JANUS: Fast and Flexible Deep Learning via Symbolic Graph Execution of Imperative Programs

Eunji Jeong, Sungwoo Cho, Gyeong-In Yu, Joo Seong Jeong, Dong-Jin Shin, Byung-Gon Chun
**Introduction**

**Challenge**

**Solution**

**Results**

---

**Deep Neural Networks**

Images From:
- http://www.intid.com/
- Short-Term Load Forecasting Using EMD-LSTM Neural Networks with a Xgboost Algorithm for Feature Importance Evaluation, Energies 2017
Deep Neural Networks

Symbolic DL Frameworks

Imperative DL Frameworks

Introduction

Challenge

Solution

Results

1. Introduction
2. Challenge
3. Solution
4. Results

Deep Neural Networks
Symbolic DL Frameworks

✓ Build a Symbolic Graph
✓ Execute the Graph

```python
def build_graph(g):
    x = g.placeholder(float)
    linear = g.add(g.mul(W, x), b)
build_graph(graph)
run_graph(graph, x_data)
```

Imperative DL Frameworks

✓ Directly Execute the Computations

```python
def linear(x):
    return W * x + b
linear(x_data)
```
<table>
<thead>
<tr>
<th>Pros</th>
<th>Symbolic DL Frameworks</th>
<th>Imperative DL Frameworks</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Easy to Optimize</td>
<td>+ Direct Execution:</td>
</tr>
<tr>
<td></td>
<td>+ Compiler Optimization</td>
<td>Easy to Program &amp; Debug</td>
</tr>
<tr>
<td></td>
<td>+ Parallel Execution of</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Operations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+ Deploy on GPU, Cluster, Mobile, ...</td>
<td></td>
</tr>
<tr>
<td>Cons</td>
<td>- Decoupled View:</td>
<td>- Hard to Optimize</td>
</tr>
<tr>
<td></td>
<td>Hard to Program &amp; Debug</td>
<td></td>
</tr>
</tbody>
</table>
JANUS: Combining the Best of Both Worlds

Imperative DL Program

```python
def foo(x):
    tmp = mul(3, x)
    return add(tmp, 2)
```

Transparent Conversion

Symbolic DL Graph

“Easy Programmability”

“High Performance”
Imperative DL Program with Dynamic Features

```python
for item in sequence:
    state = Cell(state, item)
    outputs += [state]
```

- Dynamic Control Flow
- Dynamic Types
- Impure Functions
- ...

Symbolic DL Graph
Imperative DL Program with **Dynamic** Features

```python
for item in sequence:
    state = Cell(state, item)
outputs += [state]
```
**Imperative DL Program with Dynamic Features**

```python
for item in sequence:
    state = Cell(state, item)
outputs += [state]
```
Solution: Speculative Graph Generation and Execution

- **[Performance]** Speculatively Specialize the Graph
  - Make reasonable assumptions based on the execution history (Profiling)
  - Run specialized graph (Common Case)

- **[Correctness]** Validate Assumptions
  - *Fallback* if an assumption is broken (Rare Case)
Imperative DL Program

```python
for item in sequence:
    state = rnn(state, item)
    outputs += [state]
```

Imperative Executor

- Python Interpreter
- Pre-defined DL Operations
Imperative DL Program

for item in sequence:
    state = rnn(state, item)
    outputs += [state]

Imperative Executor

Profiler

Python Interpreter

Pre-defined DL Operations
Symbolic Graph Executor

Imperative DL Program

len:3

for item in sequence:
    state = rnn(state, item)
    outputs += [state]

Symbolic DL Graph

Graph Generator

state

Cell

Cell

len == 3

Assert

Symbolic Graph Executor

Python Interpreter

Pre-defined DL Operations
Imperative DL Program

```python
for item in sequence:
    state = rnn(state, item)
outputs += [state]
```

Symbolic DL Graph

- `state` to `len == 3` to `Assert`
- `Cell` to `Cell` to `Cell`

Symbolic Graph Executor

- Python Interpreter
- Pre-defined DL Operations

Assumption Failure
Imperative DL Program

```python
for item in sequence:
    state = rnn(state, item)
outputs += [state]
```

Symbolic DL Graph

Symbolic Graph Executor

Python Interpreter

Pre-defined DL Operations
Imperative Executor

for item in sequence:
    state = rnn(state, item)
    outputs += [state]

Imperative DL Program

Profiler

Python Interpreter

Pre-defined DL Operations

Overall Workflow on JANUS

Introduction

Challenge

Solution

Results
**Imperative DL Program**

```python
for item in sequence:
    state = rnn(state, item)
    outputs += [state]
```

**Symbolic DL Graph**

- `state`
- `i<N`
- `Merge`
- `Next`
- `Switch`
- `Cell`

**Symbolic Graph Executor**

- Python Interpreter
- Pre-defined DL Operations
ImageNet Test Error with ResNet50

- **Symbolic**
- **JANUS**
- **Imperative**
- **3.4x Faster Convergence**

36 GPUs
### Normalized Training Throughput

<table>
<thead>
<tr>
<th>Category</th>
<th>Models</th>
<th>Single Machine</th>
<th>Improvements</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>LeNet, ResNet-50, Inception-v3</td>
<td>2x, 3x, 4x, 5x, 6x, 7x, 8x</td>
<td>47.6x over Imperative</td>
</tr>
<tr>
<td>RNN</td>
<td>LSTM, LM</td>
<td>2x, 3x, 4x, 5x, 6x, 7x, 8x</td>
<td>96.0% of Symbolic</td>
</tr>
<tr>
<td>TreeNN</td>
<td>TreeRNN, TreeLSTM</td>
<td>2x, 3x, 4x, 5x, 6x, 7x, 8x</td>
<td></td>
</tr>
<tr>
<td>DRL</td>
<td>A3C, PPO</td>
<td>2x, 3x, 4x, 5x, 6x, 7x, 8x</td>
<td></td>
</tr>
<tr>
<td>GAN</td>
<td>AN, PIX2PIX</td>
<td>2x, 3x, 4x, 5x, 6x, 7x, 8x</td>
<td></td>
</tr>
</tbody>
</table>

**JANUS**

Introduction

Challenge

Solution

Results

**Single Machine**
Thank You!