

DLシステムにおける実際の欠陥の分類法の構築

Taxonomy of Real Faults in Deep Learning Systems

Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota,
Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. 2020.
In Proceedings of the ACM/IEEE 42nd
International Conference on Software Engineering (ICSE '20).
Association for Computing Machinery, New York, NY, USA, 1110–1121.
DOI:<https://doi.org/10.1145/3377811.3380395>

背景

- 深層学習(DL)は様々な分野で利用されている
 - 音声→テキスト, 翻訳
 - クレジットカードの不正利用検知, 病気の診断と治療, 自動車の自動走行等
- セーフティクリティカルなシステムのDLネットワークの依存が高まっている
 - 上記のシステムで発生する欠陥の種類が重要なテーマになっている
- DLシステムは従来ソフトウェアと比べて複雑
 - コードにバグがなくとも, 学習段階で導入された欠陥により期待される動作から逸脱する可能性がある
 - 学習パラメータのミス
 - バランスの取れてない/代表性のない学習データセットの使用

関連研究

○ TensorFlowを使用して開発されたDLアプリケーションを研究^[27]

- SOとGithubから175件のTensorFlow関連のバグに関する情報を収集
- 開発者が直面する課題や障害を検出して特定するために使用する戦略を明らかに
- 7種類の一般的な根本原因と4種類の症状に分類

○ 関連研究との違い

- 扱うフレームワークが違う (本研究はTensorFlow, PyTorch, Keras)
- プロジェクト数が違う (先行研究は11, 本研究は564)
- DLに特化したバグが多い (先行研究は計175件, 本研究はDL特化のみで375件)

[27] Yuhao Zhang, Yifan Chen, Shing-Chi Cheung, Yingfei Xiong, and Lu Zhang. 2018.

An Empirical Study on TensorFlow Program Bugs.

In Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2018).

DLシステムを構築する開発者の支援

→ DLシステムにおける実際の欠陥の分類法を構築する

- よくある落とし穴を回避できる
- 特定の欠陥タイプに対応するテストシナリオを定義する動機付けとなるチェックリストとして使用できる

研究の流れ

○ 分析データの収集

- Github, StackOverflow

○ 構造化されていない情報源の手動分析

- 問題の背後にある根本的な原因を特定することが目的

○ インタビュー

- 20人の研究者や開発者を対象にDLフレームワークの使用に関する経験を収集
- 著者全員でオープンコーディングを行い,
インタビュー対象者が言及した欠陥のカテゴリーを特定

○ 分類法の検証 (アンケート)

- 21名の研究者・実務家を対象に最終的な分類法を検証

分析データセット収集 | Github

○ Github APIを用いて分析データを収集

- 「tensorflow」「keras」「torch」を含むpythonファイルを所持しているリポジトリ
- 貢献者が5人未満、公開issueがない、コミット数100以下、スターフォーク数10未満を削除
- 上記の検索でヒットしたが関連するフレームワークを使用していないものを著者が目視で除去

○ プロジェクトごとに各情報を収集

- MLに関連するissue, PR → bug, fault, errorタグがあるもの
- コミット → リポジトリの変更ログを調査し, [13]を含むメッセージを持つもの

IssueやPRが多いプロジェクトをサンプルとして中身を見ると
DLフレームワークの使用に無関係なissue/PR/commitであった
→ 更なるクリーニングを行う必要がある

[13] Michael Fischer, Martin Pinzger, and Harald C. Gall.
2003. Populating a Release History Database from Version Control and Bug Tracking Systems.
In 19th International Conference on Software Maintenance (ICSM 2003).

分析データセット収集 | Github

○ DLに関連する単語リストを作成

- 上記のデータから単語の辞書(12k語)を作成し, 出現回数が10回以下を削除
- 著者らが目視で検査を行い, DLの関連語リストを作成
(複製パッケージ[7]に含まれており, layer, train, tensorなど105単語)

最終的にコミット 1,981件, issue/PR 1,392件

分析データセット収集 | StackOverflow

○ StackExchange Data Explorer [8] を使用

- 投稿のタグに各フレームワークの名前がある
 - 「how」「install」「build」という単語が含まれている投稿をフィルタリング
 - 解決策が確定している投稿
- Tensorflow 9,935 件, Keras 3,116 件, PyTorch 1,007 件

閲覧数が多い上位各1,000件を分析対象とする

[8] <https://data.stackexchange.com/stackoverflow/query/new>

手動ラベリング

○ 収集したデータに対し手作業でラベリングを行う

- オープンコーディングの手順に従って著者全員で行う
- 2人で意見が分かれた時は、3人目を導入
- それでも意見が分かれた時は、全員で議論を行う

○ 厳密な手順で行なった

- DLシステムではなくフレームワーク自体に問題があるものを誤検出とした
- 修正対象がDLシステムであっても不具合自体が一般的なプログラミングエラーなら一般的であるというラベルを付与
- DLシステムの問題であっても評価者が問題の根本原因を追跡できない場合は不明瞭とした
- ドキュメントを見る際、特定の箇所だけではなく、関連する全てのものを見ている

インタビュー

○ 参加者20名

- 参加者のDL経験に関する詳細な情報は[7]

○ 半構造化インタビュー

- 自由形式の質問 → 予想外の情報を引き出すため
- 具体的な質問 → インタビューの範囲内にとどめ、具体的な質問をする際に役立つ
- 具体的な質問も[7]

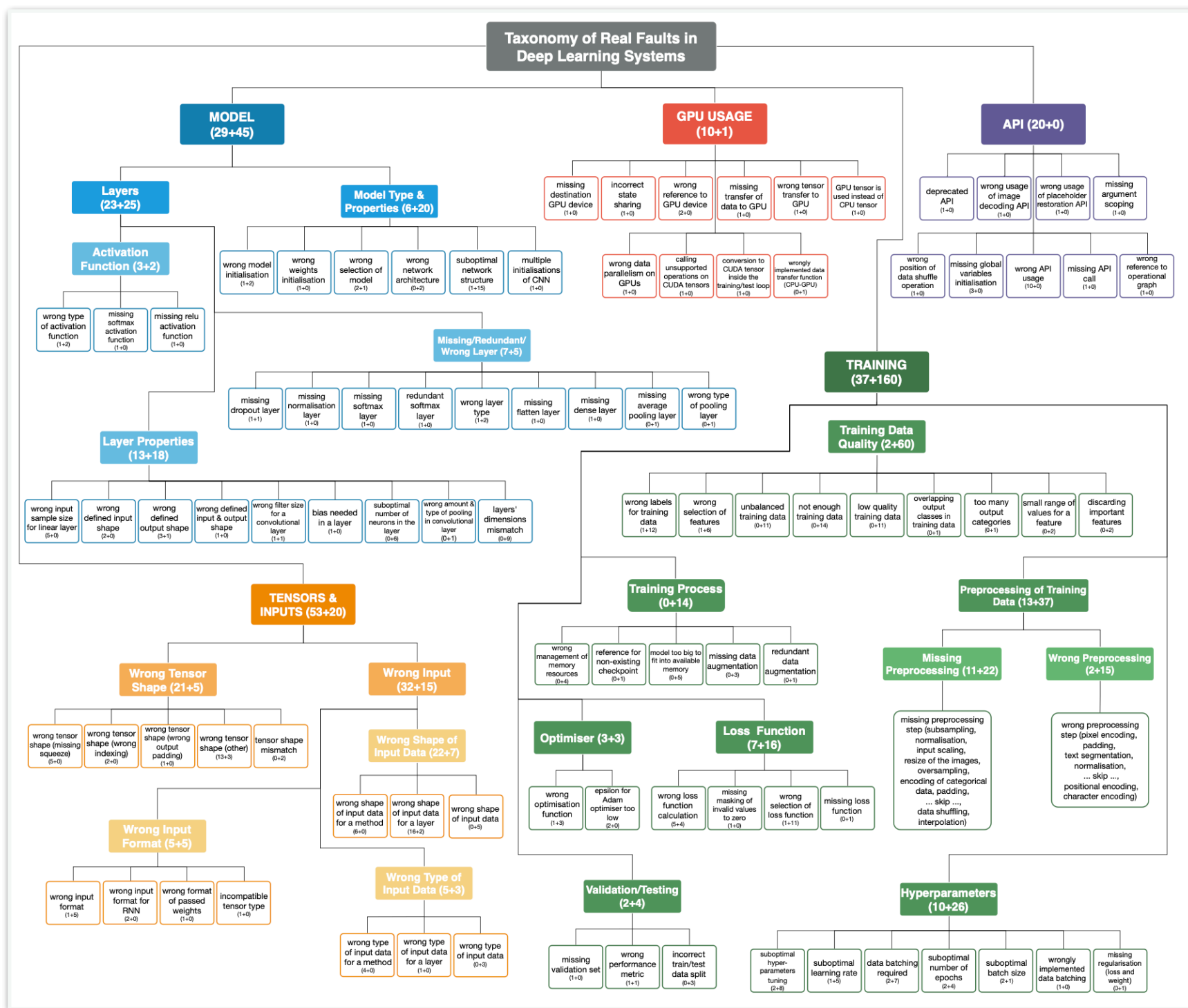
[7] <https://github.com/dlfaults/taxonomy>

○ インタビューで聞き取りした事案に対してラベル付け

- モデレータ1名と評価者2名
- 最終的に226個のタグが分類法に使用

最終的な欠陥分類

左の数字：手動ラベリングで割り当てられた数
右の数字：インタビューで該当した数



MODEL (29+45)

モデル全体に影響を与える欠陥

TRAINING (37+160)

トレーニングプロセスのあらゆる側面に
関連する幅広い問題

- トレーニングデータの品質や前処理
- ハイパーパラメータの調整
- 適切な損失/最適化関数の選択

API (20+0)

フレームワークのAPIの使用に起因する問題
- 最も多いのが間違ったAPIの使用

TENSORS & INPUTS (53+20)

互換性のないフォーマット、タイプ、または
形状のデータがレイヤーやメソッドの入力と
して使用することで生じる不具合

GPU USAGE (10+1)

DL作業中のGPUデバイスの使用に関連する
あらゆる種類の不具合

- 類似概念に対応するタグをカテゴリ分類
- 親カテゴリを作成し、is-a関係になるように

それぞれの情報源から上位5件を比較する

○ Github + StackOverflow

- 間違ったテンソルの形状 (21+5)
- レイヤーの入力データの形状の違い (16+2)
- 前処理の欠落 (11+22)
- 間違ったAPIの使用 (10+0)
- メソッドの入力データの形状の違い (6+0)

○ インタビュー

- 前処理の欠落 (11+22)
- 最適でないネットワーク構造(1+15)
- 間違った前処理(2+15)
- 十分なトレーニングデータがない(0+14)
- トレーニングデータの間違ったラベル(1+12)

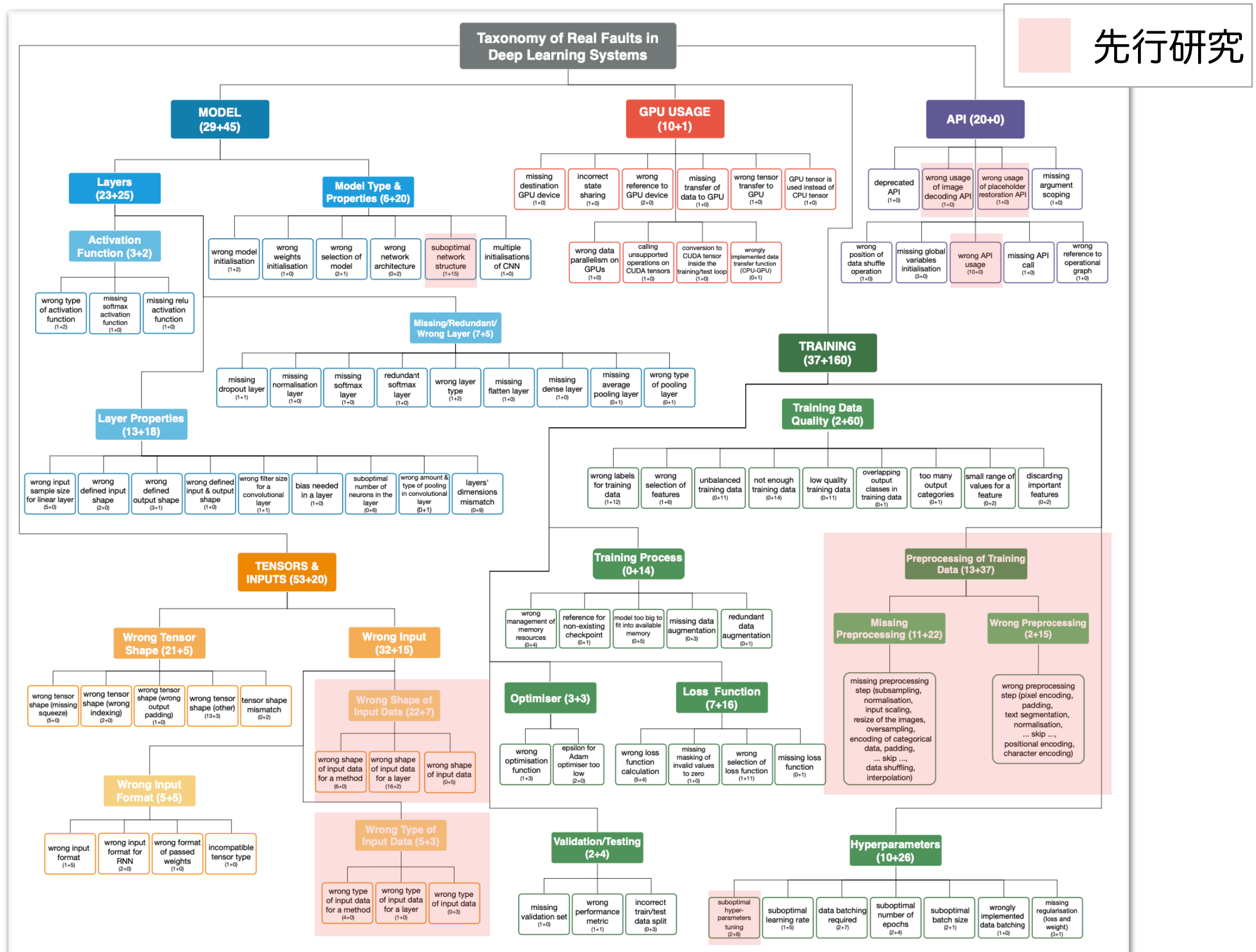
選択された情報源が
補完的であることを示している

各不具合カテゴリーの重大性の認識と カテゴリーの不具合を特定して修正するために必要な労力の認識

Category	Response		Severity			Effort Required		
	Yes	No	Minor	Major	Critical	Low	Medium	High
Hyperparameters	86%	14%	44%	44%	11%	22%	33%	44%
Loss Function	65%	35%	15%	54%	31%	23%	46%	31%
Validation & Testing	60%	40%	33%	42%	25%	50%	17%	33%
Preprocessing of Training Data	86%	14%	56%	17%	28%	28%	39%	33%
Optimiser	57%	43%	75%	25%	0%	58%	33%	8%
Training Data	95%	5%	6%	33%	61%	6%	17%	78%
Training Process	68%	32%	31%	23%	46%	31%	15%	54%
Model Type & Properties	81%	19%	44%	44%	13%	38%	44%	19%
Missing/Redundant/Wrong Layer	24%	76%	60%	40%	0%	60%	40%	0%
Layer Properties	76%	24%	44%	44%	13%	63%	31%	6%
Activation Function	43%	57%	33%	67%	0%	67%	22%	11%
Wrong Input	62%	38%	62%	31%	8%	69%	31%	0%
Wrong Tensor Shape	67%	32%	57%	21%	21%	71%	29%	0%
GPU Usage	52%	48%	55%	18%	27%	55%	18%	27%
API	67%	33%	43%	29%	29%	36%	43%	21%

- 分類表の中で参加者が一度も遭遇したことがない不具合のカテゴリーはない
- 特にTraining Dataに必要な労力を払っている

考察 | 先行研究との比較



まとめ

- DLシステムを構築する開発者の支援するために
DLシステムにおける実際の欠陥の分類法を構築する
- Github + StackOverflow とインタビューの2種類情報源から
欠陥分類を作成
- 上記の2つの情報源は補完的であるため,
提案した欠陥分類は包括的に欠陥を網羅できていると言える

所感

- 手作業多くてめっちゃ大変そう
- インタビューの形式等も細かく書かれており，妥当性が損なわれないように最新の注意がなされているなと感じた
- githubやstackoverflowから集めた情報とインタビューから収集した生の声が両方あるので勉強する時等にも使えそうだなと感じた