Developer Study on Reproducibility of Deep Learning Bugs

Hello there!

Welcome to this survey! We are a group of researchers from Dalhousie University, Canada. Recently, we conducted an empirical study involving 85 reproducible bugs from Stack Overflow posts. Our aim was to understand two main aspects: (1) the edit actions that can be employed to complete code snippets for bug reproduction and (2) the information that enhances the reproducibility of bug reports. Our investigation has yielded several interesting findings, and we are seeking your feedback on them. We will share our findings with you once the study is completed.

* Indicates required question

We reproduced 85 bugs and found out that they could be reproduced using 10 edit actions. To enhance their reproducibility, there are 5 main information categories that need to be present. The edit actions and information categories are described below.

Edit Actions

1. Input Data Generation: Generating input data which simulates the data used for training the model.
2. Neural Network Construction: Reconstructing or modifying the neural network based on the information provided
3. Hyperparameter Initialization: Initializing the hyperparameters for training, such as batch size and number of epochs
4. Import Addition and Dependency Resolution: Determining the dependencies in the code snippet and adding the missing imports.
5. Logging: Adding appropriate logging statements to capture relevant information during reproduction
6. Obsolete Parameter Removal: Removing outdated parameters or functions to match the parameters of the latest library versions
7. Compiler Error Resolution: Debugging and resolving compiler errors that arise due to the errors in the provided code snippet.
8. Dataset Procurement: Acquiring the datasets and using them to train the model
9. Downloading Models & Tokenizers: Fetching pre-trained models and tokenizers from external sources.
10. Version Migration: Updating the code to adapt the changes introduced in newer library or framework versions.

Information Categories:

Data: Shape of the input data, type of data, data distribution.

Model: Neural network architecture, number of layers, neurons, activation function for layers.

Hyperparameters: Batch size, epochs, optimizers, loss function.

Code Snippet: Training code snippet, evaluation script, data preprocessing, and transformation operations.

Logs: Compiler error logs, training error logs

Demographics

As part of this developer study, we will require certain demographic information from you to facilitate further analysis.

1. Q1: What is your relevant work experience with deep learning? *

   Mark only one oval.

   - ◯ <1 Year
   - ◯ 1-5 Years
   - ◯ 5-10 Years
   - ◯ >10 Years

2. Q2: What is your relevant experience with deep learning bug fixing?

   Mark only one oval.

   - ◯ <1 Year
   - ◯ 1-5 Years
   - ◯ 5-10 Years
   - ◯ >10 Years

3. Q3: What is your current occupation?

Mark only one oval.

- ◯ Software Practitioner (Software Engineer, Deep Learning Engineer, Machine Learning Engineer etc.)
- ◯ Researcher (Masters/Doctoral Student, PostDoc, Faculty)

4. Q4: What are the deep learning frameworks you have worked with?

Tick all that apply.

- ☐ Tensorflow
- ☐ PyTorch
- ☐ Keras
- ☐ Other: _____

5. Q5: What challenges are associated with reproducing deep learning bugs in your day-to-day activities?

_____

_____

_____

_____

Bug #1

Given the issue description, and the code snippet. Please reproduce the bug, select the most appropriate edit operations and critical information needed to reproduce this bug.

To help the reproduction process, we have provided the sample edit operations here.

Original Issue Report: https://stackoverflow.com/questions/69549126/tensorflow-typeerror-cannot-convert-1e-12-to-eagertensor-of-dtype-int32

Description:
I have a multiclass classification machine learning application for which I want to calculate the f1 score using tensorflow. The predicted and actual values are stored in pandas dataframes y_pred and y_act respectively. Both are populated with 1's and 0's. So I do something like this (Code Snippet provided below):
However I get the following error

TypeError: Cannot convert 1e-12 to EagerTensor of dtype int32

There must be something with the type casting from pandas to tensorflow which is throwing the error. I have tried a series of mitigations to no avail.

I tried converting the numpy arrays to tensors like so: pred_tf = tf.convert_to_tensor(pred_numpy, numpy.int32)

I tried ensuring the pandas dataframe has no 1e-12 instances with: y_pred = y_pred.replace(1e-12, 0)

I tried converting to numpy without the numpy.int32 option.

However I still get the same error. Any tips for converting from pandas to tensors successfully without getting this error?

Code Snippet: You can use this Colab notebook as the base notebook to start the reproduction process: https://colab.research.google.com/drive/14yuLcsJ6vg0ECahKeUcQ1BuORi3kkJsZ?usp=sharing

Hint: Focus on the statement "Both are populated with 1's and 0's.".

6. Q1: What are the edit operations that could be used to reproduce this bug? *

Tick all that apply.

- [ ] Input Data Generation
- [ ] Neural Network Construction
- [ ] Hyperparameter Initialization
- [ ] Import Addition and Dependency Resolution
- [ ] Logging
- [ ] Obsolete Parameter Removal
- [ ] Compiler Error Resolution
- [ ] Dataset Procurement
- [ ] Downloading Models and Tokenizers
- [ ] Version Migration
- [ ] Other: _____

7. Why do you think these edit operations could prove useful in reproducing the bug? *

8. Q2: What are the critical information components that could help the reproducibility of this bug? *

Tick all that apply.

- [ ] Data
- [ ] Hyperparameters
- [ ] Model
- [ ] Code Snippet
- [ ] Logs
- [ ] Other: _____

9. How do you think the selected critical information could be useful in reproducing the bug? *

10. Q3: Did you implement any additional operations or actions beyond those suggested by us? Please let us know your thoughts.

Bug #2

Given the issue description, and the code snippet. Please reproduce the bug, select the most appropriate edit operations and critical information needed to reproduce this bug.

To help the reproduction process, we have provided the sample edit operations here.

Original Issue Report: https://stackoverflow.com/questions/64576751/neural-network-typeerror-unsupported-operand-types-for-dense-and-str

Description:

I am trying to use a neural network to predict the price of houses. Here is what the top of the dataset looks like:

Price	Beds	SqFt	Built	Garage	FullBaths	HalfBaths	LotSqFt
485000	3	2336	2004	2	2.0	1.0	2178.0
430000	4	2106	2005	2	2.0	1.0	2178.0
445000	3	1410	1999	1	2.0	0.0	3049.0

...
I am using the ReLU activation function. When I try to evaluate my model on my test data, I get this TypeError: unsupported operand type(s) for +=: 'Dense' and 'str'.

I looked at the types of the columns from my original dataframe, and everything looks fine.

```
print(df.dtypes)
## Output
#Price         int64
#Beds          int64
#SqFt          int64
#Built         int64
#Garage        int64
#FullBaths     float64
#HalfBaths     float64
#LotSqFt      float64
#dtype: object
```

I'm not sure if I am messing something up in my neural network to cause this error. Any help is appreciated

Code Snippet: You can use this notebook as the base notebook to start the reproduction process: https://colab.research.google.com/drive/127nDAw8gnh9yDpZmHSFDRdDP1Mv7hX6q?usp=sharing

Hint: Focus more on the data types of the columns rather than the actual values, and try to simulate the reproduction environment.

11. Q1: What are the edit operations that could be used to reproduce this bug? *

Tick all that apply.

- [ ] Input Data Generation
- [ ] Neural Network Construction
- [ ] Hyperparameter Initialization
- [ ] Import Addition and Dependency Resolution
- [ ] Logging
- [ ] Obsolete Parameter Removal
- [ ] Compiler Error Resolution
- [ ] Dataset Procurement
- [ ] Downloading Models and Tokenizers
- [ ] Version Migration
- [ ] Other: _____

12. Why do you think these edit operations could prove useful in reproducing the bug? *

_____

_____

_____

_____

_____

13. Q2: What are the critical information components that could help the reproducibility of this bug? *

Tick all that apply.

- [ ] Data
- [ ] Hyperparameters
- [ ] Model
- [ ] Code Snippet
- [ ] Logs
- [ ] Other: _____

14. How do you think the selected critical information could be useful in reproducing the bug? *

_____

_____

_____

_____

_____

15. Q3: Did you implement any additional operations or actions beyond those suggested by us? Please let us know your thoughts.

_____

_____

_____

_____

_____