PTMTorrent: A Dataset for Mining Open-source Pre-trained Model Packages

Wenxin Jiang *§ Nicholas Synovic†§ Purvish Jajal* Taylor R. Schorlemmer* Arav Tewari*
Bhavesh Pareek* George K. Thiruvathukal† James C. Davis*
*Purdue University and †Loyola University Chicago

***Abstract*— Due to the cost of developing and training deep learning models from scratch, machine learning engineers have begun to reuse pre-trained models (PTMs) and fine-tune them for downstream tasks. PTM registries known as "model hubs" support engineers in distributing and reusing deep learning models. PTM packages include pre-trained weights, documentation, model architectures, datasets, and metadata. Mining the information in PTM packages will enable the discovery of engineering phenomena and tools to support software engineers. However, accessing this information is difficult — there are many PTM registries, and both the registries and the individual packages may have rate limiting for accessing the data.**

We present an open-source dataset, PTMTorrent, to facilitate the evaluation and understanding of PTM packages. This paper describes the creation, structure, usage, and limitations of the dataset. The dataset includes a snapshot of 5 model hubs and a total of 15,913 PTM packages. These packages are represented in a uniform data schema for cross-hub mining. We describe prior uses of this data and suggest research opportunities for mining using our dataset.

The *PTMTorrent* dataset (v1) is available at: https://app.globus.org/file-manager?origin_id=55e17a6e-9d8f-11ed-a2a2-8383522b48d9&origin_path=%2F%7E%2F.

Our dataset generation tools are available on GitHub: https://doi.org/10.5281/zenodo.7570357

***Index Terms*—Open-Source Software, Data Mining, Machine learning, Empirical software engineering**

I. Introduction

Modern software systems reuse Deep Neural Networks (DNNs) to build intelligent and adaptive systems [1, 2]. Engineering a DNN from scratch is challenging for many reasons, including the variation in deep learning libraries [3, 4] and the high expense of training models [5]. Organizations and developers can address some of these challenges and reduce the cost and effort associated with DNN development by reusing *pre-trained DNN models* (PTMs) [6, 7]. PTMs are shared via *deep learning model registries*, which are modeled on traditional software package registries such as NPM [8]. These PTM packages include reusable components, such as model architectures, weights, licenses, and other metadata. Deep learning model registries enable engineers to develop their models with re-usability in mind [9, 10]. Although PTM reuse is still in its early stages, the most popular PTMs are downloaded millions of times each month [11, 12].

As PTM reuse becomes more widespread, the engineering community will benefit from research into PTM reuse practices, challenges, and tools [11, 12]. By analogy to traditional software, mining PTM software repositories can help us understand development trends [13–15] and usage patterns [16, 17]. However, mining the software repositories associated with PTM packages is difficult for three reasons related to *data availability*. First, researchers must look in many places — PTM packages are distributed across many competing PTM registries [11]. Second, researchers must access the packages — PTMs include complex DNN models and weights with sizes over 1 TB, and access to these packages may be hindered by throttling or rate limiting [18]. Third, for scientific replicability, this large-scale data needs to be hosted long-term.

To enable mining of PTM packages, we share *PTMTorrent*, the first many-hub dataset of PTM packages. PTMTorrent contains 15,913 PTMs from 5 different PTM registries identified in our prior work [11]: Hugging Face [19], Model Zoo [20], PyTorch Hub [21], ONNX Model Zoo [22], and Modelhub [23]. Our dataset is hosted on a high-performance storage system (HPSS) maintained by Purdue University's Research Computing center. The dataset includes the metadata of each PTM and the package histories for each GitHub repository. These packages are represented in a uniform data schema for cross-hub mining. Out dataset supports many directions for further research, including studies of the PTM supply chain, PTM package evolution, PTM mining tools, and DNN architectural trends.

II. The PTMTorrent Dataset

A. Data Source

In prior work we mapped the major model hubs and indicated that there exist open, gated, and commercial hubs [11]. Open and gated hubs tend to be larger and

§Authors contributed equally.

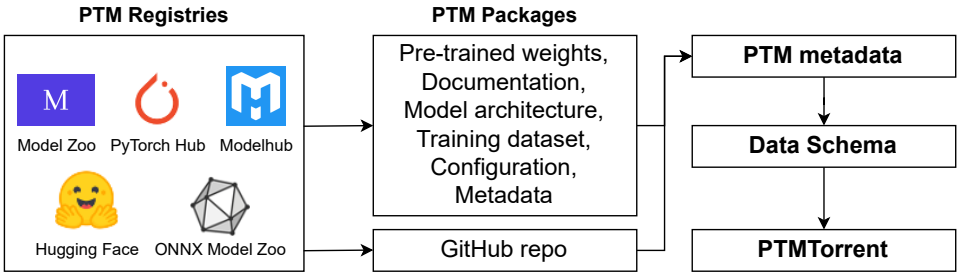

Fig. 1: Data collection and preprocessing workflow for PTMTorrent. We standardize the PTM metadata by using a data schema, collecting it from PTM packages and the corresponding GitHub repository.

more widely used because they accept contributions from anyone, and can be accessed by anyone. Commercial hubs are offered by individual companies to share vetted models with their clients. Due to the limited access to commercial model hubs, we only provide a snapshot of the open hubs (Hugging Face) and some of the gated hubs (Model Zoo, PyTorch Hub, Modelhub, and ONNX Model Zoo).

The PTMTorrent dataset contains the repository histories of 15,913 PTM packages available as of January 2023. They are provided as complete git clones, resulting in a compressed footprint of ~61TB. Each PTM package was cloned at its most recent version, including the model card, architecture, weights, and other information provided by the maintainers (*e.g.,* training configuration, hyper-parameters).

Figure 1 indicates the collection and preprocessing approaches of our dataset.

We collected PTM packages from all open and gated model hubs per Jiang *et al.* [11], excluding TensorFlow Hub because it does not support version control features. We downloaded all PTM packages from Model Zoo, PyTorch Hub, ONNX Model Zoo, and Modelhub. Due to the size of Hugging Face, we downloaded only the top 10% most-downloaded PTMs.[1] Overall, our dataset contains 15,913 packages from 5 PTM registries, distributed as described in Table I.

TABLE I: Details about the PTMTorrent content for each of the 5 model registries we collected.

Name	# Models	Data Size
Hugging Face [24]	12,401	61TB
Model Zoo [20]	3,245	115GB
PyTorch Hub [21]	49	1.5GB
ONNX Model Zoo [22]	185	441MB
Modelhub [23]	33	721MB
PTMTorrent	**15,913**	**~61TB**

[1]Although we collected a small amount of the full Hugging Face registry, this "top 10%" snapshot includes all Hugging Face PTMs with over 30 downloads.

B. Data Schema

Figure 2 shows the overview of the data schema we used to standardize the dataset. We extracted common entities into a general PTM schema. Each PTM registry has some custom features, so we customized the schema slightly for each model registry. The full data schema is encoded following the JSON Schema format,[2] and is available in the GitHub repository associated with this project.

C. Data Storage

As shown in Table I, the entire PTMTorrent dataset (v1) needs ~61TB of storage space. A cost-effective storage system is required to serve this dataset. Commercial services are cost-prohibitive at this scale, *e.g.,* we estimated a monthly cost of over $1000 to store and serve this dataset from Amazon Web Services. We opted instead for an internal resource available at Purdue University: the Purdue Fortress tape-based hierarchical storage system.[3] To facilitate external distribution of our dataset, we offer a Globus share [25] named *PTMTorrent*.

D. Maintainability and Extensibility

The sizes of PTM registries are increasing rapidly. For example, Hugging Face provided 63,182 public PTM packages on August 2022, and now it provides 124,427 packages. We believe the number of open-source PTM packages will increase in the foreseeable future. Therefore, maintainability and extensibility are two important properties of PTMTorrent.

The PTMTorrent dataset is designed to be maintainable by re-running our scripts to gather any additional changes that may have been made to the PTM registry since its last collection. Expect a biannual update.

For extensibility, new model hubs can be incorporated into the dataset. We follow an open-source model

[2]See https://json-schema.org/draft/2020-12/json-schema-core.html

[3]For more information about Fortress, see https://www.rcac.purdue.edu/knowledge/fortress/overview. Our GitHub repository includes a guide on how to access data stored in Globus.

and will review Issue and Pull Request contributions on GitHub. The PTMTorrent data schema captures most elements of a PTM package, though some specialization is needed. The downloaders for a new model hub can be developed based on the examples of the already-supported model hubs in our open-source data collection tools. An extender must provide 2-4 scripts following the pattern we used on the other hubs.

III. Originality and Relevance

Prior works have extracted information from open-source projects to a dataset and provide it for future analysis, such as GHTorrent [26], TravisTorrent [27], and RTPTorrent [28]. These datasets can be used for further mining software repository researches and help the community better understand open-source software projects [14, 15, 29, 30].

Similarly, our dataset captures the open-source PTM packages from many model hubs. The structure of our dataset imitates prior datasets that were focused on traditional open-source software [26, 28]. Compared to prior work, PTMTorrent focuses on PTM packages, including the metadata, architecture, dataset, and performance metrics. Our dataset provides a way for users to efficiently download and access large amount of data on PTM packages and relevant repositories.

IV. Usage Examples

A. Prior Usage in the Literature

In prior work, we used a part of PTMTorrent (the Hugging Face part) to measure potential risks in the Hugging Face model registry [12]. We measured the dependencies of model architecture and datasets, PTM documentation, and GPG commit signing in Hugging Face PTMs. Our analysis identified potential software supply chain concerns facing PTM reusers, including spoofing, tampering, and repudiation.

In prior work, we also used metadata from Hugging Face to measure model discrepancies and maintainers' reach [11]. Our analysis showed that existing defenses appear insufficient for ensuring the security of PTMs.

The PTMTorrent dataset provides more opportunities for mining PTM data by covering more PTM registries and providing greater structure. We believe that these large amount of PTM packages can be analyzed in similar ways as traditional packages [31–33].

B. Applying an Existing MSR Tool

Since PTMTorrent consists of git repositories, it is possible to use existing software repository mining tools on the PTM packages. Our GitHub repository includes a demonstration of this. We used our PRIME tool [34] to analyze software process metrics on a subset of the dataset.

V. Limitations

PTMTorrent is incomplete. It is biased towards the top 10% most-downloaded PTMs in HuggingFace (though this is almost all PTMs with any downloads, cf. §II-A). There are other model hubs, such as Papers With Code [35], PINTO Model Zoo [36], and Jetson Zoo [37]. Beyond these, there are other deep learning-specific registries that lack versioning or packaging features. The initial PTMTorrent release provides PTMs from model hubs that are similar to traditional software packages, as defined by Jiang *et al.* [11]. We leave their capture for future work.

Another limitation of our data is the non-standardized granularity. The current version of PTMTorrent lacks detailed metadata and does not provide uniform information, *e.g.,* datasets, model architectures. During the data collection, we notice that the provided information from PTM registries can be quite different and we use customized data schemas for each PTM registry. As a result, it is difficult to analyze all the PTM packages under the same umbrella when using our dataset.

For example, Hugging Face provides detailed documentation and structured metadata, as well as relevant configuration files for each PTM, while ONNX Model Zoo provides PTM metadata through unstructured Markdown files. Thereby making metadata extraction challenging. To mitigate this problem, we have a parent data schema for all the PTM registries and child schemas for each specific registry that represents their custom data.

VI. Future Work

In addition to the risk measurements presented by Jiang *et al.* [11, 12], the PTMTorrent dataset can be used in different ways. We suggest three research directions: PTM supply chain analysis, tools for PTM reuse, and mining tool development.

A. Supporting Future PTM Supply Chain Analysis

Prior work has focused on understanding the characteristics of package registries and their supply chains. Zimmermann *et al.* analyzed the metadata of NPM packages and identified the potential threats on downstream users [31]. Ladisa *et al.* proposed an attack taxonomy on open-source supply chains, including code contributions to package distribution [38]. Similar studies are also important in PTM supply chain alongside studies focused on PTM-specific aspects. We propose that future studies can analyze PTMTorrent dataset to understand the characteristics of the PTM supply chain, including the dependency analysis [31], vulnerabilities [39], and code knowledge transfer [40].

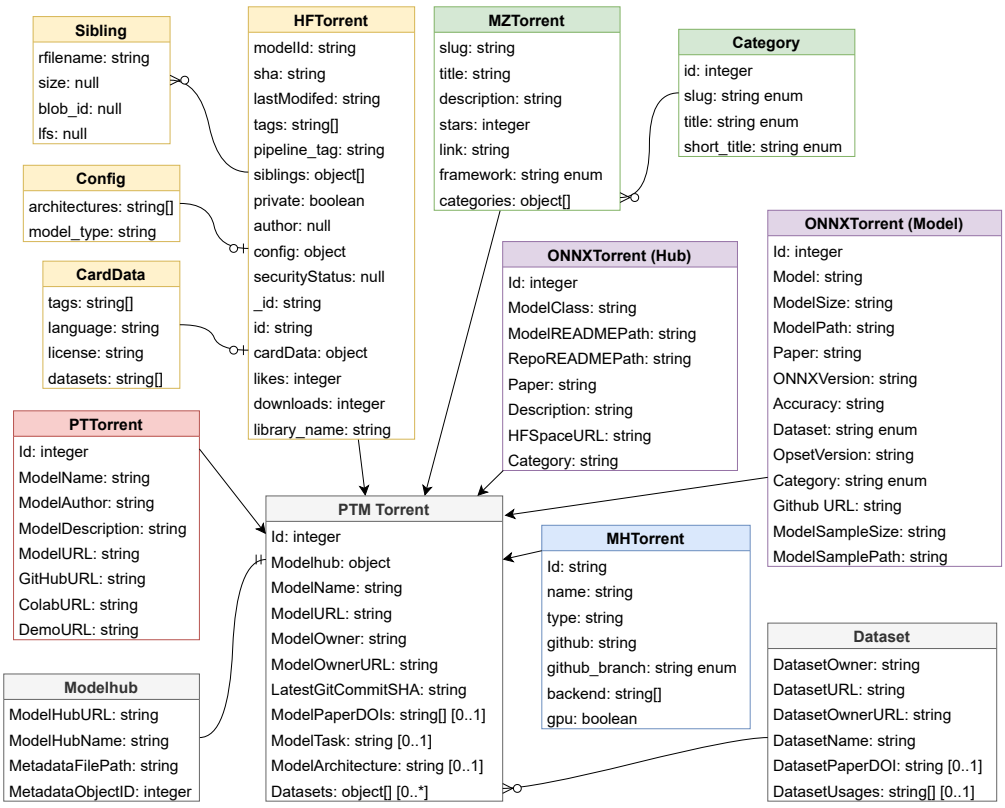

Fig. 2: An overview of PTMTorrent's data schema. Each model hub shares a general schema (*grey boxes*), with hub-specific data stored in customized schema (*colored boxes*). The full schema is available in JSON in the dataset generation repository.

Recent advances in AI, such as ChatGPT [41], that clearly build upon composing various PTMs strongly suggest that being able to study how PTMs and are composed to build more complex systems (a trait shared with traditional software) will become more important. We hope our dataset will aid in performing such analyses.

B. Expanding PTM Model Registry Analysis

Researchers can extract more information from these model registries by reusing or developing software metrics for PTM packages, including provenance, reproducibility, and portability [12]. PTM registries can help us develop comprehensive attributes and provide these details in the PTM dashboard, similar to the measured attributes from NPM [42] and PyPi [43].

Our prior study has indicated that engineers can have trouble finding the best PTM that matches their requirements, and it can therefore be hard to identify the portability and reproducibility of the open-source PTMs [12]. Montes *et al.* shows that there exist notable discrepancies among different model zoos [44]. With more detailed and comprehensive metadata provided for each PTM and the corresponding usage patterns on downstream tasks, it will be possible to develop a recommender system to help engineers find the right set of PTMs for a given application and requirements [45]. PTM Registry contributors can develop sophisticated visualization tools—with the aid of our dataset—that help PTM users understand the strengths and limitations of each model.

C. Furthering the State of Mining Tool Development

Given the lack of standardization among different PTM registries (§V), it was challenging to standardize all the metadata. PTMTorrent may not have everything needed for every type of analysis. Researchers can augment the dataset during the data collection and processing stage for other subsequent mining needs. We have included the relevant GitHub pages of each PTM in out dataset, and therefore the extraction can be done either based on the provided documentation from PTM registries [46] or source code from the underlying repositories [47].

VII. Acknowledgements

This work was supported by gifts from Google and Cisco and by NSF awards #2107230, #2229703, #2107020, and #2104319.

References

[1] S. Amershi, A. Begel, C. Bird, R. DeLine, and H. Gall, "Software Engineering for Machine Learning: A Case Study," in *International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2019.

[2] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, "A Literature Review of Using Machine Learning in Software Development Life Cycle Stages," *IEEE Access*, 2021.

[3] H. V. Pham, S. Qian, J. Wang, T. Lutellier, J. Rosenthal, L. Tan, Y. Yu, and N. Nagappan, "Problems and Opportunities in Training Deep Learning Software Systems: An Analysis of Variance," in *ASE*, 2020.

[4] V. Banna, A. Chinnakotla, Z. Yan, A. Vegesana, N. Vivek, K. Krishnappa, W. Jiang, Y.-H. Lu, G. K. Thiruvathukal, and J. C. Davis, "An experience report on machine learning reproducibility: Guidance for practitioners and tensorflow model garden contributors," 2021.

[5] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon Emissions and Large Neural Network Training," 2021. [Online]. Available: https://arxiv.org/abs/2104.10350

[6] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A Survey on Deep Transfer Learning," *IEEE Transactions on knowledge and data engineering*, 2018.

[7] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, 2010.

[8] NPM, "npm," 2022. [Online]. Available: https://www.npmjs.com/

[9] J. Gordon, "Introducing TensorFlow Hub: A Library for Reusable Machine Learning Modules in TensorFlow," 2018.

[10] T. Wolf, L. Debut, V. Sanh, and others, "Transformers: State-of-the-Art Natural Language Processing," in *Conference on Empirical Methods in Natural Language Processing*, 2020.

[11] W. Jiang, N. Synovic, R. Sethi, A. Indarapu, M. Hyatt, T. R. Schorlemmer, G. K. Thiruvathukal, and J. C. Davis, "An empirical study of artifacts and security risks in the pre-trained model supply chain," in *ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*, 2022.

[12] W. Jiang, N. Synovic, M. Hyatt, T. R. Schorlemmer, R. Sethi, Y.-H. Lu, G. K. Thiruvathukal, and J. C. Davis, "An empirical study of pre-trained model reuse in the hugging face deep learning model registry," in *ICSE*, 2023.

[13] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, "A Large Scale Study of Programming Languages and Code Quality in Github," in *IEEE International Conf. on Software Maintenance*, 2009.

[14] F. Zampetti, S. Scalabrino, R. Oliveto, and others, "How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines," in *MSR*, 2017.

[15] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *ICSE*, 2014.

[16] P. Anbalagan and M. Vouk, "On predicting the time taken to correct bug reports in open source projects," in *International Conf. on Software Maintenance*, 2009.

[17] S. Malinen, "Understanding user participation in online communities: A systematic literature review of empirical studies," *Computers in Human Behavior*, 2015.

[18] H. Face, "Hugging face documentations," 2022. [Online]. Available: https://huggingface.co/docs

[19] Hugging Face, "Hugging face – the ai community building the future." 2021. [Online]. Available: https://huggingface.co/

[20] Y. K. Jing, "Model Zoo - Deep learning code and pretrained models," 2021. [Online]. Available: https://modelzoo.co/

[21] Pytorch, "Pytorch hub," 2021. [Online]. Available: https://pytorch.org/hub/

[22] ONNX, "Onnx model zoo," 2022. [Online]. Available: https://github.com/onnx/models

[23] Computational Imaging and Bioinformatics Lab, "Modelhub," 2022. [Online]. Available: http://modelhub.ai/

[24] H. Face, "Hugging Face – The AI community building the future." 2021. [Online]. Available: https://huggingface.co/

[25] K. Chard, J. Pruyne, B. Blaiszik, and others, "Globus data publication as a service: Lowering barriers to reproducible science," in *IEEE International Conference on e-Science*, 2015.

[26] G. Gousios and D. Spinellis, "GHTorrent: Github's data from a firehose," in *MSR*, 2012.

[27] M. Beller, G. Gousios, and A. Zaidman, "TravisTorrent: Synthesizing Travis CI and GitHub for Full-Stack Research on Continuous Integration," in *MSR*, 2017.

[28] T. Mattis, P. Rein, F. Dürsch, and R. Hirschfeld, "RTPTorrent: An Open-source Dataset for Evaluating Regression Test Prioritization," in *MSR*, 2020.

[29] M. Beller, G. Gousios, and A. Zaidman, "Oops, My Tests Broke the Build: An Explorative Analysis of Travis CI with GitHub," in *MSR*, 2017.

[30] D. Elsner, F. Hauer, A. Pretschner, and S. Reimer, "Empirically Evaluating Readily Available Information for Regression Test Optimization in Continuous Integration," 2021.

[31] M. Zimmermann, C.-A. Staicu, and M. Pradel, "Small World with High Risks: A Study of Security Threats in the npm Ecosystem," in *USENIX Security Symposium*, 2019.

[32] N. Zahan, T. Zimmermann, P. Godefroid, B. Murphy, C. Maddila, and L. Williams, "What are Weak Links in the npm Supply Chain?" in *ICSE*, 2022.

[33] A. Decan, T. Mens, and E. Constantinou, "On the impact of security vulnerabilities in the npm package dependency network," in *International Conf. on Mining SW Repositories*, 2018.

[34] N. Synovic, M. Hyatt, R. Sethi, S. Thota, A. J. Miller, W. Jiang, E. S. Amobi, A. Pinderski, K. Laufer, N. J. Hayward, N. Klingensmith, J. C. Davis, and G. K. Thiruvathukal, "Snapshot Metrics Are Not Enough: Analyzing Software Repositories with Longitudinal Metrics," in *ASE-Tools*, 2022.

[35] Meta AI Research, "Papers with code," 2023. [Online]. Available: https://paperswithcode.com/

[36] PINTO0309, "Pinto model zoo," 2023. [Online]. Available: https://github.com/PINTO0309/PINTO_model_zoo

[37] eLinux, "Jetson zoo," 2022. [Online]. Available: https://elinux.org/Jetson_Zoo

[38] P. Ladisa, H. Plate, M. Martinez, and O. Barais, "Taxonomy of Attacks on Open-Source Software Supply Chains," 2022. [Online]. Available: http://arxiv.org/abs/2204.04008

[39] M. Alfadel, D. E. Costa, and E. Shihab, "Empirical Analysis of Security Vulnerabilities in Python Packages," in *International Conf. on SW Analysis, Evolution and Reengineering*, 2021.

[40] Y. Ma, T. Dey, C. Bogart, and y. . . others, "World of code: enabling a research workflow for mining and analyzing the universe of open source VCS data," *Empirical Software Engineering*.

[41] OpenAI, "Introducing chatgpt," https://openai.com/blog/chatgpt, 2022.

[42] A. Cruz and A. Duarte, "Tensorflow hub," 2022. [Online]. Available: https://npms.io/about

[43] PyPI, "Python package index," 2022. [Online]. Available: https://pypi.org

[44] D. Montes, P. Pongpatapee, J. Schultz, C. Guo, W. Jiang, and J. Davis, "Discrepancies among Pre-trained Deep Neural Networks: A New Threat to Model Zoo Reliability," in *ESEC/FSE-IVR*, 2022.

[45] M. Robillard, R. Walker, and T. Zimmermann, "Recommendation Systems for Software Engineering," *IEEE Software*, 2010.

[46] J. Slankas and L. Williams, "Automated extraction of non-functional requirements in available documentation," in *International Workshop on Natural Language Analysis in SW Engineering*, 2013.

[47] M. Allamanis, H. Peng, and C. Sutton, "A Convolutional Attention Network for Extreme Summarization of Source Code."