Reusing Deep Learning Models: Challenges and Directions in Software Engineering

George K. Thiruvathukal, *Professor and Chairperson*
Loyola University Chicago, Computer Science Department
Visiting Computer Scientist, Argonne National Laboratory
(w) https://gkt.sh (e) gkt@cs.luc.edu

Co-Authors: James C. Davis*, Purvish Jajal*, Wenxin Jiang*, Taylor Schorlemmer*, Nicholas Synovic+, and George K. Thiruvathukal+

*Purdue University +Loyola University Chicago

We acknowledge financial support from Google, Cisco, and NSF awards 2229703, 2107230, 2107020, and 2104319.

Talk overview

1. Software Engineering Refresher
2. Machine Learning and Pre-Trained Models
3. Reuse Challenges
4. Future Directions
5. Additional Reading

Software Engineering Refresher

Retrospective: What is Software Engineering?

"Software Engineering is (1) the application of a **systematic, disciplined, quantifiable approach** to the development, operation, and maintenance of software; that is, the application of engineering to software

and (2) **the study of approaches** as in (1)"

IEEE Standards Collection: *Software Engineering*, IEEE Standard 610.12-1990 and Pressman, Software Engineering, 7th Ed.

This forward-looking definition of SE from 1993 (IEEE!) speaks to the importance not only of software methodology but also **the study of how software comes to be**.

Machine learning software is still software by any other name and is therefore worthy of further study and community attention.

Two Primary Kinds of SE Research

Observations

Open problems

Solutions

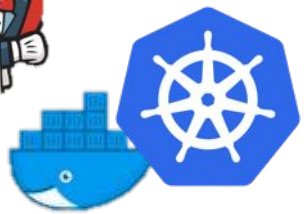


{JSON}
protobuf

SUBVERSION
git
CVS

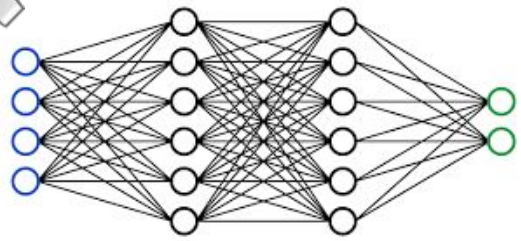


MySQL
mongoDB.



docker

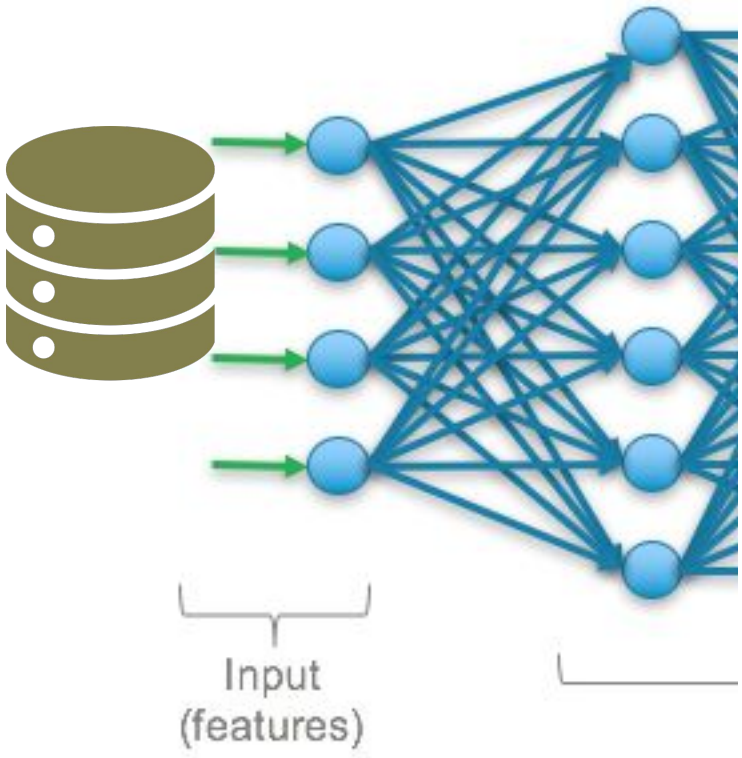
IJ
LLVM

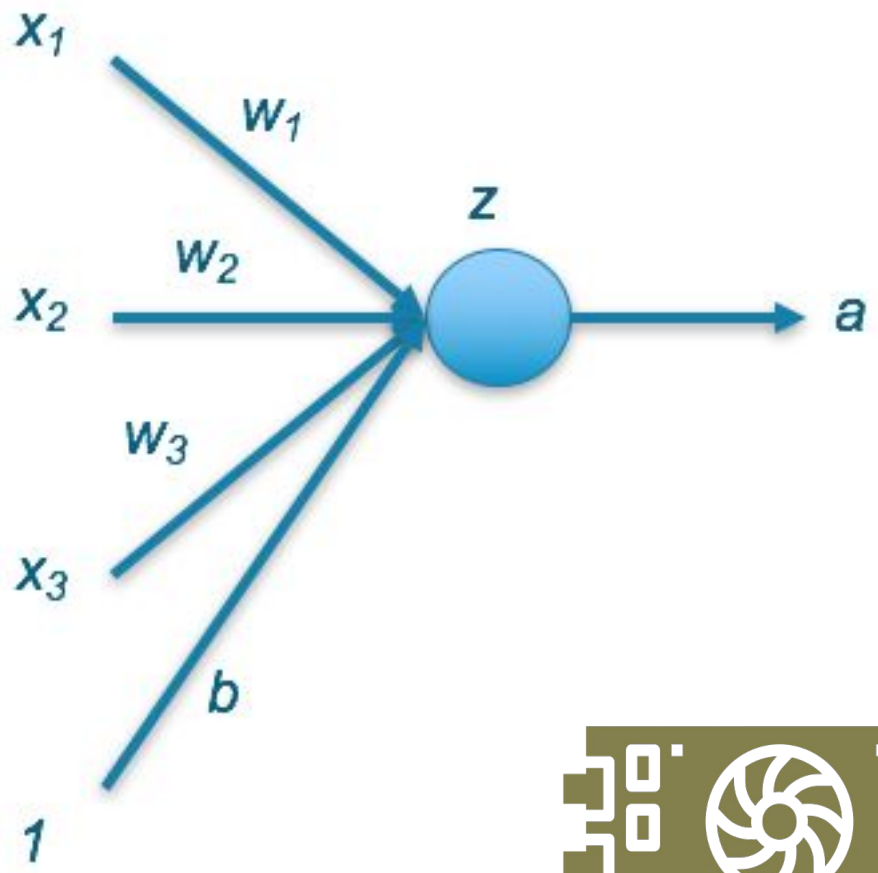




Machine Learning and Pre-Trained Models

What is a Neural Network?

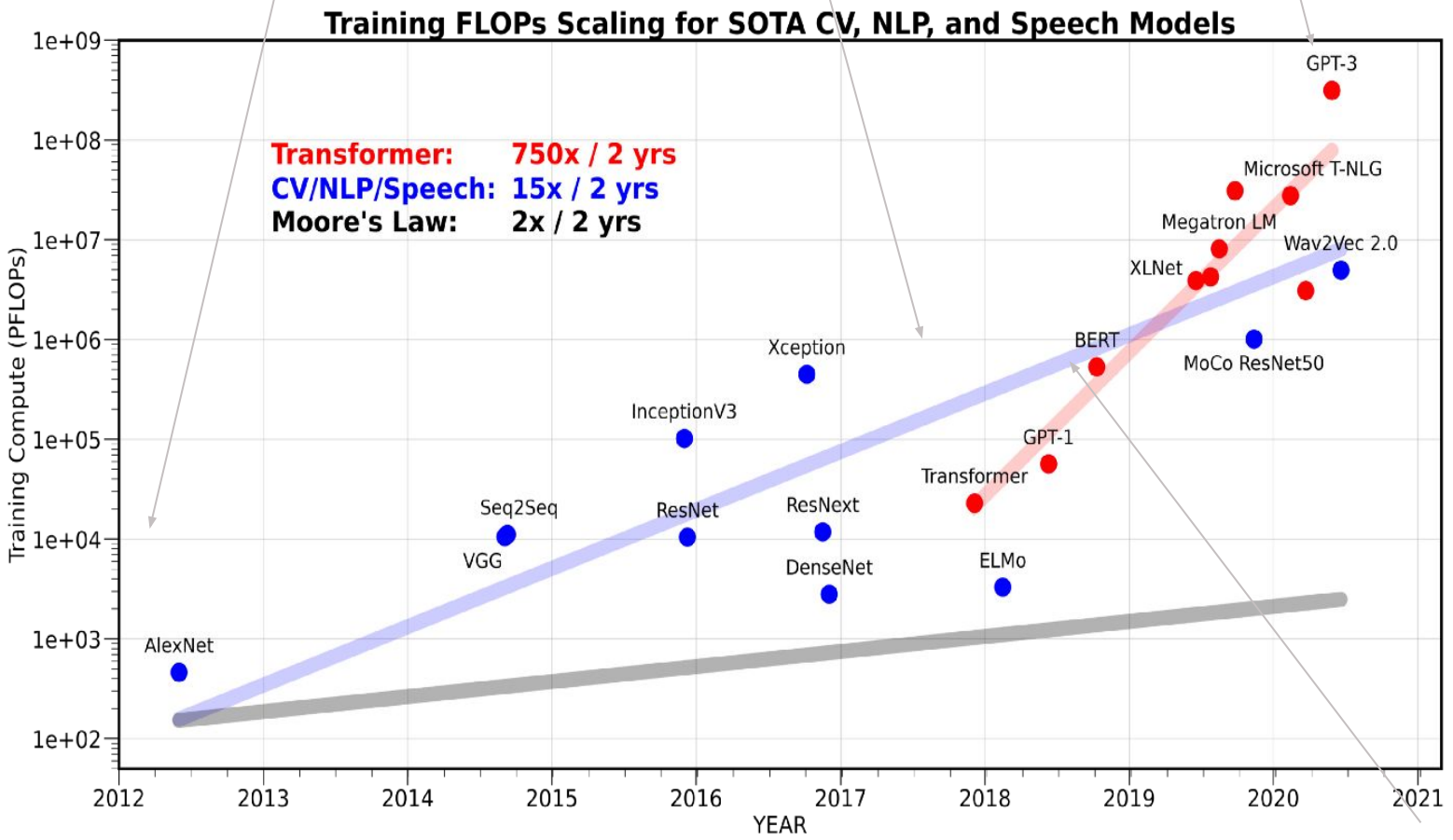




https://srnghn.medium.com/deep-learning-overview-of-neurons-and-activation-functions-1d98286cf1e4

Huge models, huge training costs (and growing)

60M parameters ***110M parameters*** ***175B parameters***



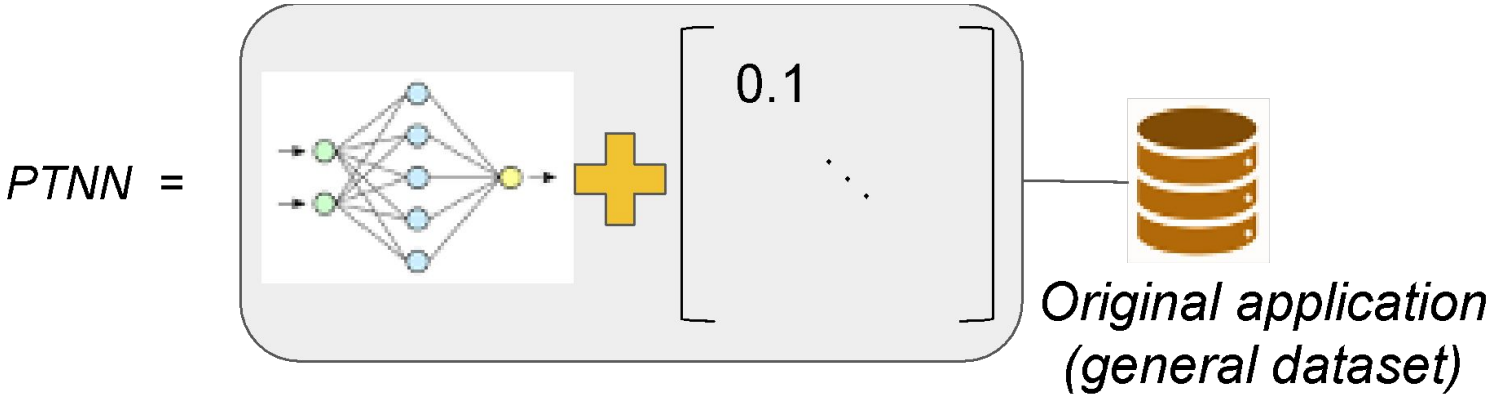
GPT-4 ~170T parameters

Carbon footprint costs measured using BERT in 2019

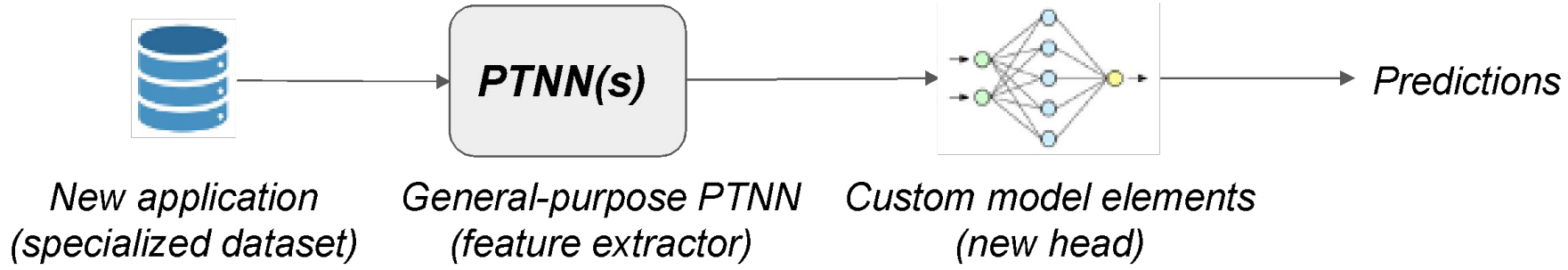
Patterson et al., 2021 "Carbon emissions and large neural network training." *arXiv preprint arXiv:2104.10350* (2021).
Gholami et al. 2021: https://medium.com/riselab/ai-and-memory-wall-2cb4265cb0b8
Strubell et al. 2019: https://arxiv.org/pdf/1906.02243.pdf

Pre-trained neural networks

PTNN: A neural network that is already parameterized for an application



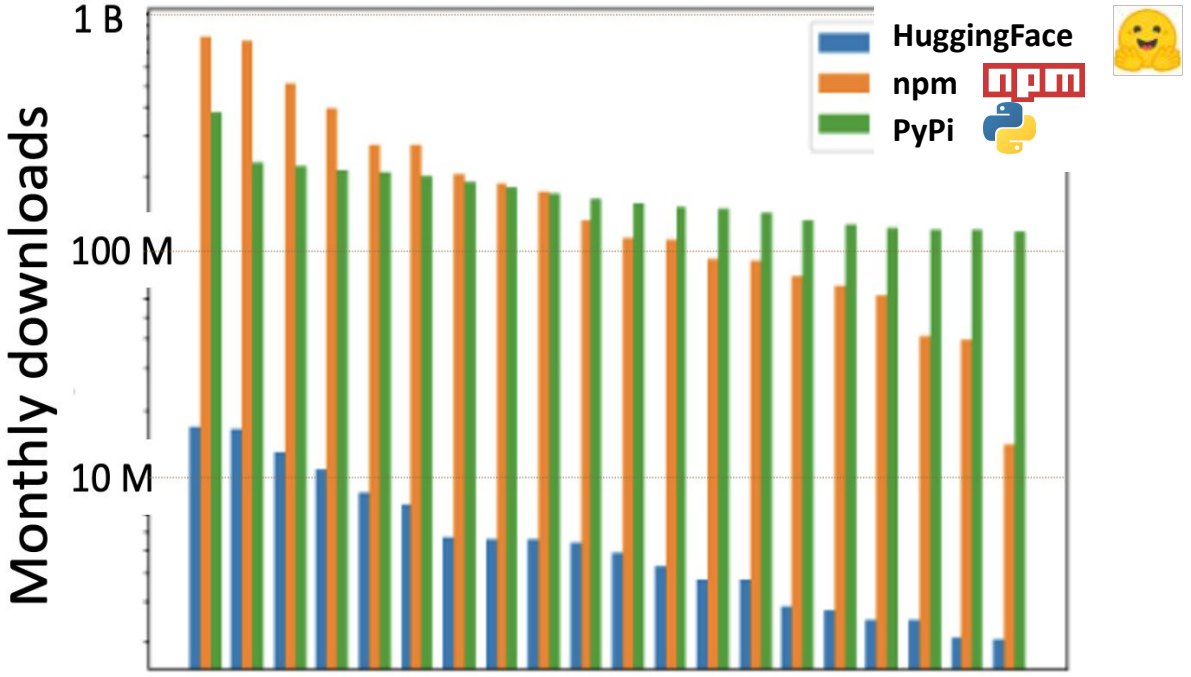
Example use: Transfer learning



PTMs vs. Traditional Software Package Reuse

pyf

```
#
# This file is autogenerate
# To update, run:
#
#    pip-compile
#
contourpy==1.0.6
    # via matplotlib
cycler==0.11.0
    # via matplotlib
```



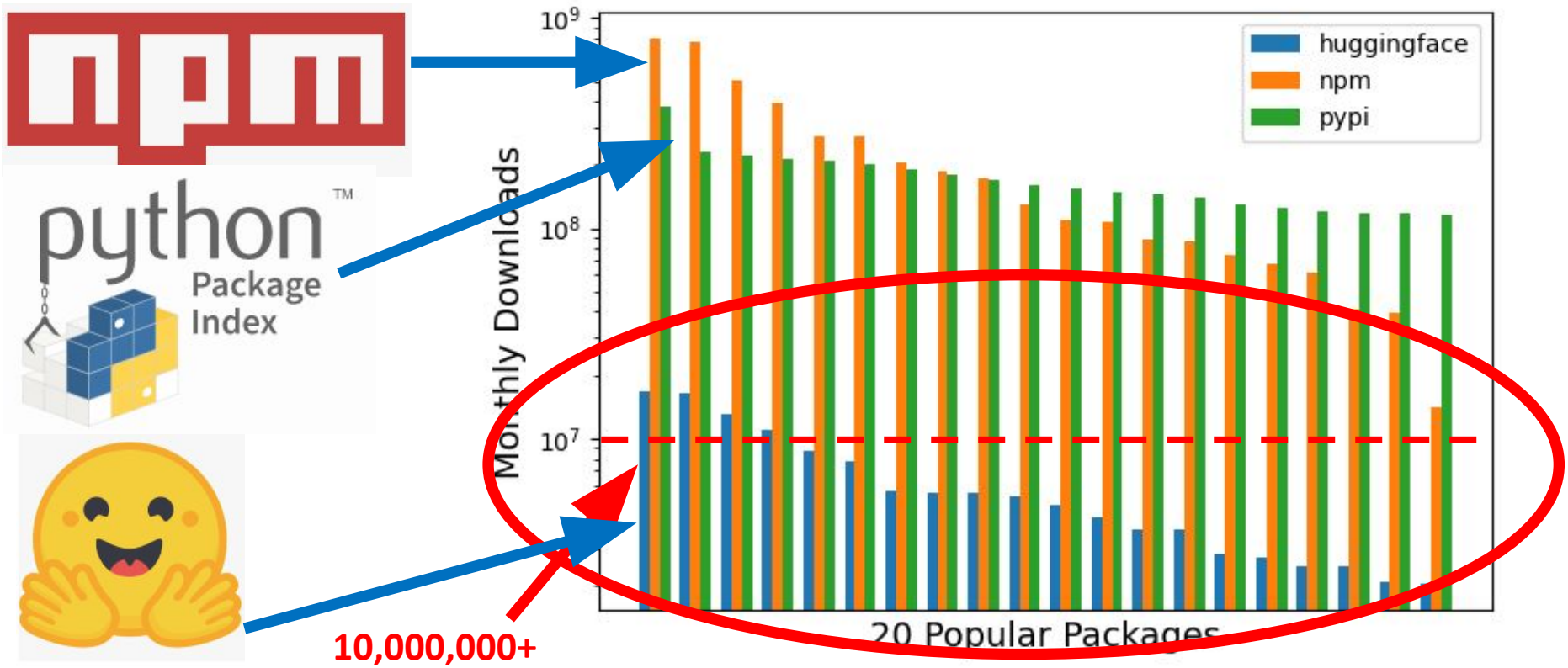
Top 20 most popular packages

```
"dependencies": {
  "markdown-toc": "^1.2.0"
}
```

Model Hubs

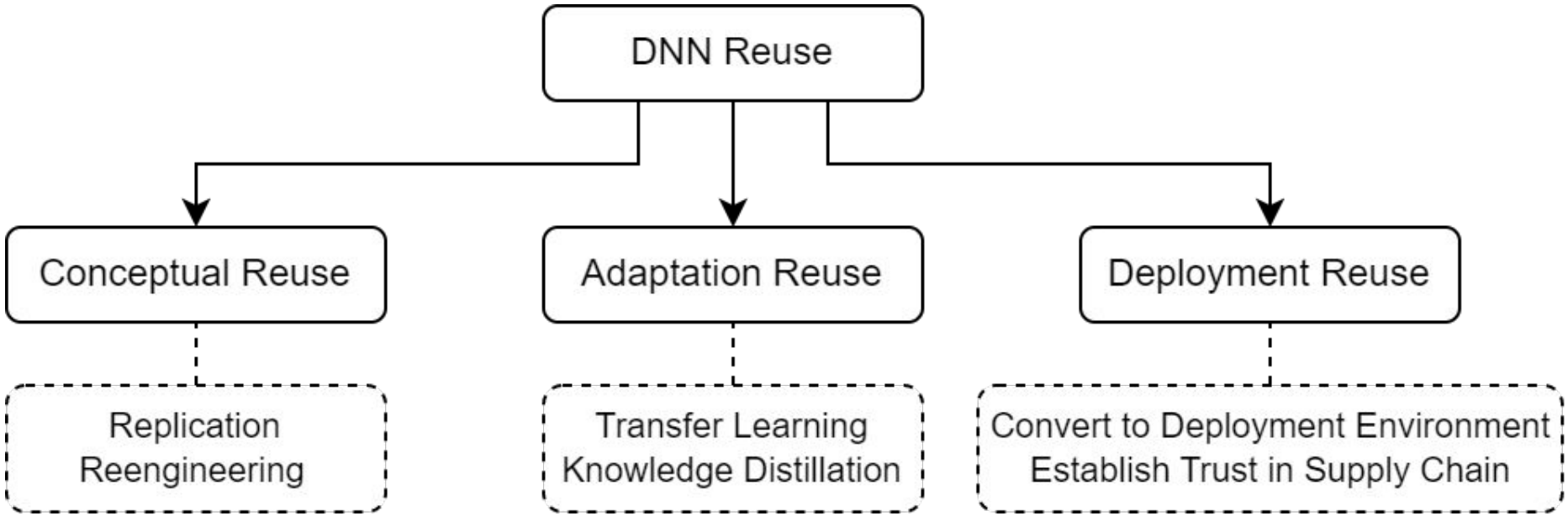
What is a model hub?

- A model hub is a hosted platform of pre-trained models (PTMs) and datasets organized by problem domain.



Reuse Challenges

Three Types of Reuse



Deep neural network reuse is the process of using existing DNN technology for another purpose.

We focus on three distinct types: *conceptual reuse*, where existing theory is repurposed; *adaptation reuse*, where existing DNN models are modified; and *deployment reuse* where existing DNN models are converted for use in a new environment.

Dashed boxes provide examples of each type.

Reuse in Deep Learning (DNNs)

Conceptual Reuse

Replicate and reengineer the algorithms, model architectures, or other concepts described in academic literature and similar sources, integrating the replication into new projects.

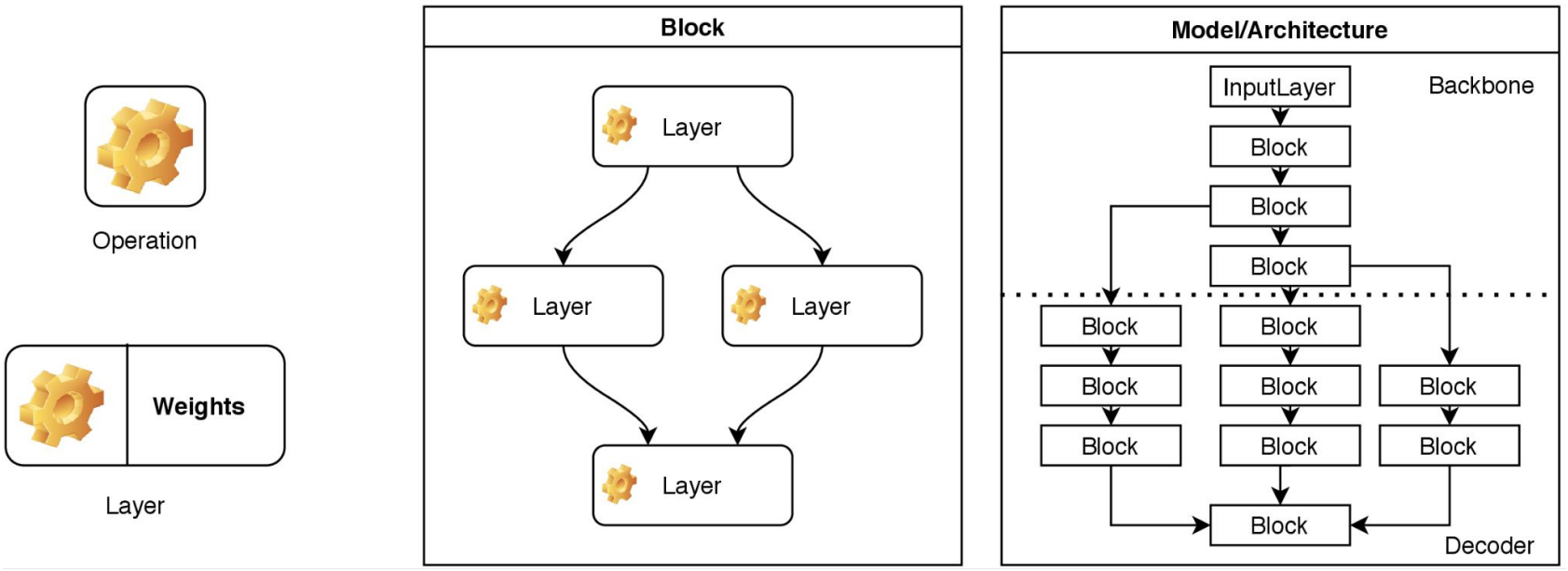
Adaptation Reuse

Leverage existing DNN models and adapt them to solve different learning tasks.

Deployment Reuse

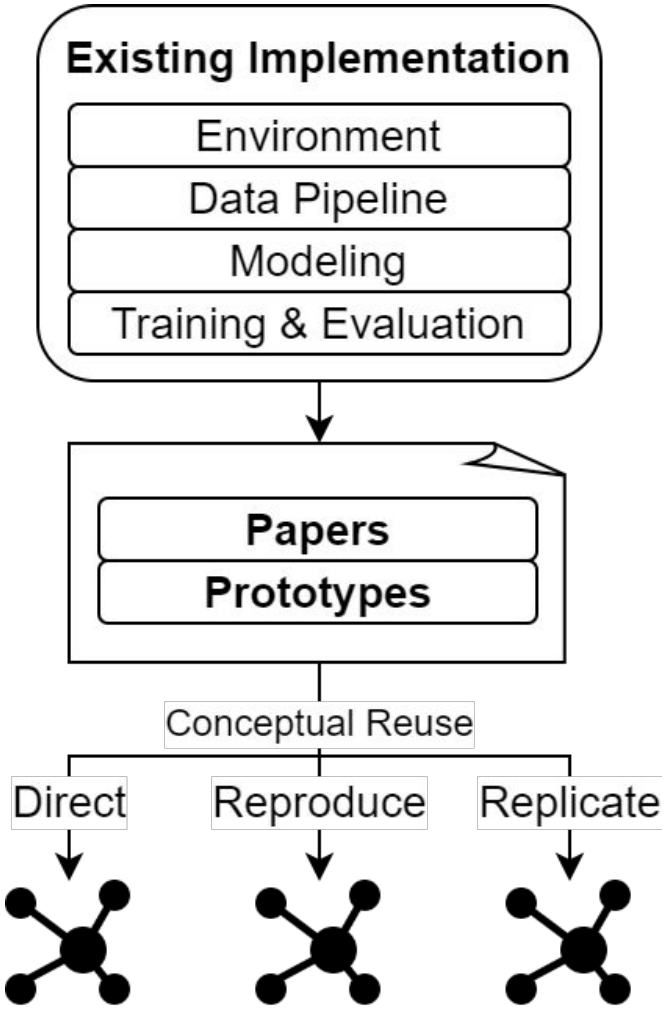
Convert and deploy pre-trained DNN models in different computational environments and frameworks.

Model Structure



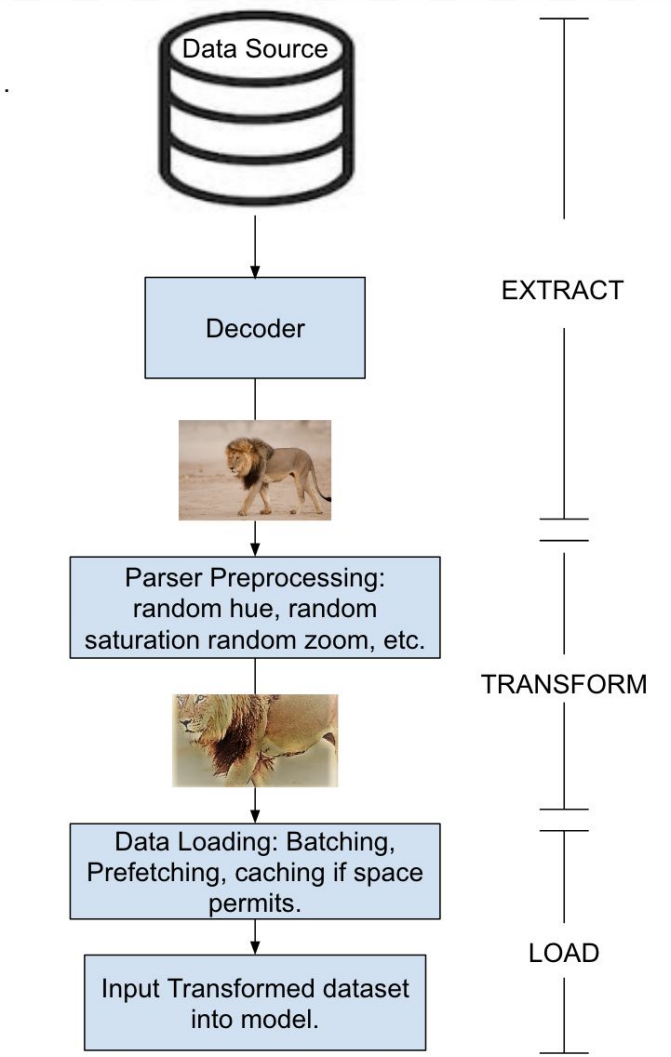
Components of a deep neural network (DNN), represented at different levels of abstraction. A DNN is a composition of weighted operations. These are combined into a *layer*; a group of layers into a *block*; and a *group of blocks into a sub-graph* such as a *backbone* or a *head*.

Conceptual Reuse



Replicate and reengineer the algorithms, model architectures, or other concepts described in academic literature and similar sources, integrating the replication into new projects.

Data Pipelines



Well established pattern from industry.

Illustration of a data pipeline following the Extract-Transform-Load design pattern.

The specific pipeline is for the You-Only-Look-Once (YOLO) model family (computer vision)

Conceptual Reuse Challenges

Reproducibility of Results

Reproducibility is considered a key quality of machine learning software, yet achieving DNN **reproducibility remains a challenging task** and continues to be a focal point within the research community.

Many SOTA models are prototype stage. This stage is typically characterized by an **absence of rigorous testing**, inadequate **documentation**, and a lack of considerations for **portability**.

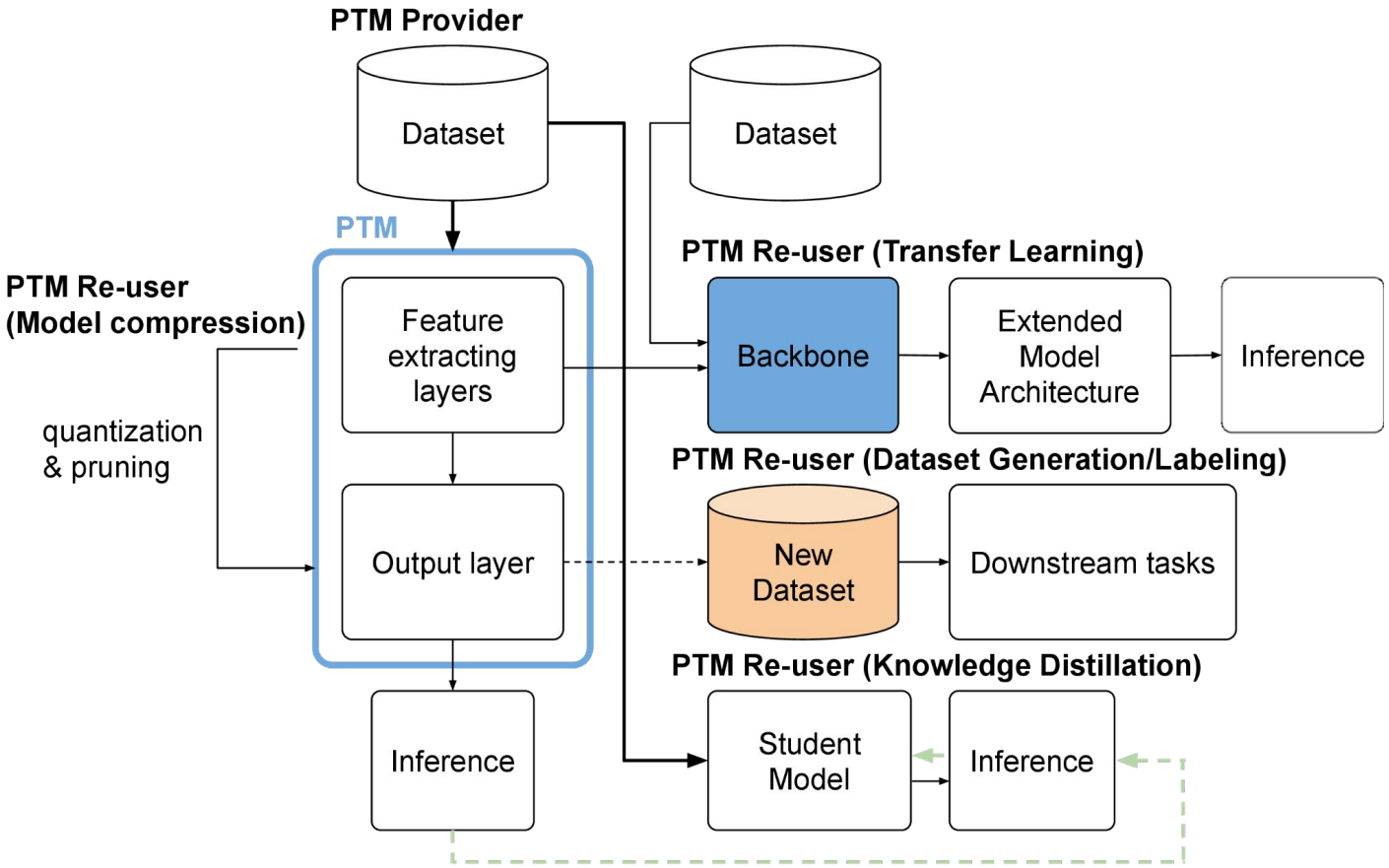
Model Replication and Reengineering

Replicating and reengineering DNNs is tricky, even when referring to the original code of the research prototypes

Our group has previously reported on three challenges of DNN replication and reengineering: model operationalization, portability of DL operations, and performance debugging.

Adaptation Reuse

Leverage existing DNN models and adapt them to solve different learning tasks.



Adaptation Reuse Challenges

Technical Adaptation

Accuracy and latency issues. **Lack of push-button solutions** for adapting DNNs across diverse hardware environments (heterogeneous compute is a thing)

Fairness and robustness is an ongoing challenge. Techniques such as local interpretability and model-agnostic methods can help

More modular designs (similar to traditional software) can help.

Decision Making

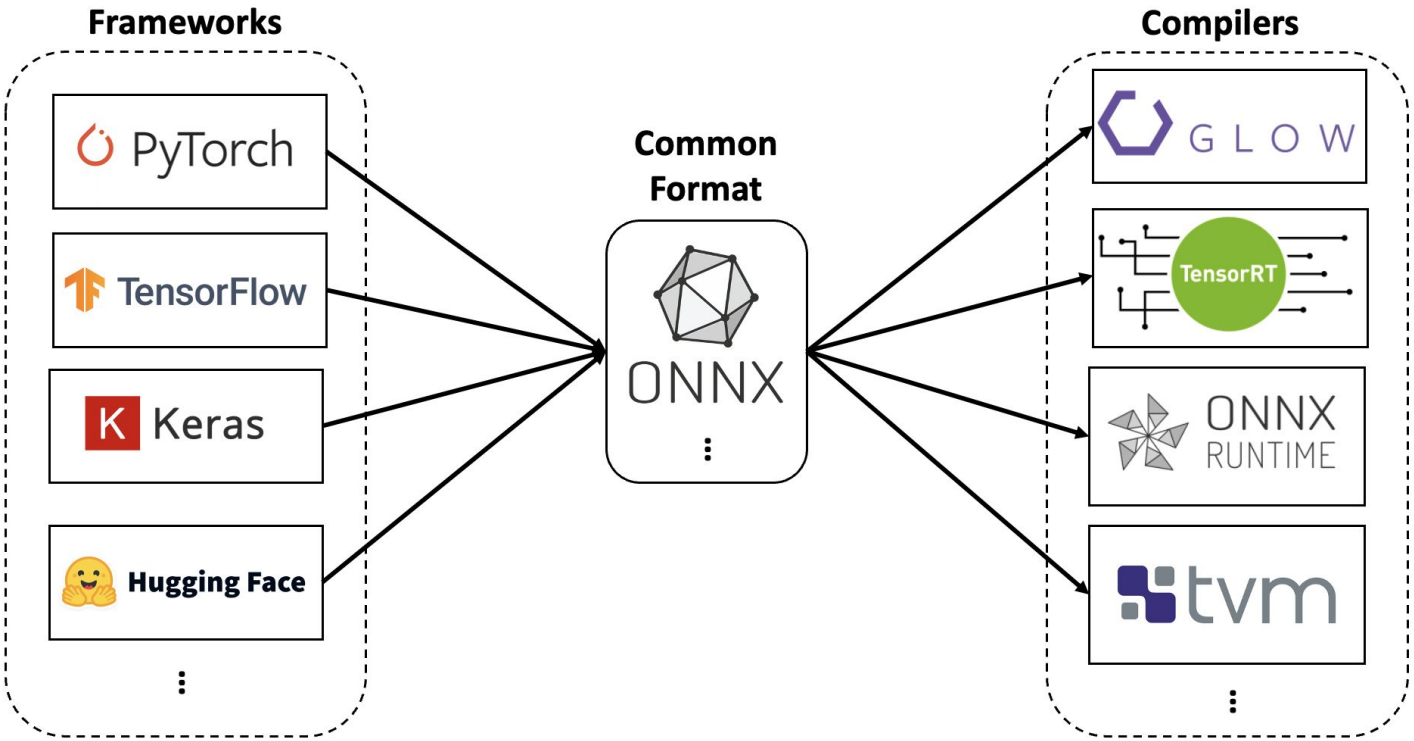
Registries often lack infrastructure and attributes helpful to the reuse process: provenance, reproducibility, and portability.

Traditional software info, e.g. **popularity, quality, and maintenance, are often emphasized** instead (not sufficient).

Security and privacy attacks (studied by our group) are a concern: train-time attacks, idle-time attacks, inference-time attacks, and traditional software supply chain attacks.

Deployment Reuse

Convert and deploy pre-trained DNN models in different computational environments and frameworks.



Deployment Reuse Challenges

Interoperability

Specialized compute platforms and novel architectures present a great challenge for deployment.

Deep learning compilers can help; however, **not all frameworks and their operators are supported** on different hardware.

We're looking at **model conversion failures** in emerging work within our group by doing failure studies. Operator conversion is a common cause of failure.

Establishing Trust in Supply Chains

Similar to traditional software supply chains (e.g. Node), **establishing trust** is a major challenge. Anyone can release anything (for the most part).

Users are often unwilling or unable to check for various possible attacks.

Traditional software methods such as Software Bills of Materials (SBOMs) and reproducible builds are more difficult for DNNs, owing to **non-determinism and training costs**.

Future Research Directions

Future Directions: Conceptual Reuse

Evaluate artifacts at conferences/journals to look beyond basic reproducibility **by including software engineering aspects from traditional software**. Consider a “checklist” approach similar to Journal of Open Source Software.

Testing tools are emerging for DNNs but need greater adoption. Make use of **checklists** to aid in assessing conceptual reuse potential.

Make use proper testing tools across the board: e.g., validation tools, unit testing, and fuzz testing (for security aspects).

Future Directions: Adaptation Reuse

Prior work found that the trustworthiness of DNNs are concerning due to the lack of DNN transparency. Future work can **measure attributes of DNNs by extracting from provided documentation, source code, and metadata**

Engineers struggle to compare different DNNs and **identify a good way to adapt to their downstream task**. To facilitate the adaptation, researchers can identify different approaches to support the model selection process. e.g., providing **enhanced documentation, similar to the badges used by GitHub**.

Open-source PTMs remain underutilized, suggesting the **need for a robust PTM recommendation system** aid engineers in adaptation reuse.

Specific **attack detection tools** are currently missing in DL model registries. Adding such tools would help to improve trust of registries.

Future Directions: Deployment Reuse

Model converters (e.g. ONNX) often produce models that are semantically not equivalent to the original models, pointing to a **need for more rigor in the intermediate representation** or **stronger type checking**.

Model converters suffer from **incompatibilities with the evolution of intermediate representation**, better focused by understanding operator popularity, and **domain-specific languages** can be used to automatically (and safely) generate converter code.

Build on existing software Supply Chain Security tech for DNNs: The software engineering community has been working on systems such as TUF and Sigstore to increase the usability and effectiveness of signatures for package managers.

Future Directions: Assessing SE Process of DNNs

There is a lack of tools that quantify what is and is not an effective SE process for DNNs. SE process must look beyond the code-based versioning model.

Mining repositories is a major challenge. Beyond source code: training dataset (sometimes prohibitively large), configuration, and documentation are all crucial to understanding PTMs.

We've created a PTMTorrent data set to aid in mining HuggingFace and other hubs for MSR 2023 with a new/richer version planned for 2024.

Papers from Our Group

Davis, James C.; Jajal, Purvish; Jiang, Wenxin; Schorlemmer, Taylor R; Synovic, Nicholas; Thiruvathukal, George K., Reusing Deep Learning Models: Challenges and Directions in Software Engineering. IEEE JVA Symposium on Modern Computing at IEEE Services 2023, doi.org/10.6084/m9.figshare.23317556 **[this paper]**

Jiang, Wenxin; Synovic, Nicholas; Hyatt, Matthew; Schorlemmer, Taylor R.; Sethi, Rohan; Lu, Yung-Hsiang; et al. (2023): An Empirical Study of Pre-Trained Model Reuse in the Hugging Face Deep Learning Model Registry. In Proceedings of International Conference on Software Engineering, 2023. doi.org/10.6084/m9.figshare.22056872

Wenxin Jiang, Nicholas Synovic, Rohan Sethi, Aryan Indarapu, Matt Hyatt, Taylor R. Schorlemmer, George K. Thiruvathukal, and James C. Davis. 2022. An Empirical Study of Artifacts and Security Risks in the Pre-trained Model Supply Chain. In Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED'22). Association for Computing Machinery, New York, NY, USA, 105–114. doi.org/10.1145/3560835.3564547

Jiang, Wenxin; Synovic, Nicholas; Jajal, Purvish; Schorlemmer, Taylor R.; Tewari, Arav; Pareek, Bhavesh; et al. (2023): PTMTorrent: A Dataset for Mining Open-source Pre-trained Model Packages. figshare. Dataset. https://doi.org/10.6084/m9.figshare.22009880