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

SEOUL NATIONAL UNIVERSITY

Demo

Deep Neural Networks

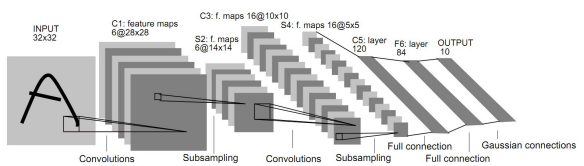
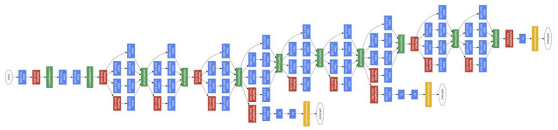
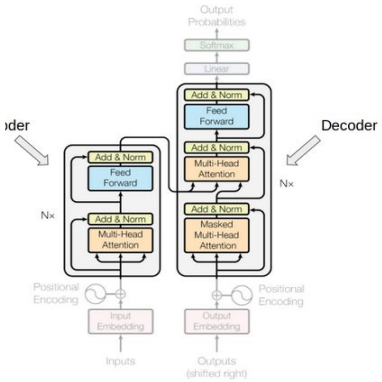
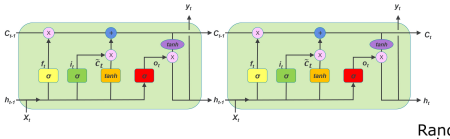
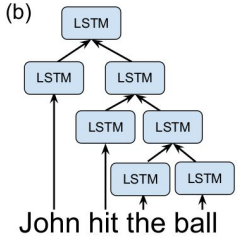


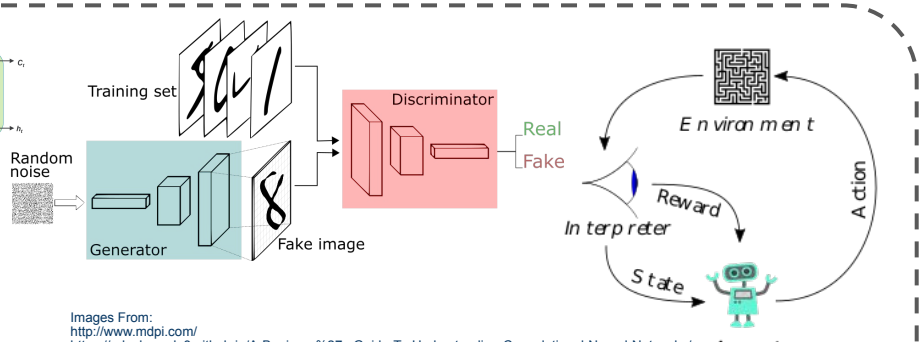
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.











Images From:
http://www.mdpi.com/
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
Going Deeper with Convolutions, 2014, https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7
Short-Term Load Forecasting Using EMD-LSTM Neural Networks with a Xgboost Algorithm for Feature Importance Evaluation, Energies 2017
https://skymind.ai/wiki/generative-adversarial-network-gan
https://en.wikipedia.org/wiki/Reinforcement_learning
https://medium.com/@Petuum/intro-to-dynamic-neural-networks-and-dynet-67694b18cb23

Deep Neural Networks

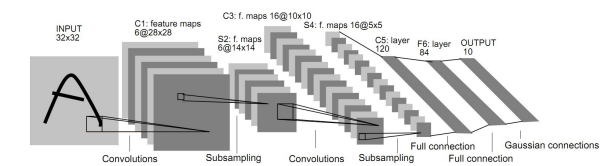
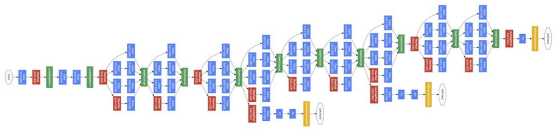
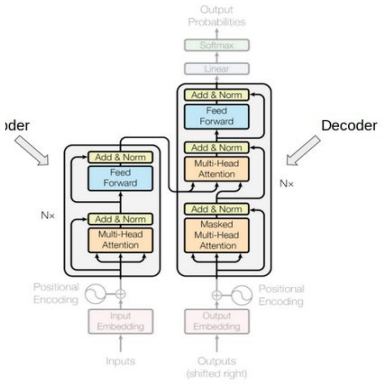
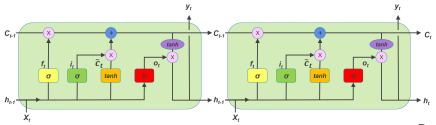
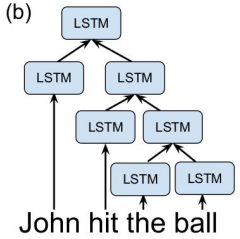


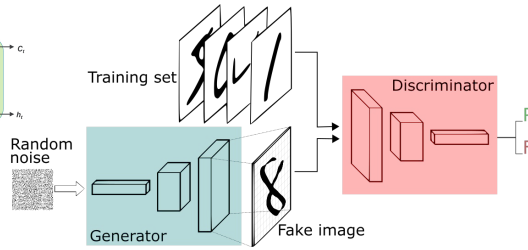
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

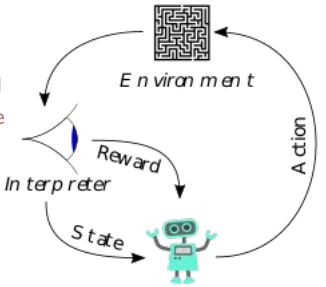












Images From:
http://www.mdpi.com/
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
Going Deeper with Convolutions, 2014, https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7
Short-Term Load Forecasting Using EMD-LSTM Neural Networks with a Xgboost Algorithm for Feature Importance Evaluation, Energies 2017
https://skymind.ai/wiki/generative-adversarial-network-gan
https://en.wikipedia.org/wiki/Reinforcement_learning
https://medium.com/@Petuum/intro-to-dynamic-neural-networks-and-dynet-67694b18cb23

Deep Learning (DL) Frameworks

TensorFlow Google
mxnet
Microsoft CNTK
Caffe2
julia
tvm
Chainer
ONNX
PyTorch facebook
K Keras
dy/net
R
theano

Deep Neural Networks

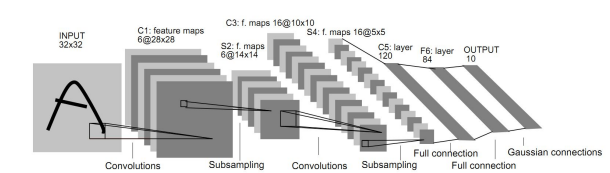
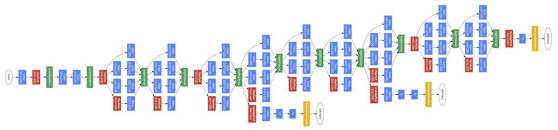
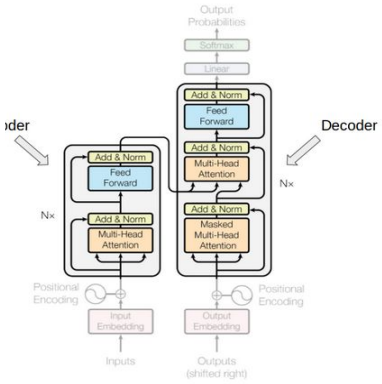
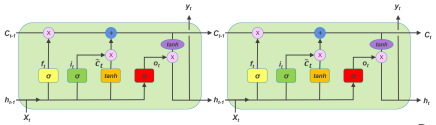
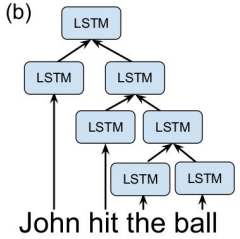


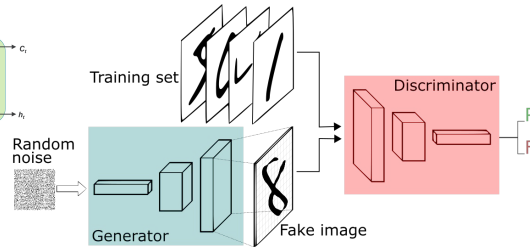
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

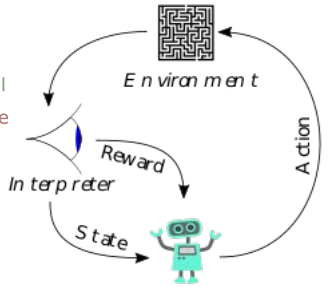












Images From:
http://www.mdpi.com/
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
Going Deeper with Convolutions, 2014, https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7
Short-Term Load Forecasting Using EMD-LSTM Neural Networks with a Xgboost Algorithm for Feature Importance Evaluation, Energies 2017
https://skymind.ai/wiki/generative-adversarial-network-gan
https://en.wikipedia.org/wiki/Reinforcement_learning
https://medium.com/@Petuum/intro-to-dynamic-neural-networks-and-dynet-67694b18cb23

Symbolic DL Frameworks

TensorFlow 1.x
Microsoft CNTK
theano
ONNX
Caffe2

Imperative DL Frameworks

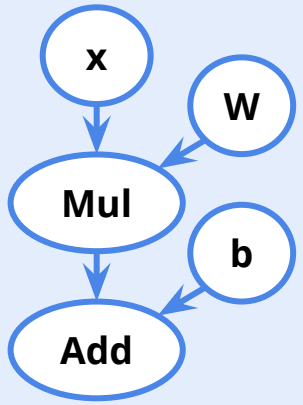
PyTorch
julia
TensorFlow 2.0
dy/net
R
Chainer

Symbolic DL Frameworks

✓ Build a Symbolic Graph
✓ Execute the Graph

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

build_graph(graph)
run_graph(graph, x_data)
```



TensorFlow 1.x Caffe2

Imperative DL Frameworks

✓ Directly Execute the Computations

```
def linear(x):
  return W * x + b
linear(x_data)
```

PyTorch TensorFlow 2.0

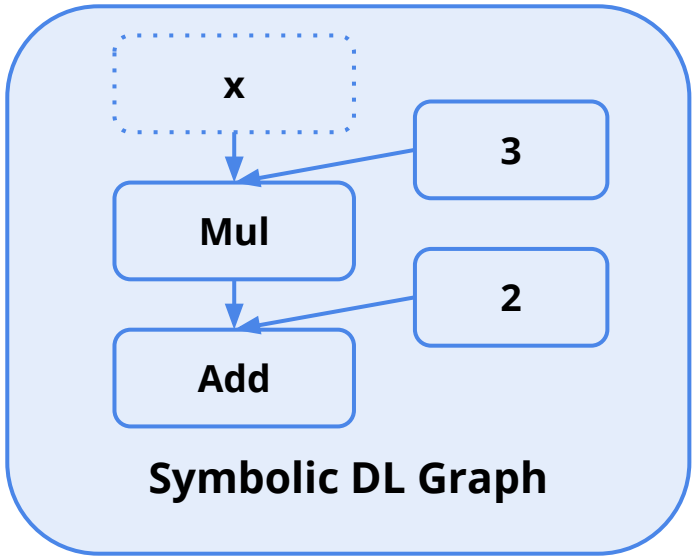
	Symbolic DL Frameworks	Imperative DL Frameworks
Pros	+ **Easy to Optimize** + **Compiler Optimization**  + **Parallel Execution of Operations** + **Deploy on GPU, Cluster, Mobile,…**	+ **Direct Execution:** **Easy to Program & Debug**
Cons	- Decoupled View: Hard to Program & Debug	- Hard to Optimize

JANUS: Combining the Best of Both Worlds

Imperative DL Program

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

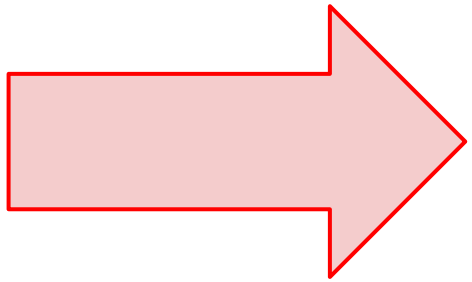
Transparent Conversion

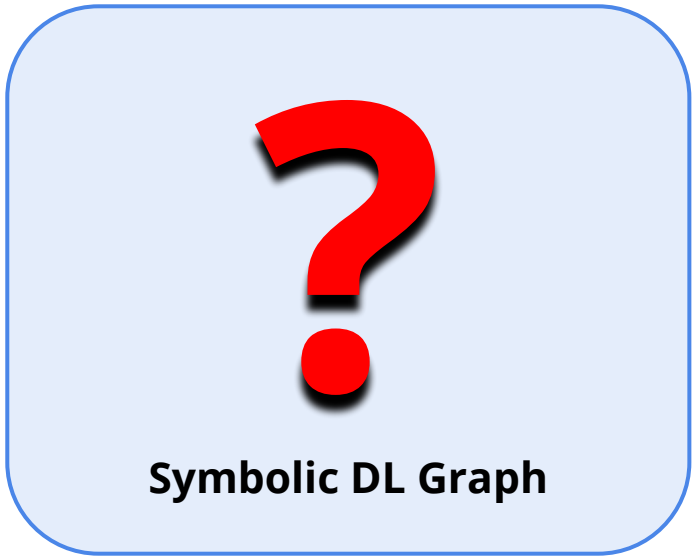


"Easy Programmability" *"High Performance"*

Imperative DL Program with Dynamic Features

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

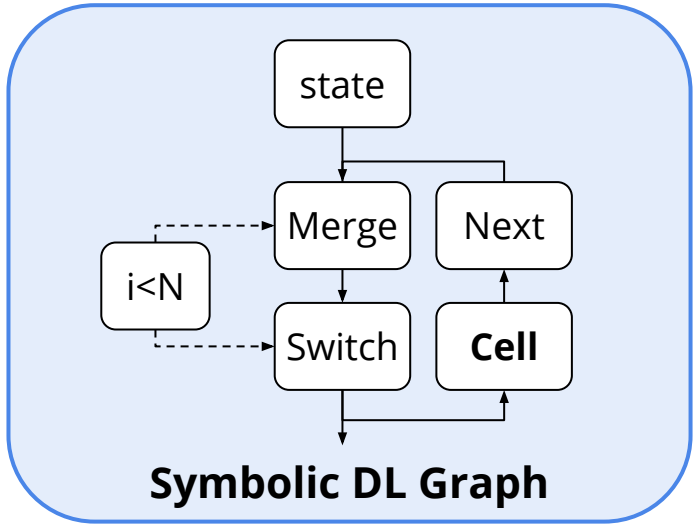




- Dynamic Control Flow
- Dynamic Types
- Impure Functions
- ...

Imperative DL Program with Dynamic Features

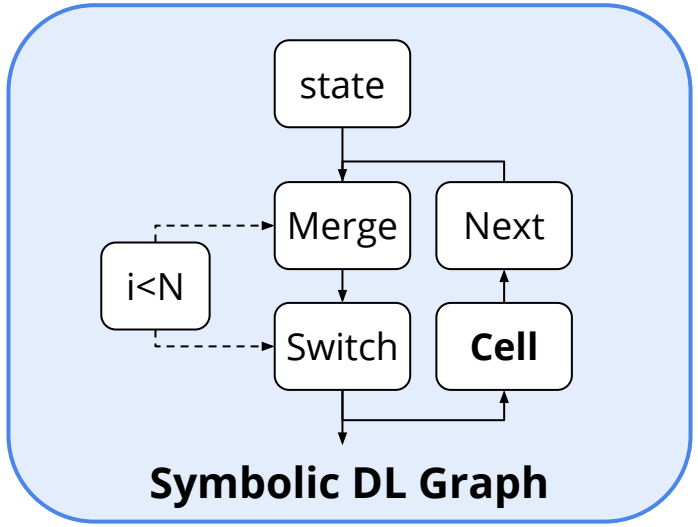
```
for item in sequence:
  state = Cell(state, item)
  outputs += [state]
```



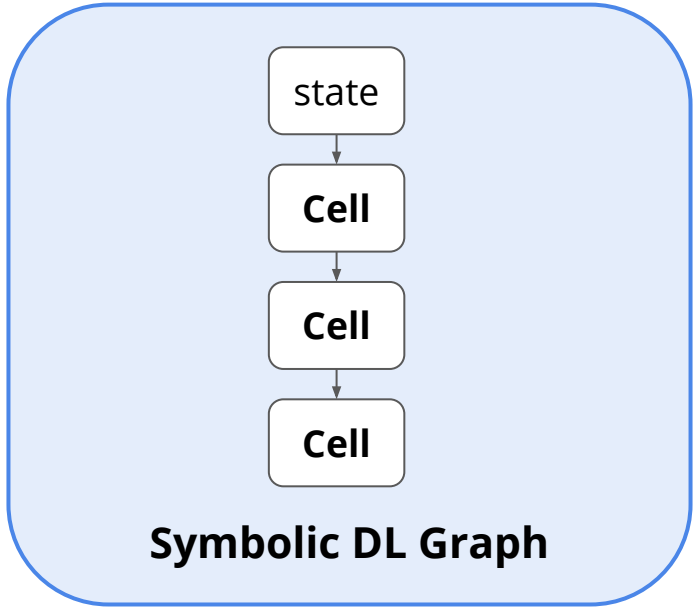
- Correct ☺
- Slow ☹

Imperative DL Program with Dynamic Features

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



- Correct ☺
- Slow ☹



- Fast ☺
- Incorrect ☹

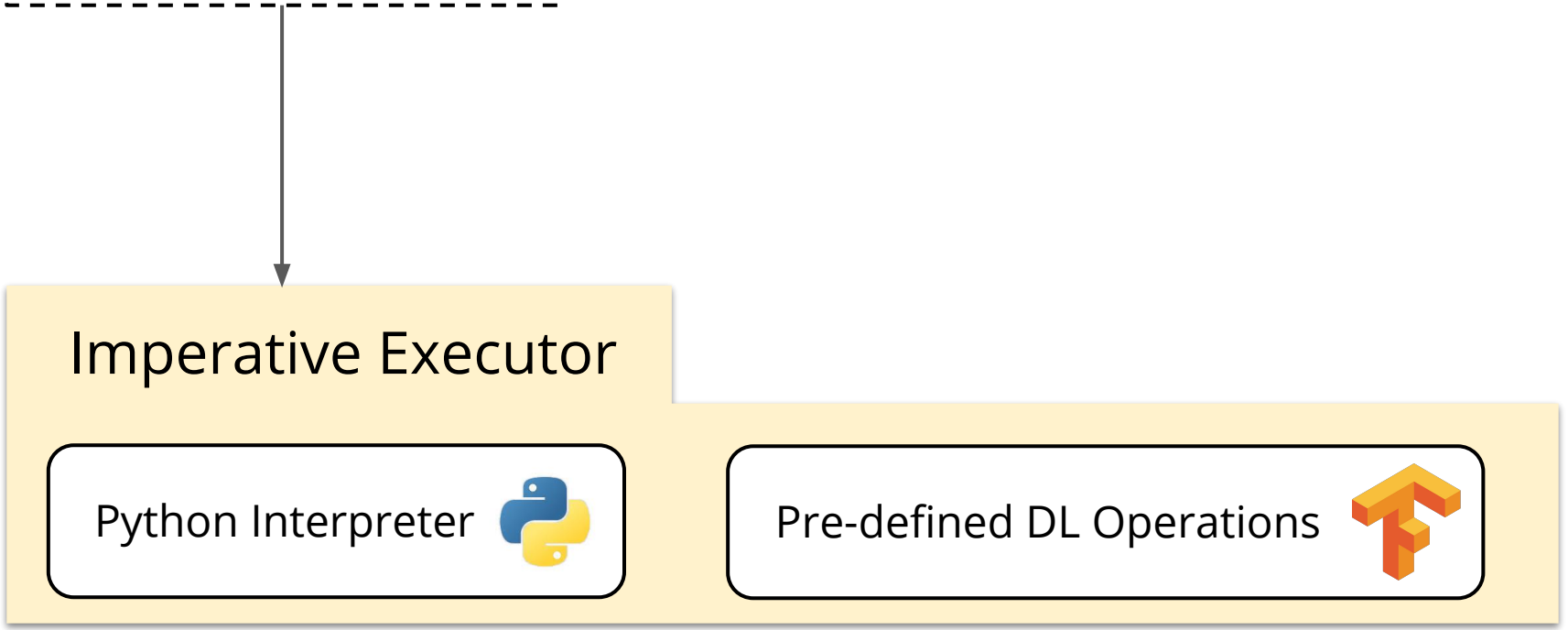
Solution: Speculative Graph Generation and Execution

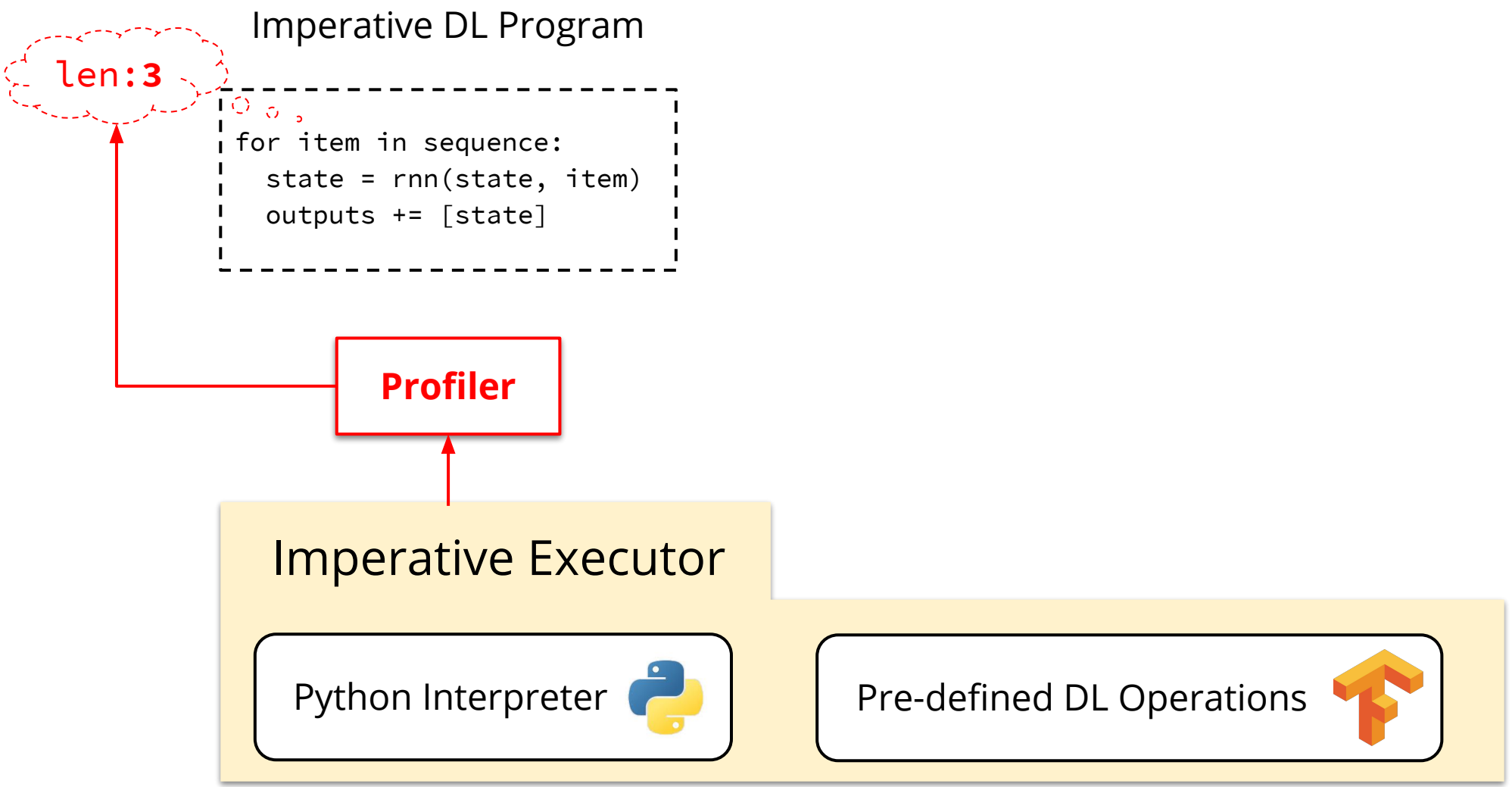
- [Performance] **Speculatively Specialize the Graph**
  - Make reasonable assumptions based on the execution history (***Profiling***)
  - Run specialized graph (Common Case)

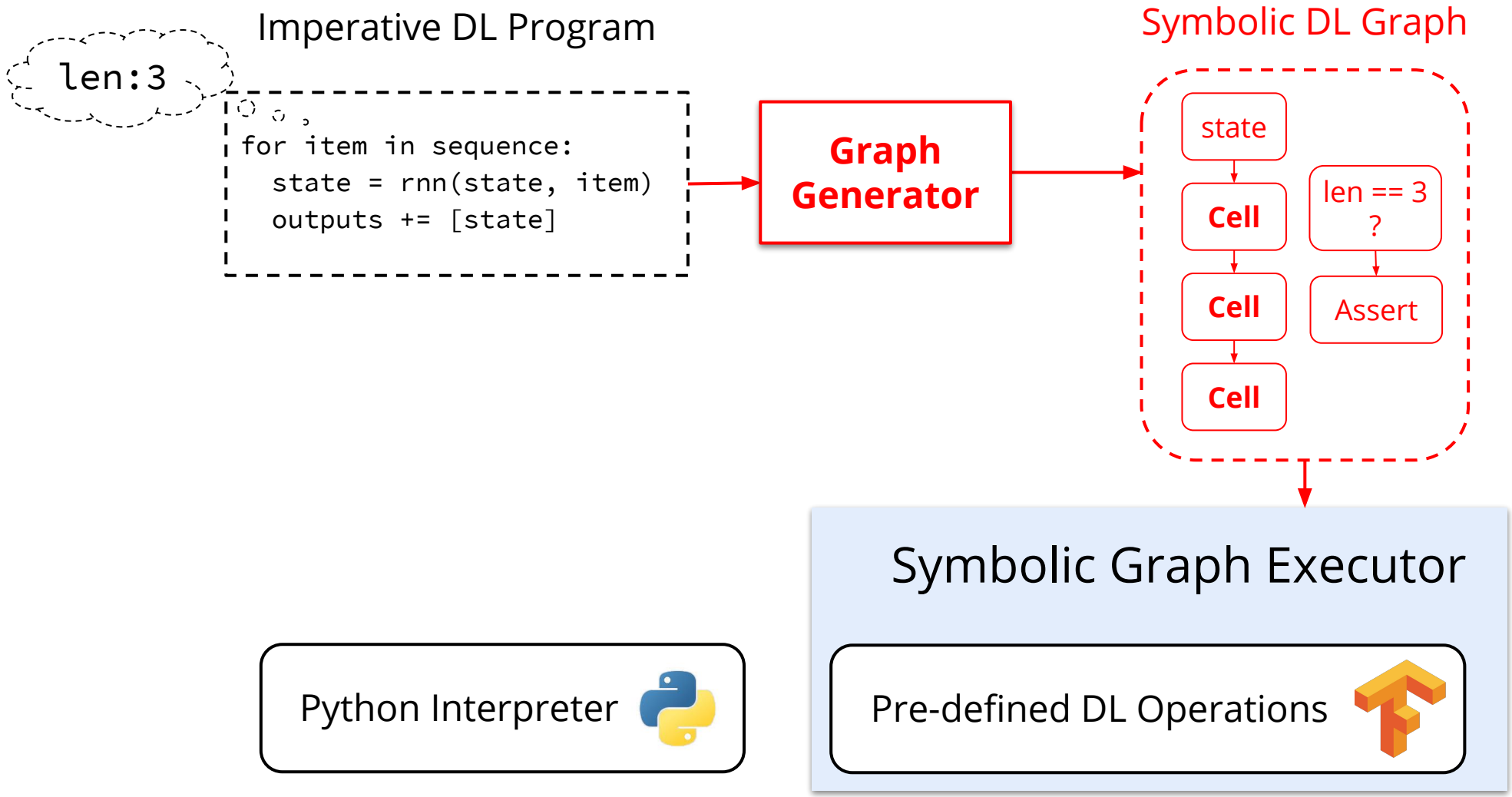
- [Correctness] **Validate Assumptions**
  - ***Fallback*** if an assumption is broken (Rare Case)

Imperative DL Program

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




Imperative DL Program
len:3
for item in sequence:
state = rnn(state, item)
outputs += [state]
Profiler
Imperative Executor
Python Interpreter
Pre-defined DL Operations

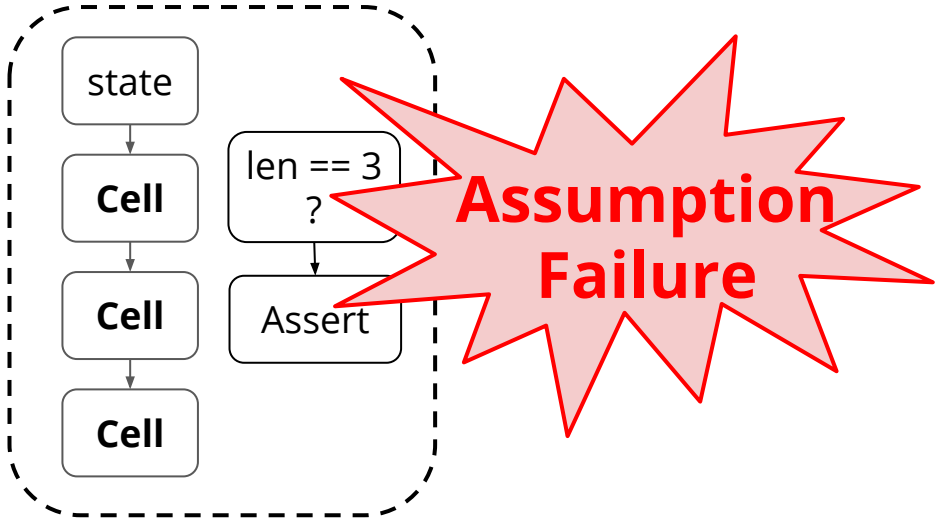

Imperative DL Program
len:3
for item in sequence:
state = rnn(state, item)
outputs += [state]
Graph Generator
Symbolic DL Graph
state
Cell
Cell
Cell
len == 3 ?
Assert
Symbolic Graph Executor
Python Interpreter
Pre-defined DL Operations

Imperative DL Program

len:3

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

Symbolic DL Graph



Python Interpreter

Symbolic Graph Executor

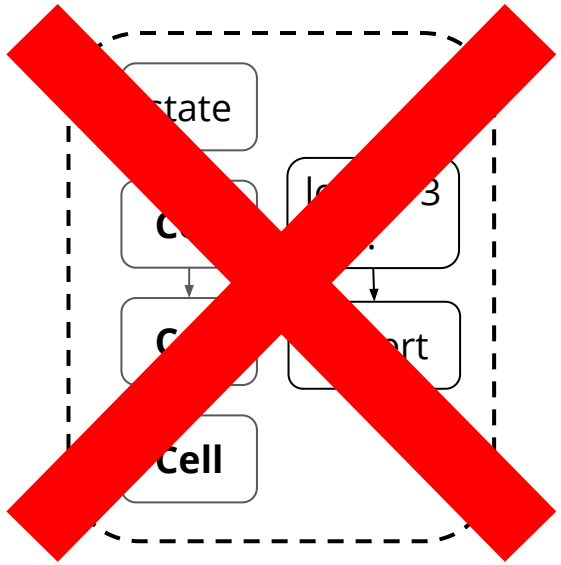
Pre-defined DL Operations

Imperative DL Program

len:3

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

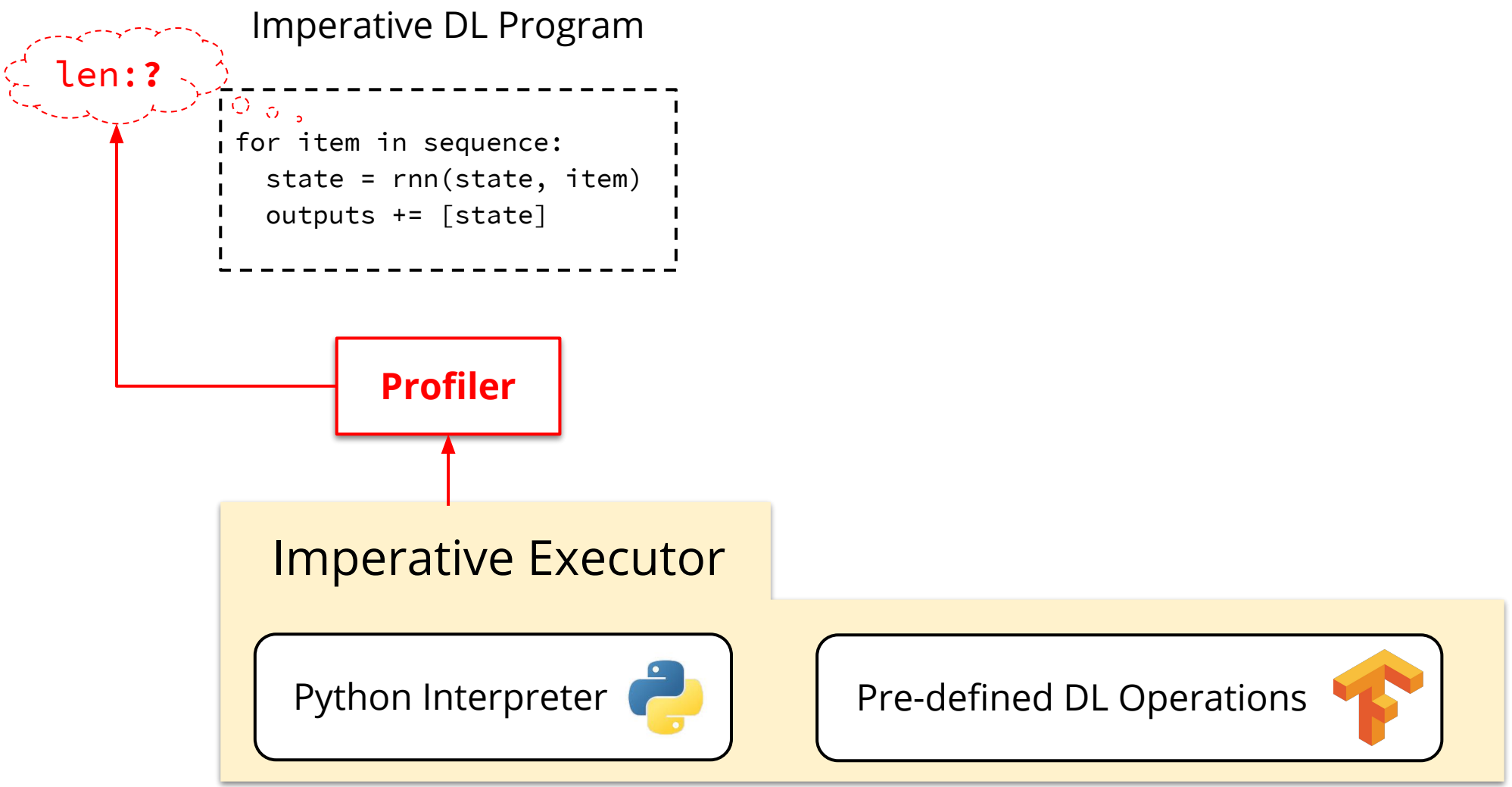
Symbolic DL Graph

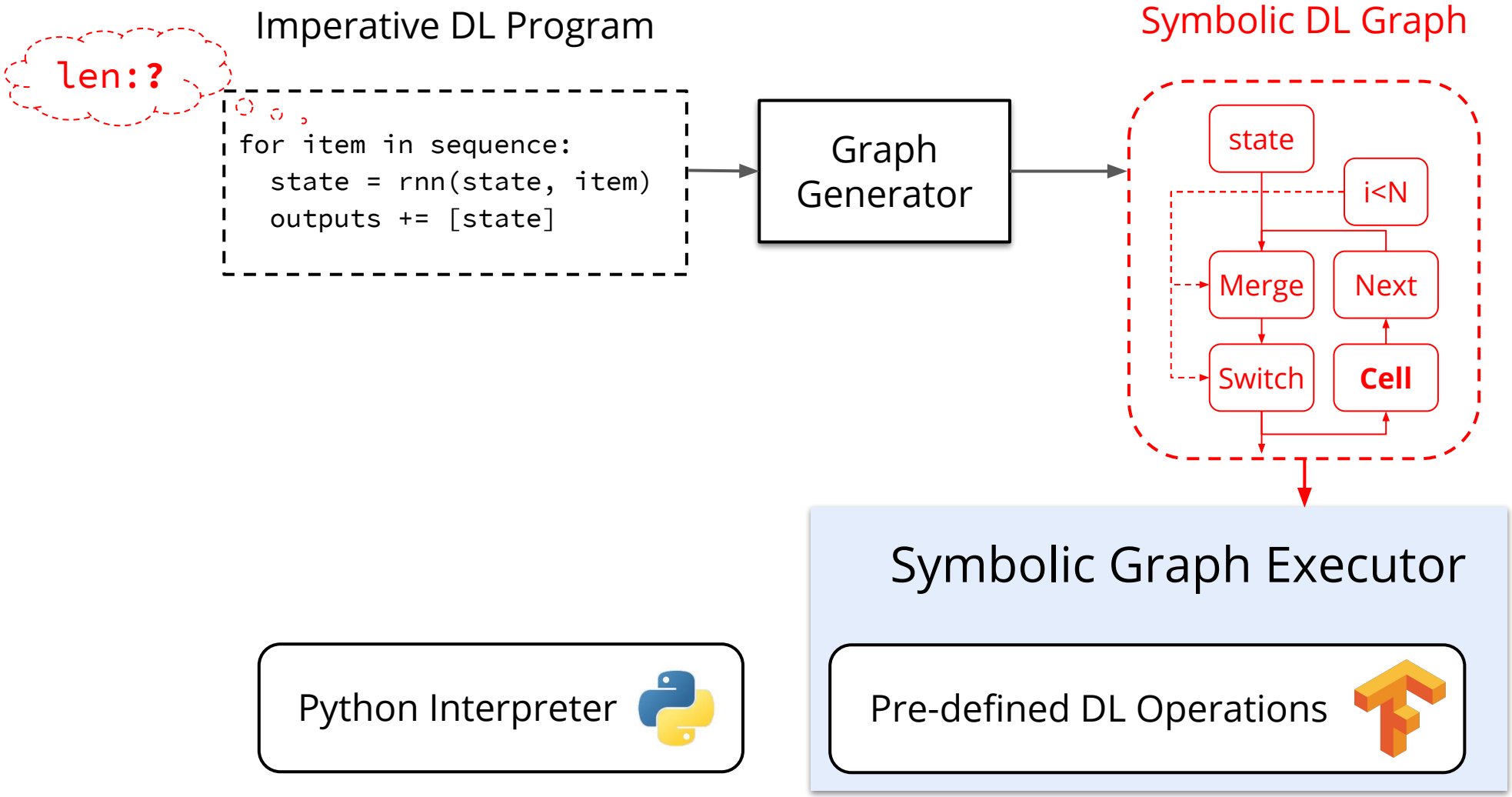


Python Interpreter

Symbolic Graph Executor

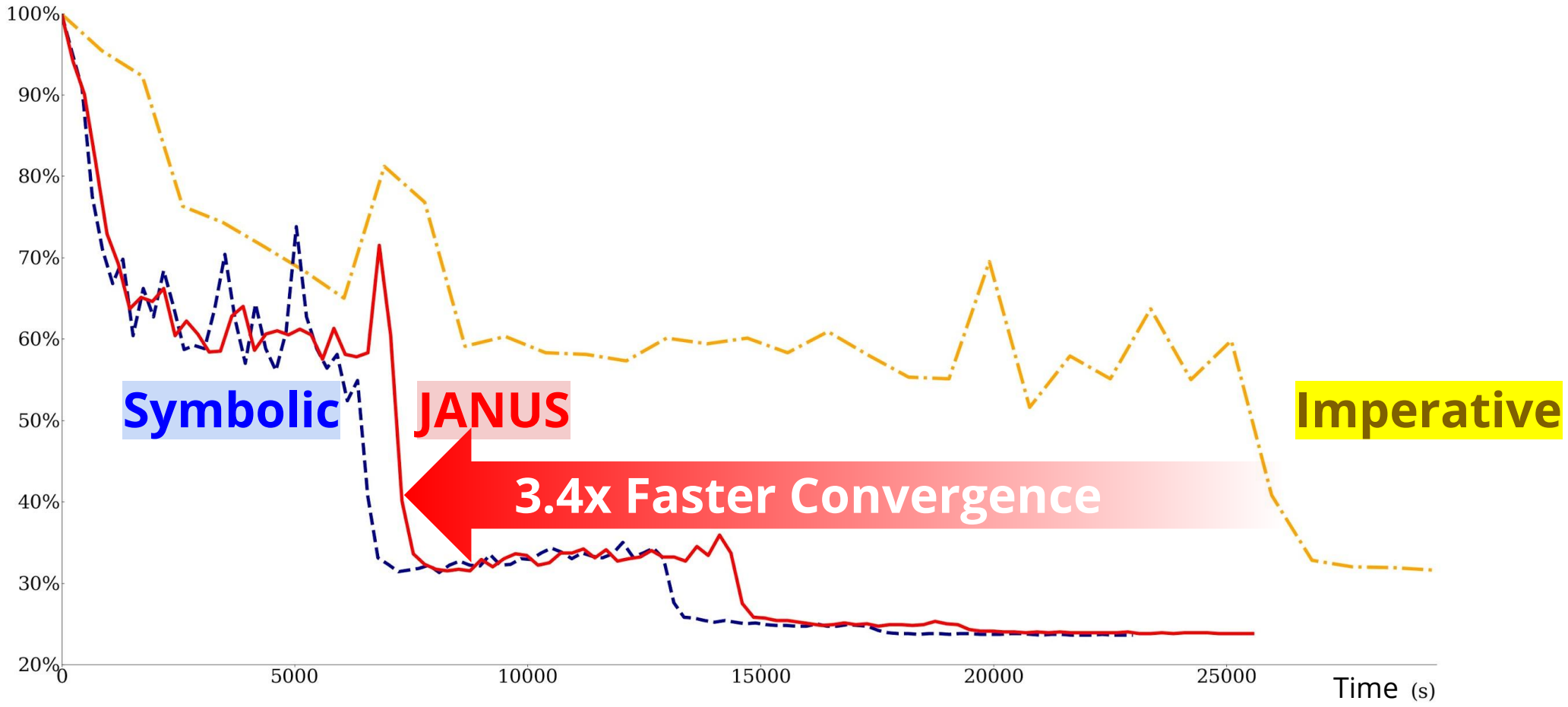
Pre-defined DL Operations


Imperative DL Program
len:?
for item in sequence:
state = rnn(state, item)
outputs += [state]
Profiler
Imperative Executor
Python Interpreter
Pre-defined DL Operations


Imperative DL Program
len:?
for item in sequence:
state = rnn(state, item)
outputs += [state]
Graph Generator
Symbolic DL Graph
state
i<N
Merge
Next
Switch
Cell
Symbolic Graph Executor
Python Interpreter
Pre-defined DL Operations

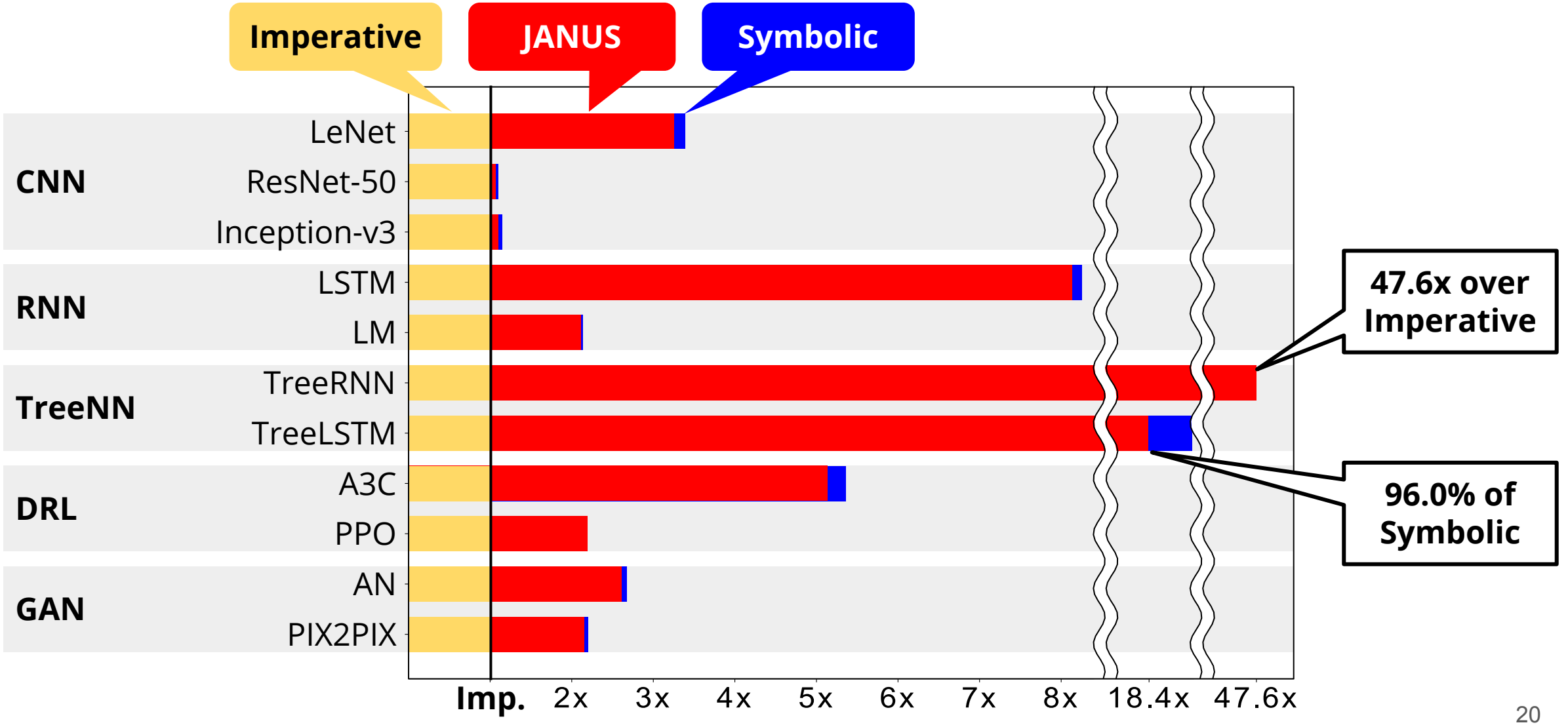
ImageNet Test Error with ResNet50

36 GPUs



Normalized Training Throughput

Single Machine



Thank You!