view
Learning Systems

- **dmilc XGBoost**
  - Data science for everyone

- **mxnet**
  - Scale up deep learning

- **tvm**
  - Deploy AI everywhere
Learning Systems

Data science for everyone

Scale up deep learning

Deploy AI everywhere

Accessible and scalable learning systems
Learning Systems

- Data science for everyone
- Scale up deep learning
- Deploy AI everywhere

Accessible and scalable learning systems
Current Learning Systems

Systems
Current Learning Systems

Systems \(\rightarrow\) Enables \(\rightarrow\) Machine Learning
Current Learning Systems

Systems → Enables → Machine Learning

Optimize
Current Learning Systems

Systems

Enables

Machine Learning

Optimize

Me
Current Learning Systems

Can we use ML to optimize systems?
Learning-based Learning Systems

Enables

Systems

Machine Learning
Learning-based Learning Systems

Systems

Enables

Optimizes

Machine Learning
Learning-based Learning Systems

Systems

Enables

Optimizes

Machine Learning
My Research

Data science for everyone
Scale up deep learning
Deploy AI everywhere

TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. **Chen et al. OSDI 18**

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

Data science for everyone

Scale up deep learning

Deploy AI everywhere

TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. Chen et al. OSDI 18

Learning to Optimize Tensor Programs. Chen et al. NeurIPS 18
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
Problem: Deep Learning Deployment

Model

Hardware
Backends
Problem: Deep Learning Deployment

Model

Deploy

Hardware

Backends
How did I Start to Work on This Problem?
How did I Start to Work on This Problem?
How did I Start to Work on This Problem?
How did I Start to Work on This Problem?

(a) Blocked convolution program with multiple thread contexts

(b) Convolution micro-coded program

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

Hardware
Deploy Deep Learning Everywhere

Huge gap between model/frameworks and hardware backends
Existing Deep Learning Frameworks

Frameworks: TensorFlow, PyTorch, MXNet, Caffe, Keras, Theano

Hardware: NVIDIA (GPGPU)
Existing Deep Learning Frameworks

Frameworks

High-level data flow graph

Hardware
Existing Deep Learning Frameworks

- High-level data flow graph
- Primitive Tensor operators such as Conv2D

Frameworks: TensorFlow, ML, PyTorch, Keras, MxNet

Hardware: NVIDIA GPU
Existing Deep Learning Frameworks

Frameworks

High-level data flow graph

Primitive Tensor operators such as Conv2D

eg. cuDNN

Offload to heavily optimized DNN operator library

Hardware

NVIDIA
Limitations of Existing Approach

Frameworks

cuDNN

NVIDIA
Limitations of Existing Approach

cuDNN

Frameworks

NVIDIA
Limitations of Existing Approach

cuDNN

Frameworks

[Images of various frameworks and NVIDIA GPU]
Limitations of Existing Approach

Frameworks

cuDNN

NVIDIA
Limitations of Existing Approach

Frameworks

- cuDNN

Limitations of Existing Approach

New operator introduced by operator fusion optimization potential benefit: 1.5x speedup
Limitations of Existing Approach

Frameworks

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

cuDNN

NVIDIA
Limitations of Existing Approach

New operator introduced by operator fusion optimization potential benefit: 1.5x speedup
Limitations of Existing Approach

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

Engineering intensive
Learning-based Learning System

Frameworks

High-level data flow graph and optimizations

Hardware
Learning-based Learning System

Frameworks

High-level data flow graph and optimizations

Hardware
Learning-based Learning System

- Frameworks
  - TensorFlow
  - PyTorch
  - MXNet
  - Caffe
  - Keras
  - Mlflow

- High-level data flow graph and optimizations

- Machine Learning based Program Optimizer

- Directly generate optimized program for new operator workloads and hardware

- Hardware
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)

\[ C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \ast B[k, x], \text{axis}=k)) \]
Problem Setting

Tensor Expression (Specification)

\[ C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \ast B[k, x], \text{axis}=k)) \]

Search Space of Possible Program Optimizations
Problem Setting

Tensor Expression (Specification)

\[ C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k)) \]

Search Space of Possible Program Optimizations

Low-level Program Variants
Example Instance in a Search Space

Tensor Expression (Specification)

```python
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)

\[
C = \text{tvm.compute}((m, n),
\text{lambda } y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k))
\]

Search Space of Possible Program Optimizations

Vanilla Code

```python
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)

```python
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

```python
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)

\[ C = \text{tvm.compute}((m, n), \]
\[ \quad \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k) \]

Search Space of Possible Program Optimizations

Map to Accelerators

```python
inp_buffer AL[8][8], BL[8][8]
acc_buffer CL[8][8]
for yo in range(128):
  for xo in range(128):
    vdl.fill_zero(CL)
    for ko in range(128):
      vdl.dma_copy2d(AL, A[ko*8:ko*8+8][yo*8:yo*8+8])
      vdl.dma_copy2d(BL, B[ko*8:ko*8+8][xo*8:xo*8+8])
      vdl.fused_gemm8x8_add(CL, AL, BL)
      vdl.dma_copy2d(CL[yo*8:yo*8+8,xo*8:xo*8+8], CL)
```
Optimization Choices in a Search Space

Tensor Expression (Specification)

\[ C = \text{tvm.compute)((m, n),} \]
\[ \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k)) \]

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)

```python
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)

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

Billions of possible optimization choices

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

\( e \) Expression

\( S_e \) Search Space
Problem Formalization

$\mathcal{S}_e$  
Expression  
\rightarrow  
AutoTVM  

$e$  
Search Space
Problem Formalization

Expression $e$ and Search Space $S_e$ are input to AutoTVM, which provides configuration $c$. The configuration is then used by the Code Generator to produce the result $x = g(e, c)$. This process is part of the optimization configuration.
Problem Formalization

\[ x = g(e, c) \]

Expression \( e \)

Search Space \( S_e \)

AutoTVM

Optimization Configuration

Code Generator

Program

\[ f(x) \]

Cost: Execute Time
Problem Formalization

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

$e$

Expression

$S_e$

Search Space

AutoTVM

Code Generator

$x$

$f(x)$
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

\[ e \rightarrow \text{Expression} \rightarrow \text{AutoTVM} \]

\[ S_e \rightarrow \text{Search Space} \]
Cost-model Driven Approach

Use cost model to pick configuration
Cost-model Driven Approach

Use cost model to pick configuration

\[ f(e, c) \]

\[ c \]

\[ \hat{f}(e, c) \]

Expression

Search Space

AutoTVM

Cost Model
Cost-model Driven Approach

Use cost model to pick configuration

\[ \hat{f}(e, c) \]

\[ S_e \]

\[ e \]

Expression

Search Space

Cost Model

AutoTVM

Code Generator

\[ x \]
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

Expression $e$

Search Space $S_e$

AutoTVM

Code Generator

Training data $D$

$f(x)$

$x$

$c$
Use machine learning to learn a statistical cost model

Statistical Cost Model

Expression

AutoTVM

Code Generator

\( \hat{f}(e, c) \)

\( c \)

\( f(x) \)

\( \mathcal{D} \)

Training data

\( e \)

Search Space

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

Expression $e$ and Search Space $S_e$ are inputs to the AutoTVM, which generates code $c$. The Statistical Cost Model learns $f(e, c)$ from the training data $D$. The Code Generator then generates code $x$.

Benefit: Automatically adapt to hardware type

Challenge: How to design the cost model
Unique Problem Characteristics

\[ e \rightarrow \text{Expression} \rightarrow \text{AutoTVM} \rightarrow c \rightarrow \text{Code Generator} \rightarrow x \rightarrow \text{Code Generator} \rightarrow f(x) \]

\[ S_e \rightarrow \text{Search Space} \rightarrow \text{AutoTVM} \rightarrow \hat{f}(e, c) \rightarrow \text{Statistical Cost Model} \rightarrow \text{Training data} \]

learning
Unique Problem Characteristics

Parameter $e$, Search Space $S_e$, Expression $E$, Code Generator $C_g$, Code $x$, Learning $\hat{f}(e, c)$, Cost Model $f(x)$, Training Data $D$

Relatively low experiment cost
Unique Problem Characteristics

- **Expression**
- **Search Space**
- **AutoTVM**
- **Code Generator**
- **Statistical Cost Model**

- Relatively low experiment cost
- Program-aware modeling

\[ f(x) \]

Training data: \( \mathcal{D} \)

\[ \hat{f}(e, c) \]

Learning process:

- \( e \)
- \( S_e \)
- \( c \)
- \( x \)
Unique Problem Characteristics

- Expression
- Search Space
- AutoTVM
- Code Generator
- Statistical Cost Model

\[ f(e, c) \]

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

- High-level Configurations
  - size/order of loops,
  - threading choices,
  - memory scope,
  ...
Vanilla Cost Modeling

- High-level Configurations
  - Flatten as feature vector
  - Feature Vector
  - size/order of loops, threading choices, memory scope, ...

$C$
Vanilla Cost Modeling

$C$

High-level Configurations

Flatten as feature vector

Feature Vector

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

Your Favorite Model

+ 

Feature Vector

+
Vanilla Cost Modeling

$C$  
High-level Configurations

Flatten as feature vector  

Feature Vector

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

Your Favorite Model

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):
        for $k$ in range(8):
            $C[y][x] = 0$
            $C[y][x] += A[k][y] \times B[k][x]$

Low-level Abstract Syntax Tree (AST)
Program-aware Modeling: Tree-based Approach

High-level Configurations

```python
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)

Statistical features of AST

```
<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>x</td>
<td>8</td>
<td>8</td>
<td>64</td>
</tr>
<tr>
<td>k</td>
<td>1</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>k</td>
<td>64</td>
<td></td>
</tr>
</tbody>
</table>
```

 touched memory
outer loop length
Program-aware Modeling: Tree-based Approach

High-level Configurations

```python
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)

```
+---
   +---
      +---
        +---
          +---
            +---
              +---
                +---
                  +---
                    +---
                      +---
                        +---
                          +---
                            +---
```

Statistical features of AST

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>x</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>k</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>64</td>
<td></td>
</tr>
</tbody>
</table>

Your Favorite Model
Program-aware Modeling: Tree-based Approach

High-level Configurations

Low-level Abstract Syntax Tree (AST)

Statistical features of AST

Your Favorite Model

Benefit: low-level AST is a common representation (task invariant)
Program-aware Modeling: Neural Approach

\[
C[y][x] = 0 \\
\text{for } k \text{ in range}(8): \\
\quad C[y][x] += A[k][y] \times B[k][x]
\]
Program-aware Modeling: Neural Approach

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

context vec of y
context vec of x
context vec of k

soft scatter

for

final embedding
## Comparisons of Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Task Invariant</th>
<th>Time Cost</th>
<th>Predictive Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla Model</td>
<td>No</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Tree-based Model</td>
<td>Yes</td>
<td>Low</td>
<td>Good</td>
</tr>
<tr>
<td>Neural Model</td>
<td>Yes</td>
<td>High</td>
<td>Good</td>
</tr>
</tbody>
</table>
## Comparisons of Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Task Invariant</th>
<th>Time Cost</th>
<th>Predictive Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla Model</td>
<td>No</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Tree-based Model</td>
<td>Yes</td>
<td>Low</td>
<td>Good</td>
</tr>
<tr>
<td>Neural Model</td>
<td>Yes</td>
<td>High</td>
<td>Good</td>
</tr>
</tbody>
</table>

Choose tree-based model by default
Effectiveness of ML based Model
Effectiveness of ML based Model

One Conv2D Layer of ResNet18 on Titan X
Effectiveness of ML based Model

One Conv2D Layer of ResNet18 on Titan X
Effectiveness of ML based Model

Number of Trials

Baseline: CuDNN

One Conv2D Layer of ResNet18 on Titan X
Effectiveness of ML based Model

One Conv2D Layer of ResNet18 on Titan X

Baseline: CuDNN
AutoTVM: Black-box Optimization
Effectiveness of ML based Model

One Conv2D Layer of ResNet18 on Titan X
### Comparisons of Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Task Invariant</th>
<th>Time Cost</th>
<th>Predictive Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla Model</td>
<td>No</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Tree-based Model</td>
<td>Yes</td>
<td>Low</td>
<td>Good</td>
</tr>
<tr>
<td>Neural Model</td>
<td>Yes</td>
<td>High</td>
<td>Good</td>
</tr>
</tbody>
</table>
## Comparisons of Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Task Invariant</th>
<th>Time Cost</th>
<th>Predictive Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla Model</td>
<td>No</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Tree-based Model</td>
<td>Yes</td>
<td>Low</td>
<td>Good</td>
</tr>
<tr>
<td>Neural Model</td>
<td>Yes</td>
<td>High</td>
<td>Good</td>
</tr>
</tbody>
</table>
Unique Problem Characteristics

Expression $e$, Search Space $S_e$, AutoTVM, Statistical Cost Model, Code Generator, Code Generator output $x$, Training data $D$, Relatively low experiment cost, Program-aware modeling, Large number of similar tasks.
Transferable Cost Model

$e_1 \rightarrow \text{Expression} \rightarrow \text{AutoTVM} \rightarrow c \rightarrow \text{Code Generator} \rightarrow x$
Transferable Cost Model

$e_1$  Expression  $\rightarrow$  AutoTVM  $\rightarrow$  Code Generator  $\rightarrow$  $x$

$S_{e_1}$  Search Space  $\rightarrow$  $D$

Historical data from related operators (tasks)
Transferable Cost Model

$e_1 \rightarrow \text{Expression} \rightarrow \text{AutoTVM} \rightarrow C \rightarrow \text{Code Generator} \rightarrow x$

$S_{e_1} \rightarrow \text{Search Space} \rightarrow \text{AutoTVM} \rightarrow D \rightarrow \text{Historical data from related operators (tasks)}$

New Tasks

$e_2 \rightarrow \text{Expression} \rightarrow \text{AutoTVM}$

$S_{e_2} \rightarrow \text{Search Space} \rightarrow \text{AutoTVM}$
Transferable Cost Model

$e_1$  
Expression

$S_{e_1}$  
Search Space

AutoTVM

$C$

Code Generator

$x$

Historical data from related operators (tasks)

Shared Cost Model

$D$

New Tasks

$e_2$  
Expression

$S_{e_2}$  
Search Space

AutoTVM
Transferable Cost Model

- $e_1$: Expression
- $S_{e_1}$: Search Space
- $e_2$: Expression
- $S_{e_2}$: Search Space
- AutoTVM
- Code Generator
- $D$: Historical data from related operators (tasks)
- New Tasks

Shared Cost Model

- $c$: Code Generator
- $x$:
Transferable Cost Model

- $e_1$: Expression
- $S_{e_1}$: Search Space
- $c$: Cost
- $x$: Code Generator
- $D$: Historical data from related operators (tasks)

New Tasks

- $e_2$: Expression
- $S_{e_2}$: Search Space

Need task invariant representation
Impact of Transfer Learning

AutoTVM: ML-based Model
Baseline: CuDNN
AutoTVM: Black-box Optimization

Number of Trials vs. Relative Speedup
Impact of Transfer Learning

Transferred ML-Model

AutoTVM: ML-based Model

Baseline: CuDNN

AutoTVM: Black-box Optimization

Number of Trials

Relative Speedup
Impact of Transfer Learning

Transferred ML-Model

AutoTVM: ML-based Model

Baseline: CuDNN

AutoTVM: Black-box Optimization

3x to 10x speedup over non-transfer case
Learning to Optimize Tensor Programs

Relatively low experiment cost

Program-aware modeling

Large number of similar tasks

Chen et al. NeurIPS 18
Device Fleet: Distributed Test Bed for AutoTVM

- **AutoTVM Experiment 1**
- **AutoTVM Experiment 2**

**Resource Manager (Tracker)**

**Resource Allocation**

**Resource Token**

**Persistent Remote Session**

- **RPC RT**
- **CUDA**
  - Nvidia GPU Server

- **RPC RT**
- **OpenCL**
  - Android Phone

- **RPC RT**
- **Bitstream**
  - Zynq FPGA Board
Device Fleet: Distributed Test Bed for AutoTVM

Scale up optimization
Resource sharing

Resource Manager (Tracker)

Resource Allocation

Persistent Remote Session

AutoTVM Experiment 1

AutoTVM Experiment 2

RPC RT

CUDA

Nvidia GPU Server

RPC RT

OpenCL

Android Phone

RPC RT

Bitstream

Zynq FPGA Board
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

AutoTVM

Device Fleet
TVM: End to End Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression and Optimization Search Space

LLVM, CUDA, Metal

Optimization

AutoTVM

Device Fleet
Tensor Expression and Optimization Search Space

Based on Halide’s compute/schedule separation
Tensor Expression and Optimization Search Space

Tensor Expression (Specification)

\[ C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k)) \]

Based on Halide’s compute/schedule separation
Tensor Expression and Optimization Search Space

Tensor Expression (Specification)

\[ C = \text{tvm.compute}((m, n), \text{lambda } y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k)) \]

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

Based on Halide’s compute/schedule separation
Tensor Expression and Optimization Search Space

Tensor Expression (Specification)

\[
C = \text{tvm.compute}((m, n),
\quad \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k))
\]

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

What is the search space

Based on Halide’s compute/schedule separation
Search Space for CPUs

- CPUs
- Compute Primitives: scalar, vector
- Memory Subsystem:
  - L1D
  - L1I
  - L2
  - L3
  - L1D
  - L1I
  - 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

Memory Subsystem

<table>
<thead>
<tr>
<th>L2</th>
<th>SM</th>
<th>SM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX/L1</td>
<td>TX/L1</td>
<td></td>
</tr>
</tbody>
</table>

scalar  •  vector  •  mixed
Search Space for GPUs

GPUs

Compute Primitives
- scalar
- vector

Memory Subsystem
- L2
- SM
- TX/L1
- RF

Shared memory among compute cores
Search Space for GPUs

Compute Primitives
- scalar
- vector

Memory Subsystem
- L2
- SM
- TX/L1
- RF

Mixed

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

Unified Buffer

FIFO

Acc
Search Space for TPU-like Specialized Accelerators

TPUs

Tensor Compute Primitives

Explicitly Managed Memory Subsystem

Unified Buffer

FIFO

Acc
Tensorization Challenge

Compute primitives
Tensorization Challenge

Compute primitives

scalar
Tensorization Challenge

Compute primitives

scalar

vector
Tensorization Challenge

Compute primitives

- scalar
- vector
- tensor
Tensorization Challenge

Challenge: Build systems to support emerging tensor instructions
Tensorization Challenge

Computation Specification (Tensor Expression)

\[ C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \ast B[k, x], \text{axis}=k)) \]
Tensorization Challenge

Computation Specification (Tensor Expression)

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

HW Interface Specification by Tensor Expression

```python
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 Challenge

Computation Specification (Tensor Expression)

\[ C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k)) \]

HW Interface Specification by Tensor Expression

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] \times B[k], \text{axis}=k))
Search Space for TPU-like Specialized Accelerators

TPUs

Tensor Compute Primitives

Explicitly Managed Memory Subsystem

- Unified Buffer
- FIFO
- Acc
Search Space for TPU-like Specialized Accelerators

TPUs

Tensor Compute Primitives

Explicitly Managed Memory Subsystem

Unified Buffer

FIFO

Acc
Software Support for Latency Hiding

Single Module
No Task-Pipelining

- load inputs
- load weights
- matrix multiplication
- store outputs
- load inputs
- load weights
- matrix multiplication
- store outputs
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

vthread0
- load inputs
- load weights
- matrix multiplication
- store outputs

vthread1
- load inputs
- load weights
- matrix multiplication
- store outputs
Software Support for Latency Hiding

Multi-threaded Program

\[ \text{vthread0} \]
- load inputs
- load weights
- matrix multiplication
- store outputs

\[ \text{vthread1} \]
- load inputs
- load weights
- matrix multiplication
- store outputs

Program with Task-level Pipeline Instructions

- load inputs
- load weights
- matrix multiplication
- store outputs
- load inputs
- load weights
- matrix multiplication
- store outputs
Summary: Hardware-aware Search Space

Tensor Expression Language

\[ C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \ast B[k, x], \text{axis}=k)) \]
Summary: Hardware-aware Search Space

Tensor Expression Language

$$C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k))$$

Primitives in prior work: Halide, Loopy

- Loop Transformations
- Thread Bindings
- Cache Locality
Summary: Hardware-aware Search Space

Tensor Expression Language

\[
C = \text{tvm.compute}((m, n), \\
\lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k))
\]

Primitives in prior work: Halide, Loopy

New primitives for GPUs, and enable TPU-like Accelerators

Loop Transformations
Thread Bindings
Cache Locality
Thread Cooperation
Tensorization
Latency Hiding
Summary: Hardware-aware Search Space

Primitives in prior work: Halide, Loopy

New primitives for GPUs, and enable TPU-like Accelerators

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

Tensor Expression Language

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

Hardware
TVM: End to End Deep Learning Compiler
End to End Inference Performance (Nvidia Titan X)

- TensorFlow
- Apache MXNet
- TensorFlow-XLA

**Time (ms)**

- **ResNet-18**
- **MobileNet**
- **LSTM LM**
- **DQN**
- **DCGAN**
End to End Inference Performance (Nvidia Titan X)

- **TensorFlow**
- **Apache MXNet**
- **TensorFlow-XLA**

Backed by cuDNN
End to End Inference Performance (Nvidia Titan X)

- **TensorFlow**
- **Apache MXNet**
- **TVM**
- **TensorFlow-XLA**

The performance is measured in milliseconds (ms) for different models:

- **ResNet-18**
- **MobileNet**
- **LSTM LM**
- **DQN**
- **DCGAN**

The chart compares the performance of TensorFlow, Apache MXNet, TVM, and TensorFlow-XLA for these models.
End to End Inference Performance (Nvidia Titan X)

- **TensorFlow**
- **Apache MXNet**
- **TVM**
- **TensorFlow-XLA**

- Competitive on standard models
End to End Inference Performance (Nvidia Titan X)

- TensorFlow
- Apache MXNet
- TVM
- TensorFlow-XLA

Bar charts showing inference times for various models:
- ResNet-18
- MobileNet
- LSTM LM
- DQN
- DCGAN
End to End Inference Performance (Nvidia Titan X)

- ResNet-18
- MobileNet
- LSTM LM
- DQN
- DCGAN

Time (ms)

- TensorFlow
- Apache MXNet
- TVM
- TensorFlow-XLA

3x better on emerging models
Portable Performance Across Hardware Platforms

- TensorFlow Lite
- ArmComputeLib
- TVM

![Comparison of performance across ARM CPU (Cortex-A53) and ARM GPU (MALI)]
Portable Performance Across Hardware Platforms

Special frameworks for the particular hardware platform

- TensorFlow Lite
- ArmComputeLib
- TVM

ARM CPU (Cortex-A53)

ARM GPU (MALI)
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

VTA Hardware/Software Interface (ISA)

VTA MicroArchitecture

Current TVM Stack

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

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

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

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

Current TVM Stack

VTA Runtime & JIT Compiler

VTA Hardware/Software Interface (ISA)

VTA MicroArchitecture

VTA Simulator

compiler, driver, hardware design
full stack open source

Moreau, Chen, et al. work in progress
TVM: End to End Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression and Optimization Search Space

LLVM, CUDA, Metal

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
- AutoTVM
- Device Fleet
TVM Impact
TVM Impact

Open source: 202 contributors from UW, Berkeley, Cornell, UCLA, AWS, Huawei, NTT, Facebook, Qualcomm, …
TVM Impact

Open source: 202 contributors from UW, Berkeley, Cornell, UCLA, AWS, Huawei, NTT, Facebook, Qualcomm, …

Used in production
TVM Enables New Research Frontiers
TVM Enables New Research Frontiers

• PL: New high-level differentiable programming IR (UW)
TVM Enables New Research Frontiers

- PL: New high-level differentiable programming IR (UW)
- Architecture: New deep learning ASICs, RISC-V (UW, Cornell)
TVM Enables New Research Frontiers

- PL: New high-level differentiable programming IR (UW)
- Architecture: New deep learning ASICs, RISC-V (UW, Cornell)
- Security: trusted enclave for private aware ML. (Berkeley)
TVM Enables New Research Frontiers

• PL: New high-level differentiable programming IR (UW)
• Architecture: New deep learning ASICs, RISC-V (UW, Cornell)
• Security: trusted enclave for private aware ML. (Berkeley)
• Machine learning: test bed for automatic optimization. (UW)
TVM Enables New Research Frontiers

- PL: New high-level differentiable programming IR (UW)
- Architecture: New deep learning ASICs, RISC-V (UW, Cornell)
- Security: trusted enclave for private aware ML. (Berkeley)
- Machine learning: test bed for automatic optimization. (UW)
TVM Enables New Research Frontiers

- **PL:** New high-level differentiable programming IR (UW)
- **Architecture:** New deep learning ASICs, RISC-V (UW, Cornell)
- **Security:** trusted enclave for private aware ML. (Berkeley)
- **Machine learning:** test bed for automatic optimization. (UW)

TVM Conference
180 attendees, 20+ talks
Learning Systems

Data science for everyone

Scale up deep learning

Deploy AI everywhere
Learning Systems

Data science for everyone

Scale up deep learning

Deploy AI everywhere

What’s Next
Learning-based Learning System
Learning-based Learning System
Learning-based Learning System

Application

Model

MobileNet-V2
Learning-based Learning System

Application

Model
MobileNet-V2

Hardware
ARM Cortex A53
Learning-based Learning System

Application

Model

MobileNet-V2

Space of Tensor Programs

Hardware

ARM Cortex A53
Learning-based Learning System

Model

MobileNet-V2

Space of Tensor Programs

Hardware

ARM Cortex A53

AutoTVM
Learning-based Learning System

Application

Model

MobileNet-V2

Space of Tensor Programs

Hardware

ARM Cortex A53

AutoTVM

Optimized Tensor Program
Full Stack Learning-based Learning System
Full Stack Learning-based Learning System

Application
Full Stack Learning-based Learning System

Application

Space of Models
Full Stack Learning-based Learning System
Full Stack Learning-based Learning System

Application

Space of Models

Space of Tensor Programs

Space of Hardware Variants
Full Stack Learning-based Learning System

Application

Space of Models

Space of Tensor Programs

Space of Hardware Variants

Full-Stack AutoTVM
Full Stack Learning-based Learning System

Application

- Space of Models
- Space of Tensor Programs
- Space of Hardware Variants

Full-Stack AutoTVM

- Auto Optimized Net
- Optimized Tensor Program
- Auto Hardware Variant
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

ML Models

System Optimizations

Hardware Design

Full-stack co-design

ML as the first-class citizen
Lifecycle of Intelligent Applications

\[ D \]
Lifecycle of Intelligent Applications

Learning System

$D$
Lifecycle of Intelligent Applications

\[ D \]

Learning System

Model
Lifecycle of Intelligent Applications

1. Data ($D$)
2. Learning System
3. Model
4. Deploy (serving)
Lifelong Learning Systems

Learning System

Environment
Lifelong Learning Systems
Lifelong Learning Systems

New Data → Learning System → Environment
Lifelong Learning Systems

New Data → Learning System → Environment → New Models
Lifelong Learning Systems

New Data → New Models → Learning System → New Tasks → Environment
Lifelong Learning Systems

Lifelong evolution of model, data and system optimizations
Challenges in Lifelong Learning Research
Challenges in Lifelong Learning Research

Model transfer as model complexity grows

Challenges in Lifelong Learning Research

Model transfer as model complexity grows


Smart data acquisition, task prioritization
Challenges in Lifelong Learning Research

Model transfer as model complexity grows

*Net2Net: Accelerating learning via knowledge transfer. Chen, et al. ICLR 16*

Smart data acquisition, task prioritization

Build real-world learning systems as test beds
Learning-based learning systems are ideal starting points
Learning-based Learning Systems

Data science for everyone
Scale up deep learning
Deploy AI everywhere
Learning-based Learning Systems

- XGBoost
- MXNet
- TVM

Data science for everyone
Scale up deep learning
Deploy AI everywhere

Full-stack Learning-based Learning System
Learning-based Learning Systems

- dmlc XGBoost
- mxnet
- tvm

Data science for everyone
Scale up deep learning
Deploy AI everywhere

Full-stack Learning-based Learning System
Life-long learning Systems
Take Aways: Elements of Future Learning Systems

Beside being accessible and scalable

Intelligent automated by machine learning

Full stack model, systems and hardware co-design

Lifelong consider the entire life-cycle of learning
Take Aways: Elements of Future Learning Systems

Beside being **accessible** and **scalable**

**Intelligent** automated by machine learning

**Full stack** model, systems and hardware co-design

**Lifelong** consider the entire life-cycle of learning

![Logos of dmlc, XGBoost, mxnet, and tvm]