Deep Learning Model Reengineering: An Exploratory Case Study on Computer Vision

PURDUE UNIVERSITY
Elmore Family School of Electrical and Computer Engineering

Wenxin Jiang, Advisor: James C. Davis, With Help from the Purdue-Loyola TFMG Team

Motivation

Re-engineering (reproducing**, **adapting**,* and ***re-using) a deep learning model is challenging for many reasons:

- Mismatches between the needs of research and practice
- Constraints and requirements of new contexts.
- Cost of implementation and testing.
- Engineers involved have different specializations.

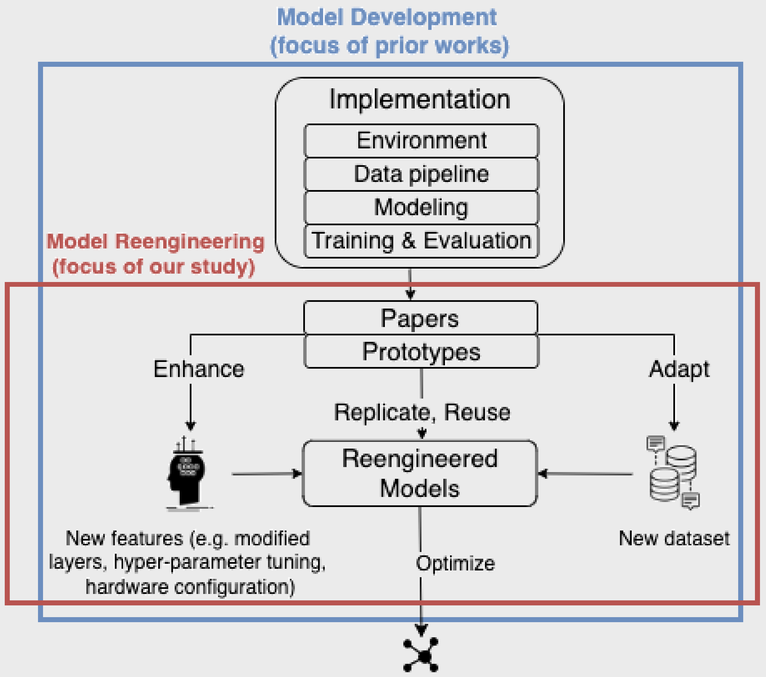

Figure 1. High-level overview of a DL model development and application life cycle.

Research Question

1. How do defects **manifest** across the reengineering process?
2. What **type** of defects are more frequent?
3. What are the **impacts** of defects?
4. What are the **root causes** of defects?
5. What are the **challenges** and **practices**?

Acknowledgement

This work was supported by a gift from Google and by NSF award #2107230.

Methods

1. ***Bug Study**: 8 Zoo repos + 14 Solo repos*
   - *Bug instrument*: Adapt existing taxonomy.
   - *Repo selection*: SOTA Computer vision, # Stars >= 1k.
   - *Bug selection*: Closed, # comments >= 10.
2. ***Case Study**:*
   - Purdue reengineering team (6 interviews).

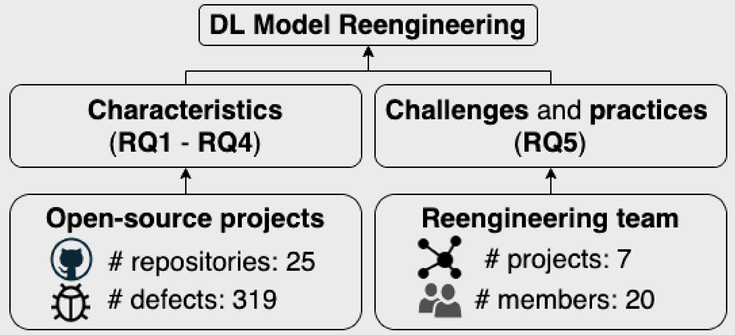

Figure 2. Relationship of research questions to methodology.

Defect manifestations

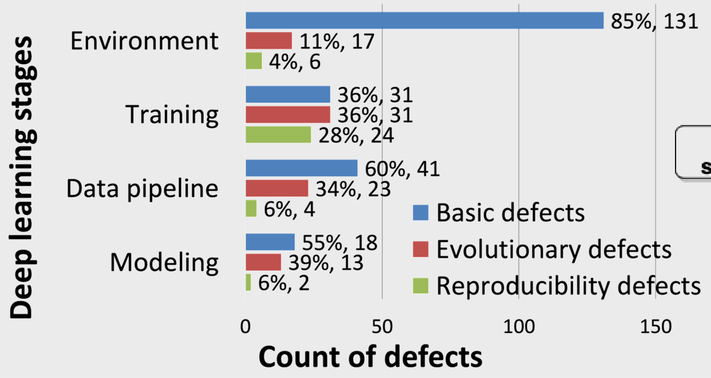

Figure 3. Manifestation in different DL stages. Most of the reproducibility defects occur during the training stage (65%).

Challenges and practices

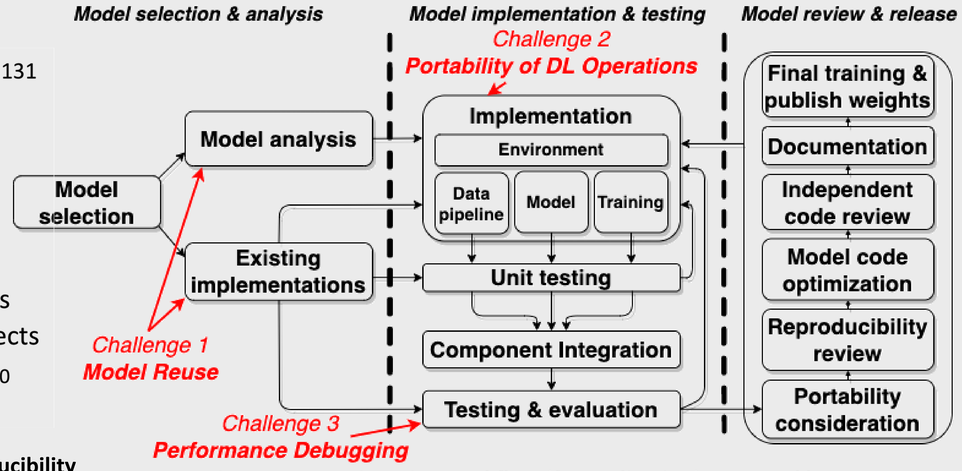

Figure 4. Reengineering workflow (for replication) proposed by our team

Impacts

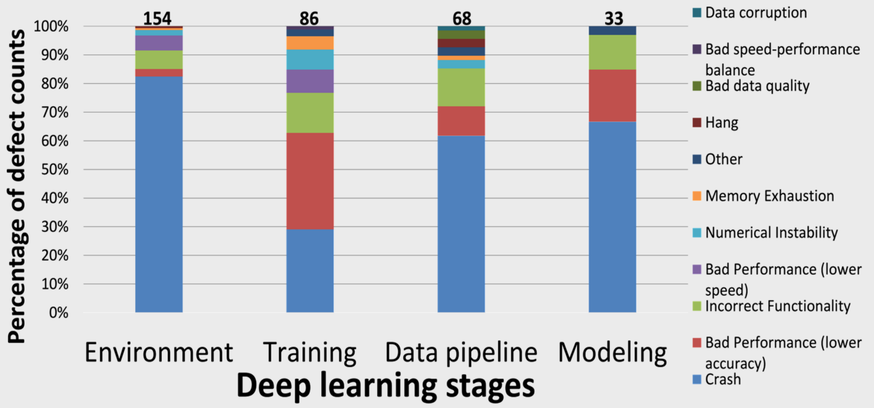

Figure 5. Defect impacts by DL stage. Across most stages, the most frequent impact is crash (62-82%). In Training, Low accuracy accounts for the largest proportion (34%) of defects.

Root causes

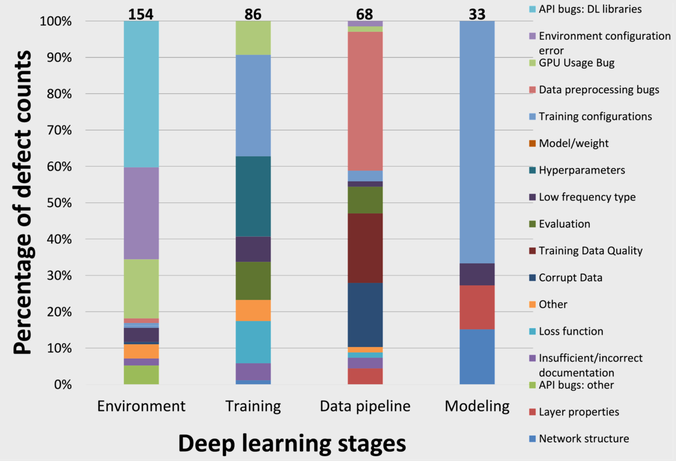

Figure 6. Root causes by DL stage. In the Data Pipeline, data preprocessing, corrupt data and data quality predominate. Most Modeling defects are due to model/weight operations and network structure. Training defects have diverse causes.

Implications: (1) Empirical justification. (2) DL Software testing. (3) Enabling model reuse. (4) Standardized practices (5) DL engineering teams