Bug Study Instrument

First please filter whether this issue contains a defect and is related to a reproducibility process. If not, move on to the next issue !!! (but remember to ***record the issue ID*** *in the spreadsheets)*

Sometimes there is a* ***one-to-many* *relationship between an issue report and the underlying defect(s). Please use multiple rows in such a case, with the same GitHub Issue ID number but* ***different details****.*

If you spend more than 10 minutes on a single question, please* ***mark the question & issue.* *We may need to discuss about that during our meetings.*

**If the comments are in another language rather than English, please use translation tools to help your understanding. Also please mark the info you got by translation, e.g. highlighting, commenting.*

1. What framework does the owner use?
    a. TensorFlow
    b. Pytorch
    c. Keras
    d. Caffe
    e. Other

2. What is the project type of the repository?
    a. Zoo
    b. Solo (Prototype)
    c. Solo (Replication)

3. What is the type of the issue reporter?
    a. Re-user:
    b. Adaptor:
    c. Enhancer:
    d. Replicator:

4. Did the work use the same data?
    a. Yes
    b. No
    c. Not mentioned

5. Did the work use the same code?
    a. Yes
    b. No

    c. Not mentioned

6. Which deep learning stage does the defect exist in?
    a. Data pipeline
    b. Modeling
    c. Training
    d. Evaluation
    e. Other
7. What are the **Impacts** of the defect? (Check all that apply)
    a. Bad Performance (lower speed)
    b. Bad Performance (lower accuracy)
    c. Bad speed – performance balance
    d. Bad data quality
    e. Numerical instability: The results are Inf, NaN or Zero which are caused by division (i.e., division by zero returns not-a-number value), logarithm (i.e., logarithm of zero returns $-\infty$ that could be transformed into not-a-number); Or the results appear random for each running.
    f. Crash: The system stops unexpectedly
    g. Data Corruption: The data is corrupted as it flows through the model and causes unexpected outputs
    h. Hang: It ceases to respond to inputs
    i. Incorrect Functionality: The system behaves in an unexpected way without any runtime or compile-time error/warning.
    j. Memory Exhaustion: The software halts due to unavailability of the memory resources. This can be caused by, either the wrong model structure or not having enough computing resources to train a particular model.
    k. Unknown
    l. Other

8. What **kind** of defect is it? (**General Code Error)**
    a. Syntax error: an error in the syntax of a sequence of characters or tokens, such that the program is not valid in the language (“It does not compile”).

    b. Algorithm/method: an error in the sequence or set of steps used to solve a particular problem or computation, including mistakes in computations, incorrect implementation of algorithms, or calls to an inappropriate function for the algorithm being implemented.

    c. Assignment/Initialization: a variable or data item that is assigned a value incorrectly or is not initialized properly or where the initialization scenario is mishandled (e.g. incorrect publish or subscribe, incorrect opening of file, etc.).

d. Checking: Inadequate checking for potential error conditions, or an inappropriate response is specified for error conditions.

e. Data Structure: Error in specifying or manipulating data items, incorrectly defined data structure, pointer or memory allocation errors, or incorrect type conversions.(i.e. Array, Linked List, Stack, Queue, Trees, Graphs)

f. External Interface: Errors in the user interface (including usability problems) or the interfaces with other systems. (e.g. API error)

g. Internal Interface: Errors in the interfaces between system components, including mismatched calling sequences and incorrect opening, reading, writing or closing of files and databases.

h. Logic: Incorrect logical conditions, including incorrect blocks, incorrect boundary conditions being applied, or incorrect expression.

i. Non-functional Defects: Includes non-compliance with standards, failure to meet non-functional requirements such as portability and performance constraints, and lack of clarity of the design or code to the reader.

j. Timing/Optimization: Errors that will cause timing or performance problems

k. Memory Exhaustion

l. Other

9. What are the **root causes** of the defect?
   a. Data Pipeline defect:
      i. Data Preprocessing Bug: If an input to the deep learning software is not properly formatted, cleaned, well before supplying it to the deep learning model.
      ii. Corrupt Data (Data Flow Bug): Due to the type or shape mismatch of input data after it has been fed to the DL model.
      iii. Training Data Quality
   b. Modeling defect:
      i. Layers
         1. Activation Function
         2. Layer Properties
         3. Missing/Redundant/Wrong Layer
      ii. Model Type & Properties
         1. Model/Weight
         2. Network structure

3. Multiple initialization

c. Training defect

i. Optimizer

ii. Loss Function

iii. Evaluation

iv. Hyperparameters

v. Other Training Process

d. API defect: Caused by APIs, this includes API mismatch, API misuse, API change, etc.

i. API – DL libraries (e.g. Pytorch, TensorFlow, Keras, CUDA, etc.)

ii. API – other data science libraries (e.g. Numpy, matplotlib, pandas, seaborn, scikit-learn, etc.)

iii. API – image IO libraries

iv. API - Other (please indicate it)

e. GPU Usage defect

f. Environment Configuration Error

g. Insufficient/Incorrect Documentation

h. Other (please indicate it below)

10. What is the **Bug Manifestation** of the work?

a. Basic errors: the code does not run (\eg it crashes, behaves very incorrectly, runs out of memory).

b. Reproducibility defects: the code using the same data runs without basic errors, but does not match the documented performance (e.g., accuracy, latency).

c. *Evolutionary defects:* the code and/or data has been changed to adapt to the user's needs. It runs without basic errors but does not match the specification/desired performance.

11. How did the engineers **fix** the bug? (Check all that apply)

a. Data Pipeline

i. Data preprocessing (before being fed into the model)

1. Data dimension: align the input data's dimension with DNN

2. Data type: change the data type of inputs to match the DNN's expectation

3. Data wrangling/cleaning: fix the form/type of the data for downstream operations without modifying its intent

4. Modify data augmentation method

5. Modify data normalization

6. Initialization modifying (i.e. categories, labels, annotations)

ii. Data Flow (after being fed into the model)

1. Modify data processing

iii. Training data quality

        1. Modify/add data augmentation
        2. Add training data
        3. Modify labeling/annotations
        4. Re-download the dataset

b. Model
    i. Network connection: change node connectivity in the DNN (e.g. change weights, remove edges, add backward propagation)
    ii. Modify layers
    iii. Layer dimension: align the input data's dimension with the layer dimension
    iv. Activation: change the activation function used in the DNN
    v. Model loading/saving/converting modification

c. Training
    i. Modify Hyperparameters (e.g. learning rate, epoch, batchsize)
    ii. Modify Loss function: add, remove or replace the loss functions
    iii. Add Monitor: add diagnostics code to monitor training
    iv. Modify Optimizer: change the optimization function used by the DNN
    v. Modify the learning rate schedular
    vi. Modify Accuracy metric: replace the accuracy metric being used to measure the correctness of a model, often to match better
    vii. Add/modify processing steps: e.g. Non-Maximum-suppression (NMS), anchors
    viii. Train from scratch
    ix. Modify training configuration

d. API
    i. API contract: fix API compositions so that the output of an API meets the preconditions of another API
    ii. Match API versions: match the versions of different API to make the program runnable, including updating, downgrading, matching.
    iii. Reinstall API
    iv. Extend/Overwrite API by with supports and other functionalities
    v. API environment configuration
    vi. Use another API

e. GPU/CPU usage
    i. Reset/Modify data parallelism on GPUs
    ii. GPU Configuration
    iii. Modify/add the conversion to CPU/GPU

f. Other
    i. Versioning: adapt the code to the new version of the library
    ii. Modify assignment/Initialization/configuration
    iii. Add/Modify checking
    iv. Read the documentation
    v. Make better documentations/tutorials
    vi. Use a stable OS

vii. Refer to another implementation

viii. Clean the cache

g. Not mentioned

12. How did the owners **confirm** their fix is correct? (Check all that apply)

a. Compare overall accuracy and loss to another implementation (including the research prototype)

b. Compare overall accuracy and loss to previous performance(e.g. after training for 100K iterations, the performance is better than it was before)

c. Compare relative improvement in accuracy and loss: use the changes in accuracy and loss between training iterations to determine correctness (e.g. the trend is improved)

d. Compare overall speed

e. Compare relative changes in learning rate

f. Replace hyper-parameters in a network

g. Examine the distribution of variable values

h. Switch the training dataset

i. Check the runnability of the program

j. Check the shape of tensors in the model

k. Use test cases

l. Check outputs

m. Check matrices

n. Check the input data

o. Not mentioned

p. Other