An Empirical Study of Pre-Trained Model Reuse in the Hugging Face Deep Learning Model Registry

Wenxin Jiang*, Nicholas Synovic†, Matt Hyatt†, Taylor R. Schorlemmer*, Rohan Sethi†,
Yung-Hsiang Lu*, George K. Thiruvathukal†, James C. Davis*
*Purdue University and †Loyola University Chicago

2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) | 978-1-6654-5701-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSE48619.2023.00206

***Abstract*— Deep Neural Networks (DNNs) are being adopted as components in software systems. Creating and specializing DNNs from scratch has grown increasingly difficult as state-of-the-art architectures grow more complex. Following the path of traditional software engineering, machine learning engineers have begun to reuse large-scale pre-trained models (PTMs) and fine-tune these models for downstream tasks. Prior works have studied reuse practices for traditional software packages to guide software engineers towards better package maintenance and dependency management. We lack a similar foundation of knowledge to guide behaviors in pre-trained model ecosystems.**

In this work, we present the first empirical investigation of PTM reuse. We interviewed 12 practitioners from the most popular PTM ecosystem, Hugging Face, to learn the practices and challenges of PTM reuse. From this data, we model the decision-making process for PTM reuse. Based on the identified practices, we describe useful attributes for model reuse, including provenance, reproducibility, and portability. Three challenges for PTM reuse are missing attributes, discrepancies between claimed and actual performance, and model risks. We substantiate these identified challenges with systematic measurements in the Hugging Face ecosystem. Our work informs future directions on optimizing deep learning ecosystems by automated measuring useful attributes and potential attacks, and envision future research on infrastructure and standardization for model registries.

***Index Terms*—Software reuse, Empirical software engineering, Machine learning, Deep learning, Software supply chain, Engineering decision making, Cybersecurity, Trust**

I. Introduction

Package reuse has transformed software engineering in programming languages such as JavaScript and Python [1, 2], and is transforming deep learning model engineering [3]. Deep Neural Networks (DNNs) are widely used in modern software systems, such as image recognition in autonomous vehicles [4]. Engineering a DNN is challenging due to the capital and operating expenses of training models [5] and variation in deep learning libraries [6]. These problems can be addressed by reusing *pre-trained DNN models* (PTMs) to amortize DNN development costs across multiple projects and organizations [7]. PTMs are shared via *Deep Learning (DL) model registries*, which are modeled on traditional software package registries such as NPM and provide packages with model architectures, weights, licenses, and other metadata. DL model registries enable reuse-driven DNN engineering [8, 9] As Figure 1 shows, PTM reuse is now appreciable [7].

Reusability and trustworthiness problems in software package registries impact the relevant ecosystems [10–12]. Much is known about the practices and challenges of reusing traditional software packages [13–15], but how this knowledge transfers to the reuse of PTM packages has not been investigated. Existing software engineering knowledge describes how large companies manage private models [16, 17]. However, we do not know how small-to-large engineering teams reuse models in DL model registries nor what challenges they experience.

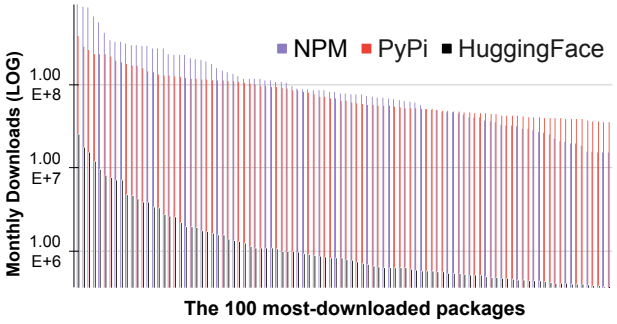

Fig. 1. Package download rates in two software package registries, NPM and PyPi, and the leading DL model registry, Hugging Face. Many Hugging Face model packages have high download rates, though with a rapid drop-off.

In this work, we present the first empirical study of pre-trained model reuse. We took a mixed-methods approach to identify diverse phenomena for future investigation [18]. We focused our study on the Hugging Face DL model registry, which is the largest PTM registry at present [19]. First, we interviewed 12 Hugging Face practitioners to understand the practices and challenges of PTM reuse. Second, we complemented this qualitative data with measurements of the Hugging Face registry. We built a dataflow model of the creation and distribution process for PTMs in Hugging Face and collected and analyzed a dataset of 63,182 PTM packages.

Our findings indicate that PTM reuse workflows are similar to those for traditional software package reuse, but that engineers follow practices and experience challenges specific to deep learning. Based on our interview data, we: present the first decision-making workflow for PTM reuse, identify useful PTM attributes and three common challenges, and discuss the extent to which existing techniques meet these challenges (§V—§VII). Our dataflow model of PTM distribution found several vectors for software supply chain attacks (§VIII). Our analysis shows unique properties of the PTM package ecosystem relative to traditional software package ecosystems,

1558-1225/23/$31.00 ©2023 IEEE
DOI 10.1109/ICSE48619.2023.00206

notably that the attributes and decision-making workflow are more complex (§X). We share the *HFTorrent* dataset of 63,182 PTM package histories for further analysis (§IX). We conclude by discussing four new research problems for further study (§X). Our contributions are:

- We depict a decision-making workflow for PTM reuse, and identify three challenges for PTM reuse (§V—§VII).
- We measure the risks of collaboration in Hugging Face. We identify several potential software supply chain concerns facing PTM reusers (§VIII).
- We publish the *HFTorrent* dataset of 63,182 PTM packages for future analysis (§IX).
- We identified unique properties of PTM package reuse to guide future research on model audit, infrastructure, standardization, and attack detection (§X).

Significance: PTM reuse reduces the engineering costs of employing DNNs in industry. This paper describes the first investigation of PTM reuse from a software engineering perspective. We are the first to (1) capture the decision-making workflow and challenges for PTM reuse; (2) determine attributes of PTMs that facilitate reuse; and (3) measure risks of PTM reuse in the Hugging Face DL model registry. Our findings can help PTM maintainers and registries improve the quality of their offerings, and show opportunities for software engineering tools to support PTM reusers in this process.

II. Background and Related Work

A. Software Package Reuse

Software package registries store versioned packaged software, associated metadata, documentation, and configurations [20]. Similarly, deep learning (DL) model registries distribute PTMs with metadata, a model card (*i.e.,* documentation), relevant configurations, and versions of pre-trained weights [21]. DL model registries are an important component of the DL ecosystem [22]. As shown in Figure 2, PTM packages may contain more component than traditional packages, including weights, datasets, and performance metrics.

Evaluating and selecting software packages is a difficult, but essential, activity for package reuse [13]. Prior work shows that engineers may improve their software selection with insights into the decision-making process and an understanding of relevant factors [13, 23, 24]. Existing literature focuses on practices in traditional software package registries, such as NPM [25] and Maven [26, 27]. The extent to which reuse practices for traditional software will transfer to the reuse of PTM packages is unclear.

Reproducibility is another important aspect of software package reuse [28]. In traditional software packages, Goswami *et al.* found that 38% of explored NPM package versions are non-reproducible [29]. Similarly, Vu *et al.* highlighted existing discrepancies at different levels of granularity in PyPi [30]. Following the machine learning scientific research community [31], the software engineering community has just begun to study concerns in DL model registries [32]. We offer an early software engineering view on this topic.

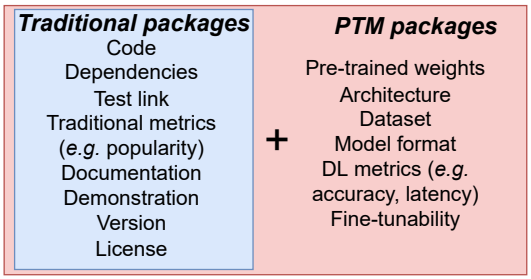

Fig. 2. Components of traditional packages [20] and PTM packages [21]. A PTM package includes all the components of a traditional package, plus some DL-specific parts.

B. Pre-Trained Model Reuse

PTM reuse is necessitated by the emergence of large-scale models, and is enabled by learning and compression techniques, including *transfer learning* [33], *quantization and pruning* [34], *knowledge distillation* [35], and *data labeling* [36]. Through transfer learning, DNNs can be pre-trained on large datasets and fine-tuned to solve specialized tasks, leveraging a PTM's knowledge of one task to better teach it a similar task [33, 37]. Using quantization and pruning methods, PTMs can be optimized for latency- or energy-sensitive contexts, such as on edge devices, without compromising accuracy [34]. Via knowledge distillation, PTMs can be used to teach a smaller model, yielding good performance and reduced computational costs [35]. Engineers can also use PTMs to automatically label datasets [36].

Practitioners from major technology companies report challenges in model management and model reliability [16, 17]. Schelter *et al.* summarized model validation challenges, including decisions on model retraining, metadata querying, and adversarial settings [16]. Rahman *et al.* highlighted that the behavior of ML models can be easily affected because of their data-driven nature [38]. To better reuse the PTMs, it is important to monitor the performance of deployed models, track changes in data characteristics, and to retrain and revalidate them frequently.

One way to address the management problems is to use a **DL model registry**, which is defined as: *a collaborative model hub where teams can share DL models* [21, 39]. The DL model registry concept imitates traditional software package registries such as NPM [40] and PyPi [41]. Through web searches, we identified several prominent DL model registries, including Hugging Face [42], TensorFlow Hub [43], PyTorch Hub [44], and ONNX Model Zoo [45]. Among all registries we examined [42–48], Hugging Face offers the largest and most diverse set of PTMs — it hosts over 60,000 PTMs, fifty times as many as the next largest DL model registry, as well as many types of models and datasets.

C. Deep Learning Trustworthiness

The trustworthiness (*e.g.,* reproducibility, explainability) of DL software grows in importance as DL techniques

are deployed in sensitive contexts such as autonomous vehicles [49, 50]. For example, DL traceability is hampered because authors often omit training logs and documentation [38, 51]. Wing urges the DL community to explore a combination of approaches to achieve trustworthy DL [52]. To improve the trustworthiness of ML systems, prior work recommends considering aspects including provenance, reproducibility, and portability [38, 52–54], as defined in Table I. Some researchers have investigated the performance variances tied to DL frameworks [55, 56], which threatens DL reliability.

Adversarial attacks and defences are also important to DL trustworthiness [57, 58]. Gu *et al.* proposed the general term *BadNet* for models that perform well on benchmark datasets but poorly on attacker-defined inputs [59]. Kurita *et al.* showed that it is possible to construct *BadNets* from weight poisoning attacks by injecting PTM with vulnerabilities that expose backdoors after fine-tuning [60]. Additionally, Goldblum *et al.* discussed that it is also possible to attack a model indirectly via malicious labels in its training dataset (data poisoning) [61]. Wang *et al.* described an *EvilModel* where a PTM has malware bytes hidden inside its neurons' parameters to be extracted and assembled into malware at run-time [62]. These attacks are not all covered by existing malware detection techniques and raise potential risks to DL model registries [63].

III. Research Questions

Summarizing the literature: Much is known about software engineers' practices and challenges in reusing traditional software packages, but little about DL software packages (PTMs). Reuse and trust are unexamined in DL model registries.

We studied the reusability of PTM packages in DL model registries, examining qualitative and quantitative aspects. We focused on one DL model registry, Hugging Face, as it is by far the largest registry at present [19]. For PTM reuse in the Hugging Face ecosystem, we ask:

RQ1 *How do engineers select PTMs?*
RQ2 *What PTM attributes facilitate PTM reuse?*
RQ3 *What are the challenges of PTM reuse?*
RQ4 *To what extent are the risks of reusing PTMs mitigated by Hugging Face defenses?*

RQ1-2 are focused on current software engineering practice, priming the participants to describe their challenges in RQ3. RQ4 complements this data with quantitative measurements.

IV. Methodology

To answer our research questions, we used a mixed approach that combined two perspectives [18]. We first explore qualitative insights by interviewing practitioners, then we substantiate our findings with systematic measurements in Hugging Face ecosystem. The relationship between our questions and methods is shown in Figure 3.

A. Qualitative Study: Interviews with PTM Reusers

Our interview study follows a four-step process modeled on the *framework analysis* methodology [64, 65]:

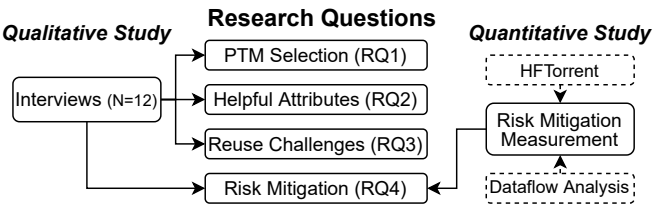

Fig. 3. Relationship of research questions to methodology.

(1) Data Familiarization and Framework Identification Our initial thematic framework is based on three themes from our literature review (§II): model selection, PTM attributes, and PTM trustworthiness. For *model selection*, the identified considerations were the PTM reuse issues and factors affecting the decision-making process [19, 66]. For *attributes*, we saw both traditional attributes (*i.e.,* popularity, quality, maintenance) [67, 68], and DL-specific attributes, viz. provenance, reproducibility, and portability [52–54], shown in the first three columns in Table I. For *trustworthiness*, we considered the aspects assumed trustworthy plus possible discrepancies [19].

(2) Interview Design We designed a semi-structured interview protocol with questions that explore the three identified themes of PTM reuse and trust. We conducted three pilot interviews. We then revised our framework and interview protocol, adding some PTM attributes and factors and clarifying definitions.

The final interview protocol took 30–45 minutes. We compensated interview participants with a $20 gift card. The protocol is available in our artifact (§XII).

(3) Recruitment We recruited users from the Hugging Face ecosystem [42], who presumably have experience in developing and reusing PTMs. According to the Hugging Face website [69] there are 18,348 Hugging Face users, 690 of whom have PRO accounts and 17,658 of whom have regular accounts. We sorted the lists of PRO and regular users by the number of models they have contributed to Hugging Face, and contacted the first 50 users of each type. We interviewed the 12 respondents described in Table II. This was a (response rate of 24%, of whom 9 had PRO accounts. Our participants contributed between 4 and 2500 models to Hugging Face.

(4) Analysis We transcribed the interview recordings. Two researchers performed memoing [70], mapping the transcripts to the pre-defined themes. Each memo had a quote for one of the themes. Multiple researchers analyzed 4 transcripts and had high agreement on the memos extracted for each theme. Agreement was because the pre-defined themes had clear definitions, but we did not measure the agreement precisely. A single researcher memoed the remaining 8 transcripts.

Then we organized the memos in a matrix by theme. Two researchers used the matrix to develop a thorough understanding of the larger picture. Then we answered each RQ by our understanding and reference to the matrix.

As part of our analysis, we measured saturation from our interview transcripts by analyzing the number of cumulative unique codes by participant [71]. Saturation was achieved after

TABLE I
DEFINITION AND EXAMPLES OF DL-SPECIFIC ATTRIBUTES, AND THE RELEVANT FACTORS MENTIONED BY MULTIPLE PARTICIPANTS.

Attribute	Definition	Example	Identified Factors
Provenance	A measure of model lineage or traceability.	Noting the original paper	(1) Dataset details (2) Performance table (3) Architecture details (4) Training logs
Reproducibility	The ability of a DL practitioner to produce the same accuracy and latency from a PTM as defined in its paper, source code, or group.	Providing details of environment configuration	(1) Hardware specification (2) Training configuration (scripts, hyper-parameters) (3) Demos (4) Documentation (5) Environment image
Portability	The ease with which an engineer can take a PTM and reuse it in another environment, software project, or other application domain.	Hardware accelerator information helps engineers know whether a model can run on their devices.	(1) Hardware specification (2) Latency (3) Quantized model (4) Environment image (5) Framework support (6) Fine-tuning instructions (7) License (8) Cost estimation

TABLE II
PARTICIPANT DEMOGRAPHICS. PARTICIPANTS GENERALLY IDENTIFY AS SOFTWARE, ML, OR NLP ENGINEERS, WORK FOR SMALL, MEDIUM, AND LARGE TECHNOLOGY COMPANIES, AND CLAIM INTERMEDIATE OR EXPERT SKILL IN DEEP LEARNING (DL) AND SOFTWARE ENGINEERING (SE). ELEVEN PARTCIPANTS WORK WITH NATURAL LANGUAGE PROCESSING (NLP) MODELS, SIX WITH COMPUTER VISION (CV) MODELS.

ID	Role	Org. size	DL skill	SE skill	Domain
P1	Tech lead	Small	Expert	Interm.	NLP, CV
P2	Tech lead	Small	Interm.	Interm.	NLP
P3	Engineer	Small	Expert	Interm.	CV
P4	Engineer	Medium	Expert	Interm.	NLP
P5	Tech lead	Large	Interm.	Interm.	NLP, CV
P6	Engineer	Small	Interm.	Interm.	NLP, CV
P7	Engineer	Small	Interm.	Expert	NLP
P8	Engineer	Small	Interm.	Interm.	NLP
P9	Data scientist	Large	Interm.	Interm.	NLP
P10	Engineer	Medium	Expert	Expert	NLP, CV
P11	Engineer	Medium	Expert	Expert	NLP, CV
P12	Engineer	Small	Interm.	Interm.	NLP

7 participants so we did not continue to recruit participants.

B. Quantitative Study: Risk Mitigation Measurement

Our qualitative findings identified a variety of challenges and risks in PTM reuse. We measured these risks and mitigations in the Hugging Face ecosystem with the STRIDE methodology for threat modeling and risk assessment [72].[1] STRIDE was proposed by Microsoft as a security analysis technique and is widely used [72–75]. STRIDE focuses on trust assumptions related to data, making it suitable for PTMs.

STRIDE is a two-step process. First, the system under consideration is modeled using a dataflow diagram, and trust boundaries and the actors involved are identified. Second, each boundary is analyzed for the threats of the STRIDE acronym.

Following the STRIDE methodology, we started by developing a dataflow diagram for PTMs on Hugging Face. Two researchers analyzed Hugging Face's public documentation. After internal iteration, we settled on one primary trust boundary: user control vs. Hugging Face internal control. We identified threats and six risk-mitigating features across this boundary. Owing to the nature of the available data source (public documentation), we limited our analysis to a subset of the STRIDE threats: Spoofing, Tampering, Repudiation, and Elevation of Privilege. The completeness of our dataflow diagram was ensured by having two researchers review the documentation. These same researchers checked the soundness of the model by creating and using models on Hugging Face both as individual accounts and organization contributors.

[1]STRIDE is a mnemonic for Spoofing identity, data Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of privilege.

V. RQ1: HOW DO ENGINEERS SELECT PTMS?

Finding 1: The participants share a similar decision-making process (Figure 4). Among the four PTM reuse scenarios in the research literature, our participants reported using only two: transfer learning and quantization techniques. When reusing, participants find PTMs from DL model registries easier to adopt than PTMs from GitHub projects.

A. Reuse scenarios

Most interview participants take PTMs from model registries and apply transfer learning techniques to the model. They either "*fine-tune an existing PTM*" by (optionally) extending architecture and training on a task-specific dataset, or "*build a new model on top of the pre-trained one*". Commonly, they select PTMs from leading technology companies (*e.g.,* Google, Meta) because "*the datasets are carefully cleaned and [the models] are straightforward to fine-tune*".

The other three reuse scenarios discussed in the research literature (§II-B) were far less common in our interviews. *P5* described using quantized models. No participants described using PTMs for knowledge distillation or for data labeling.

B. Decision-making process

To understand how engineers select PTMs, we asked participants to summarize their decision-making processes. We found similarities between participant responses. We followed Michael *et al.* [66] in adapting a general software engineering reuse process [76] to integrate our findings into a unified model (Figure 4). Our model contains 4 stages: (1) *Reusability assessment*, (2) *Model selection*, (3) *Downstream evaluation*, and (4) *Model deployment*. We discuss each in turn.

Reusability Assessment Engineers begin the decision-making process with a *reusability assessment*. Before selecting a model, engineers must identify an ML task and determine

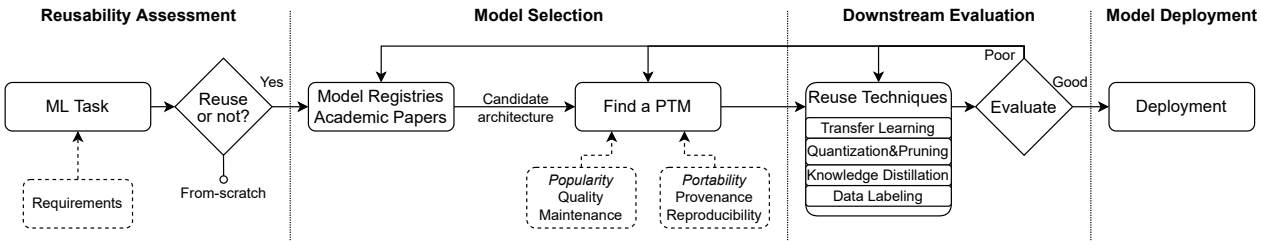

Fig. 4. Summarized decision-making progress of PTM reuse. Back edges indicate the possible changes of model selection.

if model reuse is appropriate. An ML task is composed of requirements including model input and output, latency, size, and licensing. Engineers must then decide if they should reuse a PTM or create a solution from scratch since "*PTMs do not work for every use case.*" Task parameters influence this decision. For example, three participants (*P8, P9, P11*) reuse PTMs because they "*do not have enough computational resources*". In a similar manner, participants (*P2, P5, P8, P9, P11*) note that DL model registries provide inference APIs to simplify reuse — PTMs are "*easy to use and test*".

Model Selection Once engineers decide to reuse a model, they must select an existing architecture and an associated PTM. Engineers search for candidate architectures "*built for the problem that [they are] trying to solve*", browsing model registries or relevant papers. Most study participants prefer to search through model registries. For example, participants (*P1, P4, P6*) said they can "*easily find a model*" in model registries due to classification at the domain (*e.g.,* computer vision, natural language processing) and task (*e.g.,* text generation, image classification) levels. *P10* noted that standardizing PTM reuse increases model registries' popularity.

Once engineers select a candidate architecture, they must find a particular PTM to use. All study participants, including those who select architectures from papers, prefer PTMs from model registries. "*Ease of use is very important*" to engineers that do not think it is worth "*spend[ing] much time on trying to understand the script[s] from the GitHub models*". *P8* noted models from Hugging Face are "*plug and play.*" Since multiple PTMs might implement the same architecture, engineers select from among candidates based on PTM attributes (§VII). For example, most participants use popularity as a factor to select a PTM because it indicates "*[community] trust in the model*". As another example, multiple participants (*P2, P3, P8, P11*) choose NLP PTMs trained on appropriate datasets (*e.g.,* models trained on datasets with the correct language).

Downstream Evaluation After selecting a PTM, engineers conduct a *downstream evaluation* for their specific task. Engineers have the option of assessing more than one candidate PTM in this stage. They download "*a few models,*" "*finetune them,*" "*test them,*" then "*compare them.*" When engineers select a candidate PTM, they first apply reuse techniques to fit the model to their specific application (cf. §II-B and §V-A). This procedure is not necessarily straightforward because some models "*don't really work too well directly, even with their own datasets*". Furthermore, 11 out of 12 of the participants observed a lack of adequate documentation or discrepancies within existing documentation.

After applying reuse techniques, engineers evaluate the model to see if it is ready for *model deployment*. When evaluating the trade-off between performance and architecture, *P1*, *P8* and *P10* state that these two factors are "*tightly relevant to each other*" and should be considered in a "*fifty-fifty split*". Some participants prioritize one of these factors over the other. For example, *P3* and *P6* compare multiple models to maximize task performance. On the other hand, *P7* will not use a "*weird architecture*" even if its documented performance is higher.

Model Deployment Finally, engineers deploy their models. Deployment may depend on model characteristics and deployment environments. All participants mention that certain characteristics such as PTM size, robustness, and documentation significantly impact the deployment of models to other environments (*i.e.,* different hardware or software configurations than what is used in development). Participant (*P8*) describes that the rapid increase in model size makes it "*impossible for most [low-resourced teams] to actually run these models on their systems.*" Participant (*P5*) states that most "*models are on PyTorch or TF*" and are therefore more difficult to deploy on mobile devices. Participant (*P4*) notes that documentation "*for running a model on multiple GPUs*" is "*not clear.*"

VI. RQ2: What PTM attributes facilitate PTM reuse?

Finding 2: For *Traditional attributes*, Popularity is most helpful. Provenance, Reproducibility, and Portability are the three *DL-specific attributes* we should consider.

Here we would like to learn about what sort of information is useful to engineers who reuse PTMs. We asked about two types of attributes here: traditional and DL-specific attributes.

A. Traditional Attributes

In the interview, we asked about whether the traditional attributes as offered by traditional package registries, such as NPM [68], are helpful in DL model registries.

Almost all participants highlighted the importance of *popularity* in DL model registries. For example, *P12* stated that a

PTM with "*lots of downloads*" means that "*it could be a good start point to try*". *P5* mentioned that "*popularity usually goes side by side with maintenance and can indicate the quality*".

Some participants thought that quality and maintenance are also very useful. *P2* said it is important to "*know that it is constantly maintained and does not have many open issues*", and pointed out that good *maintenance* means "*if the code is being updated or if you raise a bug, then someone will help you out*". This is highly important because "*you are relying on someone else*" and "*you want to build that trust factor*".

However, some participants think that maintenance and quality are less useful. Recall from §V that most reuse scenarios are fine-tuning on new datasets or tasks. Provided that the model is fine-tunable, some participants mentioned that maintenance and quality metrics are "*less useful in downstream tasks*". *P12* suggested that maintenance may be less relevant because of the cost of making changes — "*it is really hard to modify the PTM...there were some issues during pre-training*" in a large language model, but the model has not been retrained "*because it is too large*".

B. DL-specific Attributes

We added three DL-specific attributes: provenance, reproducibility, and portability. The participants provided the relevant factors for each attribute. Table I also indicates the factors for each attribute which were mentioned by multiple participants. Most participants mentioned that these three attributes can cover all the aspects they would consider.

Provenance Some Hugging Face PTMs provide many *provenance* metrics, such as information about original paper, dataset, and architecture. However, these are not detailed enough to fully address model reuse challenges. *P3* and *P9* would like to see "*visualization of model architecture*" and the "*explanation of changes*" compared to the paper. *P2, P4, P6, P8* mentioned the importance of more details of datasets because "*different authors process data differently, so it cannot be easily compared*". *P10 and P12* highlighted the importance of training logs because they would like to see "*how the PTM was generated*" based on the training checkpoints and scripts.

Reproducibility Reproducibility is the most problematic aspect of PTMs. The reproducibility issues mainly come from two aspects: (1) the configuration of training (2) the understanding of model. For the training configuration, the participants tend to care more about the hardware specification (*e.g.,* types, memory), training configuration (*e.g.,* training scripts, hyper-parameters). As a result, they think environment image would be helpful to help them easily configure the settings and make the model more reproducible. In terms of understanding of the models, different kinds of *demos* (*e.g.,* Notebook demo, Fine-tune demo) and *better documentation* would be helpful.

Portability Different models have different deployment constraints, which makes understanding the portability of PTMs helpful for engineers. Similar to *reproducibility*, the portability factors include hardware specification and environment. Moreover, for deployment, *latency* and *framework support* aspects are essential. For example, the inference time and cost of computational resources could be different in different platforms, as mentioned by *P2* and *P6*. This information can help engineers understand the portability of PTMs. *P3* mentioned that "*automate[d] creation of other formats of the model for different hardware*" could also be very helpful for the deployment. The quantized model is also "*helpful for continuous deployment and fine-tuning*", as mentioned by *P5*. As a result, he also suggested the development of automated quantization. Some participants also mentioned the fine-tuning instructions, which help them determine whether a model can be used in specific tasks. For example, *P12* would like to adapt language models to handle programming languages to improve their software testing, and the fine-tuning instructions can help him on deciding which model they should consider. *P6* also mentioned that if the model registries can provide a "*cost estimation for different servers*" (*e.g.,* different machine classes on a Cloud service provider). Moreover, *P1* and *P3* both said that "*licensing should be explicit to industry users*".

VII. RQ3: What are the challenges of PTM reuse?

Finding 3: Three common challenges for PTM reuse are missing attributes, discrepancies between claims and actual performances, and model risks (*e.g.,* privacy issues and unethical models) (Table III).

Missing Attributes Missing attributes are identiifed as the most challenging problem. Almost every participant mentioned that there are missing details in the model registries, including datasets, licensing, model details, robustness, and interpretability. The attributes are missing for multiple reasons: Insufficient documentation is one reason. *P5* and *P7* observed "*missing details of models*" in model registries. For example, *P1* and *P11* found the "*performances of the published models are unclear*" in the model registries. *P8* suggested that the reason for under-documentation in Hugging Face is that PTM authors can upload any model; Hugging Face does not enforce any form of documentation. Another reason is that the PTM authors occasionally do not measure the robustness and explainability of the models—and model registries do not provide an automated approach to measure such attributes.

Discrepancies Existing discrepancies are another key challenge mentioned by most participants. *P7* pointed out that "*some of the models are over-promising*". *P2* indicated another reproducibility issue: the "*model names are not named correctly and sometimes the provided scripts are broken*". These discrepancies could result in a waste of time and efforts. *P5* pointed that another reason for this kind of problem is that training configuration details (*i.e.,* hyper-parameters) are hard to find. *P6* also mentioned that some authors only provide a script instead of providing the actual fine-tuned model and corresponding performances in Hugging Face due to the sensitivity of these results. *P8* and *P9* indicated that they sometimes follow the provided steps, including the models and datasets—even the hardware configurations—and still could not reproduce the results claimed by the PTM authors.

TABLE III
CHALLENGES ASSOCIATED WITH PTM REUSE. THIRD COLUMN SHOWS HOW MANY PARTICIPANTS (OF 12) MENTIONED THE CHALLENGE.

Challenge	Description	# Participants
Missing attributes	There exist missing details in the model registries.	10
Discrepancies	The claimed performances (accuracy, latency) are often different from the actual performances.	9
Security and privacy risks	There could exist malicious models which are harmful for security and privacy aspects.	3
Model search	Sometimes the engineers know the model they want but cannot find it.	1
Model application	It is hard for new users to know the correct application of a model.	1
Model flexibility	Some of the model architectures are inflexible to change for downstream tasks.	1

Model Risks There exist potential risks for PTMs in the model registries, including privacy and ethics aspects. We discussed in §II that prior studies have identified many risks of PTMs. The participants mentioned both internal and external problems in DL model registries. Internal risks often involve privacy problems of models and data. *P2* mentioned that when using the models from model registries, "*the model deployment and data transmission are not in their control*". They could not directly deploy the model provided by Hugging Face because it is "*unreliable to send*" their sensitive dataset by Hugging Face inference APIs. *P8* mentioned that if a model "*is trained with malicious intents. It could have a lot of consequences in the real world*". This indicates the potential risks of a malicious model being uploaded to model registries.

A PTM can be used for unethical or nefarious purposes. *P8* gave an example of a chatbot created by training on a racist discussion forum. This model "*created a huge mayhem*" because it was publicly released [77]. *P10* observed that it is hard to know "*what exactly generated the model because most training logs are missing*" — the internal biases are concerning and could potentially make the model a BadNet [59].

VIII. RQ4: TO WHAT EXTENT ARE THE RISKS OF REUSING PTMS MITIGATED BY HUGGING FACE DEFENSES?

Finding 4: Although Hugging Face offers mitigations for many risks, these mitigations are incomplete or not widely adopted (Table IV). Model information can be missing or inaccurate due to the self-reporting nature of model metrics (Figure 7). These risks make the existence of malicious models possible in the model registries.

Our interview data identified a range of challenges (§VII). Incomplete or inaccurate data about PTMs was most common. Engineers also expressed concern about malicious or unethical models, *e.g.,* "BadNets" [59]. These findings led us to examine the Hugging Face defenses that mitigate these risks. We adapted the STRIDE methodology [72] to systematically measure the potential risks in Hugging Face, beginning with an analysis of the dataflow involved in collaborating on Hugging Face. The identified risks are shown in Table IV.

A. Hugging Face PTM Dataflow Model

Based on our analysis of Hugging Face's Hub and client libraries documentation, we made a dataflow diagram as shown in Figure 5, to represent the how models are created and shared. This allows us to visualize the dataflow from model contributors to users, which involves the developing, releasing, and accessing of PTMs on Hugging Face.

TABLE IV
MITIGATION AND THE CORRESPONDING STRIDE RISKS.

Mitigation	STRIDE risks	Details
Organization verification	Spoofing	Low adoption (3.19%)
Dependencies	Tampering	High dependency (one dataset has 1,673 downstream PTMs)
PTM Documentation	Repudiation	Less than 0.1% provide machine-parseable claims
GPG commit signing	Spoofing, Tampering, Repudiation	Low adoption (0.21%)
User permission model	Elevation of privileges	"Organizations" violate the principle of least privilege
ClamAV	N/A	Zero packages were flagged

The common unit of reuse on Hugging Face is the *repository*, classified into *datasets* (input/output data for supervised or unsupervised learning) and *models* (PTM architecture, weights, and configuration, cf. Figure 2).

All Hugging Face repositories have automated and manual risk mitigations to limit the spread of malware or malicious models. These include organization verification, user permission models, commit hash checkout, and model cards (Figure 5). Additionally, some PTMs depend on other datasets or PTMs which can expose themselves to outside risks.

B. Risk Analysis

Hugging Face implements six risk mitigations. These are organization verification, PTM documentation, GPG commit signing, a user permission model, automatic malware scanning (ClamAV), and utilizing a commit hash to checkout a model. While commit hash checkouts are not easily measured (performed by users on their own machines), we measured the use and scope of the others within the Hugging Face ecosystem. We detail our results on *organization verification*, *PTM documentation*, *user permission model*, and *GPG commit signing*. Following the risk analysis of traditional package registries [10], we also examine the potential impact of *dependencies*. The result for ClamAV was uninteresting as no (zero) PTM packages were flagged by ClamAV.

Organization Verification Hugging Face allows an organization to increase trust by verifying its identity, demonstrating

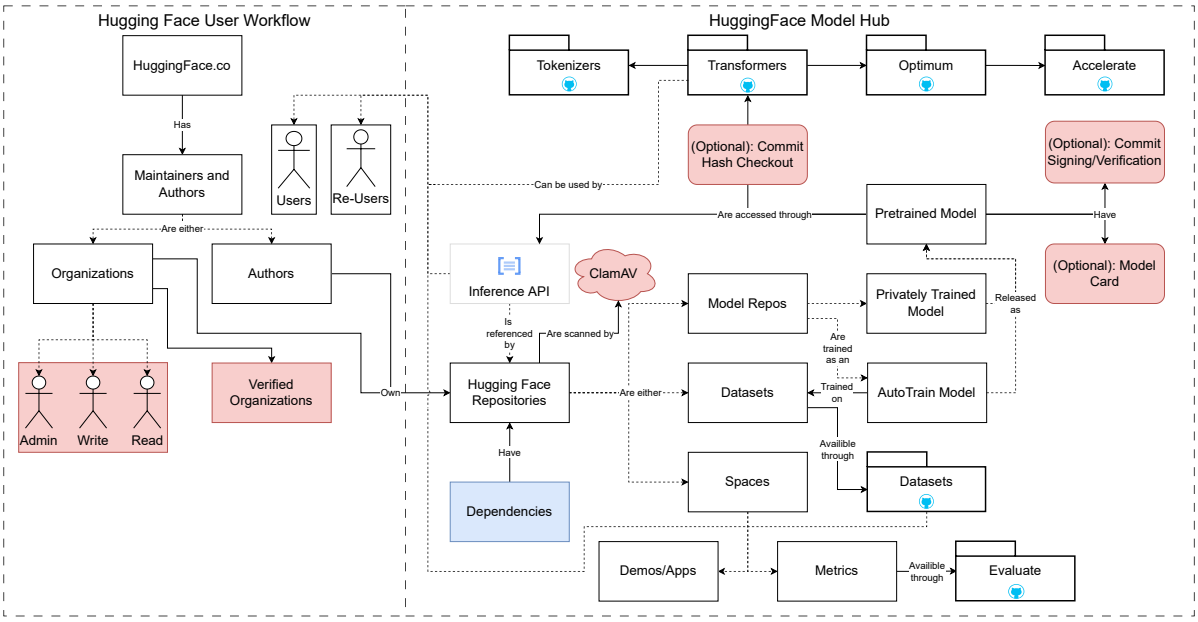

Fig. 5. Dataflow diagram for PTMs on Hugging Face. Security features and dependencies appear as red and blue blocks. Libraries with the GitHub logo are open source GitHub repositories. The large dashed boxes indicate a trust boundary between users and Hugging Face. Solid connections indicate required paths, and dashed connections represent alternatives.

to Hugging Face that an organization controls an associated web domain. We counted the number of verified organizations via web scraping. Out of 6,243 organizations, only 199 (3.19%) were verified. With such a small percentage of verified organizations, users cannot determine the legitimacy of unverified organizations. The low adoption rate raises the risk of *Spoofing*: malicious users may masquerade as real organizations, similar to typosquatting [10, 78].

This lack of adoption is concerning because organizational reputation was cited as a factor under the Provenance attribute (Table I). The risk of such squatting attacks can be greater for PTM packages than for traditional ones, because new niches in the ecosystem accompany every new state-of-the-art model. A malicious actor could identify missing PTMs in the cross product of (architecture, dataset) and provide *EvilModels* [62] in that niche, pretending to be a legitimate organization.

Dependencies The dependencies of PTMs pose potential risks. Malicious models can be injected directly via data manipulation [61], or indirectly via weight poisoning [60]. Models released through Hugging Face may be vulnerable through their dependencies on model architectures, the associated weights for those architectures, and datasets. *P5*, *P8*, *P10* stated that they fine-tune PTMs on a daily basis, implying that an attack could have a rapid impact.

Figure 6 shows that the `Universal Dependencies` dataset [79] is the most popular dataset on Hugging Face, with 6,834 models depending upon it. The distribution of models that depend on a particular architecture is similar to Figure 6. We found that the `BERT` [80] architecture is the most popular architecture on Hugging Face presently, with 10,247 models depending upon it. Hugging Face models have the potential to be trained off of multiple datasets as well. Our analysis of the existing Hugging Face models shows that many models depend on the works of others that can be maliciously tampered with. In tampering with these dependencies, *BadNets* [59] and unethical models [77] can be created, which could affect downstream PTMs.

PTM Documentation Figure 7 shows the distribution of missing documentation in Hugging Face. The highest proportion of models that make performance claims in machine-parseable documentation (YAML) was from the `token-classification` task, with 17% of models meeting the criteria. We found that 26,192 models belong to various tasks (represented by `other`) where only 12 (0.05%) PTMs provide machine-parseable claims about the PTM. Due to the sparse usage of performance claim reporting, there can be potential risks of Repudiation: model performance can be obscured, misreported, or misleading.

Some models report their performance in plain texts or tables, and are hard to identify by the users. It is also common for documentation to omit any performance claims or to refer the user to read an associated research paper for more information about performance, without assurance that this is the same model tested in the research paper. Some popular PTMs are poorly documented as well. The language model `SpanBERT/spanbert-large-cased` is the 9th most downloaded PTM on Hugging Face and receives 6.9 million monthly downloads, yet has no model card. Figure 7 supports that *missing attributes* is a real challenge existing in the Hugging Face DL model registries. The lack of transparency reduces the trustworthiness of the models and increase the potential risks of malicious models.

GPG Commit Signing Hugging Face provides a verification

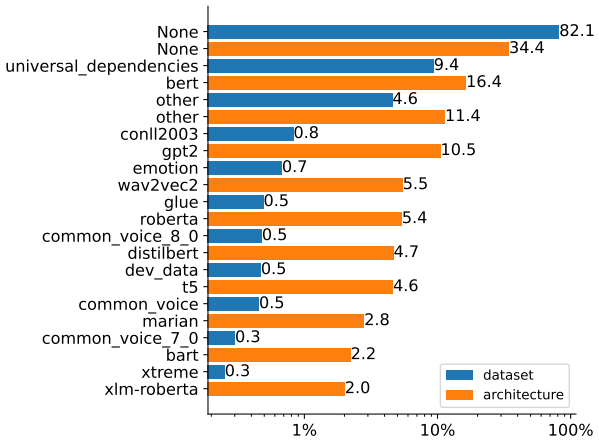

Fig. 6. Number of models trained on a specific dataset (blue) and trained with a specific model architecture (orange). 1,207 models were trained on the most popular dataset. 1,673 models depend on the most popular architecture.

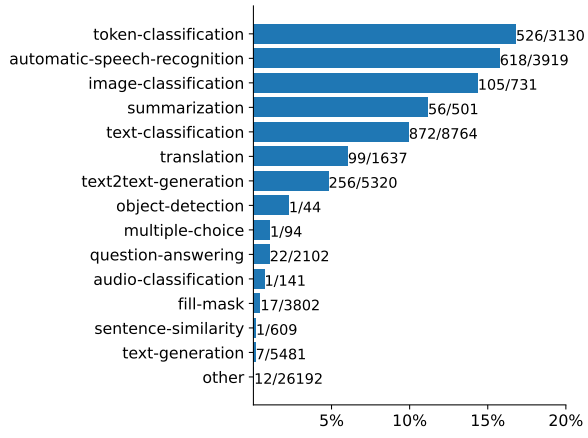

Fig. 7. The proportion of models with standardized claims (Hugging Face's YAML format) for each type of PTM.

tag for commits that have been signed with a GPG key. By signing with a GPG key, users are providing verification that they have signed their commits, and not as someone masquerading as them. This feature allows users to accurately trace back the changes to the PTM packages. Using the HFTorrent dataset (§IX), we analyzed the usage of commit signing within Hugging Face model repositories. Out of 63,182 model repositories, we found 132 (0.21%) repositories within the dataset implemented signed commits. Additionally, only 2 verified organizations have at least one repository with signed commits. This indicates that potential attackers can be contributing malicious code, implementing a *BadNet* [59] or *EvilModel* [62] under a pseudonym, or manipulating the `git` commit history to hide malicious activity.

The limited usage of GPG commit signing exposes models hosted on Hugging Face to *Spoofing*, *Tampering*, and *Repudiation* risks. First, unsigned Hugging Face models are vulnerable to *Spoofing* since attackers could make commits under the alias of a legitimate maintainer. Second, these models face the risk of *Tampering* because attackers could contribute malicious code or edit commit history. Finally, unsigned models could also risk *repudiation* since a lack of verified commit history allows an individual to deny actions within a repository.

User Permission Model Hugging Face has a standard approach of users and organizations (Figure 5). One shortcoming of the permission model is that Hugging Face organizations violate the principle of *least privilege* [81]: an organization member with *Write* privilege can modify any PTM owned by the organization. Therefore, an attacker could contribute malicious code, thereby raising the risk of *Elevation of Privileges*.

IX. The HFTorrent Dataset

Our analysis in §VIII relied in part on measurements of the PTM packages in Hugging Face. To reduce the impact on the Hugging Face service during our measurements, we took a snapshot of 63,182 PTM packages in the Hugging Face registry. This snapshot, the *HFTorrent Dataset*, is included in the artifact accompanying this paper. An improved version of this dataset is now available [82]. We hope these data facilitate further research on PTM packages, similar to the impact of the GHTorrent [83] and SOTorrent [84] datasets.

Creation process We initiated a copy of all PTM packages in the Hugging Face registry. Copies were taken between August 15th and 16th, with rate limiting to avoid abuse of the Hugging Face registry. 186 (0.3%) of the copies failed, caused, we believe, by concurrent changes in package names.

Dataset contents The HFTorrent dataset contains the repository histories of 63,182 of the PTM packages available on Hugging Face as of August 2022. They are provided as bare git clones to reduce space, resulting in a compressed footprint of ∼20 GB. Each PTM package can be reconstructed to its most recent version, including the model card, architecture, weights, and other elements provided by the maintainers (cf. Figure 2). Further information is available in our artifact.

X. Discussion and Implications

A. Integrating the findings

Our qualitative and quantitative studies provide a deeper understanding of the practices and challenges for DL model registries. In §V we obtained a general reuse process (Figure 4) which is complemented by the specific details for Hugging Face (Figure 5). In §VI we studied how the theoretical attributes of reproducibility and provenance can affect the decision-making process of PTM reuse (Figure 4). These attributes are partially operationalized in Hugging Face by aspects measured in §VIII: organizations and verification, PTM documentation, GPG commit signing, and dependencies. For example, our results in §VIII imply that although PTM reusers value the provenance of models, this provenance is actually untrustworthy in Hugging Face due to the low adoption of verified organizations and commit signing. In §VIII we measured risks based on the identified challenges from §VII. For example, in §VII we qualitatively learned that documentation may be missing or have discrepancies. In

TABLE V
MAIN DIFFERENCES BETWEEN DL MODEL REGISTRIES AND TRADITIONAL (*trad.*) PACKAGE REGISTRIES.

Aspect	Observed difference from trad. registries
Decision-making process	Selecting PTMs needs a more complicated assessment of requirements and trade-offs, and several iterations of downstream evaluation.
Attributes	PTM reusers have to consider additional DL-specific attributes.
Potential risk	There exist traditional risks and additional PTM risks in model registries.

§VIII we quantitatively measured that model documentation is missing or inadequate for 80% of PTMs.

B. Comparison to traditional package registries

Our qualitative analysis in §V—§VIII sheds light on the differences between model registries and traditional registries, in terms of decision-making, attributes, and potential risks.

Decision-making Our results indicate that the decision-making process of PTMs (Figure 4) is more complex than traditional packages [23, 85, 86], both in terms of the assessment and evaluation. Traditional software package reuse is integrated throughout the software development process [87] to improve productivity [85]. Our result shows that DL engineers behave similarly with PTMs. However, PTM reusers have to perform more complicated assessment and evaluation during decision-making process. PTMs are hard to evaluate and compare with others, as indicated by *P3, P6, P8, P10*. Moreover, Figure 4 involves three back edges (loops between selection and assessment), while the decision-making process for the traditional software reuse reportedly involves fewer iterations [13, 23]. Moreover, we can see in the selection stage of Figure 4, different factors are considered, including the dataset availability and cost. Traditional software reuser tend to consider more about the ease of use, and such detailed information are less needed by PTM reusers. Wang *et al.* suggest that developers should use the usage statistics to guide the evolution [88]. Similarly, PTM providers should consider the practices and challenges based on our qualitative data, and utilize it as a guidance to improve the PTMs in DL model registries.

Attributes Our interview data indicated the significance of traditional attributes (*i.e.,* popularity, quality, and maintenance) and the requirements for DL-specific attributes. For *Traditional attributes*, *Popularity* is most helpful for PTM reuse, while *Quality* and *Maintenance* are less useful compared to traditional packages (§VI). This differs from traditional software reusers, who consider quality and maintenance of packages as important as popularity when selecting and reusing the packages [25, 68, 89]. This difference comes from the expensive cost and data-driven nature in PTMs. It is hard for PTM users to directly obtain and employ an existing model. In contrast with taking a package from NPM or PyPI and directly reuse it in the codes, PTM reuse requires a deeper understanding and knowledge of how a model works. However, "*there are not quite reliable methods to measure the explainability of PTMs yet*", as indicated by *P10*. Therefore, to better reuse the PTMs, the engineers need more information from the model registries, which result in the requirements for DL-specific attributes.

Potential risks The dependencies, maintainers, and reported issues can help analyze the security risks in software registries [10]. Prior work indicates that developers should manage the upstream dependencies and minimize the impact on downstream tasks [86]. Recently, Jiang *et al.* empirically studied the maintainers' reach of model registries [19]. However, the results here suggest that there are different risks within model registries, such as Spoofing, Tampering, Repudiation, and Elevation of privileges (Table IV). These risks can have varying degrees of impacts on the reusability of PTMs: The unaware discrepancies "*do play a huge role*" (*P8*, *P11*, *P12*). They may not only "*hinder developer progress*", but also change the accuracy and robustness of downstream PTMs [61]. Moreover, our STRIDE analysis (§VIII) indicates multiple risks and the potential to introduce vulnerabilities [59, 60].

C. Implications

Compared to well-studied traditional package registries (*e.g.,* NPM, PyPi), the use and study of DL model registries is still in its infancy. Our empirical study of Hugging Face model registry informs future directions on model audit, infrastructure, PTM standardization, and adversarial attack detection.

Model audit Our results in §VI and §VII shows that one major challenge of PTM reuse is the missing attributes in model registries. The most important attribute is the performance of PTMs which would significantly affect the reusability of models. Though recently Hugging Face released their automated validation tool [90], it is still not able to satisfy the requirement of engineers. *P8* stated that "*Robustness and Explainabiltiy are very key factors to consider when you are deploying ML models in the real world.*" which are important for the eventual goal of model transparency [91].

The proposed DL-specific attributes should also be measured and provided. Our study indicates the importance of these attributes (§VI) and identify the corresponding factors. We inform researchers on developing formulas and automated tools to automatically calculate the score of each attribute and integrate them into the model registries, similar to the *pqm* scores (*i.e.,* popularity, quality, and maintenance) in [40].

We envision that future directions should consist of large-scale measurements of PTMs or of encouraging model registries to change their PTM release requirements so that it would be easier for users to audit models by themselves.

Infrastructure The infrastructure of model registries can be improved from different aspects. *P2* mentioned that the badge mechanism would be helpful for communicating the missing attributes (*e.g.,* reproducibility), similar to continuous integration badges provided by GitHub [92]. *P6* mentioned a unified fine-tuning process could also assist engineers. Moreover, multiple participants highlighted the importance of tools for automatically creating quantized models or converting models

into different formats. These tools could improve PTM portability and thus support model deployment.

Some interview participants (*P6, P9*) mentioned that they select PTMs first based on their experience, and then based on evaluation metrics. *P10* mentioned that he reuses PTMs to understand the "*the generalizing behavior of fine-tuned models*". This envisions development of PTM recommendation systems, which can sort the models by scores calculated by model performance in downstream tasks, or predicted by another ML model. A similar system can help to reduce the work for ML engineers and could be integrated as part of the AutoML pipeline for PTM reuse [93].

PTM Standardization Our results on §VI shows that, the model registries should include the training logs and corresponding checkpoints. This information can help reusers better understand the PTM and improve the provenance and reproducibility. As associated research opportunities, such artifacts can be costly to create and store, and it is unclear how engineers can best apply them.

Beyond model provenance, another opportunity for standardization is in the model format itself. *P5* said that "*many PTMs are on PyTorch or TensorFlow*" but they would like to use ONNX format models which could make the deployment easier for them. However, due to the rapid appearance of new operators [94], ONNX could not support all of them, especially for state-of-the-art models [95]. Knowing the compatibility of PTMs in model registries with standardized formats such as ONNX would also engineers make better decisions and save their time. We suggest future work examining development challenges in the ONNX framework.

Adversarial attack detection One of the major challenges for PTM reuse are adversarial attacks, as shown in §V. Attacks can be harmful to both the PTM reusers and the downstream application users. Although Hugging Face employs ClamAV [96] for malware scanning, this only detects traditional attacks, not new attacks such as *BadNets* [59]. As a result, we suggest future studies working on automated detection of toxic models and poisoned datasets. Integrating these detectors into model registries can largely improve their trustworthiness.

XI. Threats to Validity

Internal Threats Our choice of research methods potentially threatens the validity of our investigation of PTM reuse practices and challenges (RQ1-3). Our results here are derived solely from interview data but not generalized via a survey instrument. As a mitigation, our measurements of the Hugging Face ecosystem (RQ4) substantiate many of the interview participants' concerns. In addition, we note that the participants of greatest interest are those who also make the greatest use of PTMs, *i.e.,* presumably those with PRO accounts on Hugging Face. 9 of our 12 participants have PRO accounts, representing 1% of the PRO user population.

Another internal threat is the reliability of our framework analysis on the interview data. Our framework analysis might be biased by our understanding and transfer of concepts from traditional software to PTMs. This conceptual framework helps us tease out similarities and differences in the PTM context, although we recognize that our interviews might reflect our biases and perspectives and therefore bias participants to a certain way of thinking. To mitigate bias, we asked if the participant had anything to add in terms of each theme in our framework throughout the interview, and some did so.

External Threats Our study examines only one DL model registry, Hugging Face. We note, however, that examining a single package registry, *e.g.,* NPM, is fairly common in the literature. For PTMs, focusing on Hugging Face is sensible, since it is the only open DL model registry and has an order of magnitude more models than other registries. Our results may not generalize to other DL model registries, but given the relative importance and growing influence of Hugging Face it is unclear whether this is a concern.

Another external threat is the saturation of our interview study because of the interview study's sampling approach (one PTM registry) and size (12 participants). Within our sample, we saw a high degree of agreement. The saturation of our interview study was achieved after 7 participants (§IV-A)

ML researchers identified many uses of PTMs (§II-B), but our participants only employed a subset related to fine-tuning: transfer learning, quantization and pruning. This may pose a threat to external validity. Our random sampling approach may bias us towards high-probability use cases. One interpretation of our data is that fine-tuning is a popular approach in practice, which would motivate greater study of PTM fine-tuning relative to more theoretical applications.

XII. Conclusion

We conducted the first empirical study of PTM reuse in the Hugging Face DL model registries. Based on interviews with 12 practitioners, we defined the decision-making workflow for PTM reuse, and identified three challenges, including missing attributes, discrepancies, and model risks. To substantiate our qualitative data, we further investigated into useful attributes and potential risks in the Hugging Face ecosystem. We unveiled risky engineering practices in the Hugging Face ecosystem, particularly a lack of signatures in the PTM supply chain. Our empirical data motivates future research on PTM audit, automated PTM attribute measurements, improved infrastructure for PTM reuse, and PTM standardization.

Reproducibility and Research Ethics

Our artifact is available at https://doi.org/10.5281/zenodo.7555469. Within it, we provide the anonymized interview data used to answer RQ1-3, the HFTorrent dataset used to answer RQ4, and software for the measurements described in RQ4. Human subjects work was approved by institutional IRB.

Acknowledgments

The authors thank the reviewers; and A. Grigorescu, D. Montes, A. Indarapu, and A. Tewari for their input. This work was supported by Google and Cisco and by NSF awards #2107230, #2229703, #2107020, and #2104319.

REFERENCES

[1] E. Raymond, "The cathedral and the bazaar," *Knowledge, Technology & Policy*, vol. 12, no. 3, pp. 23–49, 1999.

[2] R. Abdalkareem, O. Nourry, S. Wehaibi, S. Mujahid, and E. Shihab, "Why do developers use trivial packages? an empirical case study on npm," in *European Software Engineering Conference/Foundations of Software Engineering (ESEC/FSE)*, 2017.

[3] N. K. Gopalakrishna, D. Anandayuvaraj, A. Detti, F. L. Bland, S. Rahaman, and J. C. Davis, ""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)*, 2022.

[4] J. Garcia, Y. Feng, J. Shen, S. Almanee, Y. Xia, and a. Q. A. Chen, "A comprehensive study of autonomous vehicle bugs," in *International Conference on Software Engineering (ICSE)*, 2020.

[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] 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 *International Conference on Automated Software Engineering (ASE)*, 2020.

[7] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang, W. Han, M. Huang, Q. Jin, Y. Lan, Y. Liu, Z. Liu, Z. Lu, X. Qiu, R. Song, J. Tang, J.-R. Wen, J. Yuan, W. X. Zhao, and J. Zhu, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, 2021.

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

[9] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-Art Natural Language Processing," in *Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020.

[10] 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.

[11] N. Zahan, T. Zimmermann, P. Godefroid, B. Murphy, C. Maddila, and L. Williams, "What are Weak Links in the npm Supply Chain?" in *International Conference on Software Engineering (ICSE)*, May 2022. [Online]. Available: https://www.microsoft.com/en-us/research/publication/what-are-weak-links-in-the-npm-supply-chain/

[12] A. Zerouali, T. Mens, A. Decan, and C. De Roover, "On the impact of security vulnerabilities in the npm and RubyGems dependency networks," *Empirical Software Engineering (EMSE)*, 2022.

[13] A. S. Jadhav and R. M. Sonar, "Evaluating and selecting software packages: A review," *Information and Software Technology*, 2009.

[14] A. Decan, T. Mens, and P. Grosjean, "An empirical comparison of dependency network evolution in seven software packaging ecosystems," *Empirical Software Engineering (EMSE)*, 2019.

[15] C. Okafor, T. R. Schorlemmer, S. Torres-Arias, and J. C. Davis, "Sok: Analysis of software supply chain security by establishing secure design properties," in *ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED'22)*, 2022.

[16] S. Schelter, F. Biessmann, T. Januschowski, D. Salinas, S. Seufert, and G. Szarvas, "On Challenges in Machine Learning Model Management," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 2018.

[17] 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.

[18] R. B. Johnson and A. J. Onwuegbuzie, "Mixed Methods Research: A Research Paradigm Whose Time Has Come," *Educational Researcher*, 2004.

[19] 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, pp. 105–114.

[20] A. e. a. Monteil, "Nine Best Practices for Research Software Registries and Repositories: A Concise Guide," Dec. 2020, arXiv:2012.13117 [cs]. [Online]. Available: http://arxiv.org/abs/2012.13117

[21] S. Oladele, "Ml model registry: What it is, why it matters, how to implement it," 2022. [Online]. Available: https://neptune.ai/blog/ml-model-registry

[22] M. J. Smith, C. Sala, J. M. Kanter, and K. Veeramachaneni, "The Machine Learning Bazaar: Harnessing the ML Ecosystem for Effective System Development," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, Jun. 2020.

[23] A. S. Jadhav and R. M. Sonar, "Framework for evaluation and selection of the software packages: A hybrid knowledge based system approach," *Journal of Systems and Software (JSS)*, 2011.

[24] V. del Bianco, L. Lavazza, S. Morasca, and D. Taibi, "A Survey on Open Source Software Trustworthiness," *IEEE Software*, 2011.

[25] A. Zerouali, T. Mens, G. Robles, and J. M. Gonzalez-Barahona, "On the Diversity of Software Package Popularity Metrics: An Empirical Study of npm," in *International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2019.

[26] D. Mitropoulos, V. Karakoidas, P. Louridas, G. Gousios, and D. Spinellis, "The bug catalog of the maven ecosystem," in *Working Conference on Mining Software Repositories (MSR)*, 2014.

[27] C. Soto-Valero, A. Benelallam, N. Harrand, O. Barais, and B. Baudry, "The Emergence of Software Diversity in Maven Central," in *International Conference on Mining Software Repositories (MSR)*, 2019.

[28] J. Pineau, P. Vincent-Lamarre, K. Sinha, V. Lariviere, and A. Beygelzimer, "Improving Reproducibility in Machine Learning Research," *Journal of Machine Learning Research*, 2020.

[29] P. Goswami, S. Gupta, Z. Li, N. Meng, and D. Yao, "Investigating The Reproducibility of NPM Packages," in *International Conference on Software Maintenance and Evolution (ICSME)*, 2020.

[30] D.-L. Vu, F. Massacci, I. Pashchenko, H. Plate, and A. Sabetta, "LastPyMile: identifying the discrepancy between sources and packages," in *European Software Eng. Conf. and Symp. on the Foundations of Software Eng. (ESEC/FSE)*, 2021.

[31] M. Hutson, "Artificial intelligence faces reproducibility crisis," *American Association for the Advancement of Science*, vol. 359, no. 6377, pp. 725–726, 2018.

[32] W. Jiang, N. Synovic, and R. Sethi, "An Empirical Study of Artifacts and Security Risks in the Pre-trained Model Supply Chain," *Los Angeles*, p. 10, 2022.

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

[34] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, pp. 370–403, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221010894

[35] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," Mar. 2015, arXiv:1503.02531 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1503.02531

[36] P. Dube, B. Bhattacharjee, S. Huo, P. Watson, and B. Belgodere, "Automatic Labeling of Data for Transfer Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019, pp. 122–129.

[37] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A Comprehensive Survey on Transfer Learning," Jun. 2020. [Online]. Available: http://arxiv.org/abs/1911.02685

[38] S. Rahman, E. River, F. Khomh, Y. G. Guhneuc, and B. Lehnert, "Machine learning software engineering in practice: An industrial case study," in *International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2019.

[39] Databricks, "Mlflow model registry," 2022. [Online]. Available: https://databricks.com/product/mlflow-model-registry

[40] NPM, "npm," 2022. [Online]. Available: httnps://www.npmjs.com/

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

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

[43] TensorFlow team, "Tensorflow hub," 2022. [Online]. Available: https://www.tensorflow.org/hub

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

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

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

[47] NVIDIA, "NVIDIA NGC: AI Development Catalog," 2022. [Online]. Available: https://catalog.ngc.nvidia.com/

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

[49] M. Injadat, A. Moubayed, A. B. Nassif, and A. Shami, "Machine learning towards intelligent systems: applications, challenges, and opportunities," *Artificial Intelligence Review*, 2021.

[50] D. Marijan and A. Gotlieb, "Software Testing for Machine Learning," *AAAI Conference on Artificial Intelligence*, 2020.

[51] K. Rasheed, A. Qayyum, M. Ghaly, A. Al-Fuqaha, A. Razi, and J. Qadir, "Explainable, trustworthy, and ethical machine learning for healthcare: A survey," *Computers in Biology and Medicine*, 2022.

[52] J. M. Wing, "Trustworthy AI," *Communications of the ACM*, 2021.

[53] M. Mora-Cantallops, S. Sánchez-Alonso, E. García-Barriocanal, and M.-A. Sicilia, "Traceability for Trustworthy AI: A Review of Models and Tools," *Big Data and Cognitive Computing*, 2021.

[54] L. Floridi, "Establishing the rules for building trustworthy AI," *Nature Machine Intelligence*, 2019.

[55] S. Shams, R. Platania, K. Lee, and S.-J. Park, "Evaluation of deep learning frameworks over different hpc architectures," in *International Conference on Distributed Computing Systems (ICDCS)*, 2017.

[56] L. Liu, Y. Wu, W. Wei, W. Cao, S. Sahin, and Q. Zhang, "Benchmarking deep learning frameworks: Design considerations, metrics and beyond," in *International Conference on Distributed Computing Systems*, 2018.

[57] N. Akhtar and A. Mian, "Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[58] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning Attack on Neural Networks," in *Network and Distributed Systems Security (NDSS) Symposium*, 2018.

[59] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain," 2019. [Online]. Available: http://arxiv.org/abs/1708.06733

[60] K. Kurita, P. Michel, and G. Neubig, "Weight Poisoning Attacks on Pre-trained Models," arXiv, Tech. Rep., 2020. [Online]. Available: http://arxiv.org/abs/2004.06660

[61] M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Madry, B. Li, and T. Goldstein, "Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[62] Z. Wang, C. Liu, and X. Cui, "EvilModel: Hiding Malware Inside of Neural Network Models," in *IEEE Symposium on Computers and Communications (ISCC)*, 2021.

[63] ö. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," *IEEE Access*, 2020.

[64] J. Ritchie and L. Spencer, "Qualitative data analysis for applied policy research," in *Analyzing qualitative data*. Routledge, 2002, pp. 187–208.

[65] A. Srivastava and S. Thomson, "Framework analysis: A qualitative methodology for applied policy research," *Journal of Administration and Governance (JOAAG)*, vol. 4, 2009.

[66] L. G. Michael, J. Donohue, J. C. Davis, D. Lee, and F. Servant, "Regexes are Hard: Decision-Making, Difficulties, and Risks in Programming Regular Expressions," in *International Conference on Automated Software Engineering (ASE)*, 2019.

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

[68] A. Abdellatif, Y. Zeng, M. Elshafei, E. Shihab, and W. Shang, "Simplifying the Search of npm Packages," *Information and Software Technology*, 2020.

[69] Hugging Face, "Hugging face users," 2022. [Online]. Available: https://huggingface.co/users

[70] J. Saldana, *Fundamentals of qualitative research*. Oxford University Press, 2011.

[71] G. Guest, A. Bunce, and L. Johnson, "How Many Interviews Are Enough?: An Experiment with Data Saturation and Variability," *Field Methods*, 2006. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1525822X05279903

[72] Microsoft, "The STRIDE Threat Model," 2021. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)

[73] D. Boyd, "How to approach threat modeling," 2021. [Online]. Available: https://aws.amazon.com/blogs/security/how-to-approach-threat-modeling/

[74] L. Conklin, "Threat Modeling Process," 2023. [Online]. Available: https://owasp.org/www-community/Threat_Modeling_Process

[75] A. Shostack, "Experiences threat modeling at microsoft," *the Workshop on Modeling Security*, vol. 413, 2008.

[76] R. S. Pressman, *Software engineering: a practitioner's approach*. Palgrave macmillan, 2005.

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

[78] N. P. Tschacher, "Typosquatting in programming language package managers," Ph.D. dissertation, Universität Hamburg, Fachbereich Informatik, 2016.

[79] J. Nivre, M.-C. De Marneffe, F. Ginter, Y. Goldberg, J. Hajic, C. D. Manning, R. McDonald, S. Petrov, S. Pyysalo, N. Silveira *et al.*, "Universal dependencies v1: A multilingual treebank collection," in *International Conference on Language Resources and Evaluation (LREC)*, 2016, pp. 1659–1666.

[80] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv*, 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[81] J. Saltzer and M. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975. [Online]. Available: http://ieeexplore.ieee.org/document/1451869/

[82] W. Jiang, N. Synovic, P. Jajal, T. R. Schorlemmer, A. Tewari, B. Pareek, G. K. Thiruvathukal, and J. C. Davis, "PTMTorrent: A Dataset for Mining Open-source Pre-trained Model Packages," 2023. [Online]. Available: https://doi.org/10.6084/m9.figshare.22009880

[83] G. Gousios and D. Spinellis, "GHTorrent: Github's data from a firehose," in *International Working Conference on Mining Software Repositories (MSR)*, 2012.

[84] S. Baltes, "SOTorrent: Reconstructing and Analyzing the Evolution of Stack Overflow Posts," in *International Conference on Mining Software Repositories (MSR)*, 2018.

[85] R. Abdalkareem, O. Nourry, S. Wehaibi, S. Mujahid, and E. Shihab, "Why do developers use trivial packages? an empirical case study on npm," in *European Software Engineering Conference/Foundations of Software Engineering (ESEC/FSE)*, 2017.

[86] C. Bogart, C. Kästner, and J. Herbsleb, "When It Breaks, It Breaks: How Ecosystem Developers Reason about the Stability of Dependencies," in *International Conference on Automated Software Engineering Workshop (ASEW)*, 2015.

[87] M. Anasuodei, Ojekudo, and N. Akpofure, "Software Reusability: Approaches and Challenges," *International Journal of Research and Innovation in Applied Science*, vol. 06, no. 05, 2021.

[88] Y. Wang, B. Chen, K. Huang, B. Shi, C. Xu, X. Peng, Y. Wu, and Y. Liu, "An Empirical Study of Usages, Updates and Risks of Third-Party Libraries in Java Projects," in *International Conference on Software Maintenance and Evolution (ICSME)*, 2020.

[89] S. Mujahid, R. Abdalkareem, and E. Shihab, "What are the Characteristics of Highly-Selected Packages? A Case Study on the NPM Ecosystem," *SSRN Electronic Journal*, 2022.

[90] L. Tunstall, A. Thakur, T. Thrush, S. Luccioni, L. v. Werra, N. Rajani, O. Piktus, O. Sanseviero, and D. Kiela, "Announcing Evaluation on the Hub." [Online]. Available: https://huggingface.co/blog/eval-on-the-hub

[91] R. Hamon, H. Junklewitz, and J. I. Sanchez Martin, "Robustness and explainability of artificial intelligence," *Publications Office of the European Union*, vol. 207, 2020.

[92] GitHub, "Adding a workflow status badge," 2022. [Online]. Available: https://docs.github.com/en/actions/monitoring-and-troubleshooting-workflows/adding-a-workflow-status-badge

[93] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowledge-Based Systems*, 2021.

[94] Y. Liu, C. Chen, R. Zhang, T. Qin, X. Ji, H. Lin, and M. Yang, "Enhancing the interoperability between deep learning frameworks by model conversion," in *European Software Engineering Conference/Foundations of Software Engineering (ESEC/FSE)*, 2020.

[95] "Portability between deep learning frameworks – with ONNX," Aug. 2019. [Online]. Available: https://blog.codecentric.de/en/2019/08/portability-deep-learning-frameworks-onnx/

[96] Cisco, "ClamAV," 2022. [Online]. Available: https://www.clamav.net/