Technical Report - Applying Revised Machine Learning Issues Taxonomy

Tuan Dung Lai, Anj Simmons, Scott Barnett, Jean-Guy Schneider, Rajesh Vasa

I. Introduction

This technical report describes the application of the taxonomy proposed by Humbatova et al. (2020) followed by the method in the registered report Lai et al. (2022). 147 machine learning (ML) issues collected from Github were labelled against the taxonomy. Our aim is to evaluate and revise (if needed) the taxonomy by Humbatova et al. and develop a robust protocol for consistently applying the taxonomy, ensuring the reliability and replicability.

We randomly sample 30 issues from our dataset of 147 ML issues, and the first three authors independently label the issues using the taxonomy. Subsequently, we will calculate the inter-rater reliability to assess the agreement among the three raters. We will repeat the process and refine the taxonomy until we achieve a moderate level of agreement.

II. Background

We validate our ML issues dataset against an existing taxonomy of real faults in deep learning (DL) systems by Humbatova et al. (2020). The paper introduces a large taxonomy of faults in DL systems containing five top-level categories, three of which are further divided into subcategories. The authors manually analysed 1,059 artefacts gathered from GitHub commits and issues of projects that use the most popular DL frameworks (TensorFlow, Keras, and PyTorch) and from related Stack Overflow posts. They also conducted structured interviews with 20 researchers and practitioners describing the problems they have encountered in their experience, which enriched their taxonomy with a variety of additional faults that did not emerge from the other two sources. The final taxonomy was validated with a survey involving an additional set of 21 developers, confirming that almost all fault categories (13/15) were experienced by at least 50% of the survey participants. The paper uses faceted classification, i.e., it created the categories/subcategories of its taxonomy in a bottom-up way, by analysing various sources of information about DL faults.

III. Method

We manually label 147 ML issues in the ML issue dataset against an existing taxonomy Humbatova et al. (2020) using an iterative approach. We randomly sampled 30 ML issues from the ML issues dataset. Three researchers independently applied the taxonomy on the 30 issues and calculated the inter-rater reliability. We repeated the labelling process and made adjustments to the taxonomy until a moderate level of agreement was reached (Kappa score above 0.61 Landis and Koch (1977)). The inter-rater agreement of the labelling process is measured by the Light's Kappa metric Light (1971), which equals the average of all possible combinations of bivariate Kappas between raters. This metric indicates an overall index of agreement. A low Light's Kappa metric will be addressed, by refining the taxonomy and repeating the process until the agreement is substantial. After that, the first author independently labelled the rest of the ML issues dataset.

IV. Iteration 1

In the first iteration, three researchers independently labelled a random sample of 30 ML issues using the issue title, description, comments, pull requests, file changes and repository descriptions. The protocol used to label the 30 ML issues in iteration 1 is as follows:

1) Download the csv file containing 30 issues for iteration 1: Link to dataset
2) For each issue, open the issue URL and use the following attributes for the labelling process:
   a. Issue title
   b. Issue description
   c. Comments section
3) Label each issue against the lowest colored box in the taxonomy in figure 1.
   If an issue does not fit any categories in the taxonomy, use "N/A" and put a note aside, think of a new category that the issue will fit in.
   If an issue is not a bug (i.e. enhancement, feature request) use "-"

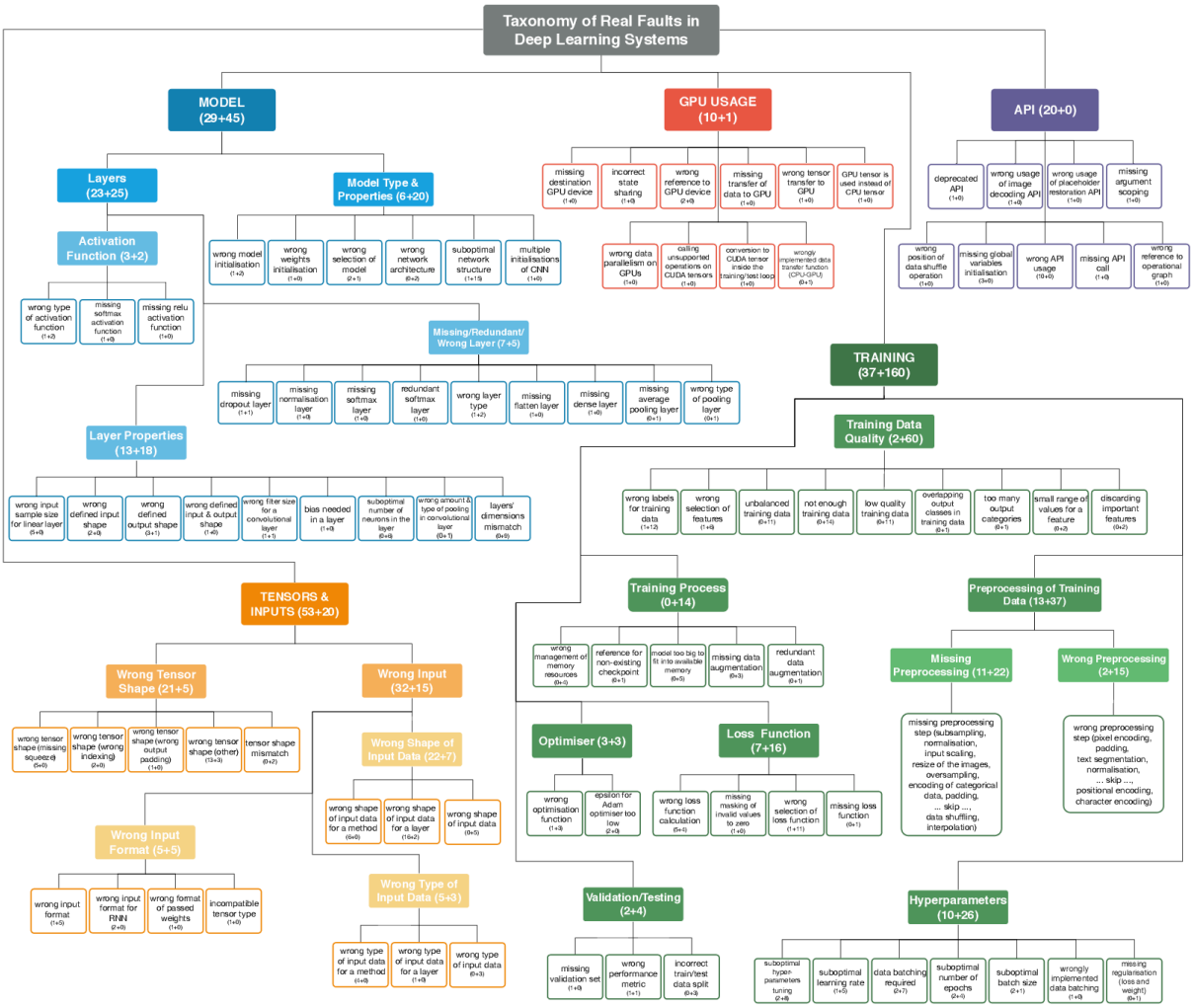

Fig. 1: Taxonomy of deep learning faults by Humbatova et al. (2020)

A bug is a flaw or error in the program that produces unexpected or incorrect results, it stops the program from being able to run end-to-end.

There are 18 categories in total as follows, the definitions of each category can be found in Section 4.1 in Humbatova et al. (2020) study.

- Model Type & Properties
- Activation Function
- Missing/ Redundant/ Wrong Layer
- Layer Properties
- GPU Usage
- API
- Wrong Tensor Shape
- Wrong Input Format
- Wrong Shape of Input Data
- Wrong Type of Input Data
- Training Data Quality
- Training Process
- Optimiser
- Loss Function
- Validation/ Testing
- Missing Preprocessing
- Wrong Preprocessing
- Hyperparameter

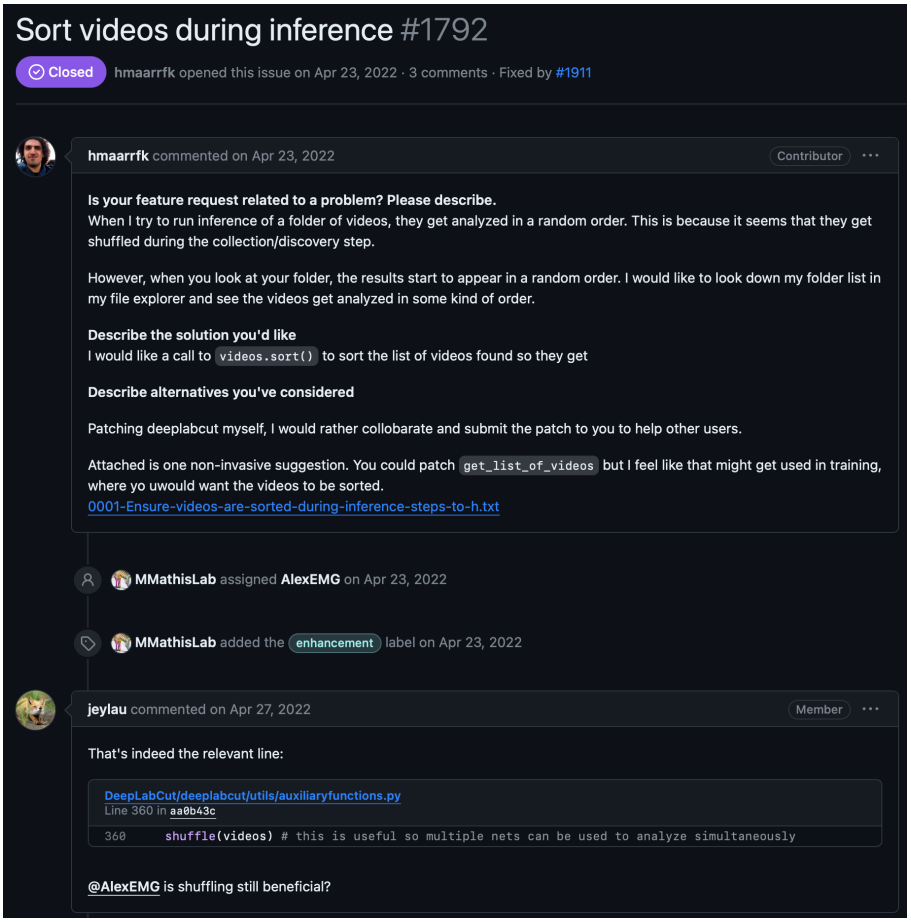

Fig. 2: Non-critical issue in the form of enhancement, issue #1792 from DeepLabCut/DeepLabCut project

This resulted in no agreement using the inter-rater reliability test Landis and Koch (1977) (Kappa score = .15). After that, we discussed each issue and came to a consensus, we modified the definitions of the taxonomy to avoid ambiguity in the second iteration, merging all subcategories in the "Tensor and Inputs" categories because they are hard to distinguish. Figure 2 shows the issue #1792 from DeepLabCut/DeepLabCut project which indicates an enhancement feature request, i.e. the ability to sort the input data, in this case, the sorting of videos at inference time. This issue did not cause the program to stop running, the original taxonomy also does not have a category for issues related to inference time. To mitigate this problem, we modify the definition of the categories to include enhancement and non-critical issues. This applies to issue #21 in brendanhasz/probflow project. Figure 3 shows issue #135 from qubvel/efficientnet project, the issue is related to incorrect hyperparameter leading to incorrect drop connection rate. This issue can be labelled as one of the following sub categories: Training - Training Process, Training - Loss Function, Model - Layer - Layer property, or Training - Hyperparameter category. Another example is issue #84 from NRCan/geo-deep-learning, certain max pooling property is not supported, however, the issue can fit into both Model - Layer - Layer Properties or Model - Model Type & Properties categories. To mitigate this issue, first, we established a labelling protocol where the categories are listed in an ordered list and the first category that is a match will be chosen. Secondly, we modified the definitions of the Model category so that it only contains issues related to the layer, activation functions and model initialization, architecture. The Tensor and Input category only contains issues related to data.

Figure 4 shows the issue #37 from the onnx/onnxmltools project, this issue is related to a third-party library called Keras. The original taxonomy from Humbatova et al. do not have a category to external libraries issues. The closest category we thought was a match is API category, however, the definition of the API category is problems arising from the framework's API usage, for example, the developers using an API in a way that does not conform to the logic set out by developers of the framework. According to the definition, only issue belongs to the API of the framework itself is included, however, our selections of projects includes applied AI projects which use other framework's API, therefore, we extend the definition of the API category to third-party usage: Inappropriate usage of third-party programs or libraries, such as TensorFlow, PyTorch, Keras.

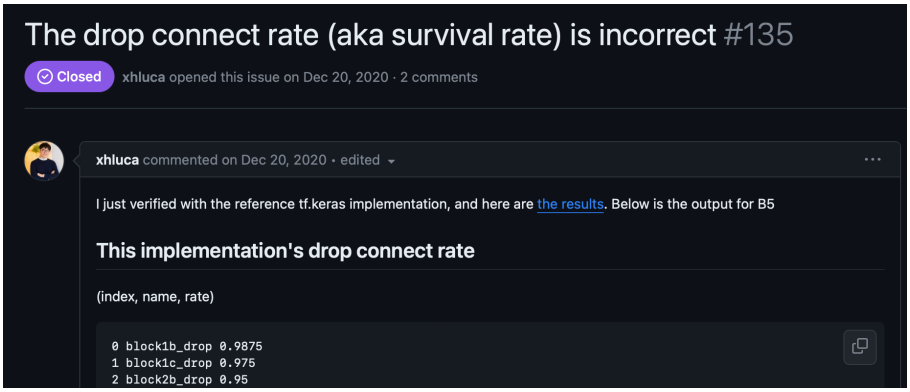

Fig. 3: An issue that suits multiple categories: Training Process, Loss Function, Layer property, Hyperparameter

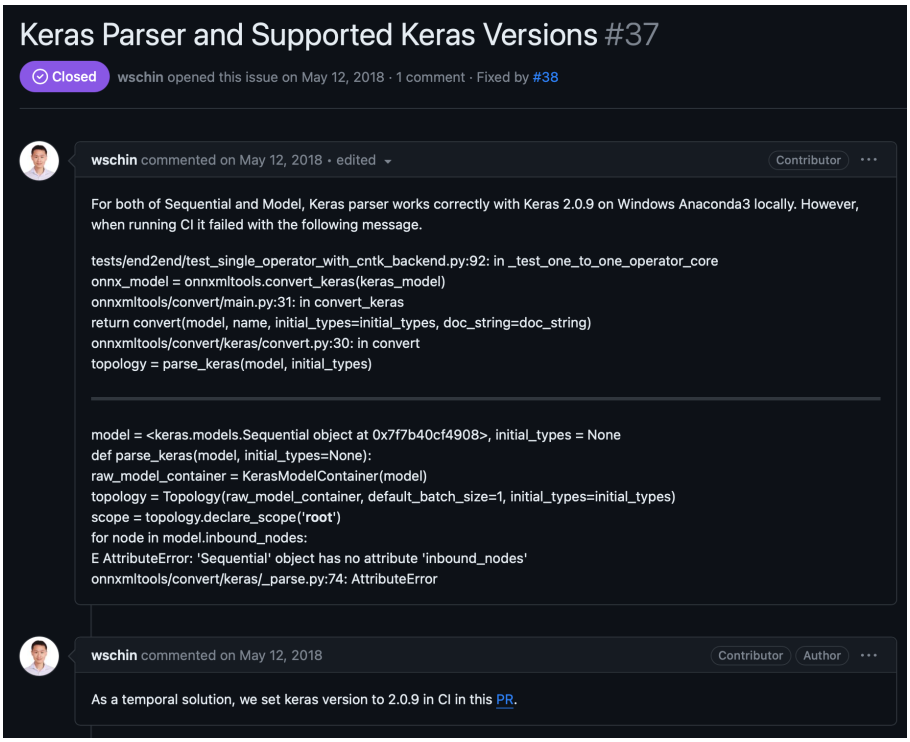

Fig. 4: An issue related to an ML framework called Keras, classified as third-party usage category.

Here is the change log to the taxonomy after the first iteration:

- Change API to Third-party usage. If the repositories is a library, API issues can be related to issues when calling a function, this can cause overlapping with other categories, for this reason, we change the group to “Third-party usage”, this only include inappropriate usage of third-party programs or libraries, such as TensorFlow, PyTorch, Keras.
- Merge everything under Tensor & Input into Tensor & Input: Avoid confusion, as the categories underneath Tensor & Inputs can be overlapping.
- Merge Training - Preprocessing of Training Data - Missing Preprocessing and Training - Preprocessing of Training Data - Wrong Preprocessing: This is for generalisation
- Model - Model Type & Properties: Extend this to include inappropriate model initialisation, for example, loading the .pth file when it’s not needed.
- Generalise the definition of each category so that it include issues that does not cause the program to stop. For example, in appropriate choice of processing steps that cause the program to runs slower and less efficient.

Categories	Definitions
Model - Layer - Activation Function	Inappropriate activation function relate to the use of such functions such as ReLu, Sigmoi, Tanh for a neural network layer.
Model - Layer - Layer Properties	Incorrect or inappropriate properties for a neural network layer, such as number of neurons, input/output shape.
Model - Layer - Missing/ Redundant/ Wrong Layer	Adding, removing or changing neural network layers inappropriately.
Model - Model Type & Properties	Inappropriate model type or model properties, such as the number of layers, architecture. Inappropriate model initilisation.
GPU Usage	Error caused by incorrect or inefficient usage of GPUs, wrong reference to GPU device, failed parallelism, incorrect state sharing between subprocesses, faulty transfer of data to a GPU device.
Third-party Usage	Inappropriate usage of third-party programs or libraries, such as TensorFlow, PyTorch, Keras.
Training - Training Data Quality	Low quality or insufficient training data, such as noisy data, imbalanced data, insufficient data.
Training - Training Process	Inappropriate or inefficient training processes, such as inappropriate batch sizes, learning rates.
Training - Optimiser	Inappropriate optimiser or not optimising the neural network model efficiently.
Training - Loss Function	Inappropriate choice loss function, or not using it efficiently during training.
Training - Validation/ Testing	Incorrect or insufficient validation/testing procedures or not evaluating the model correctly.
Training - Preprocessing of Training Data	Incorrect or missing preprocessing of training data, such as scaling, normalisation, feature engineering, etc.
Training - Hyperparameters	Inappropriate or suboptimal hyperparameters, such as learning rate, dropout rate, number of epochs, etc.
Tensors & Inputs	Incorrect shapes of input, wrong dimensions, size, inappropriate file type, encoding, selection of data format.

TABLE I: Definitions of the 14 ML issue categories used in iteration 2

V. Iteration 2

After the first iteration, we ran the labelling process again on another set of 30 random sampled ML issues, in which 2 issues were presented in iteration 1.

Here is the protocol for the labelling process for iteration 2:

1) Download the csv file containing 30 issues for iteration 2: Link to dataset
2) Label each issue using the issue title, issue description, comments, code change, pull request, and project description to label. The 14 categories and their definitions used for iteration 2 are shown in Table I.
   If an issue does not belong to any of the categories above, use N/A.
   Enhancement and non-critical issues (issues that do not crash the program) can be labelled using the categories above (e.g. inappropriate use of data format).
   If an issue does not belong to any of the categories above, use N/A.
   If it's related to software issues (e.g. function needs new parameter) which are not related to third-party libraries, fit them in one of the 14 categories, if not appropriate, label them as N/A.

Iteration 2 resulted in a weak agreement (Kappa score = .43). Figure 5 shows the issue #43 from ProGamerGov/neural-style-pt project, the issue is related to a missing parameter and can be categorised in both Training - Training Process or Training - Optimiser category. The subcategory in Training. Figure 6 shows issue #19 from brendan-hasz/probflow project, the issue is related to the ability to change the learning rate while the training process is running, this issue can be categorised to either Training - Training Process or Training - Hyperparameter. This problem repeated itself in issue #2144 from ludwig-ai/ludwig project as shown in Figure 7. To mitigate this problem, we decided to merge the subcategories in the parents node and change the definition of the Training category to Training Process to ensure the issues are all related to the Process and not the preprocessing of data. The data-related issues which were originally belonged to the Training category were merged into Tensor and Inputs category.

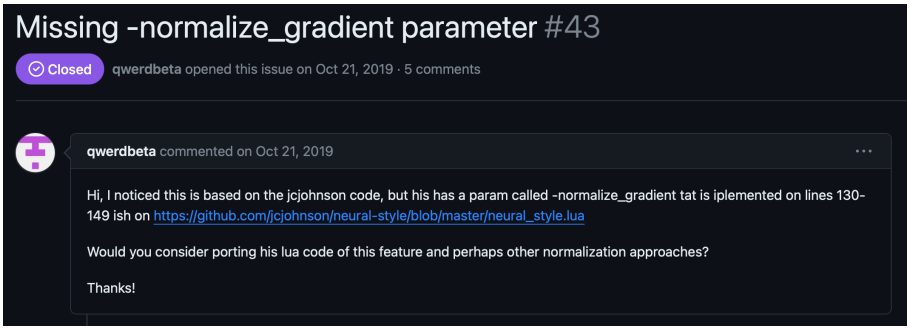

Fig. 5: Training process related issue

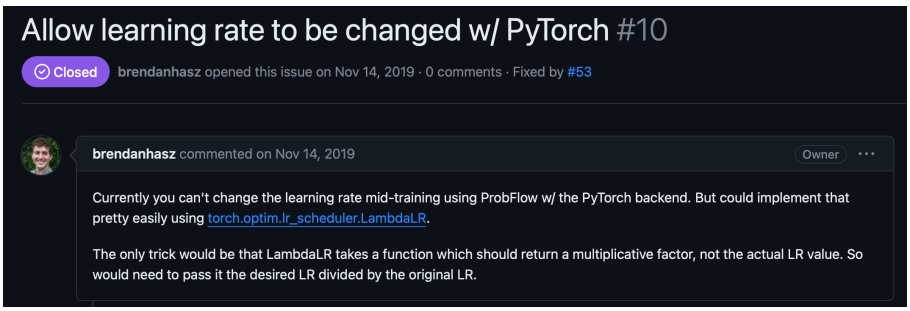

Fig. 6: Training process and hyperparameter related issue

VI. Iteration 3

The first two labelling iterations resulted in low Kappa scores which is evidence that taxonomy is inadequate to be used for real bugs in applied AI projects. In Humbatova et al. (2020) study, the authors show the creation process of the taxonomy, but do not show how the taxonomy is validated or applied in practice. For those reasons, we narrowed down the number of categories to 6 and also added a protocol for the labelling process:

The protocol for labelling 30 ML issues in iteration 3 is as follow:

- Download the csv file containing 30 issues for iteration 3: Link to download
- Open the Issue URL, read the issue title, issue description, comments, code change, PR, project description to label.
  Go through the list of 6 categories in Table III from top to bottom and pick the first one you think it's a match.
  Enhancement and non-critical issues (issues that do not crash the program) can be labelled using the categories above (e.g. inappropriate use of data format).
  Code refactoring can be in the first 5 categories, if not, label as "Other".

The third iteration of the labelling process was conducted on another set of 30 randomly sampled ML issues (all the 30 issues are excluded from the first 2 iterations), this iteration resulted in a moderate agreement among the three raters (Kappa = .67).' The new definition of the 6 categories is described in Table III.

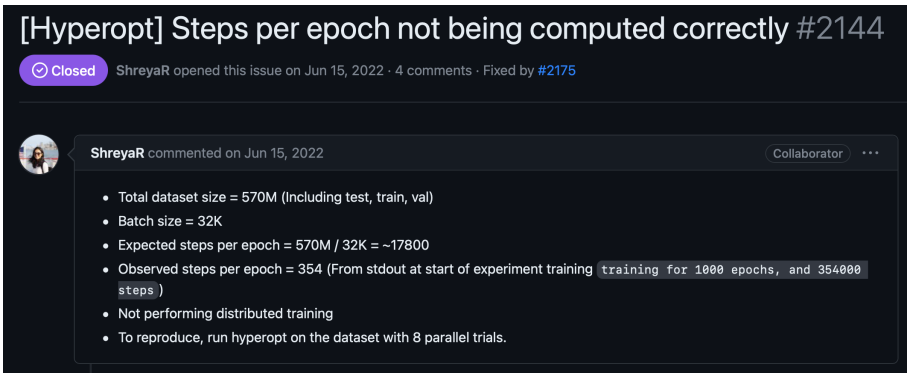

Fig. 7: Training process and hyperparameter related issue

	Iteration 1	Iteration 2	Iteration 3
Number of categories	19	15	6
Number of issues	30	30	30
Kappa score	.15 (No agreement)	.43 (Weak)	.67 (Moderate)

TABLE II: Inter-rater reliability between 3 raters when labelling ML issues

Categories	Definitions
GPU Usage	Incorrect or inefficient usage of GPUs, wrong reference to GPU device, failed parallelism, incorrect state sharing between subprocesses, faulty transfer of data to a GPU device.
Model	Inappropriate, inefficient or incorrect model initialisation, choice of architecture. Inappropriate use of activation function, incorrect properties for a neural network layer, missing, redundant or wrong layer. Errors occur during inference.
Tensor and Input	Error or inefficiencies in data quality such as low-quality data, noisy data, imbalanced data, and insufficient data. Inappropriate preprocessing of data such as scaling, normalisation, and feature engineering. Incorrect shapes of input, wrong dimensions, size, inappropriate file type, encoding, and selection of data format.
Training Process	Inappropriate or inefficient training processes excluding data-related problems, such as inappropriate batch sizes, and learning rates. Hyperparameter issues such as learning rate, dropout rate, and number of epochs. Inappropriate optimiser, inappropriate choice of loss function when using during training. Inefficient or incorrect validation/ testing procedure.
Third-party Usage	Inappropriate usage of third-party programs or libraries, such as TensorFlow, PyTorch, Keras, and Numpy.
Other	Documentation issues or anything unrelated to the 5 categories above.

TABLE III: Extended definitions of 6 ML issue categories for iteration 3

Compared to the original taxonomy, we have changed “API” to “Third-party usage” because some issues belong to other libraries, we generalised it to cover more cases. Additionally, data quality issues such as wrong data format and preprocessing occurring in the training process can either be labelled in the “Training” or “Tensor and Inputs” category, for this reason, we pull data-related categories “Training data quality” and “Preprocessing of training data” out of their original parent category “Training” and add them to “Tensor and Input”. For clarity, we changed “Training” to “Training Process” to emphasise that the issues must happen in the process of training.

VII. Results

Table II shows the Kappa score from each labelling iteration. After two iterations, we were able to achieve a moderate level of agreement among the three raters (Kappa = .67).

VIII. Taxonomy

Table III shows the extended definitions of the revised taxonomy.

IX. Conclusion

In conclusion, we were able to establish a robust protocol to apply the revised taxonomy. The revised taxonomy with the new definitions can be used to reliably label ML issues from open-source applied AI projects on Github. The revised taxonomy has 6 categories, documentation, enhancements and non-critical issues can be classified using the taxonomy. Our labelling protocol indicates that the first category from the list matches the issues, the higher level category will be more favoured than the one below it. Based on the purpose of the research, a Multi-label protocol can be used to eliminate this issue.

REFERENCES

Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. 2020. Taxonomy of real faults in deep learning systems. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 1110–1121.

Tuan Dung Lai, Anj Simmons, Scott Barnett, Jean-Guy Schneider, and Rajesh Vasa. 2022. Comparative analysis of real bugs in open-source Machine Learning projects–A Registered Report. *arXiv preprint arXiv:2209.09932* (2022).

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

Richard J Light. 1971. Measures of response agreement for qualitative data: some generalizations and alternatives. *Psychological bulletin* 76, 5 (1971), 365.

APPENDIX

Here are the definitions of the categories used in the labelling process iteration 1. These definitions are taken directly from section 4.1 in Humbatova et al. (2020) study:

Model: This category of the taxonomy covers faults related to the structure and properties of a Deep Learning (DL) model.

Model Type & Properties: This category considers faults affecting the model as a whole, rather than its individual aspects/components. One such fault is a wrong selection of the model type, for example, when a recurrent network was used instead of a convolutional network for a task that required the latter. In addition, there are several cases of incorrect initialisation of a model, which result in the instability of the gradients. Another common pitfall from this category is using too few or too many layers, causing suboptimal network structure, which in turn leads to poor performance of the model. An example was provided by one of our interviewees: ”when we started, we were thinking that we needed at least four layers in the encoder and the decoder and then we ended up having half of them, like actually very shallow model and it was even better than the bigger deeper model”.

Layers: Faults in this category affect a particular layer of a neural network. This is a large taxonomy category that was further divided into the three inner subcategories described below:

Missing/Redundant/Wrong Layer: These faults represent cases where adding, removing or changing the type of a specific layer was needed to remedy the low accuracy of a network. This is different from the suboptimal network structure of Model Type & Properties category, as here the solution is local to a specific layer, rather than affecting the whole model. An interviewee described such a fault, which was related ”not to the wrong architecture as whole, but more usually to the wrong type of layer, because usually in our field people have applied type of layers which were not suited for the type of input which they are processing”.

Layer Properties: This category represents faults due to some layer’s incorrect inner properties, such as its input/output shape, input sample size, number of neurons in it. As per interviewee’s description, ”we set too large number of neurons and we had like very slow training and validation”.

Activation Function: Another important aspect of a neural network is the activation function of neurons. If not selected properly, it can dramatically ruin the model’s performance. One interviewee noted that ”when I changed sigmoid activations into linear activations in the speech recognition, it gave me a gain”.

Tensors & Inputs: This category deals with problems related to the wrong shape, type or format of the data. We encountered two different classes of faults in this category:

Wrong Tensor Shape: A faulty behaviour manifests during some operation on tensors with incompatible shapes or on a single tensor with incorrectly defined shape. There is a number of possible causes for a wrong tensor shape, e.g., missing output padding, missing indexing, or, as it was provided in one of the interviews, a case when a developer ”was using a transposed version of the tensor instead of the normal one”.

Wrong Input: A faulty behaviour is due to data with incompatible format, type or shape being used as an input to a layer or a method. A wrong input to a method is a problem frequently observed in traditional software, as well as in DL programming. However, in DL, these faults happen to be of a specific nature, ranging from the input having unexpected datatype (e.g., string instead of float) or shape (a tensor of size 5x5 instead of 5x10) to cases when the input has a completely wrong format (e.g., a wrong data structure). One interesting example of a wrong input format was provided by our interviewee: ”my data was being loaded in with channel access first instead of last. So that actually was a silent bug and it was running and I actually don’t understand how it even ran but it did”.

Training: This is the largest category in the taxonomy and it includes a wide range of issues related to all facets of the training process, such as the quality and preprocessing of training data, tuning of hyperparameters, the choice of appropriate loss/optimisation function. It also accounts for the faults occurring when testing/validating a previously trained model.

Hyperparameters: Developers face a large number of problems when tuning the hyperparameters of a DL model. The most reported incorrect hyperparameters are learning rate, databatch size, and number of epochs. While suboptimal values for these parameters do not necessarily lead to a crash or an error, they can affect the training time and the overall performance achieved by the model. An example from the interviews is "when changing learning rate from 1 or 2 orders of magnitude, we have found that it impacts the performance of about up to 10% to 15% in terms of accuracy".

Loss Function: This category contains faults associated with the loss function, specifically its selection and calculation. Wrong selection of the loss function or usage of a predefined loss function may not adequately represent the optimisation goals that a model is expected to achieve. In its turn, a wrong calculation of a loss function occurs when a custom loss function is implemented and some error in the implementation leads to the suboptimal or faulty behaviour. As one interviewee noted, they needed "to get a more balanced loss function than just something that can predict one class very well and then screws up the other ones".

Validation/Testing: It includes problems related to testing and validating a trained model, such as the bad choice of performance metrics or faulty split of data into training and testing datasets.

Preprocessing of Training Data: Preprocessing of a training dataset is a labour-intensive process that significantly affects the performance of a DL system. This is reflected in the large number of elements in this category and in the high variety and number of its leaves. At the high-level we have separated the faults in this category into two groups: missing preprocessing and wrong pre-processing. The former refers to cases when a preprocessing step that would lead to a better performance has not been applied at all. In the latter case, the preprocessing step has actually been applied, but either it was of an unsuitable type or was applied in an incorrect way. Examples of the most frequent issues are missing normalisation step, missing input scaling or subsampling, and wrong pixel encoding. It is important to remark that preprocessing steps for training data are heavily dependent on an area of application. This explains the large variety of leaf tags in this category. We had to omit some of them from the taxonomy figure, due to the lack of space.

Optimiser: This category is related to the selection of an unsuitable optimisation function for model training. Wrong selection of the optimiser (e.g., Adam optimiser instead of stochastic gradient descent) or suboptimal tuning of its parameters (too low epsilon for Adam optimiser) can ruin the performance of a model.

Training Data Quality: In this group fall all the aspects relevant to the quality of training data. In general, issues occur due to the complexity of the data and the need for manual effort to ensure a high quality of training data (e.g., to label and clean the data, to remove the outliers). More specific cases of data collection challenges include privacy issues in the medical field and constantly changing user interfaces of web pages, from which the data is gathered automatically. All of this leads to the most frequent issue in this category, which is not enough training data. A variant of this problem is unbalanced training data, where one or more classes in a dataset are underrepresented. Moreover, to get a good classification model, it is important to ensure the provision of correct labels for training data. However, in the interviewees' experience getting wrong labels for training data is a common and an annoying issue. The set of other issues related to the quality of training data, such as the lack of a standard format, missing pieces of data or the presence of unrelated data (e.g., images from other domains) are gathered together under a rather general tag low quality of training data, because specific issues depend on the area of application.

Training Process: This category represents the faults developers face during the process of model training, such as wrong management of memory resources or missing data augmentation. It also contains leaves representing the exploitation of models that are too big to be fitted into available memory or reference to non-existing checkpoints during model restoration. Regarding the data augmentation, one of the interviewees noted that it helped "to make the data more realistic to work better in low light environments", while the other said that sometimes "you add more pictures to dataset" and as a result you can face "the overfitting of the network problem, so sometimes data augmentation can help, sometimes it can damage".

GPU Usage: This top-level category gathers all kinds of faults related to the usage of GPU devices while working with DL. There is no further division in this case as all the examples we found represent very specific issues. Some highlights from this category are: wrong reference to GPU device, failed parallelism, incorrect state sharing between subprocesses, faulty transfer of data to a GPU device.

API: This part of the taxonomy represents a broad category of problems arising from framework's API usage. The most frequent is wrong API usage, which means that a developer is using an API in a way that does not conform to the logic set out by developers of the framework. Another illustrating example could be a missing or wrongly positioned API call.