1 Paper Overview

"Hidden Technical Debt in Machine Learning Systems", written by D.Scully, Gary Holt, et al. from Google, is published in NeurIPS 15'. Within this paper, the technical and system maintenance concerns relating with the rapid development of machine learning system are addressed. Given the unique characteristics of ML systems, new challenges arise in terms of system development and routine issues arise, comparing with traditional software engineering. The rapid deployment, fast prototyping and heterogeneous data issues incur design and development shortages, which further cause massive ongoing maintenance costs in real world [3]. Several ML-specific risk factors in ML system design, including component entanglements, data dependencies, system feedback loops, etc., are covered. Potential consequences and possible experienced solutions are also discussed. Regarding the company background, after summarizing the technique concerns and bringing the discussion to the level of team cultures, this paper gives an experienced conclusion:

1. It is worthy and necessary to both measure and monitor the possible system evolution and also detect the latency shortages brought by previous fast development and deployment.
2. The efforts devoted to reducing the long-term system risks should be recognized and rewarded.

1.1 Problem Summary

Comparing with traditional software engineering, the special capacity of generating long term costs, i.e. technical debt, of ML system is argued in this paper, due to the natural development characteristics and external dependencies, which might trigger sever problems in future development and system maintainability. Through combining thoughts from both traditional software engineering and ML-specific consideration, this paper examines the potential long-term problems in the height of system-level interactions and interfaces of ML systems.

1.2 Related Works

- Refactoring: improving the design of existing code [1].
- Searching for build debt: Experiences managing technical debt at google [2].

2 Paper Strengths

The company background of this paper make it valuable and it no longer focuses merely on academic or merely on engineering aspect but on both. The aspect is unique and especially trustworthy from teams in Google, given their industrial and academia positions. It is a great reflection of present ML systems. Lots of potential problems, regarding routine maintenance, development flaws and intrinsic ML system attributes, are addressed in a separated way, instead of mixing with traditional software engineering. Moreover, this paper borrows several thoughts from traditional software engineering and develops those terms in the context of ML problems, including technique debt, glue code and anti-patterns. By establishing connection with traditional software developing, this paper is inspired and connected. Also, several experienced solutions or possible directions for addressed issues are given. At the end, the reflection of team cultures and present system considerations is another worthy point. In general, the strengths of this reflection paper are as follows:

1. It subjects the traditional software engineering concerns to the unique ML context and makes extensions.
2. It is interesting to see concerns of both production and development environments.
3. The team culture reflections show its unique trustworthiness, given the company background of this paper.

3 Paper Weaknesses

Several solutions for the brought-up potential problems are included in this paper. However, those solutions tends to be experienced ones, rather than some rigid or concrete ones. Also, there is no detailed metrics, no matter in measuring system changes or monitoring system deviation, leading this paper as only a generalized illustration. Also, the diversified examples mentioned in this paper are more or less separated ML systems and the possible solutions are from either academic fields or experiences, lacking of industrial aspects. Although the conclusion of this paper values the importance of team efforts devoting to technique debt and it advocates the shift of team culture, there is no detailed analysis or case study to support this conclusion. In general, the weaknesses of this reflection paper are as follows:

1. There is no available metrics given in this paper.
2. The possible solutions for the diversified examples are either experienced or academical.
3. There is few analysis or case study to support the very last appeals.

References

[1] Martin Fowler. *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 2018.

[2] J David Morgenthaler, Misha Gridnev, Raluca Sauciuc, and Sanjay Bhansali. Searching for build debt: Experiences managing technical debt at google. In *2012 Third International Workshop on Managing Technical Debt (MTD)*, pages 1–6. IEEE, 2012.

[3] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in neural information processing systems*, pages 2503–2511, 2015.