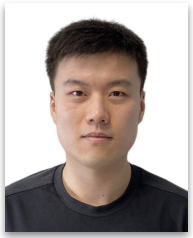
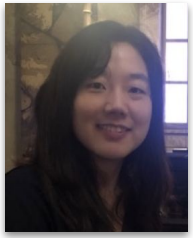
DocTer: Documentation-Guided Fuzzing for Testing Deep Learning API Functions

Danning Xie[1]



Yitong Li[2]



Mijung Kim[3]

Hung Viet Pham[2]

Lin Tan[1]

Xiangyu Zhang[1]

Michael W. Godfrey[2]

1 PURDUE UNIVERSITY

2 UNIVERSITY OF WATERLOO

3 UNIST ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Deep learning (DL) systems are widely deployed

Self-driving cars

Machine translation

Cancer detection

Testing DL APIs is difficult

- It is hard to generate valid inputs for DL APIs,

```
> torch.as_strided(input = [1,1,1], ...)

TypeError: as_strided(): argument 'input' must be Tensor, not list
```

Testing DL APIs is difficult

- It is hard to generate valid inputs for DL APIs,

A tensor is a type of data structure representing a multi-dimensional array.

```
> torch.as_strided(input = [1,1,1], ...)

TypeError: as_strided(): argument 'input' must be Tensor, not list
```

The exceptions prevent us from reaching the core functionality and testing the function deeper.

Testing DL APIs is difficult

- It is hard to generate valid inputs for DL APIs, because they must follow:
  - DL-specific data structures

```
> torch.as_strided(input = [1,1,1], ...)

TypeError: as_strided(): argument 'input' must be Tensor, not list
```

Testing DL APIs is difficult

- It is hard to generate valid inputs for DL APIs, because they must follow:
  - DL-specific data structures

```
> torch.as_strided(input = [1,1,1], ...)
```
❌ **Fail**

```
> torch.as_strided(input = torch.tensor([1,1,1]), ...)
```
✅ **Pass!**

Testing DL APIs is difficult

- It is hard to generate valid inputs for DL APIs, because they must follow:
  - DL-specific data structures

tf.nn.max_pool3d

```
tf.nn.max_pool3d(input, ksize, strides, padding, ...)
```

Tensor

```
> tf.nn.max_pool3d(input = tf.ones((1,1,1,1,1)), ...)
```

```
InvalidArgumentError: tensor_in must be 5-dimensional [Op:MaxPool3D]
```

Testing DL APIs is difficult

- It is hard to generate valid inputs for DL APIs, because they must follow:
  - DL-specific data structures

tf.nn.max_pool3d

```
tf.nn.max_pool3d(input, ksize, strides, padding, …)
```

5 dimensional

```
> tf.nn.max_pool3d(input = tf.ones((1,1,1,1,1)), …)
```

Testing DL APIs is difficult

- It is hard to generate valid inputs for DL APIs, because they must follow:
  - DL-specific data structures

tf.nn.max_pool3d

```
tf.nn.max_pool3d(input, ksize, strides, padding, ...)
```

Zero padding

```
> tf.nn.max_pool3d(input = tf.ones((1,1,1,1,1)), …, padding = "zero")

InvalidArgumentError: Value for attr 'padding' of "zero" is not in the
list of allowed values: "SAME", "VALID"
```

Testing DL APIs is difficult

- It is hard to generate valid inputs for DL APIs, because they must follow:
  - DL-specific data structures
  - DL-specific data structure properties

tf.nn.max_pool3d

```
tf.nn.max_pool3d(input, ksize, strides, padding, …)
```

5 dimensional

"SAME" or "VALID"

```
> tf.nn.max_pool3d(input = tf.ones((1,1,1,1,1)), …, padding = "zero")
```

```
InvalidArgumentError: Value for attr 'padding' of "zero" is not in the
list of allowed values: "SAME", "VALID"
```

Opportunities: API documents contain constraints

- Challenge: valid inputs must follow two types of constraints:
  - DL-specific data structures, and
  - DL-specific data structure properties
- Observation: input constraints are described in API documents
- Solution: extract constraints from document to guide input generation
- **Existing techniques: insufficient for extracting DL-specific constraints**

`tf.nn.max_pool3d`

```
tf.nn.max_pool3d(input, ksize, strides, padding, ...)
```

Args	
`input`	A 5-D `Tensor` of the format specified by `data_format`.
`ksize`	An int or list of ints that has length `1`, `3` or `5`. The size of the window for each dimension of the input tensor.
`strides`	An int or list of ints that has length `1`, `3` or `5`. The stride of the sliding window for each dimension of the input tensor.
`padding`	A string, either `'VALID'` or `'SAME'`....

A running example of **DocTer**

tf.nn.max_pool3d

```
tf.nn.max_pool3d(input, ksize, strides, padding, ...)
```

Args

`input`	A 5-D `Tensor` of the format specified by `data_format`.
`ksize`	An int or list of ints that has length `1`, `3` or `5`. The size of the window for each dimension of the input tensor.
`strides`	An int or list of ints that has length `1`, `3` or `5`. The stride of the sliding window for each dimension of the input tensor.
`padding`	A string, either `'VALID'` or `'SAME'`....

$C(p = \textbf{input}) = p_D \in \{5\} \wedge p_S \in \{\text{tensor}\}$

$C(p = \textbf{ksize}) = p_T \in \{\text{int}\} \wedge p_S \in \{\text{list}\} \wedge p_{SP} \in \{[1],[3],[5]\}$

$C(p = \textbf{strides}) = p_T \in \{\text{int}\} \wedge p_S \in \{\text{list}\} \wedge p_{SP} \in \{[1],[3],[5]\}$

$C(p = \textbf{padding}) = p_T \in \{\text{string}\} \wedge p_0 \in \{\text{VALID,SAME}\}$

```
tf.nn.max_pool3d (
input = tf.constant([[[[[1.0]]]]])
ksize = [0]
strides = [1]
padding = "VALID" )
```

TensorFlow `max_pool3d` bug detected by DocTer

- DocTer extracts **5,908** constraints for **911** TensorFlow APIs from documents.
- DocTer generates valid inputs following the constraints and triggers a **floating point exception** in `tf.nn.max_pool3d`.
- This bug is **confirmed and fixed by TensorFlow developers.**

floating point exception in tf.nn.avg_pool3d and tf.nn.max_pool3dwhen ksize=0 #46834

Closed DNXie opened this issue on Feb 1, 2021 · 3 comments · Fixed by #46838

```
+  bool non_negative = std::all_of(ksize_.begin(), ksize_.end(),
+                                  [](int k) {return k > 0; });
+  OP_REQUIRES(context, non_negative,
+              errors::InvalidArgument("Sliding window ksize field must "
+                                      "have non-negative dimensions"));
```

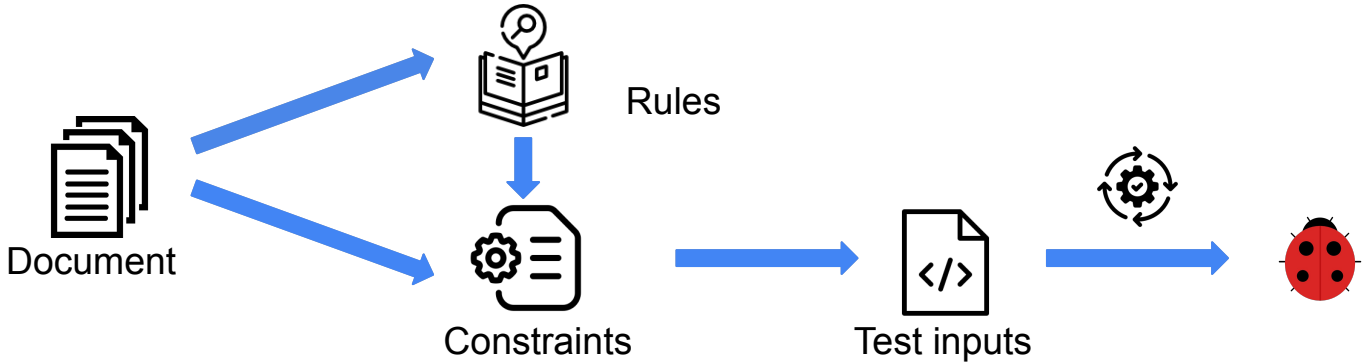
Our contributions

- A novel **rule-construction technique** to build rules that map syntactic pattern to parameter constraints.
- A novel **fuzz-testing tool DocTer** that extracts **4 categories** of constraints from API documentation to guide the input generation for testing.

- An evaluate on 3 most popular DL libraries: TensorFlow, PyTorch, & MXNet.
  - DocTer extracts **16,035 constraints** for **2,415 APIs** from documents with **85.4% precision**.
  - DocTer detects **94 bugs** in **174 APIs**, including **63 unknown bugs**, and one previously unknown **CVE security vulnerability**.

- We release the source code of DocTer, the dataset, and the bug list.

Overview



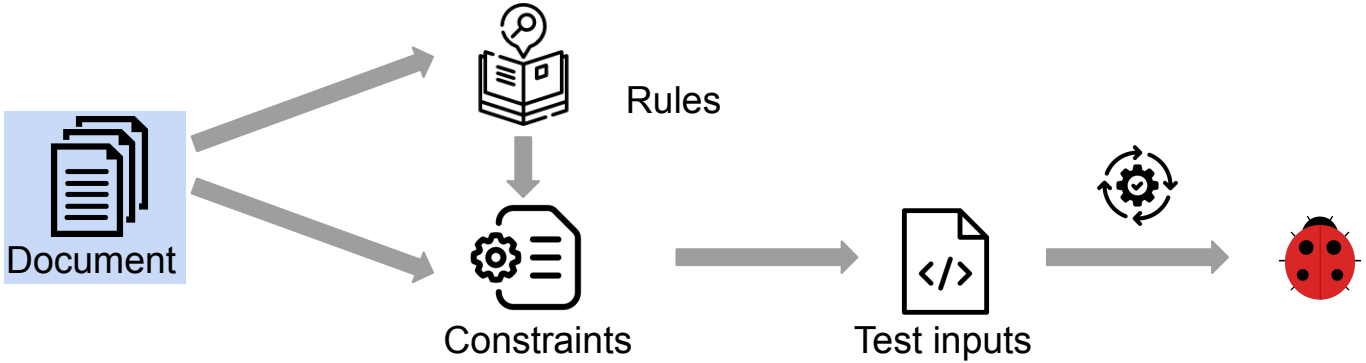

```
value: A 4-D `Tensor` of type `float`
```

```
data type: float
structure: tensor
dimension: 4
```

```
value:
 tensor([[[[1]]]],dtype=float)
...
```

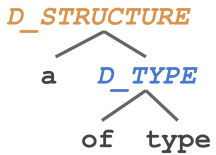
```
tf.nn.atrous_conv2d(…)
```

Approach - Preprocess

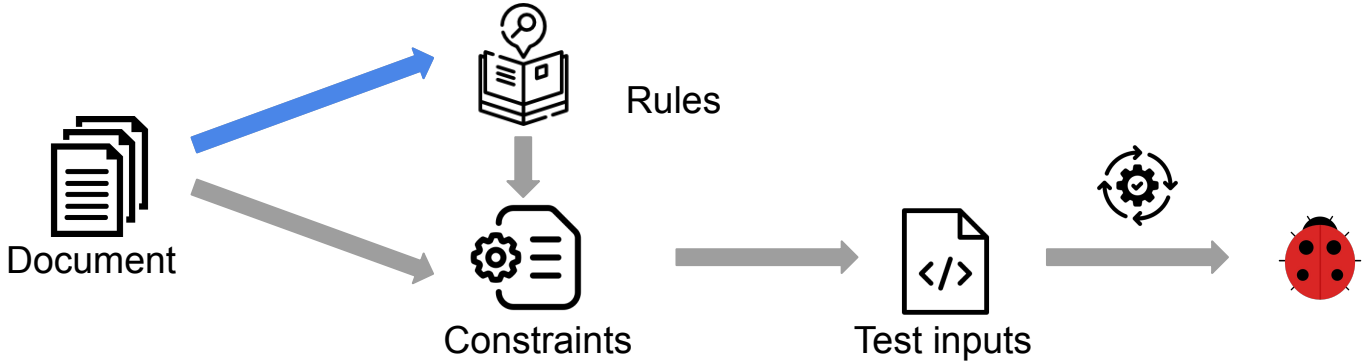



```
max_filter:
      A `Tensor` of type `float32`
```

```
A D_STRUCTURE of type D_TYPE
```



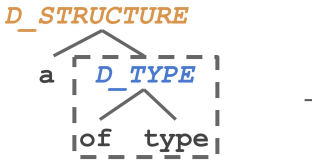
Approach - Rule Construction




```
max_filter:
      A `Tensor` of type `float32`
```

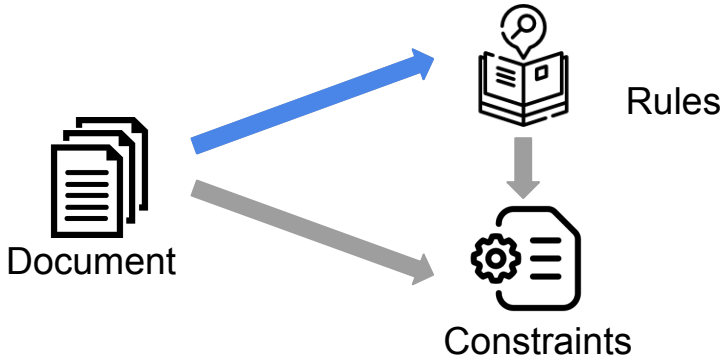
A D_STRUCTURE of type D_TYPE

1. structure: D_STRUCTURE
2. data type: D_TYPE



Subtree	Constraint
D_TYPE of type	data type: D_TYPE

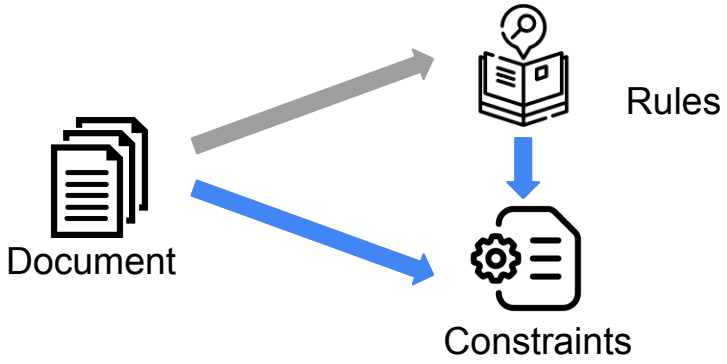
Approach - Rule Construction



Rules

	Subtree	Constraint
1	D_TYPE of type	data type: D_TYPE
2	D_STRUCTURE CONSTAINT_NUM d	structure: D_STRUCTRUE
3	D_STRUCTURE CONSTAINT_NUM d	dimension: CONSTANT_NUM
...	...	...

Approach - Constraints Extraction

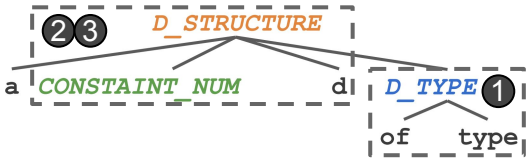


Rules

	Subtree	Constraint
1	D_TYPE of type	data type: D_TYPE
2	D_STRUCTURE CONSTAINT_NUM d	structure: D_STRUCTRUE
3	D_STRUCTURE CONSTAINT_NUM d	dimension: CONSTANT_NUM
...	...	...

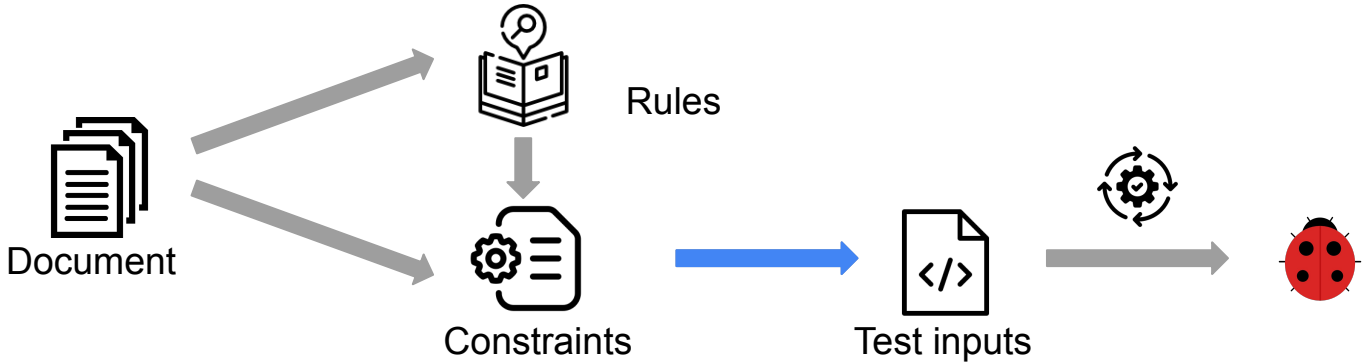
```
value:
      A 4-D `Tensor` of type `float`
```

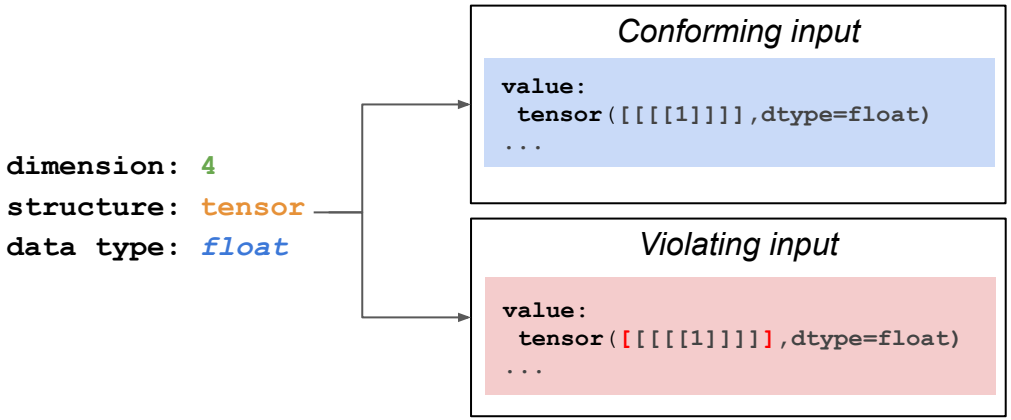
```
A <CONSTANT_NUM>-D D_STRUCTURE of type D_TYPE
```



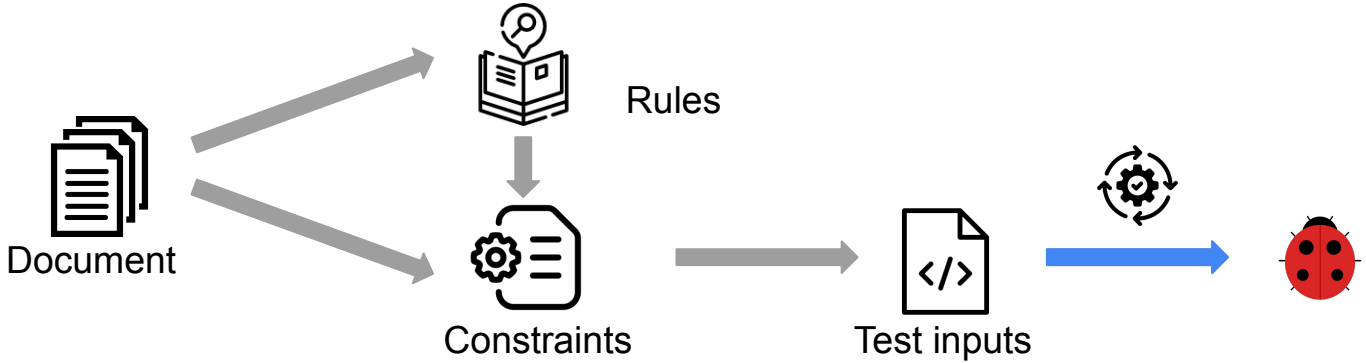

```
dimension: <CONSTANT_NUM>          dimension: 4
structure: <D_STRUCCTURE>  ──►     structure: tensor
data type: <D_TYPE>                data type: float
```

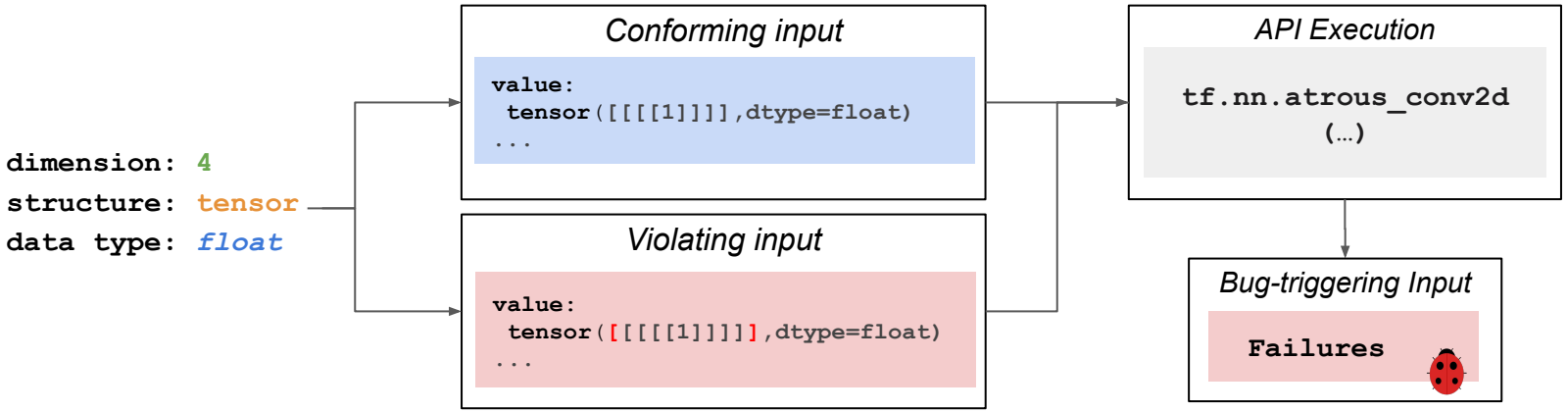
Approach - Test Inputs Generation





Approach - Testing





Experimental settings

3 libraries 

2,558 DL library APIs

2,000 inputs per APIs

0.14 seconds on average per input

Constraint extraction results

2,415 out of 2,558 APIs with constraints extracted

1,338 rules automatically constructed

16,035 constraints extracted in total

6.6 constraints extracted per API

603 parameters manually evaluated

85.4% precision achieved

Experimental design

- **Baseline**: **unguided** input generation tool

- **DocTer**: **constraint-guided** input generation tool, where constraints are extracted from API documents

How many bugs does DocTer detect

	TensorFlow	PyTorch	MXNet	Total
Baseline	41	7	11	59
DocTer	61	18	15	**94**

From **174** APIs,
63 of them are new,
54 of **63** are confirmed or verified.

How many bugs does DocTer detect

	TensorFlow	PyTorch	MXNet	Total
Baseline	41	7	11	59
DocTer	61	18	15	**94**

We only consider Segmentation Fault, Floating Point Exception, Abortion, and Bus Errors.

How many bugs does DocTer detect

2,000 inputs

2,000 inputs

1,000 inputs

1,000 inputs

	TensorFlow	PyTorch	MXNet	Total
Baseline	41	7	11	59
DocTer	61	18	15	**94**
DocTer - CI	47	14	14	75
DocTer - VI	51	18	12	81

CI: Conforming input
VI: Violating input

A new PyTorch bug detected by DocTer

high priority

- DocTer generates valid inputs following the constraints and triggers a floating point exception in `torch.nn.functional.avg_pool2d`
- This bug is **confirmed and fixed by PyTorch developers after we reported it.**

```
torch.nn.functional.avg_pool2d(input, kernel_size, stride=None, padding=0,
ceil_mode=False, count_include_pad=True, divisor_override=None) → Tensor
```

```
> torch.nn.functional.avg_pool2d(
            input = torch.tensor(np.random.rand(10, 3, 20, 20)),
            kernel_size = 4,
            stride = 0)
```

tensor

a single integer or a tuple

Fix:

```
+  TORCH_CHECK(stride != 0, "stride should not be zero");
```

Conclusions

- We propose **DocTer** to automatically extract constraints from DL API documentation to guide the input generation for testing.
- We evaluate DocTer on three of the most popular DL libraries: TensorFlow, PyTorch, and MXNet.
- DocTer extracts **16,035 constraints** automatically from **2,415 API** documentation with **a precision of 85.4%**.
- DocTer detects **94 bugs** in **174 APIs**, including **63 unknown bugs**, and one previously unknown **CVE security vulnerability**.
- We make the source code of DocTer, the dataset, and the bug list publicly available.