An Empirical Study of Artifacts and Security Risks in the Pre-trained Model Supply Chain

Wenxin Jiang
Purdue University
West Lafayette, IN, USA
jiang784@purdue.edu

Nicholas Synovic
Loyola University Chicago
Chicago, IL, USA
nsynovic@luc.edu

Rohan Sethi
Loyola University Chicago
Chicago, IL, USA
rsethi1@luc.edu

Aryan Indarapu
University of Illinois
Urbana-Champaign
Champaign, IL, USA
awi2@illinois.edu

Matt Hyatt
Loyola University Chicago
Chicago, IL, USA
mhyatt@luc.edu

Taylor R. Schorlemmer
Purdue University
West Lafayette, IN, USA
tschorle@purdue.edu

George K. Thiruvathukal
Loyola University Chicago
Chicago, IL, USA
gkt@cs.luc.edu

James C. Davis
Purdue University
West Lafayette, IN, USA
davisjam@purdue.edu

Abstract

Deep neural networks achieve state-of-the-art performance on many tasks, but require increasingly complex architectures and costly training procedures. Engineers can reduce costs by reusing a pre-trained model (PTM) and fine-tuning it for their own tasks. To facilitate software reuse, engineers collaborate around *model hubs*, collections of PTMs and datasets organized by problem domain. Although model hubs are now comparable in popularity and size to other software ecosystems, the associated *PTM supply chain* has not yet been examined from a software engineering perspective.

We present an empirical study of artifacts and security features in 8 model hubs. We indicate the potential threat models and show that the existing defenses are insufficient for ensuring the security of PTMs. We compare PTM and traditional supply chains, and propose directions for further measurements and tools to increase the reliability of the PTM supply chain.

CCS Concepts

• **Computing methodologies → Machine learning**; • **General and reference → Empirical studies**; • **Security and privacy**;

Keywords

Software supply chain; Machine learning; Deep neural networks; Model hubs; Software reuse; Empirical software engineering

ACM Reference Format:
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), November 11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3560835.3564547

1 Introduction

Deep Neural Networks (DNNs) are widely used, from image recognition in autonomous vehicles [24] to detecting anomalies in system logs [16]. Making or training DNNs is challenging for reasons including the high cost of model training and evaluation [30], variation in DL libraries [60], and mismatch between the needs of research and practice [74]. Additionally, training large DNNs incurs a large carbon footprint [59]. *Reusing* pre-trained models (PTMs) can address some of these problems, enabling engineering teams across different organizations to share the economic and environmental burden [72, 86]. PTMs are reused through *Model Hubs — collections of PTMs and datasets organized by problem domain.* Hugging Face is the largest model hub with 60,904 public PTMs [19]. As shown in Figure 1, the most popular PTMs in Hugging Face are downloaded at rates comparable to the popular packages in npm [54] and PyPI [61]. This indicates that the model hubs also make up a significant portion of software supply chain.

However, DL model hubs are in their infancy. Despite the increasing importance of PTMs (Figure 1), there has been no systematic investigation of the artifacts and security risks in the PTM supply chain. Prior works have examined the security risks in traditional software registries such as NPM [47, 85, 88]. On the other hand, the machine learning community has studied adversarial attacks on DNNs [1, 38] and PTMs [31]. In this work, we synthesize techniques from traditional package ecosystem security studies and DL attacks to understand the risks of the PTM supply chain.

To characterize the PTM supply chain, we an empirical study of artifacts (*i.e.,* the contents of model hubs) and security risks in popular model hubs. As a first step, we filtered 8 model hubs by searching for key words in a major search engine. Furthermore, we

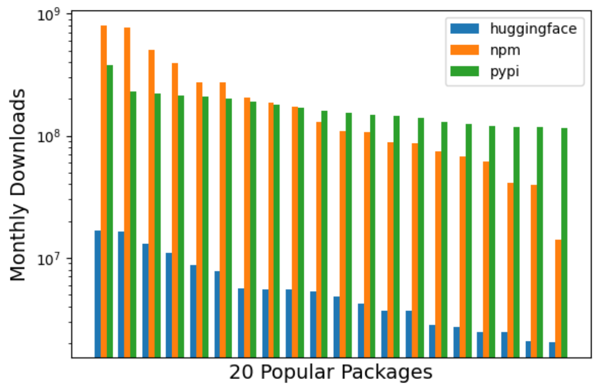

Figure 1: Monthly downloads of top-20 packages in Hugging Face, NPM, and Pypi. Download rates of Hugging Face PTMs are comparable to popular traditional package registries.

measured the artifacts in these model hubs, identified the similarities and differences between different kinds of model hubs, and depicted the PTM supply chain. Finally, we studied the security features in place to improve the security of the model hubs, and analyzed the potential threat.

Our results introduce thee types of model hubs (open, gated, and commercial), with varying security properties by type. We summarized two threat models and measured potential risks in the form of model discrepancies and maintainer reach. We observed two main differences between the PTM supply chain and the traditional software supply chain: versioning and security properties.

Our contributions are:

- We measure the artifacts in 8 model hubs, identify their typical structures, and depict the PTM supply chain.
- We indicate the security features on different model hubs, and summarize the threat models.
- We show the differences of versioning and security risks between PTM supply chain and traditional supply chain.

2 Background and Related works

In this section we describe background on pre-trained models and software supply chains.

2.1 Pre-Trained Models (PTMs)

Various factors have led to the rise of PTMs. Large PTMs, such as BERT [14] and GPT [6], are recent milestones for the DL community; they can be used off-the-shelf and tuned to more specific tasks with little effort [36]. Meanwhile, compared to traditional software, deep learning software is hard and expensive to build [30] and reproduce [2, 60] from scratch. Deep learning models also present unique debugging and testing challenges beyond traditional software, owing to the probabilistic nature of machine learning [3, 5, 15]. PTMs partially address these issues [68].

In light of these properties, engineers have begun to rely on *model hubs* offering a collection of PTMs and datasets organized by problem domain [37, 82]. We define a *Model Hub Ecosystem* as: *the interaction of a set of actors on top of the model hub that results in a number of machine learning solutions or services* [50]. Since 2018, ML engineers have been building model hubs to better reuse these costly artifacts [28, 37, 82]. These hubs offer PTMs applicable to many DL domains including reinforcement learning [12], computer vision [71], and natural language processing [10, 63]. However, there has not yet been a study that explores the characteristics of model hubs. We explore their characteristics and potential security issues.

2.2 Pre-Trained Model Re-use

Engineers reuse DL models in various ways, including in transfer learning, dataset labeling, and knowledge distillation [29, 34, 84, 87] PTM reuse reduces the need for users to train their own models and allows models to be quickly developed for a greater variety of tasks [34, 51]. Figure 2 displays methods of PTM reuse. Initially, A PTM provider trains a DL model on a dataset to create a model checkpoint (*i.e.,* PTM) with its associated architecture and weights. A PTM reuser can implement one of the methods below to transfer information to the new model. This model can be reused for either the same task or a novel application. The new model can be smaller, more manageable, and resource-efficient [29, 34].

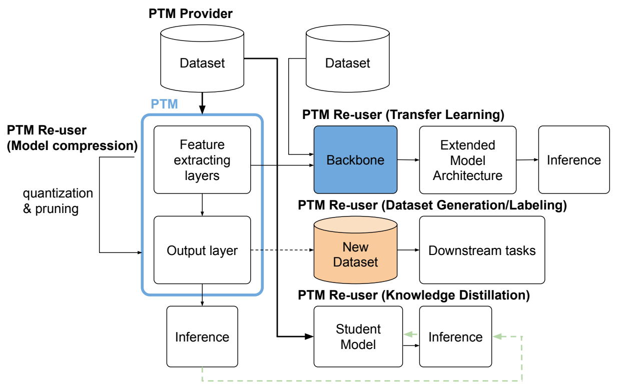
PTM Provider
Dataset
Dataset
PTM
PTM Re-user (Model compression)
quantization & pruning
Feature extracting layers
Output layer
Inference
PTM Re-user (Transfer Learning)
Backbone
Extended Model Architecture
Inference
PTM Re-user (Dataset Generation/Labeling)
New Dataset
Downstream tasks
PTM Re-user (Knowledge Distillation)
Student Model
Inference

Figure 2: A PTM can be used in four different ways, including transfer learning, dataset generation and labeling, quantization and pruning, and knowledge distillation.

Another milestone for DL model reuse is the introduction of transfer learning through knowledge transfer [58, 77]. Transfer learning can reduce computation cost by avoiding expensive data-labeling tasks [58]. Han *et al.* indicated that adopting a PTM's feature extraction layers for downstream tasks rather than learning models from scratch has become the consensus of the AI community [34]. For example, ResNet [35] pre-trained on ImageNet [44] can quickly advance various tasks [40, 64, 65].

PTMs can also be reused for automatically generating data. Automated dataset labelling can reduce the cost of dataset creation, while increasing the quantity of data, and can result in a high degree of accuracy dependent upon the models performance [18, 42]. Generative models can also help generate new data [9, 70].

Model compression approaches can make the model portable to edge devices [27]. Reusers can apply quantization or pruning techniques on the PTMs directly to make it more portable [76]. By training a student model to mimic the behavior of its teacher, knowledge distillation can receive richer information from its loss function and can learn quicker than it normally would [29]. For smaller mobile applications and systems with limited computational

Name	# Models	# Datasets	Categories	# Tasks	Contribution	Distribution
Hugging Face [19]	60,904	7,759	Computer Vision, Natural Language Processing, Audio, Multi-modal, Tabular, Reinforcement Learning	28	Open	Hub APIs
TensorFlow Hub [75]	1,273	87	Image, Text, Video, Audio	39	Gated	Hub APIs
Model Zoo [41]	1,211	0	Computer Vision, Natural Language Processing, Audio, Generative, Reinforcement Learning, Unsupervised Learning, Graph, Self-supervised Learning, Health and Bioscience	8	Gated	GitHub
Pytorch Hub [62]	48	0	Computer Vision, Natural Language Processing, Audio, Generative, Scriptable	N/A	Gated	Hub APIs
ONNX Model Zoo [57]	11	0	Computer Vision, Natural Language Processing, Audio	10	Gated	GitHub
Modelhub [46]	6	0	Computer Vision, Natural Language Processing	3	Gated	Hub APIs
NVIDIA NGC [55]	450	1	Computer Vision, Natural Language Processing, Audio, Other	10	Commercial	Hub APIs
MATLAB Model Hub [52]	60	0	Computer Vision, Natural Language Processing, Audio, Lidar	N/A	Commercial	Hub APIs

Table 1: Contents and artifacts of 8 popular model hubs. The hubs are grouped by contribution type, and sorted by the number of PTMs. Categories define the applications/uses of the machine learning models (i.e. computer vision, natural language processing). Tasks are actions that are required by categories (i.e. image classification, text generation).

capacity, knowledge distillation allows for a comparatively smaller model to learn from a larger network.

2.3 Secure PTM Software Supply Chains

The aforementioned types of model reuses result in a PTM supply chain, where new models rely on the extension of existing artifacts. Some of the most popular models on Hugging Face (*i.e.,* one of the most popular model hubs), such as *bert-base-uncased*[1] and *gpt2*[2], are PTMs that are frequently fine-tuned for a downstream task because they excel at extracting important features from natural language. As a result, the PTM supply chain exists, because any PTM Re-users can apply reuse techniques on open-source PTMs and datasets (Figure 2) [4] which could then be deployed in downstream applications, such as autonomous vehicles [25]. The vulnerability of PTM supply chain can pose a hazard to the real-world systems, but there is no study about the secure PTM supply chain yet.

Prior work has been done to study the inheritance of security vulnerabilities in software supply chains. Zimmermann *et al.* analyzed the metadata of 5,386,239 versions of NPM packages and reported that NPM based supply chains have single points of failure. Furthermore, they depicted the *maintainers' reach* (*i.e.,* the number of packages per maintainer) graph and identified that unmaintained packages threaten substantial downstream users [88]. We also depicted the maintainers' reach graph to explore the potential risks of PTM supply chain. Ladisa *et al.* proposed a general taxonomy for attacks on open-source supply chains which covers all supply chain stages from code contributions to package distribution. They then validated the taxonomy via a user survey of 17 domain experts and 134 software developers [47]. Zahan *et al.* proposed six signals of security weaknesses and identified 11 malicious packages from the install scripts signal [85]. However, the security vulnerabilities of PTMs could be different due to the probabilistic nature [81] and huge training cost [30, 32]. We are building on knowledge from the traditional software supply chain and package registries, and determining the characteristics of them in the PTM context.

Model hub use is increasing (Figure 1), but the security features surrounding the usage of PTMs are unclear [31]. Only Tan *et al.* have conducted an exploratory study on the deep learning supply chain. Their data is focused on deep learning frameworks, not PTMs [73]. However, model hubs also contribute to a significant part of DL software supply chain (Figure 1). With that in mind, our work aims to help the community better understand the practices of pre-trained model supply chain and inform further studies.

2.4 Adversarial attacks on and with PTMs

Following the reuse cases in Figure 2, an attacker can target either datasets or PTMs. First, it is possible to attack a model indirectly, via its training dataset in a *dataset poisoning* attack, which is an essential part in Figure 2. This attack involves mislabelling data manually or by a malicious PTM, resulting in future models that train off of this dataset to have poisoned values [26]. Second, some attacks directly target PTMs and can either modify the models' I/O behavior[3] (known as a *backdoor/Trojan* attack [48, 66]) or add side-effects. It is possible to construct *weight poisoning* attacks by injecting pre-trained weights with vulnerabilities that expose *backdoors* after fine-tuning, even with limited knowledge of the training dataset [45]. This form of attack is further discussed by Gu *et al.* as a generalized malicious model, called a *BadNet* [31], and it can also affect the I/O behavior of downstream models. Another malicious attack, the *EvilModel*, produces side-effects instead. Wang *et al.* show a PTM with malware bytes embedded into its neurons'

[1]See https://huggingface.co/bert-base-uncased.
[2]See https://huggingface.co/gpt2.

[3]The I/O behavior means the prediction of specific input patterns can be affected.

parameters, which are extracted and assembled into malware at runtime [79]. Both *BadNets* and *EvilModels* can cause performance drops. However, recent studies have also shown that it is possible to attack the PTMs without affecting their performance [49, 80].

These attacks can be mapped to real-world examples. Compromised PTMs can bypass the anti-malware protections provided by Hugging Face [43]. Bias in AI systems [23] may result from dataset poisoning [26]. Beyond the malicious modification of PTMs and training datasets, bad actors can also abuse the widespread availability of PTMs in problematic applications. For example, a PTM was trained to be rude and then deployed as a conversation bot on a web forum [22]. PTMs have also been used to refine phishing attacks [69]. However, we note that compared to abuses of traditional software, the abuse of PTMs (either in supply chain or in application) appears to be relatively rare. The PTM supply chain is still in its infancy and we expect the relevant issues will increase soon. We report on security aspects of the PTM supply chain and indicate potential threats in the model hubs.

3 Research Questions

To the best of our knowledge, the characteristics and practices of secure PTM supply chain have not been studied before, as current work either focuses on traditional software supply chain attacks [47, 85, 88] or supply chain for major deep learning frameworks [73].

To characterize the PTM supply chain, we ask:

RQ1 *What is the typical structure of model hubs?*
RQ2 *What practices are in place to improve security among users of the model hubs?*
RQ3 *What are the potential threats in model hubs?*

In §4, we discuss our methodology. In §5, we identify the characteristics of model hubs and depict the PTM supply chain. In §5.1, we measure these model hubs, indicate the artifacts in each of them, and analyze the possible explanations based on their characteristics. In §6, we indicate the security features in different model hubs, indicate four security defenses and potential risks. Finally, we analyze the threat models in §7 and imply future directions in §8.

4 Methodology

Model Hub Selection: First, we search for keywords[4] in a major search engine (Google). Then we use three criteria to filter the resulted pages: (1) There is a model hub matching our definition; (2) The model hub should have a website. (3) The documentation should be accessible. Finally, we got 8 model hubs: Hugging Face [19], TensorFlow Hub [75], Pytorch Hub [62], MATLAB Model Hub [52], ONNX Model Zoo [57], NVIDIA NGC [55], Modelhub [46], model zoo [41]. All of the accessible model hubs are listed in Table 1.

RQ1: To understand the typical structure of model hubs, we looked at each of them and collected the categories (*e.g.,* computer vision, natural language processing), tasks (*e.g.,* image classification, tokenization), models, and datasets. Our analysis of these artifacts appears in Table 1. We analyze the artifacts, properties, and categorize them into three different types, similar to prior work on APP stores [39]. Then we describe the *model contributing workflow*, *model distribution workflow*, and *model versioning*.

[4]Search terms were "Machine learning model hub" and "Deep learning model hub".

RQ2: We also tried to identify the existing security features of model hubs. The Hugging Face ecosystem has open access where anyone can publish their own models, thus it is more complex and more likely to have security risks than the others. Hugging Face briefly indicates their security features in documentations [21]. We looked at these documentations and investigated into the security features in Hugging Face ecosystem.

RQ3: To measure the potential security risks, we first measured the existence of discrepancies in Hugging Face. We audited 53 object detection models, 26 image classification models, and 160 sentiment analysis models from Hugging Face. We set up the test dataset and obtained the claimed metrics from the YAML file. Then we compare the actual accuracy to the claims and indicate whether there are any discrepancies.

Each repository on Hugging Face is managed by a set of maintainers. We measured the number of repositories per maintainer by looking at the commit histories from all PTMs in Hugging Face.

5 RQ1: What is the typical structure of model hubs?

Finding 1: The artifacts and properties of model hubs differ depending on their types. All model hub host models and other artifacts, and function as a research-to-practice pipeline. The contribution workflow differs by model hub types, but the mechanisms of distribution workflow and versioning are the same.

Model hubs can be implemented in different ways to achieve the goals of a particular community. To identify the collaboration model of model hubs, we first analyzed the artifacts in 8 different model hubs in §5.1. Based on our study on the artifacts, we identify the common properties of model hubs in §5.2. The differences we observe between model hub types is how models are *contributed* to the hub and *distributed* to end users. We discuss the contribution workflow in §5.3 and the distribution workflows in §5.4.

5.1 Model Hub Artifacts

Following the methodology outlined in §4, we found and analyzed 8 different deep learning model hubs, which are outlined in Table 1. We extracted the information about how each model hub operates and how users depend on these model hubs using the Web UI tools and/or available API endpoints.

Table 1 contains information about the name of categories and number of tasks offered by model hubs. We found that all model hubs had similar categories to offer (*e.g.,* Computer Vision, Natural Language Processing, etc.), but the model hubs that offered the greatest flexibility (*i.e.,* Hugging Face) in community contributions were found to have more tasks and models associated with each category, a clear advantage of open source model hubs. For example Hugging Face is an open hub that offers 60,904 models and tasks while Modelhub is a gated hub that only offers 6 models and tasks.

We found that different types of model hubs differ by the number of PTMs, datasets, and model types available. For example, Hugging Face and Model Zoo provide a breadth of machine learning model tasks and datasets for categories which are not included in other model hubs, such as reinforcement learning. While model hubs offer PTMs across a variety of tasks, the number of tasks offered does not necessarily indicate a large number of models. For example, in Hugging Face, there exists a large number of PTMs for NLP tasks (*e.g.,* 9,734 text classification PTMs), but relatively few PTMs for other model tasks (*e.g.,* only 840 image classification PTMs). Hugging Face only offers about 1% of its models for the reinforcement learning category, while it offers 53% of its models for the computer vision category.

5.2 Common properties

Based on the qualitative data obtained from the documentations, we were able to categorize these model hubs into three types: *open*, *gated*, and *commercial*. For the *open* model hub, anyone can contribute to it. The *gated* model hub allows only restricted contributions. For the *commercial* model hub, it only accept internal contributions. These model hubs share two common properties:

Common Property 1: A model hub hosts *pre-trained models*. It may host other artifacts (*e.g.,* datasets, source code).

Common Property 2: Model hubs function as a research-to-practice pipeline, more so than in the traditional software supply chain. Model contributors are either the authors of a model, or are maintaining or fine-tuning an existing model [21]. An author is the researcher(s) who created the original implementation of a model, typically released alongside an academic work [21]. A maintainer is the individual(s) who maintain and expand the properties of an author's work [88]. This could be done through resolving bugs, improving model performance and accuracy, or providing alternative implementations of the model.

5.3 Model Contribution Workflow

Depending on the model hub type, contributing a model may require an author or maintainer to undergo varying levels of scrutiny prior to a model being accepted. Models may also have to adhere to a standard before being accepted. The contribution workflow is modeled in Figure 3.

Open model hubs: Contributions are freely accepted in open model hubs. Both the uploader of a model and the model itself do not have to undergo any scrutiny or review prior to acceptance. However, user access management systems can be in place to prevent users from changing others' models without permission. In addition, these models are freely distributed to reusers.

Gated model hubs: These model hubs require models to adhere to hub specific criteria before acceptance. While the uploader is not scrutinized, the submitted PTM model must be verified by hub staff members prior to acceptance to ensure that it aligns with the hubs goals. Should the repository be approved of, it is accepted into the hub and made available for reuse.

Commercial model hubs: This is a model hub where contributions from the community are not accepted or made public. It is similar to a *gated* model hub in that models must adhere to specific criteria, but differs in that the uploader of the model must be a staff member of the model hub itself.

5.4 Model Distribution Workflow

While model hubs can be differentiated by type, we found that they distribute models similarly. Models are distributed through either client libraries or via repositories typically hosted on a cloud version control system (VCS) such as GitHub. A visualization of the distribution workflow is modeled in Figure 3.

Client libraries refer to a precompiled package that acts as an interface into a model hubs library of models. These allow for reusers to download and implement models in a standardized manner that adheres to the model hubs guidelines. They also provide additional security and verification features to ensure that the model and reuser are not malicious, indicated in §6. Examples of client libraries include TensorFlow Hub [28, 75], PyTorch Hub [62], and HuggingFace Transformers [19, 82].

As mentioned prior, model hubs can distribute models through cloud VCSs, specifically *GitHub*. In doing so, they are relying upon the end user's technical prowess to implement a model. In addition, they are relying on the cloud VCSs security and verification features to ensure that a model is what it seems to be once a reusers downloads it. Model hubs that utilize cloud VCSs include ONNX Model Zoo [57] and Model Zoo [41].

5.5 Model Versioning

Model hubs host multiple versions of the same model. This is useful for re-users as they can select a specific version to meet their needs. Additionally, depending on how models are released to the hubs, it provides users with additional security via versioning. However, open, gated, and commercial model hubs implement different model release schemes, even within the same classification.

Open model hubs (*i.e.,* Hugging Face) version their models by commit[5]. In other words, a release tag does not have to be associated with a model to be released on the hub. Gated model hubs (*i.e.,* Pytorch Hub, Tensorflow Hub) implement different model release schemes. Pytorch Hub implements a system similar to Hugging Face[6], whereas Tensorflow Hub requires new documentation to be published describing how to checkout a model prior to releasing a new model[7]. Additionally, model updates need to be approved for the Tensorflow Hub prior to release. Commercial model hubs (*i.e.,* MATLAB Model Hub) utilize develop, extend, or optimize existing PTMs to fit their hub's niche(s).

6 RQ2: What practices are in place to improve security among users of the model hubs?

Finding 2: Open model hubs are vulnerable due to its open nature and thus multiple security features have been applied. Many of the vulnerabilities of an open model hub are mitigated in gated/commercial model hubs due to controlled access.

To better understand the security features, we studied different model hubs separately. We will discuss the security features in open model hubs in §6.1 and gated/commercial model hubs in §6.2.

6.1 Security Feature in Open Model Hubs

Hugging Face is the only example of an open model hub that we identified, so we complete a case study of its ecosystem. Hugging Face provides model owners and users several security defense mechanisms in order to prevent malicious maintainers from corrupting their models. These defenses include: organization user permission handling, organization verification, automatic malware

[5]See https://huggingface.co/docs/transformers/custom_models#using-a-model-with-custom-code.
[6]See https://pytorch.org/docs/stable/hub.html#torch.hub.load.
[7]See https://www.tensorflow.org/hub/writing_documentation.

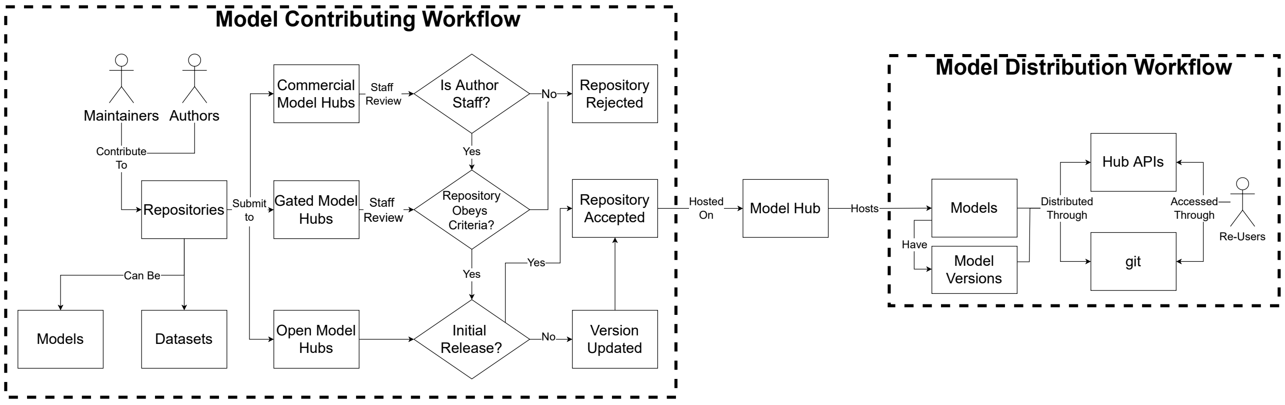

Figure 3: PTM supply chain of *open, commercial,* and *gated* model hubs.

scanning, and commit signing. However, each of these defenses present potential risks and vulnerabilities for the ecosystem's users.

6.1.1 ***Permission Model*** Hugging Face has two types of accounts:

- *Personal account*: an account owned by a single user and stores repositories developed and managed by that user.
- *Organizational account*: an account groups personal user accounts which manage shared repositories.

The different user roles in an organizational account are: *admins*, *readers*, and *writers*. The *admin* role is the most permissive role; with it, a user can view an account's private repositories, and change both repository and account settings. The *reader* role allows for an account to view private and public repositories of an account. However, they cannot directly contribute to the repository. Instead, they need to open a pull request to submit repository changes. The *writer* role is similar to the *admin* role, with the exception that it cannot change account settings. User permissions for organizational accounts are assigned globally so that users will have the same permissions to all the repositories hosted under that organization. The only user role defined in personal account is the *author*, which is the user of that personal account.

6.1.2 ***Organization verification*** This is an optional feature that organizations can opt into to provide a verification badge on their accounts. The purpose of verification is to prevent other organizations from squatting on well known companies or organizations. Furthermore, by verifying the authenticity of an organization, users of can hold an organization liable for malicious models.

To become verified, an organization submits its domain name to Hugging Face. From there, Hugging Face manually confirms that the domain name originates from the organization. Assuming proper verification, Hugging Face then assigns a *verified* badge next to an organization's name on their page. However, as this is an optional feature, many organizations have not opted into it. Nonetheless, Hugging Face provides other markers that could provide organization legitimacy. An organization can provide a link to a website and GitHub account without having to go through Hugging Face manual verification.

6.1.3 ***Commit Signing*** Hugging Face implements an optional verification feature for commits, called *commit signing*. Commit signing involves configuring both `git` and a Hugging Face profile to use a `GPG`, validating that a commit originated from a specific individual on a specific machine. These commits are marked as *verified* in a repositories history when viewed through the `Web UI`, and are marked as *signed* when viewed through the `git` interface. In addition to commit signing, users of the `Hub Client Library` and `Transformers` library can specify a particular revision of a file or PTM via a commit hash when downloading. By doing so, a user can have a consistent version of a PTM is used in development. When combined with a *signed commit*, users of a model have the ability to choose a verified commit to checkout from.

6.2 Security Features in Gated/Commercial Model Hubs

Due to the controlled access of gated/commercial model hubs, we could not study their practical security features. Therefore, we discuss the accessibility-security tradeoff they offer. Commercial and gated model hubs decrease the availability of its models to potential users. However, by the same token they mitigate many of the risks of an open model hub. Since commercial and gated model hubs restrict access to a trusted set of users. there is less risk that a change to a model is malicious. Conversely, since the set of contributors are limited, so is the potential for innovation. These tradeoffs are analogous to those of app stores, *e.g.,* Apple offers a gated store and Android has an open one.

7 RQ3: What are the potential threats in model hubs?

Finding 3: *Insider* and *outsider* attackers can both exist in the PTM supply chain, but perform different attacks. We proved the possibility of potential risks by analyzing the model discrepancies and repositories per maintainer graph.

Based on our analysis of the artifacts (§5) and security features (§6), we mapped the attacker profiles of existing threats to the PTM supply chain in §7.1. We also pointed out the potential risks by analyzing the model discrepancies and maintainers' reach.

7.1 Threat Models

To understand more about how a PTM supply chain could be potentially attacked, we should not only study the existing practices, but also the attacker profiles. Here, we outline two different attacker profiles: *insider* and *outsider* threats.

An *insider* attacker profile is one that exists within the model hub organization themselves. In the case of an open model hub, this would be a rogue employee or someone with access to hub's servers. With such permissions, the attacker could directly change files on the servers, altering model checkpoints or datasets. Since many open model hubs do not maintain a proper file checksum verification, i.e. making sure that the file downloaded is the same as the file uploaded, a malicious *insider* could easily manipulate model checkpoints. On the other hand, for other commercial and gated model hubs, an *insider* would be maintainers with access to editing model repositories and the hub itself. All three types of model hubs open the gate for *weight poisoning* attacks, *BadNets*, and *data manipulation* when an *insider* is in play.

Outsider attacker profiles are derived from the model hub users. In an open model hub, maintainers of model repositories are considered *outsiders*, whereas for other types of model hubs, outsiders are those who do not have access to the ecosystem. For an open hub, this profile can cause damage by maliciously changing data within specific artifacts to affect users and the ecosystem. This can be through the previously discussed threat models, or it can be through another method, such as typosquatting [78]. For commercial and gated model hubs, the definition of an *outsider* attacker is a bit different, and instead describes a user who do not have more than a "read-only" access to the models. An outsider in a gated or commercial model hub can perform a model stealing attack, where they could rebuild the data and model from the weights itself [67].

7.2 Potential Risks

We also conducted quantitative measurements on the risks in model hubs to identify the vulnerabilities in the PTM supply chain. We measured the existence of discrepancies of PTMs from three tasks, and depicted the effects of maintainers. Our results indicate the potential risks to the PTM supply chain.

Model discrepancies: Documented and actual performance of PTMs may be different when users run the models. These differences are described as *model discrepancies* and can be caused by either insider or outsider attackers. Attackers can easily modify a model by adding backdoor behavior or side-effects. When a model's documentation is thorough and accurate, model discrepancies are more likely to indicate potential PTM attacks. However, the lack of proper documentation prevents users from adequately assessing the models they use. We identified the existence of inaccurate documentation in Hugging Face. This could decrease users' awareness of potential risks, allowing attackers to substitute EvilModels or BadNets for the original models.

In this measurement, 8/53 object detection models, 4/26 image classification models, and 136/160 sentiment analysis models made claims we could validate. We were unable to reproduce the documented performances of a large number of PTMs, as shown in Figure 4. Some of the models we looked at had a difference of more than 5% accuracy, even when they were from major technology companies (*i.e.,* Facebook/Meta). Our result shows the presence of performance discrepancies which could either decrease the users' awareness or the detectability of PTM attacks.

Maintainers' reach: Maintainers are one of the most significant components in the PTM supply chain (Figure 3). The maintainers' misbehavior can become an insider attack and do harm to PTM users. Similar to the npm ecosystem [88], maintainer accounts with access to a large number of PTMs increase risk in the model hub ecosystem. As discussed in §2.3, a graph of the maintainers' reach can help us understand their influence on the ecosystem. We depict the correlation between maintainers and PTMs in Figure 5 by looking at the commit histories from all Hugging Face PTMs.

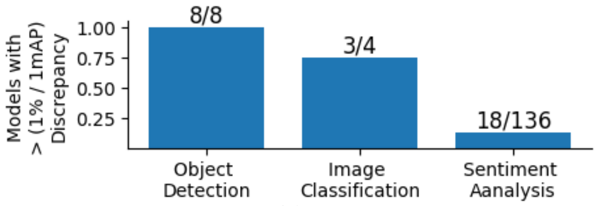

Figure 4: Discrepancies for (a) mAP in 8 object detection (b) top-1 accuracy in 4 image classification (c) accuracy in 136 sentiment analysis PTMs.

Figure 5 demonstrates how many Hugging Face repositories each maintainer has access to. Within the Hugging Face ecosystem, a small number of maintainers have access to a disproportionate number of repositories. These maintainers are hazardous to the robustness of the PTM supply chain — by compromising one of these accounts, attackers could influence hundreds of models and thus do harm to the PTM supply chain.

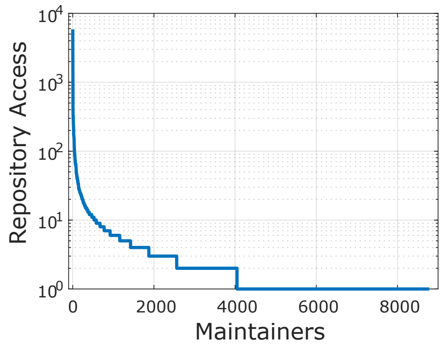

Figure 5: Maintainers' reach in the Hugging Face model hub.

8 Discussion

8.1 Comparison to Traditional Software Supply Chain

Comparing to traditional software supply chain, the PTM supply chain has different characteristics, in terms of the nature of software versioning and security risks.

The requirements for time and computational resources in the PTM training mean it is hard to regularly update the PTM versions. The time to release traditional software versions is much less since test cases can be written to validate the changes to the software. Furthermore, updates to traditional software can more closely follow semantic versioning practices since patch, minor, and major changes can be quantified easily. Based on §2.1 and §5, we notice that while PTMs can follow semantic versioning, it is more difficult to quantify the change because differences between the training datasets, methodologies, or code improvements can dramatically impact the quality of the resulting model [68]. Moreover, due to the obfuscated nature of PTMs, it is difficult to identify security issues within them [31, 79].

PTM attacks are highly variable and have significant impacts since they are deployed in safety-critical systems, such as autonomous vehicles [25]. PTM software have many important components like dataset, weights and configurations which can be attacked in different ways, including poisoning [26, 45] and backdoors [48**?**]. Moreover, the manifestation of PTM attacks can be different from traditional attacks, as shown in §2.4 and Figure 4. Therefore, the traditional scanning tools are incomplete in the PTM context. Figure 5 shows similar results as npm ecosystem [88]. It shows that the traditional attacks happened in traditional software supply chain can also happen in the PTM supply chain, such as account takeover and collusion attack. Moreover, the complexity of PTM components may make it harder to detect the threats in the PTM supply chain.

Two main differences between PTM supply chain and traditional software supply chain are versioning and security properties. The traditional software supply chain allows traceability between versions to verify the integrity and security of the software. Due to the obfuscated nature of PTMs, the PTM supply chain struggles with these validations. For security properties, PTM attacks are harder to detect and they can significantly affect the applications.

8.2 Implications

Based on our results and analysis, we highlight future directions for empirical study and automated tools.

8.2.1 ***Empirical Study***

We identified the PTM supply chain contribution and distribution workflows, and analyzed the security features. Our results indicates the insufficiency of security defenses on PTM supply chain. However, the security characteristics of PTM supply chain remain under-discovered. We call for expanding our knowledge of traditional supply chain management [17, 83] to encompass DL software supply chains. Researchers have analyzed dependencies, maintainers, and security issues in traditional software packages (*e.g.,* NPM, Pypi, RubyGems) [11, 13, 47, 56, 85, 88]. Model hub ecosystems need similar studies.

Dependencies: The dependencies of PTMs include model structures, pre-trained weights, and datasets. Measurement on each aspect would help the community understand more about the potential threats in model hub ecosystems. Moreover, the potential security risks of a ecosystem also depends on the behavior of maintainers.

Maintainers: Prior works indicated that in NPM ecosystem, few maintainers — who have access to over 3K models — could impact a much larger fraction of the repositories [88]. We hypothesize that the number of PTMs per maintainer (*i.e.,* maintainers' reach [88]) may be more evenly distributed in model hubs due to a greater complexity and cost of developing PTM packages than traditional software packages. If true, both insider and outsider attacks on model hubs may be less significant than on NPM. We suggest more studies to measure maintainers' reach on model hubs and evaluating the risk relative to that on traditional software supply chains. We also note that researchers play an unusually prominent role in model hubs compared to traditional software package registries; this may diminish the engineering quality of their packages.

8.2.2 ***Automated Tool***

Our analysis of threat models indicate that Model audit and automated model scanning could help detect the security issues in the PTM supply chain. However, there have not been any comprehensive tools developed for these issues.

Model audit: Checking of different PTM behavior (*i.e.,* model audit) is a necessary approach to validate existing PTMs. Prior works have already indicated that adversarial attacks could affect the accuracy of PTMs [1, 7]. Montes *et al.* shows preliminary results of non-trivial discrepancies existing among different model hubs [53]. Their results reveal the problems on the reliability of PTM supply chain. Hamon *et al.* proposed the three levels of AI transparency, including implementation, specifications, and interpretability [33]. Recently, Hugging Face released an auto-evaluator to audit model performance [20]. However, the tool only supports pre-defined metrics focused on accuracy. Even were the tool widely used, the third level (interpretability) of PTMs would remain under-audited. We propose future works on automated audit tools for the measurement of interpretability of PTMs.

Model scanning: In addition to model audits, specific integrated scanning tools for PTMs are needed for improving the security of model hubs. *ClamAV* is an existing practice in Hugging Face, but it originated in traditional software [8] and could not detect the DL-specific threat models. It has been proved that EvilModels can still be uploaded to Hugging Face. For example, arbitrary code could be executed when loading a model from Hugging Face[8] and there is no warning about this misbehavior. As a result, we suggest future works on more specific scanning tools for attacks on PTMs (*i.e.,* weights, datasets, configurations).

9 Conclusion

To have a better understanding of the role of model hubs in the secure DL supply chain, we examined 8 model hubs, proposed three model hub types, and summarized the *PTM supply chain.* Furthermore, we did the first study on model hubs and show that the PTM supply chain is an essential part of DL software supply chain. We also reviewed existing defenses and analyzed potential threat models. Our analysis shows that the existing defenses are insufficient for ensuring the security of PTMs distributed through model hubs since state-of-the-art attacks can be performed without detection and have effects on downstream models. We inform future research on model supply chains and suggest that a model audit and further empirical studies are needed to fully address the security vulnerabilities on model hubs.

Data availability

Our data is available at https://github.com/Wenxin-Jiang/SCORED22-PTMSupplyChain. Within it, we provide the software of measurements and the quantitative data we analyzed in §7.2.

Acknowledgments

This work was supported by gifts from Google and Cisco and by NSF awards #2107230, #2229703, #2107020, and #2104319. We thank the reviewers and our shepherd, Nicholas Boucher, for their careful reading of our paper and insightful comments.

[8]See https://huggingface.co/ykilcher/totally-harmless-model.

References

[1] Naveed Akhtar and Ajmal Mian. 2018. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access* 6 (2018), 14410–14430.

[2] Vishnu Banna, Akhil Chinnakotla, Zhengxin Yan, Anirudh Vegesana, Naveen Vivek, Kruthi Krishnappa, Wenxin Jiang, Yung-Hsiang Lu, George K. Thiruvathukal, and James C. Davis. 2021. An Experience Report on Machine Learning Reproducibility: Guidance for Practitioners and TensorFlow Model Garden Contributors. http://arxiv.org/abs/2107.00821

[3] Adrien Bibal and Benoît Frénay. 2016. Interpretability of Machine Learning Models and Representations: an Introduction. In *European Symposium on Artificial Neural Networks.*

[4] Jon M. Boyens, Celia Paulsen, Rama Moorthy, and Nadya Bartol. 2015. *Supply Chain Risk Management Practices for Federal Information Systems and Organizations.* Technical Report NIST SP 800-161. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-161

[5] Houssem Ben Braiek and Foutse Khomh. 2020. On testing machine learning programs. *Journal of Systems and Software (JSS)* 164 (2020), 110542.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. *Language Models are Few-Shot Learners.* Technical Report arXiv:2005.14165. arXiv. http://arxiv.org/abs/2005.14165

[7] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial Attacks and Defences: A Survey. https://arxiv.org/abs/1810.00069

[8] Cisco. 2022. ClamAV. https://www.clamav.net/

[9] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. 2018. Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine* 35 (2018), 53–65. https://doi.org/10.1109/MSP.2017.2765202

[10] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting Pre-Trained Models for Chinese Natural Language Processing. http://arxiv.org/abs/2004.13922

[11] James C Davis, Christy A Coghlan, Francisco Servant, and Dongyoon Lee. 2018. The impact of regular expression denial of service (ReDoS) in practice: an empirical study at the ecosystem scale. In *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering (ESEC/FSE).* 246–256. https://dl.acm.org/doi/10.1145/3236024.3236027

[12] Gabriel V. de la Cruz, Yunshu Du, and Matthew E. Taylor. 2019. Pre-training with Non-expert Human Demonstration for Deep Reinforcement Learning. *The Knowledge Engineering Review* 34 (2019). https://doi.org/10.1017/S0269888919000055

[13] Alexandre Decan, Tom Mens, and Eleni Constantinou. 2018. On the impact of security vulnerabilities in the npm package dependency network. In *International Conference on Mining Software Repositories (MSR).* 181–191. https://doi.org/10.1145/3196398.3196401

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* Technical Report arXiv:1810.04805. arXiv. http://arxiv.org/abs/1810.04805

[15] Finale Doshi-Velez and Been Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. https://arxiv.org/abs/1702.08608

[16] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS).* Association for Computing Machinery, Dallas, Texas, USA, 1285–1298. https://doi.org/10.1145/3133956.3134015

[17] Shixian Du, Tianbo Lu, Lingling Zhao, Bing Xu, Xiaobo Guo, and Hongyu Yang. 2013. Towards An Analysis of Software Supply Chain Risk Management. In *Proceedings of the World Congress on Engineering and Computer Science*, Vol. 1.

[18] Parijat Dube, Bishwaranjan Bhattacharjee, Siyu Huo, Patrick Watson, and Brian Belgodere. 2019. Automatic Labeling of Data for Transfer Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.* 122–129.

[19] Hugging Face. 2021. Hugging Face – The AI community building the future. https://huggingface.co/

[20] Hugging Face. 2022. Announcing Evaluation on the Hub. https://huggingface.co/blog/eval-on-the-hub

[21] Hugging Face. 2022. Hugging Face Hub documentation. https://huggingface.co/docs/hub/index

[22] Jon Fingas. 2022. AI trained on 4chan's most hateful board is just as toxic as you'd expect. https://www.engadget.com/ai-bot-4chan-hate-machine-162550734.html

[23] Shannon Flynn. 2020. Artificial Intelligence Bias Affects Everyone — Even You. https://rehack.com/iot/artificial-intelligence-bias/

[24] Hironobu Fujiyoshi, Tsubasa Hirakawa, and Takayoshi Yamashita. 2019. Deep learning-based image recognition for autonomous driving. *IATSS Research* 43 (2019), 244–252.

[25] Joshua Garcia, Yang Feng, Junjie Shen, Sumaya Almanee, Yuan Xia, and and Qi Alfred Chen. 2020. A comprehensive study of autonomous vehicle bugs. In *International Conference on Software Engineering (ICSE).* IEEE, Seoul, Korea (South), 385–396. https://dl.acm.org/doi/10.1145/3377811.3380397

[26] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. 2022. Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[27] Nikhil Krishna Gopalakrishna, Dharun Anandayuvaraj, Annan Detti, Forrest Lee Bland, Sazzadur Rahaman, and James C Davis. 2022. "If security is required": Engineering and Security Practices for Machine Learning-based IoT Devices. In *International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT).*

[28] Josh Gordon. 2018. Introducing TensorFlow Hub: A Library for Reusable Machine Learning Modules in TensorFlow. https://blog.tensorflow.org/2018/03/introducing-tensorflow-hub-library.html.

[29] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. , 1789–1819 pages.

[30] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2018. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. https://arxiv.org/abs/1706.02677

[31] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. https://doi.org/10.48550/arXiv.1708.06733

[32] Qianyu Guo, Sen Chen, Xiaofei Xie, Lei Ma, Qiang Hu, Hongtao Liu, Yang Liu, Jianjun Zhao, and Xiaohong Li. 2019. An empirical study towards characterizing deep learning development and deployment across different frameworks and platforms. In *IEEE/ACM International Conference on Automated Software Engineering (ASE).* IEEE, 810–822. https://doi.org/10.1109/ASE.2019.00080

[33] Ronan Hamon, Henrik Junklewitz, and Ignacio Sanchez. 2020. Robustness and explainability of Artificial Intelligence: from technical to policy solutions. *Publications Office of the European Union* (2020).

[34] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. 2021. Pre-trained models: Past, present and future. *AI Open* 2 (2021), 225–250.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR).* IEE, 770–778. https://doi.org/10.1109/CVPR.2016.90

[36] Cathal Horan. 2020. Can GPT-3 or BERT Ever Understand Language?—The Limits of Deep Learning Language Models. https://neptune.ai/blog/gpt-3-bert-limits-of-deep-learning-language-models

[37] Ahmed Hosny, Michael Schwier, Christoph Berger, Evin P. Örnek, Mehmet Turan, Phi V. Tran, Leon Weninger, Fabian Isensee, Klaus H. Maier-Hein, Richard McKinley, Michael T. Lu, Udo Hoffmann, Bjoern Menze, Spyridon Bakas, Andriy Fedorov, and Hugo JWL Aerts. 2019. ModelHub.AI: Dissemination Platform for Deep Learning Models. http://arxiv.org/abs/1911.13218

[38] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin I P Rubinstein, and J D Tygar. 2011. Adversarial Machine Learning. In *ACM workshop on Security and Artificial Intelligence.* IEEE, 43–58. https://doi.org/10.1145/2046684.2046692

[39] Slinger Jansen and Ewoud Bloemendal. 2013. Defining app stores: The role of curated marketplaces in software ecosystems. In *International conference of software business.* Springer, Berlin, Heidelberg, 195–206. http://dx.doi.org/10.1007/978-3-642-39336-5_19

[40] A. Jeyanthi Suresh and J. Visumathi. 2020. Inception ResNet deep transfer learning model for human action recognition using LSTM. *Materials Today: Proceedings* (2020).

[41] Yu Koh Jing. 2021. Model Zoo - Deep learning code and pretrained models. https://modelzoo.co/

[42] Andrew Khalel and Motaz El-Saban. 2018. Automatic Pixelwise Object Labeling for Aerial Imagery Using Stacked U-Nets. http://arxiv.org/abs/1803.04953

[43] Yannic Kilcher. 2022. Totally Harmless Model. https://huggingface.co/ykilcher/totally-harmless-model

[44] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 6. 84–90.

[45] Keita Kurita, Paul Michel, and Graham Neubig. 2020. *Weight Poisoning Attacks on Pre-trained Models.* Technical Report. arXiv. http://arxiv.org/abs/2004.06660

[46] Computational Imaging and Bioinformatics Lab. 2022. Modelhub. http://modelhub.ai/

[47] Piergiorgio Ladisa, Henrik Plate, Matias Martinez, and Olivier Barais. 2022. Taxonomy of Attacks on Open-Source Software Supply Chains. http://arxiv.org/abs/2204.04008

[48] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. In *Network and Distributed Systems Security (NDSS) Symposium.*

[49] Zeyan Liu, Fengjun Li, Zhu Li, and Bo Luo. 2022. LoneNeuron: a Highly-Effective Feature-Domain Neural Trojan Using Invisible and Polymorphic Watermarks. In *ACM SIGSAC Conference on Computer and Communications Security.* ACM, Los Angeles.

[50] Konstantinos Manikas and Klaus Marius Hansen. 2013. Software ecosystems – A systematic literature review. *Journal of Systems and Software (JSS)* 86 (2013), 1294–1306.

[51] Pedro Marcelino. 2022. Transfer learning from pre-trained models. https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

[52] MathWorks. 2022. MATLAB Deep Learning Model Hub. https://www.mathworks.com/solutions/deep-learning.html

[53] Diego Montes, Pongpatapee Peerapatanapokin, Jeff Schultz, Chengjun Guo, Wenxin Jiang, and James C Davis. 2022. Discrepancies among pre-trained deep neural networks: a new threat to model zoo reliability. In *European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE-IVR track).* ACM, Singapore. https://doi.org/10.1145/3540250.3560881

[54] NPM. 2022. npm. https://www.npmjs.com/

[55] NVIDIA. 2022. NVIDIA NGC: AI Development Catalog. https://catalog.ngc.nvidia.com/

[56] Marc Ohm, Henrik Plate, Arnold Sykosch, and Michael Meier. 2020. Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, Clémentine Maurice, Leyla Bilge, Gianluca Stringhini, and Nuno Neves (Eds.). Springer, 23–43.

[57] ONNX. 2022. *ONNX Model Zoo.* ONNX. https://github.com/onnx/models

[58] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 3 (2010), 1–40. https://doi.org/10.1109/TKDE.2009.191

[59] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon Emissions and Large Neural Network Training. https://doi.org/10.48550/arXiv.2104.10350

[60] Hung Viet Pham. 2020. Problems and Opportunities in Training Deep Learning Software Systems: An Analysis of Variance. In *International Conference on Automated Software Engineering (ASE).* 771–783. https://doi.org/10.1145/3324884.3416545

[61] PyPI. 2022. Python Package Index. https://pypi.org

[62] Pytorch. 2022. PyTorch Hub. https://pytorch.org/hub/

[63] XiPeng Qiu, TianXiang Sun, YiGe Xu, YunFan Shao, Ning Dai, and XuanJing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences* 63 (2020), 1872–1897.

[64] A. Sai Bharadwaj Reddy and D. Sujitha Juliet. 2019. Transfer Learning with ResNet-50 for Malaria Cell-Image Classification. In *International Conference on Communication and Signal Processing (ICCSP).* 0945–0949.

[65] Edmar Rezende, Guilherme Ruppert, Tiago Carvalho, Fabio Ramos, and Paulo de Geus. 2017. Malicious Software Classification Using Transfer Learning of ResNet-50 Deep Neural Network. In *International Conference on Machine Learning and Applications (ICMLA).* 1011–1014.

[66] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. 2020. Hidden Trigger Backdoor Attacks. *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (2020), 11957–11965. https://doi.org/10.1609/aaai.v34i07.6871

[67] Sunandini Sanyal, Sravanti Addepalli, and R Venkatesh Babu. 2022. Towards Data-Free Model Stealing in a Hard Label Setting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 15284–15293.

[68] Sebastian Schelter, Felix Biessmann, Tim Januschowski, David Salinas, Stephan Seufert, and Gyuri Szarvas. 2018. On Challenges in Machine Learning Model Management. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* (2018).

[69] John Seymour and Philip Tully. 2016. Weaponizing data science for social engineering: Automated E2E spear phishing on Twitter. *Black Hat USA* (2016).

[70] Connor Shorten and Taghi M. Khoshgoftaar. 2019. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data* (2019).

[71] Jonah Sokipriala. 2021. Prediction of Steering Angle for Autonomous Vehicles Using Pre-Trained Neural Network. *European Journal of Engineering and Technology Research* (2021).

[72] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. 2018. A Survey on Deep Transfer Learning. *IEEE Transactions on knowledge and data engineering* IEEE Transactions on knowledge and data engineering (2018). http://arxiv.org/abs/1808.01974

[73] Xin Tan, Kai Gao, Minghui Zhou, and Li Zhang. 2022. An exploratory study of deep learning supply chain. In *International Conference on Software Engineering (ICSE).* Pittsburgh Pennsylvania.

[74] Rachael Tatman, Jake Vanderplas, and Sohier Dane. 2018. A Practical Taxonomy of Reproducibility for Machine Learning Research. In *Reproducibility in Machine Learning Workshop at ICML.*

[75] TensorFlow. 2022. TensorFlow Hub. https://www.tensorflow.org/hub

[76] George K Thiruvathukal, Yung-Hsiang Lu, Jaeyoun Kim, Yiran Chen, and Bo Chen. 2022. *Low-power Computer Vision: Improve the Efficiency of Artificial Intelligence.*

[77] Sebastian Thrun and Lorien Pratt. 1998. Learning to learn: Introduction and overview. In *Learning to learn.* Springer.

[78] Nikolai Philipp Tschacher. 2016. *Typosquatting in programming language package managers.* Ph. D. Dissertation. Universität Hamburg, Fachbereich Informatik.

[79] Zhi Wang, Chaoge Liu, and Xiang Cui. 2021. Evilmodel: hiding malware inside of neural network models. , 7 pages.

[80] Zhi Wang, Chaoge Liu, Xiang Cui, Jie Yin, and Xutong Wang. 2022. EvilModel 2.0: Bringing Neural Network Models into Malware Attacks. *Computers & Security* (2022). https://doi.org/10.1016/j.cose.2022.102807

[81] Jeannette M. Wing. 2021. Trustworthy AI. *Commun. ACM* (2021).

[82] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Conference on Empirical Methods in Natural Language Processing: System Demonstrations.*

[83] Lei Xu, Lin Chen, Zhimin Gao, Yang Lu, and Weidong Shi. 2017. CoC: Secure Supply Chain Management System Based on Public Ledger. In *International Conference on Computer Communication and Networks (ICCCN).*

[84] Mu Yuan, Lan Zhang, Xiang-Yang Li, and Hui Xiong. 2020. Comprehensive and efficient data labeling via adaptive model scheduling. (2020), 1858–1861.

[85] Nusrat Zahan, Tom Zimmermann, Patrice Godefroid, Brendan Murphy, Chandra Maddila, and Laurie Williams. 2022. What are Weak Links in the npm Supply Chain?. In *ICSE 2022.* https://www.microsoft.com/en-us/research/publication/what-are-weak-links-in-the-npm-supply-chain/

[86] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models. *arXiv* (2022). http://arxiv.org/abs/2205.01068

[87] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2020. A Comprehensive Survey on Transfer Learning. arXiv:1911.02685 https://arxiv.org/abs/1911.02685

[88] Markus Zimmermann, Cristian-Alexandru Staicu, and Michael Pradel. 2019. Small World with High Risks: A Study of Security Threats in the npm Ecosystem. In *USENIX Security Symposium.* https://doi.org/10.5555/3361338.3361407