The 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2023)

COMPATIBILITY ISSUES IN DEEP LEARNING SYSTEMS: PROBLEMS AND OPPORTUNITIES

Jun Wang¹, Guanping Xiao¹, Shuai Zhang¹, Huashan Lei¹, Yepang Liu², Yulei Sui³

¹Nanjing University of Aeronautics and Astronautics, China

²Southern University of Science and Technology, China

³University of New South Wales, Australia



^{05/12/2023} San Francisco, USA



Traditional Software

- Mahmud T et al. Detecting Android API Compatibility Issues With API Differences. [TSE 2023]
- Xavier L et al. Historical and Impact Analysis of API Breaking Changes: A Large-Scale Study. [SANER 2017]

Deep Learning Systems:

- Data-Driven
- Require substantial computational resources







Deep Learning Field

- Humbatova N et al. **Taxonomy of Real Faults in Deep Learning** Systems. [ICSE 2020]
- Islam M J et al. A Comprehensive Study on Deep Learning Bug Characteristics. [ESEC/FSE 2019]



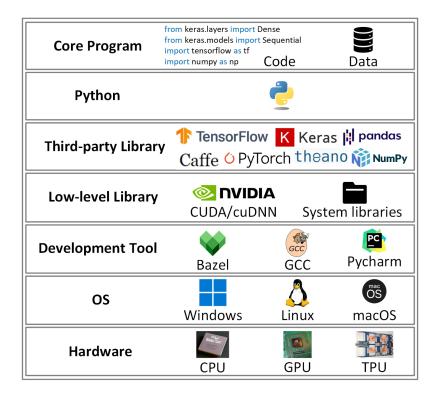
- Comprehensive taxonomies
- Library API Evolution
- Dependency conflict



Motivation



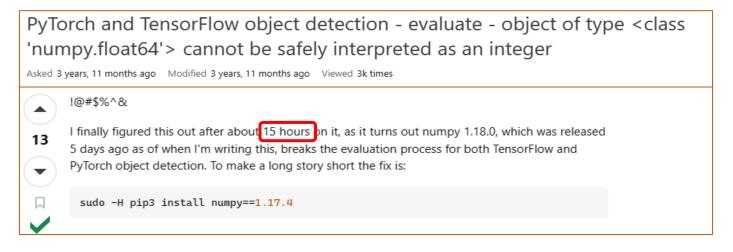
Different components of deep learning systems have complex interactions and asynchronous evolution.



Motivation Examples









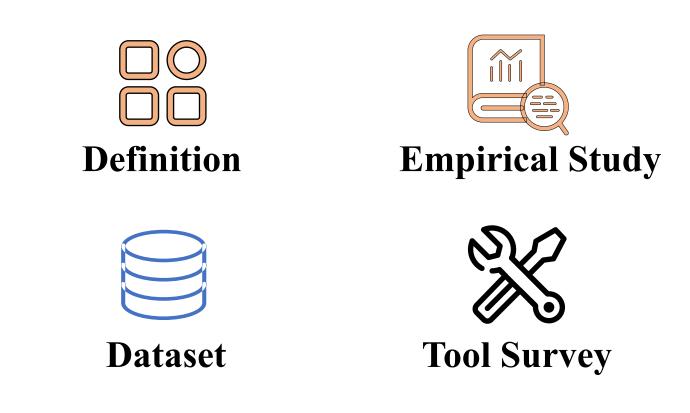
RQ1. What **types** of compatibility issues are frequently exposed in DL systems?

RQ2. What are the **root causes** of DL compatibility issues and how do developers **fix them**?

RQ3. How far are we from **automatically detecting and fixing** DL compatibility issues?

Contribution





Methodology

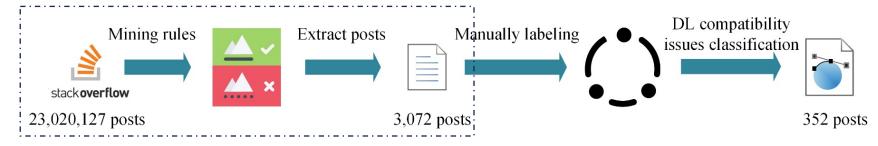


Data Collection and Classification

Dataset: *stackoverflow.com-Posts*.7*z*, *stackoverflow.com-Tags*.7*z*

Filtering Rules: ① question posts, ② open for discussion, ③ have accepted answer

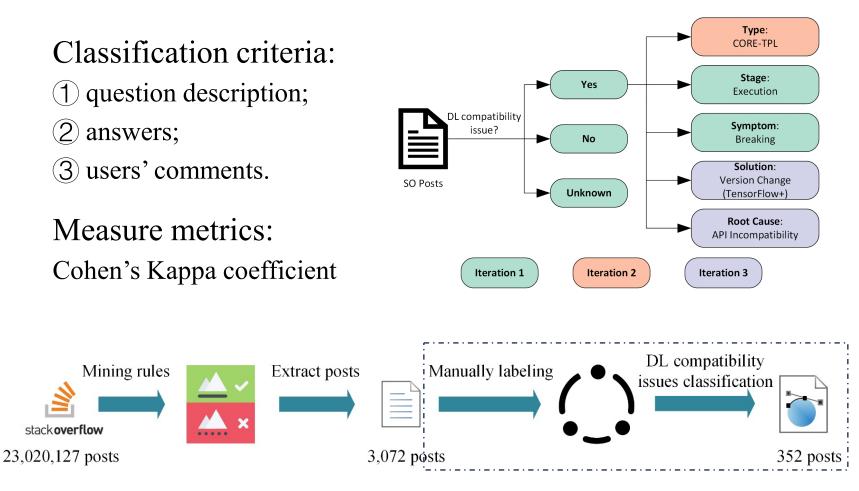
Framework	Post Numbers	412	keywo	ords, (5) 5	Frameworks
1 TensorFlow	79,018			12 Keywords	
K Keras	48,581	cuda	typeerror	importerror	attributeerror
🗘 PyTorch	18,764	cudnn	exception	compatible	modulenotfounderror
Caffe	2,845	version	evolution	compatibility	runtimeerror
theano	2,447				



Methodology

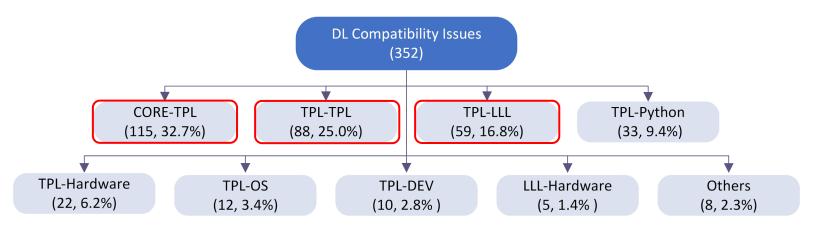


Data Collection and Classification





RQ1. <u>Types</u>, Stages and Symptoms



CORE: Core Program, TPL: Third-party Library, LLL: Low-level Library, DEV: Development Tool, OS: Operating System

Answer to RQ1: CORE-TPL is the most frequent type of DL compatibility issues. TPL-TPL and TPL-LLL are the second the third types frequently exposed.

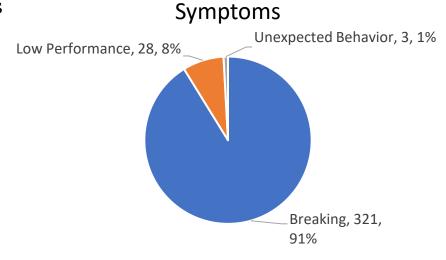


RQ1. Types, Stages and Symptoms

Table 2: Distribution of Stages and Types

Туре	Installation	Execution	Total
CORE-TPL	0	115	115
TPL-TPL	5	83	88
TPL-LLL	3	56	59
TPL-Python	28	5	33
TPL-Hardware	1	21	22
TPL-OS	6	6	12
TPL-DEV	6	4	10
LLL-Hardware	2	3	5
Others	1	7	8
Total	52	300	352

Answer to RQ1: Most of DL compatibility issues were exposed during the execution stage. CORE-TPL, TPL-TPL, TPL-LLL, and TPL-Hardware are prone to occur during execution, while TPL-Python and TPL-DEV are likely to appear in the installation.

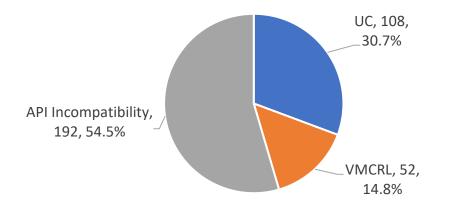


Answer to RQ1: Most of DL compatibility issues have a breaking impact on the installation stage or execution stage of DL systems.



RQ2. Root Causes and Solutions

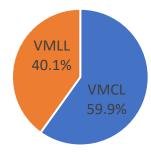
- RC1: API Incompatibility (192/352).
- RC2: Unsupported Component (UC) (108/352).
- RC3: Version Mismatch Between Compiled and Runtime Libraries (VMCRL) (52/352).





RQ2. Root Causes and Solutions

• RC1: API Incompatibility (192/352).



• Version Mismatch Between CORE and Libraries' APIs (VMCL) (115/192)

installed TensorFlow 0.12cross_entropy = tf.nn.sigmoid_cross_entropy_with_logits(labels=y_, logits=y)

TypeError: sigmoid_cross_entropy_with_logits() got an unexpected keyword argument 'labels'

The keyword argument '*labels*' only exists in **TensorFlow 1.0.0+**

• Version Mismatch Between Libraries' APIs (VMLL) (77/192)

installed TensorFlow 1.2.1, Keras 2.1.3
File "main.py", line 84, in build_discriminator
model.add(LeakyReLU(alpha=0.2))

File "/opt/libraries/anaconda2/lib/python2.7/site-packages/keras/ backend/ tensorflow_backend.py", line 2918, in relu

x = tf.nn.leaky_relu(x, alpha)

The function *leaky_relu* was added to **TensorFlow 1.4**.



RQ2. Root Causes and <u>Solutions</u>

• Version Change.

Share Follow

This error is being raised because the loader cannot find version 7.0 of the CUDA runtime on your system. <u>TensorFlow requires CUDA 7.0.</u> From the path in your question (/usr/local/cuda-5.5/...) it looks like you have CUDA 5.5

installed. The solution is to upgrade your CUDA runtime to version 7.0, which is available from NVIDIA here.

Upgrade your CUDA runtime to version 7.0

• Code Change.

6

- Change API
- Change Import
- Change Build

keras 2.0.0 from keras.layers.core import Dense, Dropout, Activation, Merge, Reshape

Change import

keras 2.0.0
from keras.layers import Merge



location ×



RQ2. Root Causes and <u>Solutions</u>

Table 6: Distribution of Solutions to API Incompatibility

Solution		V	MCL	VMLL	Total	
		forward	backward	VIVILL		
Version Change		31	6	57	94	
	Change API	9	40	1	50	
Code Change	Change Import	8	22	19	49	
	Change Build	0	0	0	0	

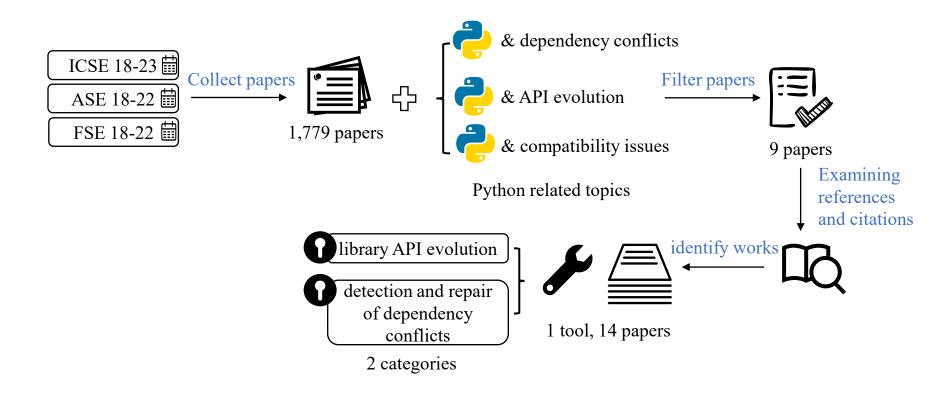
Table 7: Distribution of Solutions to UC and VMCRL

Solution		UC	VMCRL	Total
Version Change		76	52	128
Code Change	Change API	1	0	1
	Change Import	0	0	0
	Change Build	15	0	15

Answer to RQ2: For compatibility issues caused by VMCL, forward-incompatible issues tend to be fixed by changing the TPL version, while backward-incompatible issues tend to be repaired by changing the code. In addition, most of issues caused by VMLL were fixed by version change. **Answer to RQ2**: Most of the issues were caused by UC and all VMCRL-induced issues were solved by changing the component version.



RQ3. How far are we from **automatically detecting and fixing** DL compatibility issues?



Tool Collection



RQ3. How far are we from **automatically detecting and fixing** DL compatibility issues?

Table 8: Library API Evolution						
Tool	Extract Used API	Extract TPL API	Match API	Detect Changes	Repair API	
AexPy [44]		1		✓		
PyCompat [129]	✓	1	1	1		
DLocator [109]	1		 Image: A set of the set of the	1		
MLCatchUp [60]	✓		 ✓ 	1	 Image: A set of the set of the	
APIScanner [103]		1		1		
Relancer [131]				1	1	

Table 9: Detection and Repair of Dependency Conflicts

Tool	Infer TPL	Infer Python	Infer LLL
PyEGo [124]	 Image: A set of the set of the	✓	1
DockerizeMe [63]	 Image: A set of the set of the		1
SnifferDog [110]	 Image: A set of the set of the		
pipreqs [52]	✓		
PyDFix [84]	 Image: A set of the set of the		
V2 [64]	 Image: A set of the set of the		
Watchman [111]	 Image: A set of the set of the		
PyCRE [42]	 Image: A set of the set of the		
smartPip [106]	~		

Answer to RQ3: Of the six tools related to library API evolution, none of them can fully realize the entire automation process from detection to repair of DL compatibility issues caused by API incompatibility. For the resolution of dependency conflicts in Python programs, nine tools attempt to fix dependency conflicts induced by TPL-TPL, but only a few of them can infer the incompatibilities caused by the Python interpreter versions and system libraries. Currently, none of them can detect and fix compatibility issues caused by CUDA/cuDNN which are commonly used in DL systems.



#1: Ensuring Consistency Between API Usage and Installed Library Versions.

#2: Early Determination of Component Versions.

#3: Challenges in Automated Detection and Repair of API Evolution Issues.

#4: Challenges in Automated Detection and Resolution of Incompatible Component Versions.

Future Work



Automatically detect DL compatibility issues



Repair DL compatibility issues

The 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2023)

Thank you for your listening!

Q&A



05/12/2023 San Francisco, USA