ISSRE 2021 Wuhan, China

Nondeterministic Impact of CPU Multithreading on Training Deep Learning Systems

Guanping Xiao[1,2], Jun Liu[1], Zheng Zheng[3], Yulei Sui[4]

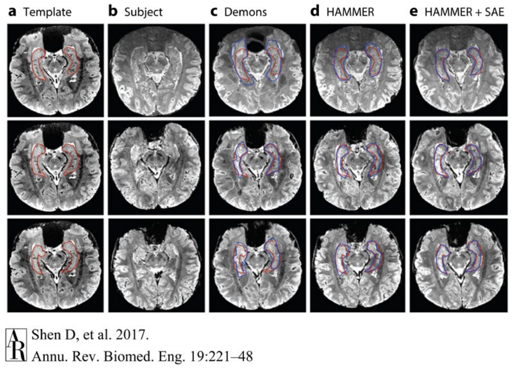
[1] Nanjing University of Aeronautics and Astronautics, China
[2] State Key Laboratory for Novel Software Technology, Nanjing University, China
[3] Beihang University, China
[4] University of Technology Sydney, Australia

Oct 27, 2021
Wuhan, China

Motivation

- Deep learning (DL) is widely used in various domains nowadays.
- A reliable DL system is crucial, especially for safety-critical applications.

Self-driving Cars



Medical Diagnosis

Motivation

- For reproducibility and stability purposes of using DL systems, it is expected that DL systems can have a deterministic behavior in identical training runs: under a fixed software-level and hardware-level experimental condition, multiple training runs produce the same model and result in identical evaluation results.

- Unfortunately, DL systems are usually nondeterministic, i.e., multiple identical training runs can generate inconsistent models and yield different evaluation results.

- Software-level and hardware-level nondeterminism factors.

weight initialization
shuffled batch ordering, etc.

GPU/CPU (**rarely been studied**)

Randomness

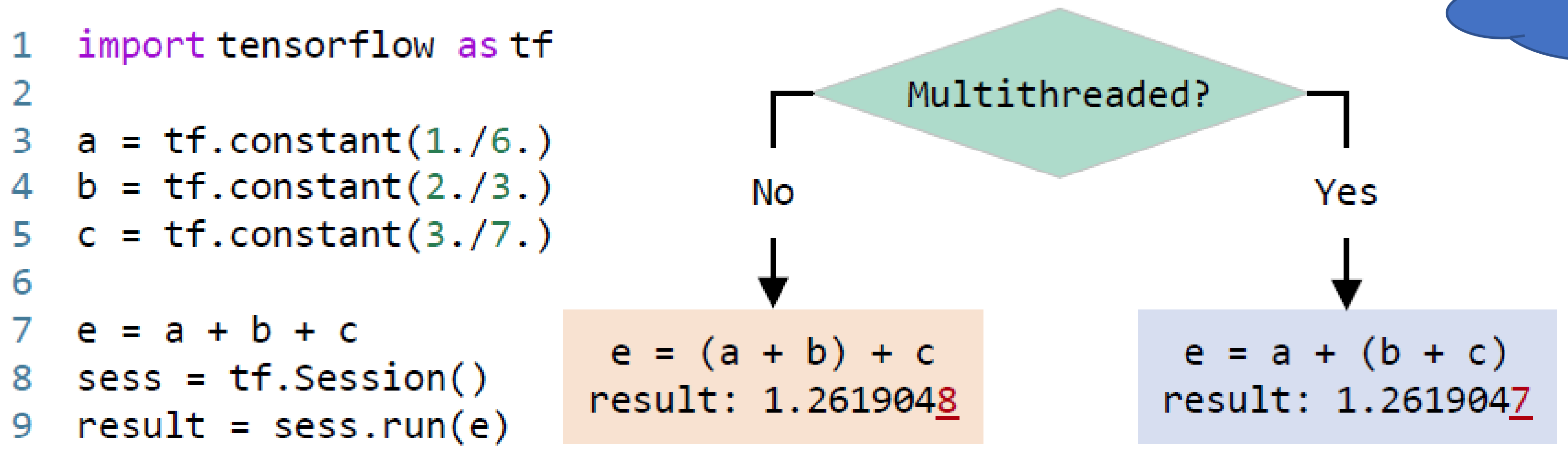
Motivation

Example: the expression e = a + b + c can be computed in two different ways:

(1) e = (a + b) + c = 1.2619048

(2) e = a + (b + c) = 1.2619047

Floating-point imprecision



```
import tensorflow as tf

a = tf.constant(1./6.)
b = tf.constant(2./3.)
c = tf.constant(3./7.)

e = a + b + c
sess = tf.Session()
result = sess.run(e)
```

Fig. 1. Floating-point difference impacted by different computing orders.

Experimental Framework

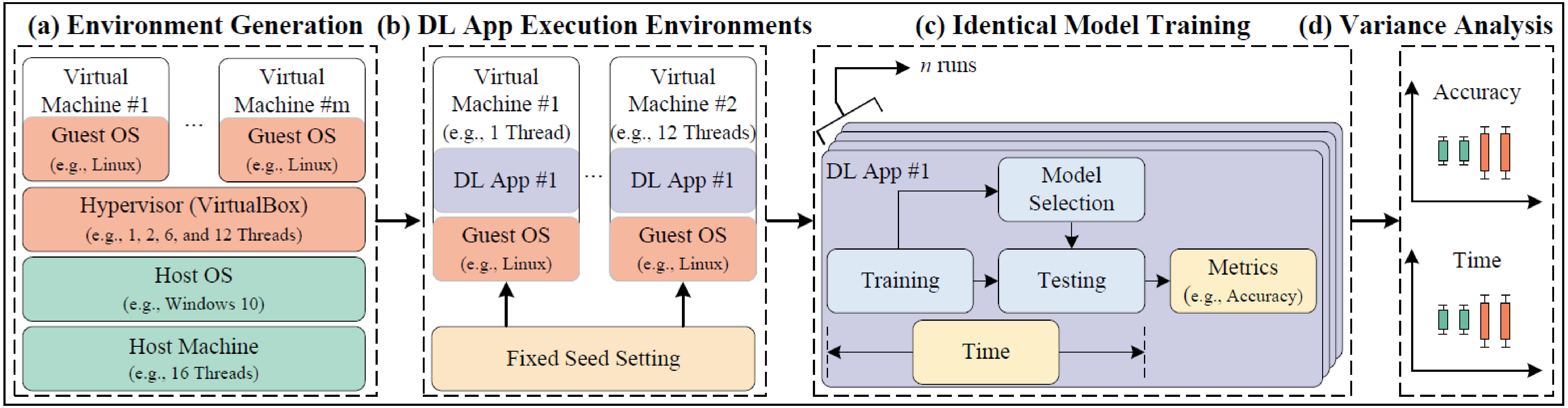

(a) Environment Generation
(b) DL App Execution Environments
(c) Identical Model Training
(d) Variance Analysis
Virtual Machine #1
Guest OS (e.g., Linux)
Virtual Machine #m
Guest OS (e.g., Linux)
Hypervisor (VirtualBox) (e.g., 1, 2, 6, and 12 Threads)
Host OS (e.g., Windows 10)
Host Machine (e.g., 16 Threads)
Virtual Machine #1 (e.g., 1 Thread)
DL App #1
Guest OS (e.g., Linux)
Virtual Machine #2 (e.g., 12 Threads)
DL App #1
Guest OS (e.g., Linux)
Fixed Seed Setting
n runs
DL App #1
Model Selection
Training
Testing
Metrics (e.g., Accuracy)
Time
Accuracy
Time

Fig. 2. Overview of our experimental framework.

Research Questions

- RQ1:
What's the impact of CPU multithreading on the effectiveness variance (e.g., accuracy) in training DL systems?

- RQ2:
What's the impact of CPU multithreading on the time variance in training DL systems?

- RQ3:
How many collected DL projects in GitHub clearly mentioned software and hardware requirements?

Contributions

- An experimental framework based on VirtualBox for analyzing the impact of CPU multithreading on training DL systems
- Six findings obtained from our experiments and examination on GitHub DL projects
- Five implications to DL researchers and practitioners according to our findings
- Released the research data (https://github:com/DeterministicDeepLearning)

Data Collection and Aggregation

To collect our research data, we manually search projects with more than 200 stars in GitHub using two keywords, i.e., deep learning and neural network. After initial filtering, we gather a total of 1,610 projects, as shown in TABLE I.

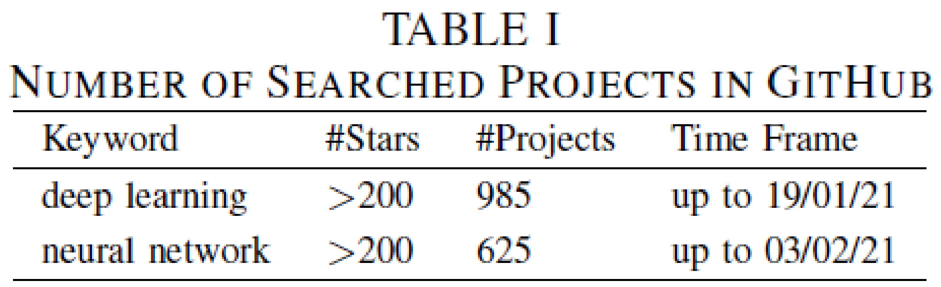


TABLE I
NUMBER OF SEARCHED PROJECTS IN GITHUB

Keyword	#Stars	#Projects	Time Frame
deep learning	>200	985	up to 19/01/21
neural network	>200	625	up to 03/02/21

Data Collection and Aggregation

- Data Clean (e.g., excluding non-DL projects)
- Examined DL Projects (e.g., networks, frameworks, and languages)
- Examine Requirements (software and hardware requirements)

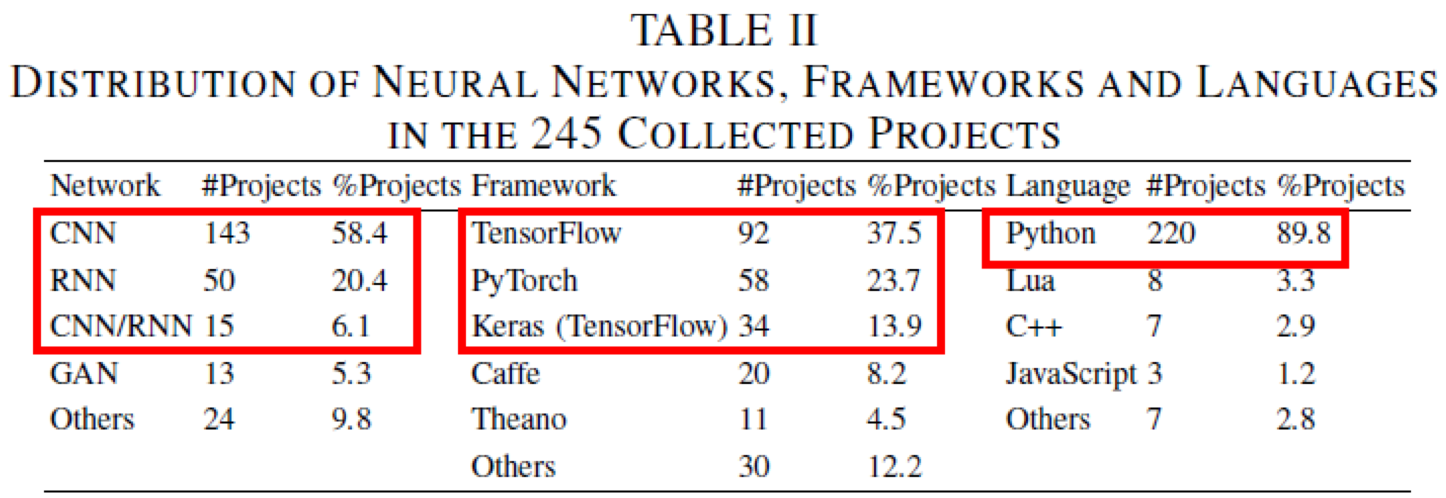


TABLE II
DISTRIBUTION OF NEURAL NETWORKS, FRAMEWORKS AND LANGUAGES IN THE 245 COLLECTED PROJECTS

Network	#Projects	%Projects	Framework	#Projects	%Projects	Language	#Projects	%Projects
CNN	143	58.4	TensorFlow	92	37.5	Python	220	89.8
RNN	50	20.4	PyTorch	58	23.7	Lua	8	3.3
CNN/RNN	15	6.1	Keras (TensorFlow)	34	13.9	C++	7	2.9
GAN	13	5.3	Caffe	20	8.2	JavaScript	3	1.2
Others	24	9.8	Theano	11	4.5	Others	7	2.8
			Others	30	12.2			

Our Experimental Framework

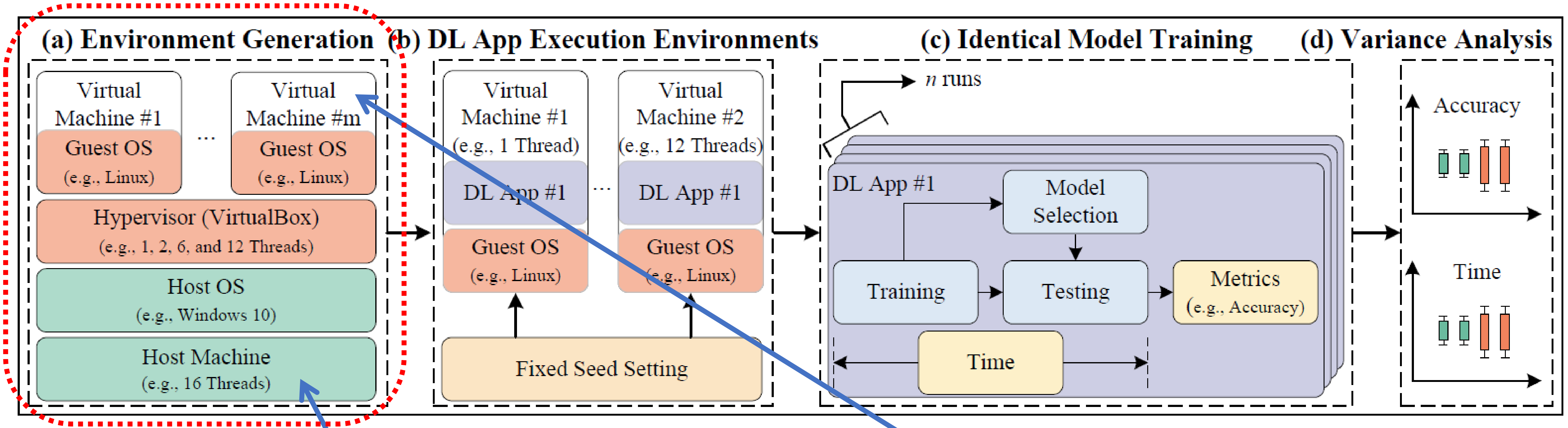
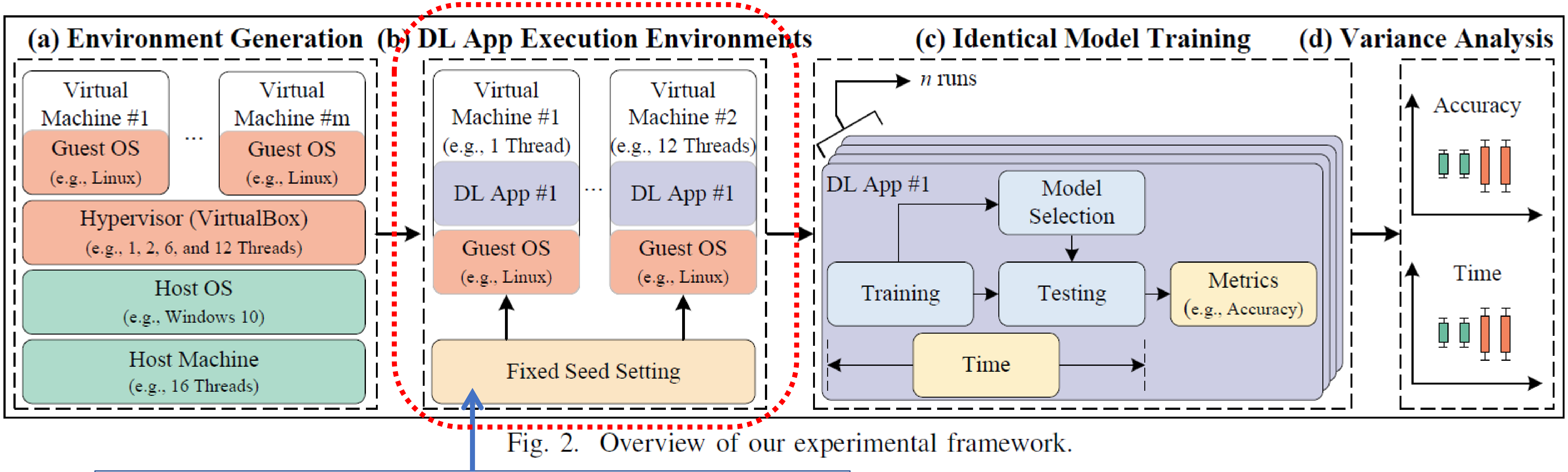

(a) Environment Generation
Virtual Machine #1
Guest OS (e.g., Linux)
...
Virtual Machine #m
Guest OS (e.g., Linux)
Hypervisor (VirtualBox) (e.g., 1, 2, 6, and 12 Threads)
Host OS (e.g., Windows 10)
Host Machine (e.g., 16 Threads)
(b) DL App Execution Environments
Virtual Machine #1 (e.g., 1 Thread)
DL App #1
Guest OS (e.g., Linux)
Virtual Machine #2 (e.g., 12 Threads)
DL App #1
Guest OS (e.g., Linux)
Fixed Seed Setting
(c) Identical Model Training
n runs
DL App #1
Model Selection
Training
Testing
Metrics (e.g., Accuracy)
Time
(d) Variance Analysis
Accuracy
Time

Fig. 2. Overview of our experimental framework.

Host Machine:
CPU: i9-9900K (8 cores 16 threads)
Memory: 64 GB
Storage: 2 TB HDD
Host OS: Windows 10 Pro (20H2)

Virtual Machine:
CPU Threads: 1/2/6/12 Threads
Memory: 24 GB
Storage: 500 GB
Guest OS: Ubuntu 18.04.5 LTS Desktop

Our Experimental Framework



(a) Environment Generation
Virtual Machine #1
Guest OS (e.g., Linux)
Virtual Machine #m
Guest OS (e.g., Linux)
Hypervisor (VirtualBox) (e.g., 1, 2, 6, and 12 Threads)
Host OS (e.g., Windows 10)
Host Machine (e.g., 16 Threads)
(b) DL App Execution Environments
Virtual Machine #1 (e.g., 1 Thread)
DL App #1
Guest OS (e.g., Linux)
Virtual Machine #2 (e.g., 12 Threads)
DL App #1
Guest OS (e.g., Linux)
Fixed Seed Setting
(c) Identical Model Training
n runs
DL App #1
Model Selection
Training
Testing
Metrics (e.g., Accuracy)
Time
(d) Variance Analysis
Accuracy
Time

Fig. 2. Overview of our experimental framework.

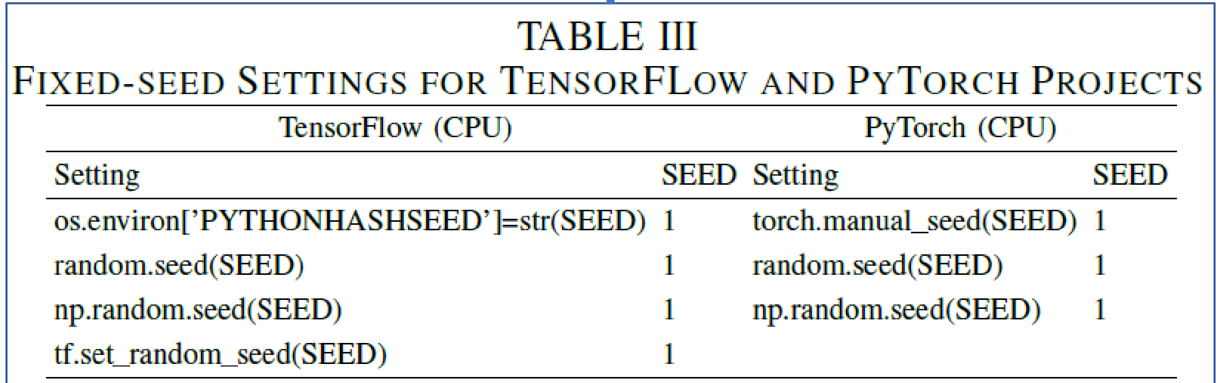


TABLE III
FIXED-SEED SETTINGS FOR TENSORFLOW AND PYTORCH PROJECTS

TensorFlow (CPU)		PyTorch (CPU)	
Setting	SEED	Setting	SEED
os.environ['PYTHONHASHSEED']=str(SEED)	1	torch.manual_seed(SEED)	1
random.seed(SEED)	1	random.seed(SEED)	1
np.random.seed(SEED)	1	np.random.seed(SEED)	1
tf.set_random_seed(SEED)	1		

Control software-level randomness
e.g., Python/Numpy/TF random seed

Our Experimental Framework

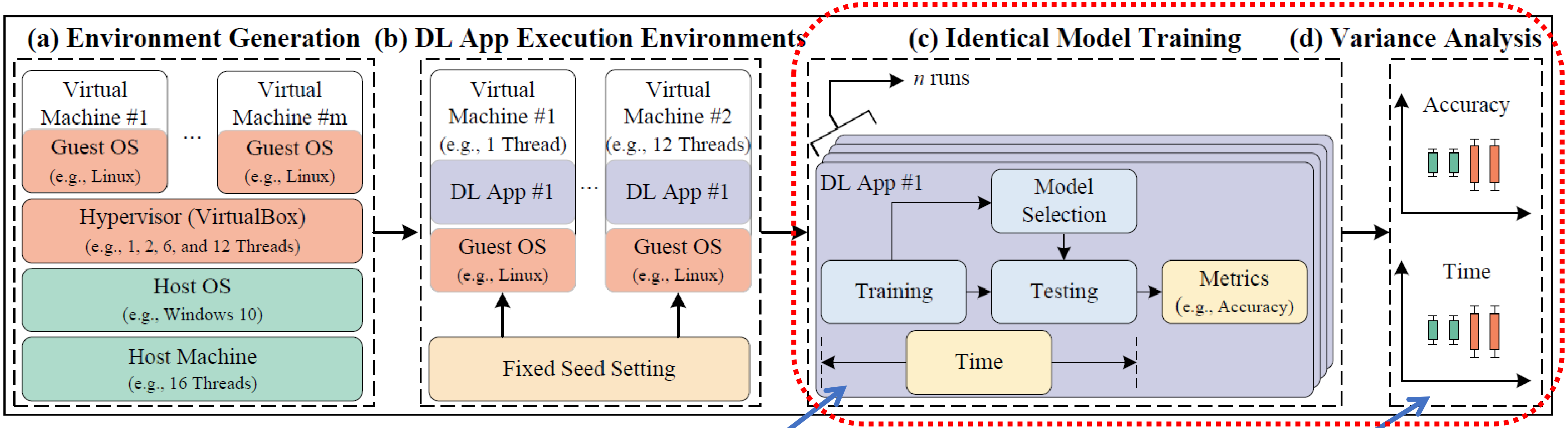


Fig. 2. Overview of our experimental framework.

Three types of identical training runs:
(1) default runs with model selection
(2) fixed-seed runs with model selection
(3) fixed-seed runs without model selection

Variance analysis:
Boxplots and statistics test

Our Experimental Framework

- Details of examined DL projects

  Five projects: two Keras, two TensorFlow, and one PyTorch

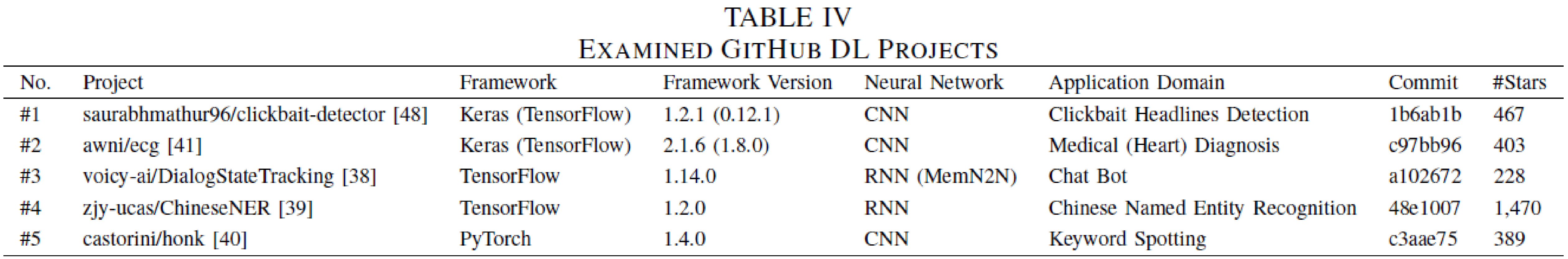


TABLE IV
EXAMINED GITHUB DL PROJECTS

No.	Project	Framework	Framework Version	Neural Network	Application Domain	Commit	#Stars
#1	saurabhmathur96/clickbait-detector [48]	Keras (TensorFlow)	1.2.1 (0.12.1)	CNN	Clickbait Headlines Detection	1b6ab1b	467
#2	awni/ecg [41]	Keras (TensorFlow)	2.1.6 (1.8.0)	CNN	Medical (Heart) Diagnosis	c97bb96	403
#3	voicy-ai/DialogStateTracking [38]	TensorFlow	1.14.0	RNN (MemN2N)	Chat Bot	a102672	228
#4	zjy-ucas/ChineseNER [39]	TensorFlow	1.2.0	RNN	Chinese Named Entity Recognition	48e1007	1,470
#5	castorini/honk [40]	PyTorch	1.4.0	CNN	Keyword Spotting	c3aae75	389

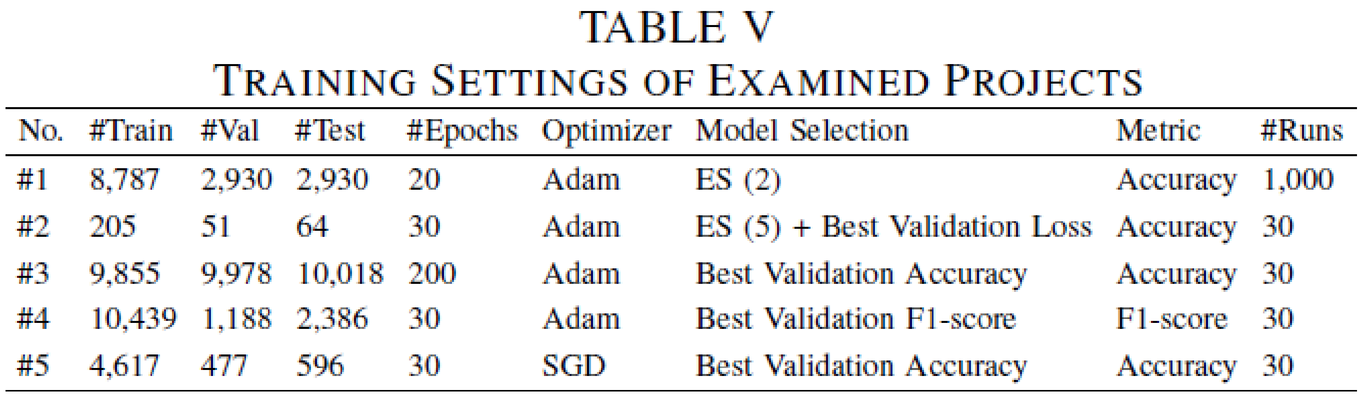
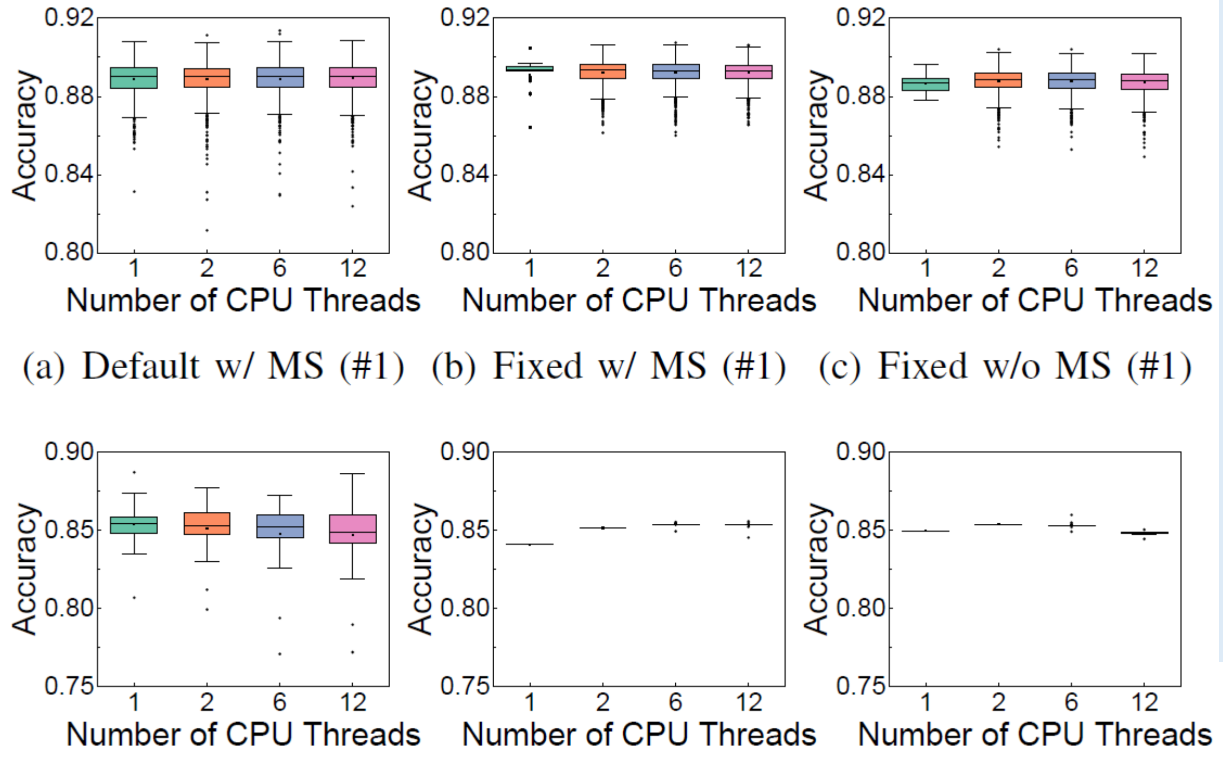


TABLE V
TRAINING SETTINGS OF EXAMINED PROJECTS

No.	#Train	#Val	#Test	#Epochs	Optimizer	Model Selection	Metric	#Runs
#1	8,787	2,930	2,930	20	Adam	ES (2)	Accuracy	1,000
#2	205	51	64	30	Adam	ES (5) + Best Validation Loss	Accuracy	30
#3	9,855	9,978	10,018	200	Adam	Best Validation Accuracy	Accuracy	30
#4	10,439	1,188	2,386	30	Adam	Best Validation F1-score	F1-score	30
#5	4,617	477	596	30	SGD	Best Validation Accuracy	Accuracy	30

Results and Analysis

- RQ1: Impact of CPU Multithreading on Effectiveness Variance



(a) Default w/ MS (#1) (b) Fixed w/ MS (#1) (c) Fixed w/o MS (#1)

(d) Default w/ MS (#2) (e) Fixed w/ MS (#2) (f) Fixed w/o MS (#2)

Fig. 4. Boxplots of accuracy variance of projects #1 and #2.

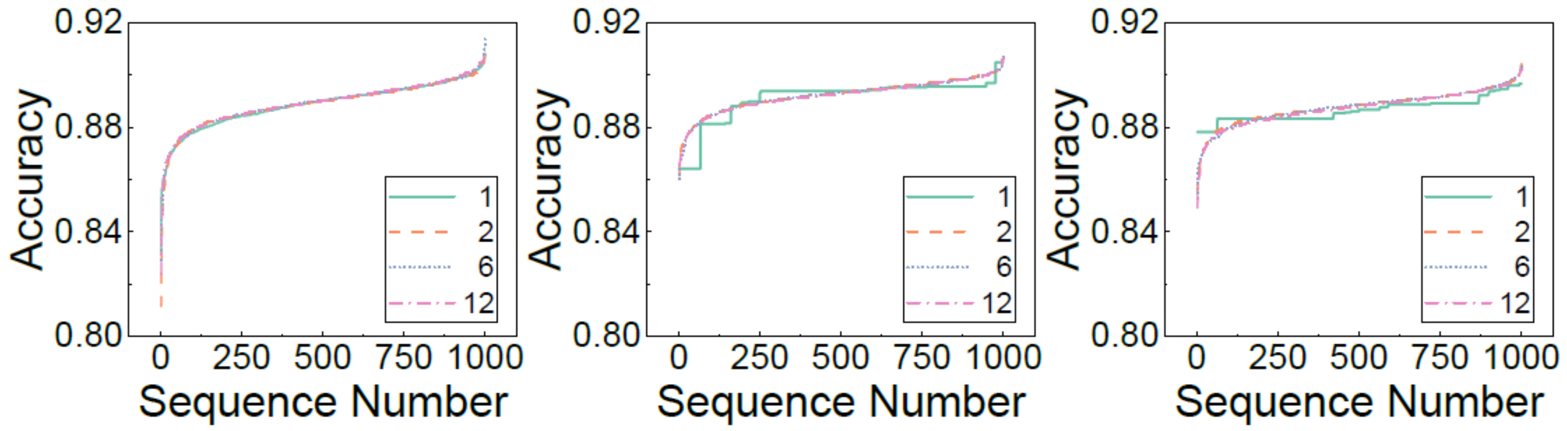
- **Finding #1:** For the examined two Keras (with TensorFlow backend) projects, the accuracy variances obtained from default identical training runs (without controlling random generators) have no significant difference in different CPU multithreaded environments. The nondeterminism factors cause accuracy differences as large as 11.43%. However, after controlling random generators, we can observe the impact of CPU multithreading on the accuracy variance (e.g., different training accuracy).

Results and Analysis

- RQ1: Impact of CPU Multithreading on Effectiveness Variance
  - **Finding #1 (cont'd):** Although variance still exists in fixed-seed identical runs, some accuracy scores keep occurring in a single-threaded environment.

Single-threaded env: 89.39%-334 times

Single-threaded env: 88.33%-358 times

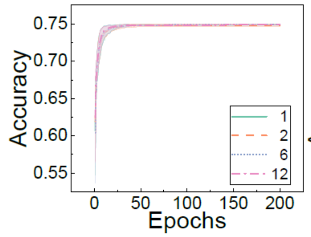


(a) Default w/ MS (b) Fixed w/ MS (c) Fixed w/o MS

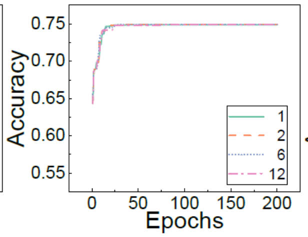
Fig. 5. Sorting of accuracy variance of project #1.

Results and Analysis

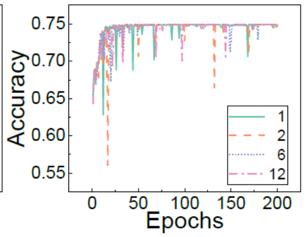
- RQ1: Impact of CPU Multithreading on Effectiveness Variance



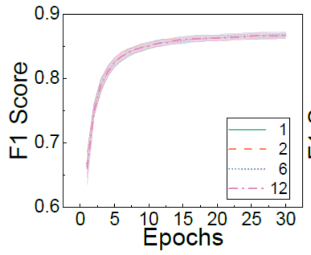
(a) Default w/ MS (#3)



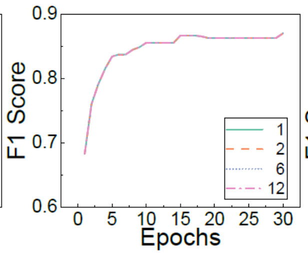
(b) Fixed w/ MS (#3)



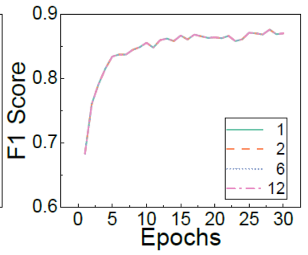
(c) Fixed w/o MS (#3)



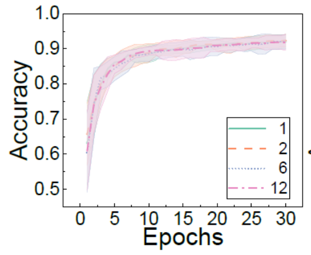
(d) Default w/ MS (#4)



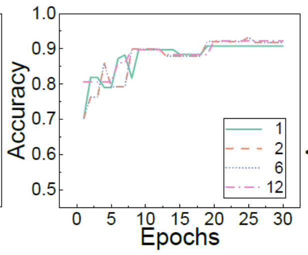
(e) Fixed w/ MS (#4)



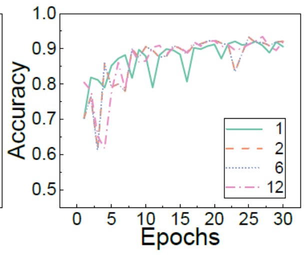
(f) Fixed w/o MS (#4)



(g) Default w/ MS (#5)



(h) Fixed w/ MS (#5)



(i) Fixed w/o MS (#5)

- **Finding #2:** Similar to the Keras projects, for the examined TensorFlow and PyTorch projects, default identical training runs produce multiple models with different evaluation results. However, fixed-seed identical training runs produce deterministic evaluation results in the environment with the same number of CPU threads. The deterministic evaluation results of one project can be different across different environments.

Results and Analysis

- RQ1: Impact of CPU Multithreading on Effectiveness Variance

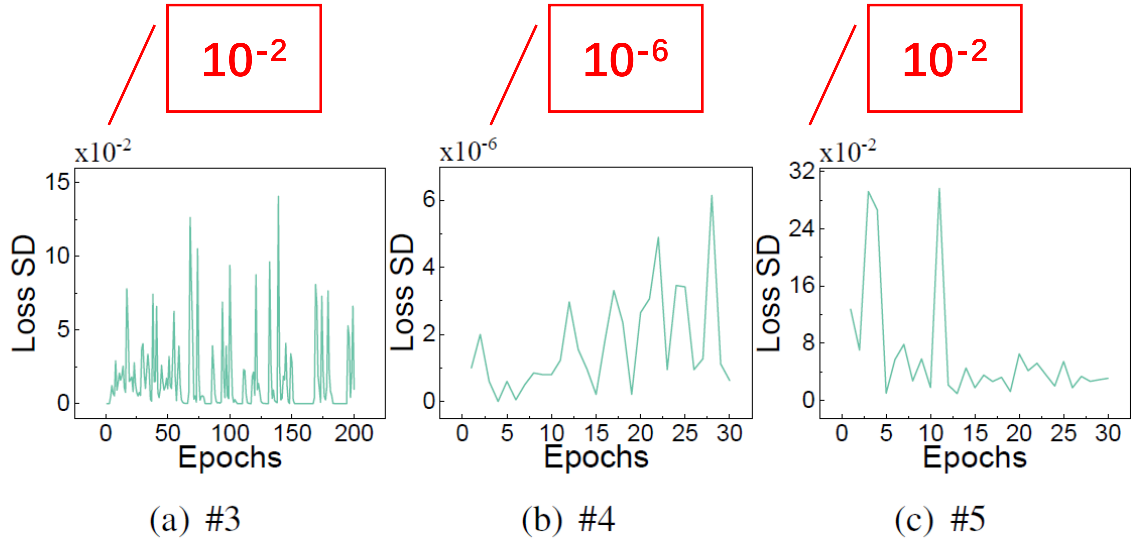
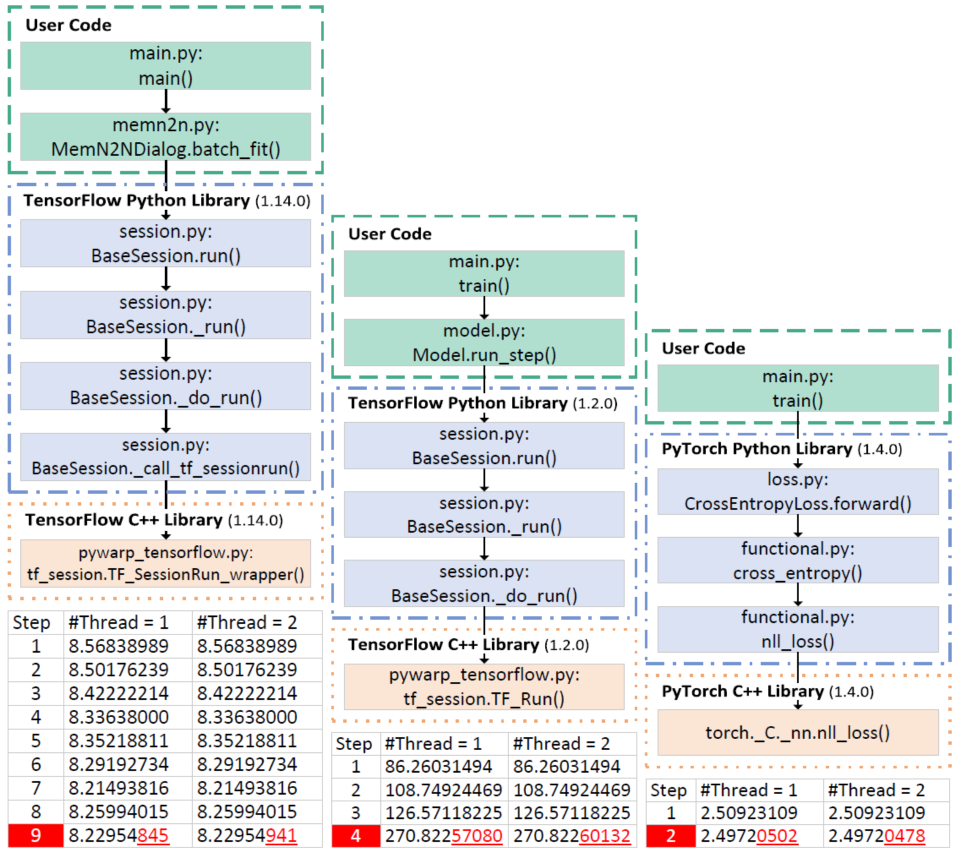


Fig. 7. Standard deviation (SD) calculated from training loss of fixed-seed runs of projects #3-5.

- **Finding #2:** Similar to the Keras projects, for the examined TensorFlow and PyTorch projects, default identical training runs produce multiple models with different evaluation results. However, fixed-seed identical training runs produce deterministic evaluation results in the environment with the same number of CPU threads. The deterministic evaluation results of one project can be different across different environments.

Results and Analysis

- RQ1: Impact of CPU Multithreading on Effectiveness Variance



User Code
main.py: main()
memn2n.py: MemN2NDialog.batch_fit()
TensorFlow Python Library (1.14.0)
session.py: BaseSession.run()
session.py: BaseSession._run()
session.py: BaseSession._do_run()
session.py: BaseSession._call_tf_sessionrun()
TensorFlow C++ Library (1.14.0)
pywarp_tensorflow.py: tf_session.TF_SessionRun_wrapper()
User Code
main.py: train()
model.py: Model.run_step()
TensorFlow Python Library (1.2.0)
session.py: BaseSession.run()
session.py: BaseSession._run()
session.py: BaseSession._do_run()
TensorFlow C++ Library (1.2.0)
pywarp_tensorflow.py: tf_session.TF_Run()
User Code
main.py: train()
PyTorch Python Library (1.4.0)
loss.py: CrossEntropyLoss.forward()
functional.py: cross_entropy()
functional.py: nll_loss()
PyTorch C++ Library (1.4.0)
torch._C._nn.nll_loss()

Step	#Thread = 1	#Thread = 2
1	8.56838989	8.56838989
2	8.50176239	8.50176239
3	8.42222214	8.42222214
4	8.33638000	8.33638000
5	8.35218811	8.35218811
6	8.29192734	8.29192734
7	8.21493816	8.21493816
8	8.25994015	8.25994015
9	8.22954845	8.22954941

(a) #3

Step	#Thread = 1	#Thread = 2
1	86.26031494	86.26031494
2	108.74924469	108.74924469
3	126.57118225	126.57118225
4	270.82257080	270.82260132

(b) #4

Step	#Thread = 1	#Thread = 2
1	2.50923109	2.50923109
2	2.49720502	2.49720478

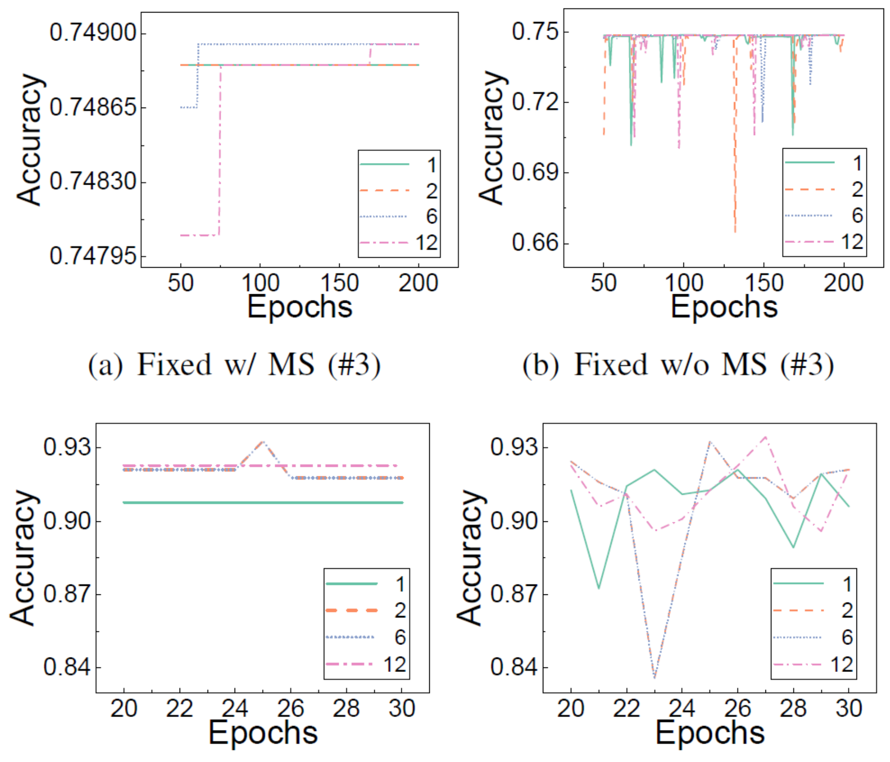
(c) #5

Fig. 8. Loss value differences in user and framework code for projects #3-5.

- **Finding #3:** The rounding errors of floating-point numbers come from low-level implementations of DL frameworks, i.e., the C++ library. Under different CPU multithreaded environments, the rounding errors are accumulated during the training process. After some training steps, the values of training loss become different.

Results and Analysis

- RQ1: Impact of CPU Multithreading on Effectiveness Variance



(a) Fixed w/ MS (#3) (b) Fixed w/o MS (#3)

(c) Fixed w/ MS (#5) (d) Fixed w/o MS (#5)

Fig. 9. Accuracy variance of project #3 from 50-200 training epochs and project #5 from 20-30 training epochs.

- **Finding #4:** For fixed-seed identical training runs without model selection, CPU multithreading causes accuracy differences as large as 8.56%. Using model selection can minimize the impact of CPU multithreading on training DL systems. The max accuracy differences are range from 0.0898% to 2.52% with model selection.

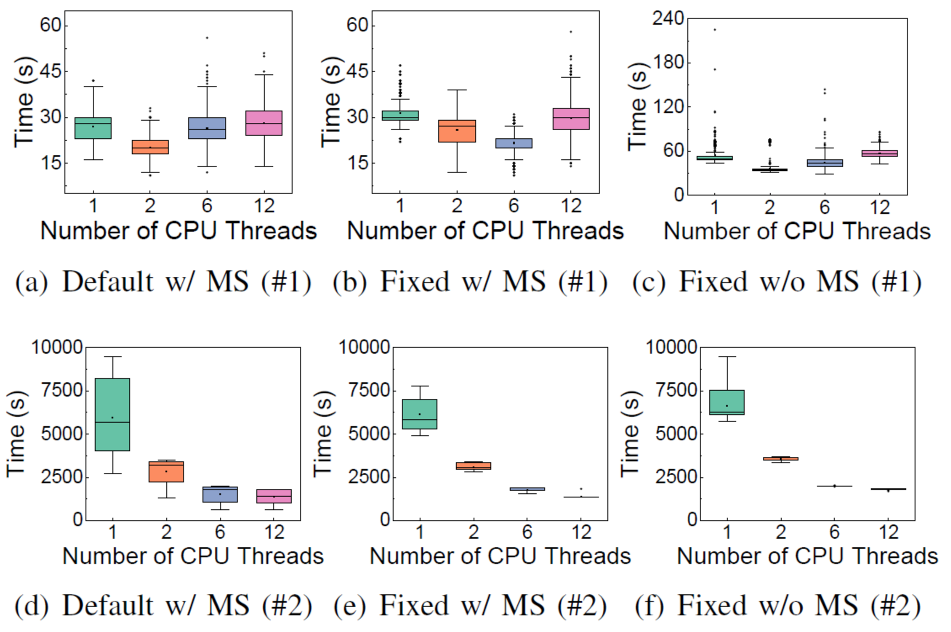
TABLE VII
MAXIMUM ACCURACY DIFFERENCE OF PROJECTS #3 AND #5

#Threads	Fixed w/ MS		Fixed w/o MS	
	#3 (Epochs 61–74)	#5 (Epoch 25)	#3 (Epoch 132)	#5 (Epoch 23)
1	0.748852066	0.907718122	0.748852066	0.921140969
2	0.748852066	0.932885885	0.664703534	0.835570455
6	0.748951887	0.932885885	0.748951887	0.835570455
12	0.748053504	0.922818780	0.748652426	0.895973146
Max Diff	0.000898383	0.025167763	0.084248353	0.085570514

Results and Analysis

- RQ2: Impact of CPU Multithreading on Time Variance

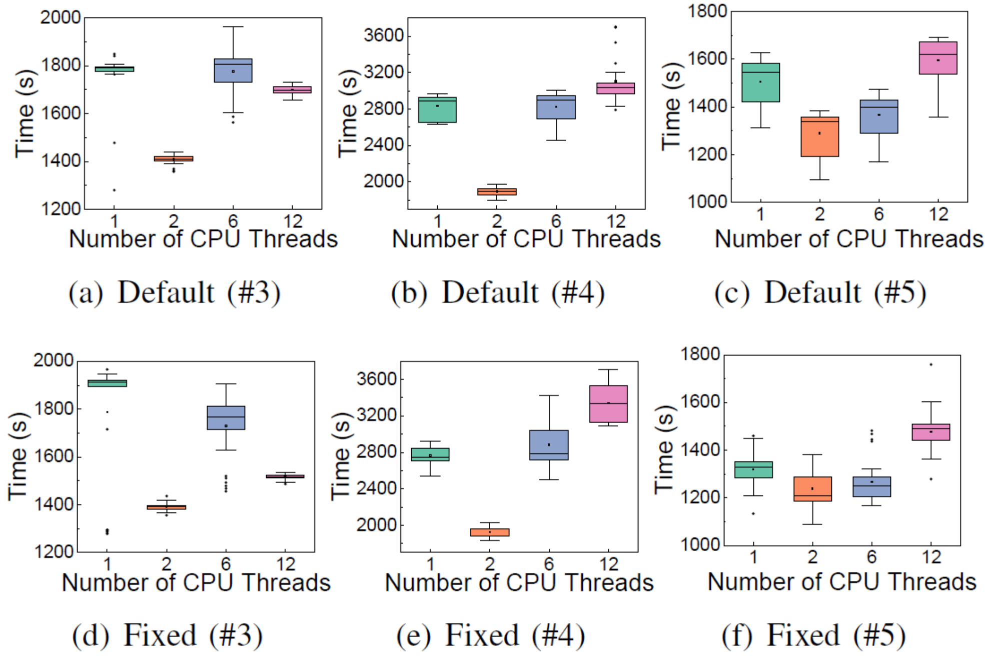
- **Finding #5:** For the examined five DL projects, only one project shows that the training time would decrease along with the increasing number of CPU threads. For most projects, using 2 threads would be a better choice to obtain a faster training efficiency.



(a) Default w/ MS (#1) (b) Fixed w/ MS (#1) (c) Fixed w/o MS (#1)

(d) Default w/ MS (#2) (e) Fixed w/ MS (#2) (f) Fixed w/o MS (#2)

Fig. 10. Boxplots of training time variance of projects #1 and #2.



(a) Default (#3) (b) Default (#4) (c) Default (#5)

(d) Fixed (#3) (e) Fixed (#4) (f) Fixed (#5)

Fig. 11. Boxplots of training time variance of projects #3-5.

Results and Analysis

- RQ3: Proportions of DL Projects that Mentioned Software and Hardware Requirements

  - **Finding #6:** Among the 245 collected GitHub DL projects, 90.6% of them provide software-level requirements, while only 22.0% mention hardware-level requirements. For the projects that have hardware requirements (54 projects), 94.4% present tested GPU environments, while 14.8% list tested CPU environments.

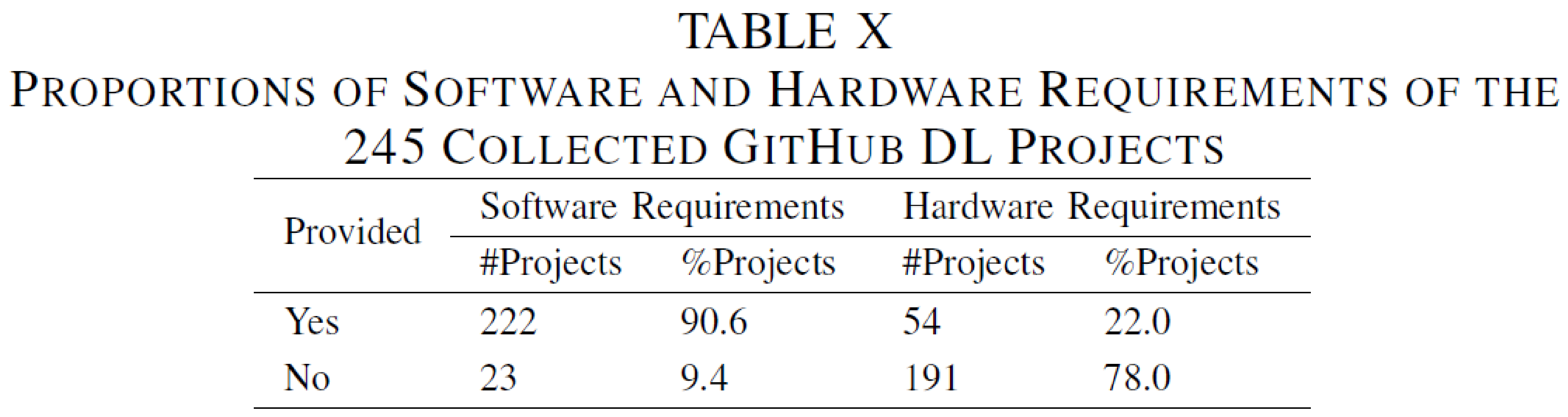


TABLE X
PROPORTIONS OF SOFTWARE AND HARDWARE REQUIREMENTS OF THE 245 COLLECTED GITHUB DL PROJECTS

Provided	Software Requirements		Hardware Requirements	
	#Projects	%Projects	#Projects	%Projects
Yes	222	90.6	54	22.0
No	23	9.4	191	78.0

Implications for DL Researchers and Practioners

- **Implication #1:** When training DL systems on CPU platforms, to obtain deterministic training results, developers should control software-level nondeterminism factors (e.g., random generators). Besides, practitioners should pay attention to the effectiveness variance under different CPU multithreaded environments.

- **Implication #2:** The effectiveness variance introduced by CPU multithreading could be huge, in particular when training DL systems without using model selection techniques. To minimize the impact of CPU multithreading on the effectiveness variance in training DL systems, developers are suggested to use model selection methods (e.g., best validation loss/accuracy). Moreover, to achieve deterministic DL systems, it is necessary to develop mitigation techniques to eliminate the inconsistency of accuracy scores introduced by CPU multithreading.

Implications for DL Researchers and Practioners

- **Implication #3**: Using more CPU threads for model training does not necessarily indicate that we can achieve faster training efficiency. DL models with a default setting (e.g., using all CPU threads) do not often take full advantage of computing capability of the underlying hardware [55]. For obtaining a promising training time, developers are suggested to use some auto-tuning tools (e.g., TensorTuner [35]) to fine-tune the built-in threading configuration in DL frameworks (e.g., TensorFlow).

- **Implication #4**: When optimizing CPU parallelism for training DL systems, developers should pay attention to the potential impact on the effectiveness variance introduced by CPU multithreading. For example, it would be necessary to consider the task as a multi-objective optimization problem, i.e., minimize the training time and maximize the effectiveness (e.g., accuracy).

Implications for DL Researchers and Practioners

- **Implication #5:** Developers are suggested to mention hardware environments used for model training and evaluation when releasing a DL project. In particular, if DL models are trained on CPU platforms, developers should provide detailed CPU models, since the number of threads used can affect model training and evaluation.

Contributions

- An experimental framework based on VirtualBox for analyzing the impact of CPU multithreading on training DL systems
- Six findings obtained from our experiments and examination on GitHub DL projects
- Five implications to DL researchers and practitioners according to our findings
- Released the research data (https://github:com/DeterministicDeepLearning)

Future Work

- Collect and examine more DL projects (models)

- Perform experiments on cloud platforms (e.g., using more powerful CPUs)

Thank you for your listening!

Q&A

Guanping Xiao

Email: gpxiao@nuaa.edu.cn
Homepage: https://guanpingxiao.github.io/