On Exploring the Sub-domain of Artificial Intelligence (AI) Model Forensics

Tiffanie Edwards(✉), Syria McCullough, Mohamed Nassar, and Ibrahim Baggili

University of New Haven Cyber Forensics Research and Education Group (UNHcFREG) and Connecticut Institute of Technology (CIT), West Haven, CT 06516, USA
{tedwa4,smccu1}@unh.newhaven.edu, {mnassar,ibaggili}@newhaven.edu

Abstract. AI Forensics is a novel research field that aims at providing techniques, mechanisms, processes, and protocols for an AI failure investigation. In this paper, we pave the way towards further exploring a sub-domain of AI forensics, namely AI model forensics, and introduce AI model ballistics as a subfield inspired by forensic ballistics. AI model forensics studies the forensic investigation process, including where available evidence can be collected, as it applies to AI models and systems.

We elaborate on the background and nature of AI model development and deployment, and highlight the fact that these models can be replaced, trojanized, gradually poisoned, or fooled by adversarial input.

The relationships and the dependencies of our newly proposed sub-domain draws from past literature in software, cloud, and network forensics. Additionally, we share a use-case mini-study to explore the peculiarities of AI model forensics in an appropriate context. Blockchain is discussed as a possible solution for maintaining audit trails. Finally, the challenges of AI model forensics are discussed.

Keywords: Digital forensics · Artificial Intelligence · AI forensics

1 Introduction

The rapid integration of Artificial Intelligence (AI) in modern technologies, services, and industries is leading AI to become a fundamental part of daily life. AI applications are found in video games [23], autonomous vehicles [18], healthcare [13], and cybersecurity [19]. While AI exists to benefit society, it is not without its challenges, which led to the development of the AI safety field [2]. Further exploration is necessary in the realm of forensics concerning AI, particularly how forensics will apply to the AI domain, to contribute to overall AI safety. This new discipline has been coined as AI Forensics [3].

AI Forensics is a subfield of digital forensics and is defined as *"Scientific and legal tools, techniques, and protocols for the extraction, collection, analysis, and reporting of digital evidence pertaining to failures in AI-enabled systems"*

P. Gladyshev et al. (Eds.): ICDF2C 2021, LNICST 441, pp. 35–51, 2022.
https://doi.org/10.1007/978-3-031-06365-7_3

[3]. Within this domain is the subdomain of AI Model Forensics, which narrows the forensic focus of failures in AI-enabled systems to information linked to AI models, usually stored in model files. Before, during, or after model deployment, AI models may suffer from malicious attacks ranging from physical adversarial samples [26], backdoors [6], malware injection [40], active learning, gradual poisoning, or replacing the authentic model by a malicious one, compromising an AI-enabled system.

For instance, [3] coined AI model authentication forensics as a subfield of AI model forensics. Recent work identified watermarking techniques to authenticate the ownership of AI models. Behzadan et al. [4] proposed a novel scheme for the watermarking of Deep Reinforcement Learning (DRL) policies. In [20,41], watermarks are generated samples, almost indistinguishable from their origins, and infused into the deep neural network model by assigning specific labels to them. The specific labels are the basis for copyright claims.

However, watermarked models can still be tampered with or forged [3], which impedes AI model authentication and verification. In addition, watermarking in itself does not hinder an adversary from publishing or deploying a new model under the name of an authentic model provider or creator. The possibility of such events leads to the need for procedures and protocols that help determine the forensic soundness of AI model digital evidence.

Within the new field of AI model forensics, we ask the questions: "What evidence is left behind in terms of AI model artifacts such as files, examined samples, label predictions, logs, etc.?", "How to examine such evidence?", and "What conclusions can be made about the type of event that occurred and its source?". We propose to establish AI model ballistics as a subfield of AI Model Forensics, focusing on digital forensic investigations that involve AI systems [3]. In general, the study of forensic ballistics refers to examining evidence left behind from firearms which would lead to conclusions about the type of firearm and its owner. Model ballistics could possibly help investigators identify information, such as the intention of the AI model, whether or not the output of the model differs from the creator's intention, the framework the model was created on, and any other relevant information that can be used in a digital forensic investigation.

Our work provides the following contributions:

- We provide the primary in-depth discussion of AI model forensics.
- We provide an overview of digital forensic practices for domains that are strongly connected to AI and are useful in initiating protocols and procedures tailored for AI investigations.
- We provide a primary case study to motivate the sub-domain of AI model forensics.
- We share a cohesive view of the intersection of AI model forensics with software forensics, cloud forensics, and network forensics.
- We enumerate primary challenges in AI model forensics.

The remainder of this paper is divided into the following sections: Sect. 2 describes the basic principles of digital forensics and establishes the motivation. Work from related forensic areas is discussed in Sect. 3. An example case study

is presented in Sect. 4. Section 5 discusses the use of Blockchain technology in AI model forensics. The subfield AI model ballistics is formally introduced in Sect. 6. Section 7 explores the challenges and limitations of AI model forensics. Lastly, we conclude our work in Sect. 8 and discuss future directions.

2 Background and Motivation

2.1 Principles of Digital Forensics Overview

Digital forensics (DF) is formally defined by the National Institute of Standards and Technology (NIST) as *the application of science to the identification, collection, examination, and analysis, of data while preserving the integrity of the information and maintaining a strict chain of custody for the data* [15]. Under this definition, many subfields exist such as cloud forensics [31], software forensics [37], and network forensics [14]. Digital forensics and its subfields come with continuously evolving policies and procedures that protect the forensic soundness of digital data during investigations. Digital data is forensically sound when the scientific process follows the five major principles [12]:

1. Authenticity: Digital data is proved to be unchanged after collection and analysis; if necessary, only minimal changes are made to the data.
2. Error: All known errors in the forensic process are thoroughly documented.
3. Reliability: All utilized procedures must be published and accepted within the scientific community.
4. Reproducibility: All procedures should produce consistent results on the digital data each time the procedure(s) is performed.
5. Experience: All investigators handling the digital data should have a sufficient amount of experience or knowledge.

AI Forensics aims at replicating these principles when creating policies and procedures that are suitable for investigating failures in AI-enabled systems. AI model forensics, considered a subset of the AI forensics, focuses on how to address AI model artifacts, such as files, logs, model authenticity, classification history, etc., and how to promote the forensic soundness of these artifacts.

AI-related crime has two major categories: *AI as a tool crime* and *AI as a target crime* [12]. *AI as a tool crime* implies that AI systems or services are used to commit physical crimes or aid in cybercrimes. [34] conducted experiments related to *AI as a tool crime* from an AI forensic standpoint. The research viewed a "malicious by design" AI system and set out to determine if an AI system caused a malicious incident and why the incident was caused. From their perspective, *AI as a tool* can become "malicious by design" through methods of tampering and then be used as a malicious tool by a perpetrator.

In contrast, *AI as a target crime* implies that vulnerabilities in AI-enabled systems are exploited or hijacked. Potential threats include but are not limited to: manipulation of training data, malicious code change or replacement, and tricking AI systems into improper operation.

With any forensic approach, the collected evidence should answer questions related to the when and where of the crime, who is the criminal, what was the crime's target, and how the crime occurred. Defining a forensic approach for model ballistics will help investigators answer these key questions about AI systems, whether it falls under the category of *AI as a tool crime* or *AI as a target crime*.

To gain a clear picture of what investigators will encounter when dealing with AI models, model files, and artifacts, it is imperative to provide insights and a general understanding about how model files are generated and how AI models are deployed in production.

2.2 Model Generation and Deployment Overview

Deploying, or serving, a model simply means to turn it into a usable model to host in a production platform. For example, the macro steps leading to the deployment of a typical Machine Learning (ML) model are depicted in Fig. 1. The outlined steps are data collection and preparation, feature selection, model training and testing, model packaging, and deployment.

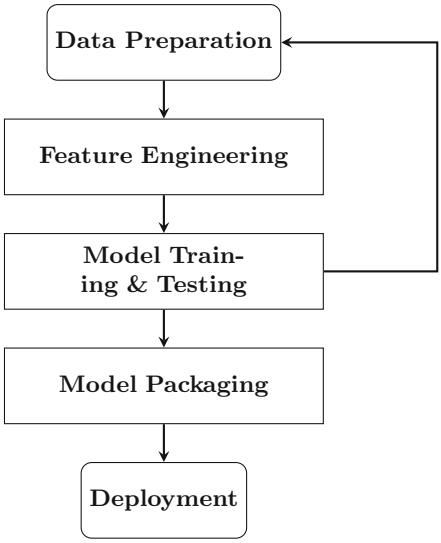

Fig. 1. Simplistic pipeline of the development and deployment of an AI model

Usually an AI model, a machine/deep learning one in particular, is saved to a single file containing the architecture and the parameters of the model, e.g. layers and weights (deep learning), support vectors (SVM), cluster centroids (K-means clustering), etc. The file is the output of the three first steps of the pipeline. Usually an inner loop that goes back to the first step may take place here if the model performance was not sufficiently satisfying.

Creators have the option to save the entire AI model, or some specific parts of it, into a file, known as the model file. Saving an AI model is useful for restoring it at any time in the future without having to repeat any previous training and parameter tuning. The saved model can be used for transfer learning, to train over new datasets, packaging, distribution, hosting, and deployment. Examples of frameworks using this approach are TensorFlow [38] and PyTorch [30]. For instance, we can restore only the architecture of a deep learning (DL) network (layers, nodes, layer types, activation functions, etc.), or the architecture along with the parameters (e.g. weights, biases, dropout, etc.).

We collected in Table 1 information about popular AI and ML platforms along with model file format, model file extension and other options of model saving. We also note that some frameworks allow models to be imported/exported from/to different formats.

Table 1. Popular AI/ML frameworks and their model formats

Framework	File format	File extension	Saving options
Tensorflow	tf or HDF5	.tf or .h5	weights & checkpoints, or entire model
Scikit-Learn	Pickled Python	.pkl	Compression
Keras	tf or HDF5	.tf or .h5	checkpoints, optimizer state, whole model
PyTorch	Torchscript	.pt or .pth	state_dict
iOS Core ML	Apple ML Model	.mlmodel	Add to App.
SparkML	model: JSON, data: Parquet	.json, .parquet	Save a Pipeline
spaCy	model:Config, metadata: JSON	.cfg, .json	Save a Pipeline and its metadata

AI and ML models are usually deployed as cloud services, and not necessarily in form of model files. Cloud deployment helps in connecting the AI model to clients and mobile applications, either for further data collection and improvement, such as in the case of federated learning [16], or for providing prediction and recommendation services for the model clients, whether they are AI agents, IoT devices, mobile applications, or browser-based users. Prediction services can be provided in batch mode or in single mode.

3 Literature Review

AI model forensics is strongly tied to other forms of forensics, and is useful when examining and analyzing model files and model deployments. We estimate that three areas in particular, namely cloud, software, and network forensics, are related to AI model forensics and may support the creation of its standards and procedures.

3.1 Cloud Forensics

The NIST defines Cloud Computing Forensic Science as “the application of scientific principles, technological practices and derived and proven methods to

reconstruct past cloud computing events through identification, collection, preservation, examination, interpretation and reporting of digital evidence [11]". Cloud forensics [32] has emerged from both the digital forensics and the cloud computing fields. Cloud forensics is in high demand today because of the importance of these two emerging fields. The Market Research Firm report [8] classifies cloud forensics as a type of digital forensics and estimates the global digital forensics market to grow to 9.68 billion by year 2022. The report correlates the prolific spread of cloud-based applications in North America with an increase in cyber-attacks and a growing sophistication of these attacks.

Challenges of Cloud Forensics. Since cloud forensics is a fairly new field, compared to its predecessors cloud computing and digital forensics, there are still many challenges in the field. Different challenges can be attributed based on the cloud service model. Three service models are identified in [22]:

- Infrastructure as a service (IaaS): Delivery of bare metal and virtual machines, e.g. AWS;
- Platform as a service (PaaS): serves application development and deployment, e.g. AWS Elastic Beanstalk;
- Software as a service (SaaS): provides packaged software such as office apps or an AI engine.

Many of these challenges are identified in [35]. Cloud resources are virtualized and shared in nature. Cloud data is described as "fragile and volatile," which makes correct data extraction a very sensitive process. Remote data collection and preservation is more difficult than its standardized counterpart for collecting physical evidence at a crime scene. The remote collection requires contact and agreements with the cloud service provider. [10] suggests using a separate cloud to store collected data due to its large amount and the peculiarities of the cloud storage structure. Maintaining the chain of custody within cloud forensics is another challenge. For instance, logs can be located within different layers of the cloud, and are sometimes volatile in nature. Lastly, during presentation, it may be difficult for the members of a jury to understand the concepts and technicalities of the cloud. We expect AI model forensics to inherit these challenges. Nevertheless, cloud forensics may help answer some of the essential questions related to AI model forensics.

3.2 Software Forensics

Software forensics is another branch of digital forensics that specifically focuses on "areas of author discrimination, identification, and characterization, as well as intent analysis" based on software source code [21]. One of the original motives for software forensics was disabling the authorship anonymity of distributed malicious code. Any remnants of the malware code left behind on the target system may point back to the author(s). Inspired by handwriting analysis techniques, software forensics adapted new methods to uniquely identify a source

code author based on distinguishing features, such as programming language, coding style, level of expertise, etc.

Source Code Analysis and Authorship Attribution. The four principal applications of software forensics according to [21] are author discrimination, author identification, author characterization, and author intent determination. *Author discrimination* is the task of deciding whether the code was written by a single author or by a number of authors. *Author identification* is the process of attributing authorship to software code, sometimes based on extracted statistics and similarity measures to other compiled code samples. *Author characterization* is similar to suspect profiling in criminal investigations. Its task is to determine personal characteristics of the software creator, such as educational background and personality traits. *Author intent determination* is the principle of determining whether or not software code failures or unexpected behaviours were written purposefully by the software developer or were the result of a human-error during the software development process. The same principles apply to AI model forensics, even though the techniques may be different. AI software is a special case of general-purpose software. Malicious AI software is, however, very different in nature than malicious general-purpose software, such as rootkits and worms. For instance, maliciousness can be hidden in the model parameters rather than being manifested in the flow control directives.

The feasibility of software forensics is questioned in [37]. Author identification may be subject to false positives due to insufficient amounts of data or recurring anti-forensic techniques, such as code reuse from other authors to increase the similarity to legitimate code. However, it has been shown that, with a sufficient amount of data and a careful choice of appropriate and distinctive characteristics, software forensics proves useful in reverse engineering and authorship attribution. [33] later proposed to build an *author profile* based on measurable authorship identifiers, namely program layout, program style, and program structure metrics. [33] also stated that combining techniques from software metric analysis and computational linguistics may lead to a more accurate plagiarism detection and author identification.

The applications of source code analysis and software forensics were extended in [9] to include plagiarism detection. The paper outlined new methods of analysis, such as discriminant analysis, neural network classification, code-based reasoning, and similarity calculation. [5] emphasized on clone detection, debugging, reverse engineering, and the visualization of analysis results. New application areas of source code analysis include middleware, software reliability engineering, and model checking in formal analysis. The paper also outlined the future challenges of software code analysis. Code written by a small number of software engineers is widely reusable by other programmers and can be combined into custom applications and model-based programming. This requires a new approach to source code analysis based on both the models itself and source codes. In a way, the need for AI model forensics was predicted in [5].

More recently, approaches to source code analysis and authorship attribution started utilizing ML and DL techniques. [1] achieved state-of-the-art results in source code authorship attribution based on automatic learning of efficient representations for Abstract Syntax Tree (AST) features. The representation learning is implemented through two types of DL models: Long Short-Term Memory (LSTM) and Bidirectional Long Short-Term Memory (BiLTSM). DL models that were deemed successful in natural language processing (NLP) tasks show good performance for authorship attribution in [17].

Challenges of Software Forensics. Source code such as notebooks, scripts, and model files can be found left after an attack event. Software forensics based on the analysis of source code files may prove difficult in the special case of AI programs because of code reuse. Moreover, AI model files eliminate the need to have access to the original source code. None of the code written by a programmer is actually stored in the model files. Without the source code, it is difficult to tie a model file to a specific author using the current software forensic approaches. This presents a research opportunity for the forensics community.

3.3 Network Forensics

Network forensics is "the use of scientifically proven techniques to collect, fuse, identify, examine, correlate, analyze, and document digital evidence from multiple, actively processing and transmitting digital sources for the purpose of uncovering facts related to the planned intent, or measured success of unauthorized activities meant to disrupt, corrupt, and or compromise system components as well as providing information to assist in response to or recovery from these activities [29]."

Similar to the digital forensics model, network forensics follows the process of preservation, collection, examination, analysis, and reporting. Evidence can be found within the network or outside of the network. Evidence that can be seized from within the network includes device logs, network traffic, and the volatile memory from the devices in question. Evidence processed from outside the network can include internet archives and logs from the domain hosting provider, domain name controller, and internet service providers [7]. The staggering amount of the evidence that can be collected from within and outside the network alludes to some of the challenges within this field. [25] acknowledges that the analysis of log data is extensive because the system has to account for all actions performed. Additionally, those logs must be processed, converted, and compared against a set of accepted misuse and attack patterns.

The future direction of network forensics corresponds to the rise of cloud-based applications and the importance of cloud forensics. [36] recommends that instead of only using packet capture files of network segments for analysis, the investigation needs to include packet capturing and analysis in cloud environments.

4 Case Study: Model Forensics of an Autonomous Need-for-Surgery Classifier

To better understand the context of AI model forensics, we present a case study where an AI-based model decides on surgery, based on patient information. Examples of patient information are: age, life quality, symptoms, past medical records, past family history, medical tests results, etc. In this scenario, the system has to decide on the need for a surgery, whether it is microsurgery, radiosurgery, or just active observation. The system helped medical staff make good decisions. Suddenly, the system started outputting bad decisions, which led to the confusion of medical staff and the occurrence of several deaths. An investigation has to take place.

A digital forensic investigator is hired to examine the system. Some of the hypotheses an investigator may consider are:

- The model has been completely replaced (Malicious). For example, a simple *wget* or *git clone* command in the system logs (or network traces) may reveal that a replacement model has been downloaded. The investigator may find the actual source code and other useful information in the git repository.
- The model has been gradually modified and poisoned (Malicious).
- The model was always malicious, but the malicious aspects remained dormant (Malicious).
- The problem occurred after an update of the libraries or some of the dependencies (Benign).
- A problem in feature extraction and prepossessing occurred following a recent code pull (Benign).
- The addition of new features to the model decreased its accuracy (Benign).

To begin, the investigator needs to determine the following:

- Where the model is hosted: locally in the internal network, or deployed as a web service on a cloud platform?
- What is the access control policy (who has access to the model, and who has access to the hosting system)?
- In the case of cloud deployment, what is the service model (e.g. PaaS, SaaS, or IaaS)?
- Is the hosting system properly protected, patched and updated?

If the model is deployed from the local system, forensic evidence about the model may reside in memory. If the model is served and deployed from the cloud, most of the evidence resides in the cloud; however, some residual evidence may also reside on the end-user machine. Methods of cloud forensics are applicable to collect evidence about the model. Figure 2 depicts possible locations where forensic evidence pertaining to AI model forensics is likely to reside.

It is also important to identify the model's framework and libraries used in order to gather information about the author's intent. The model file has to be examined to determine this information. Typically, most model files have

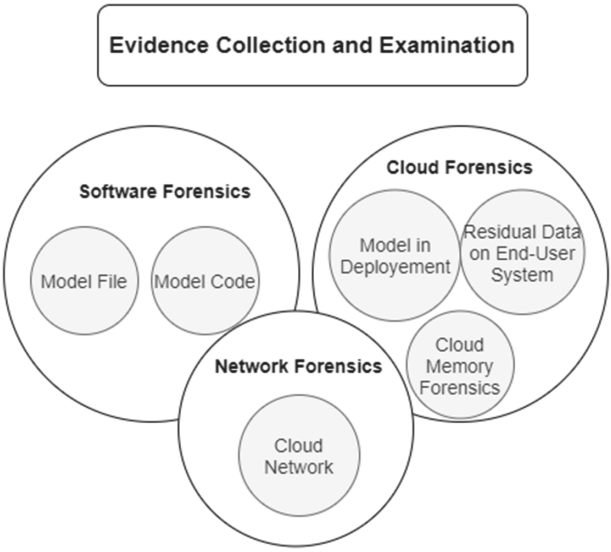

Fig. 2. Diagram of where evidence and information related to AI model forensics is likely to be located

plaintext framework information within the first few lines of the file; however, sometimes this information may not be as easy to find. For example, Fig. 3 depicts the contents of the saved iris classification model in PyTorch. From the files contents, we can see PyTorch libraries that help us correctly identify the model framework. Other information, such as parameters and function calls, can also be seen in the file.

In cases where plaintext is not found within a saved model file, we should still not only rely on the file extension or the header information. Additional fingerprinting is required to provide a clear answer. Figure 4 displays some of the most popular *AI platforms*, *AI tools & frameworks*, and *Model deployment & services*. Popular cloud providers such as Amazon AWS and Microsoft Azure have different paradigms for storing and serving AI models. The diversity, interoperability, and heterogeneity of the AI sphere is in the challenge list of AI model forensics.

The investigator has to check when the model file was last modified, and figure out whether the modification is performed by an adversary or a legitimate user. A typical defense against unsolicited model modification is to create a read-only database of cryptographic hashes for production-plan AI models, store these hashes offline, and regularly compare hashes of the active models to the stored hashes looking for changes.

The more important questions our investigator will face are determining the original intent of the model, who made the modification, was the modification authorized, what exactly was the modification, and to what degree did it alter the model's intent. Software forensic methods and techniques, such as author

```
archive/data.pklFB ZZZZZZZZZZZZZZ
_parametersq ccollections
OrderedDict
_non_persistent_buffers_setq
_backward_hooksq _forward_hooksq _forward_pre_hooksq _state_dict_hooksq _load_state_dict_pre_hooksq layer1q
ctorch.nn.modules.linear
Linear
weightq ctorch._utils
_rebuild_parameter
q ctorch._utils
_rebuild_tensor_v2
storageq"ctorch
FloatStorage
105455744q$X cpuq%K 105462944q0h%K2tq1QK )Rq4tq5Rq6
in_featuresqDK out_featuresqEK2ubX layer2qFh 132915008qJh%M tqKQK )RqNtqORqP 132915168qTh%K tqUQK )RqXtqYRqZ
)RqghDK2hEK 132915488qlh%K<tqmQK )RqptqqRqr 105462544qvh%K tqwQK
archive/data/105455744FB2 ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
archive/data/105462544FB ZZZZZZZZZZZZZZZZZZZZZZZZ
archive/data/105462944FB, ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
archive/data/132915008FB0 ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
archive/data/132915168FB ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZv: archive/data/132915488FB(
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
archive/versionFB ZZZZZZZZZZZZZZZ3
archive/data.pklPK archive/data/105455744PK archive/data/105462544PK archive/data/105462944PK
archive/data/132915008PK archive/data/132915168PK archive/data/132915488PK archive/versionPK
```

Fig. 3. Contents of the saved model file for the iris classification system written in PyTorch

identification, characterization, and intent, may help the investigator find some of the answers to the above questions. Model files offer the internal architecture, parameters of the model, and input features; however, the investigator might not have access to the training and testing dataset, the source code that generated the model, or previous versions of the same model. A list of non-comprehensive techniques that may help the investigation are:

Malware Check whether the host was infected by any kind of malware or was the target of offensive black-hat penetration testing.

Adversarial Samples The investigator may test the sensitivity of the model to adversarial samples generated by well known algorithms. If the model is vulnerable to these samples, the investigator may test logged input vectors to check if they are adversarial as well.

Visualization Techniques such as t-distributed stochastic neighbor embedding (t-SNE) and Uniform Manifold Approximation & Projection (UMAP) may help determine whether the classifier is a poor one or whether the input data is noisy and inconsistent.

XAI The investigator may employ well-known eXplainable AI (XAI) techniques to generate explanations for the inquiry decisions. The explanations can be checked by experts to see if they make sense or not. Good explainers tell a lot about possible model overfitting, underfitting, or adversarial behaviour. In case the explanations are incoherent with human expert reasoning given the input features, the investigator has to turn back to earlier parts in the AI pipeline such as data inputting and feature extraction.

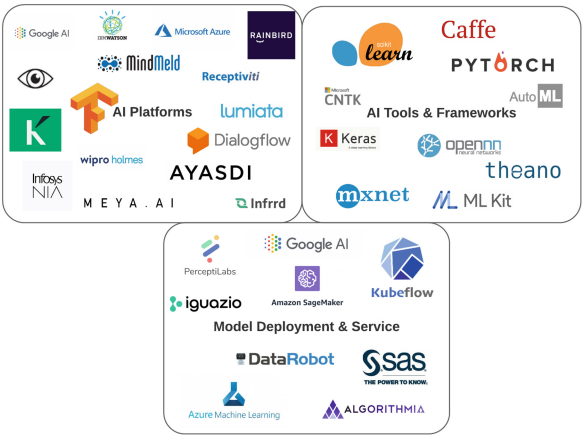

Fig. 4. Diagram of popular AI platforms, AI tools & frameworks, and model deployment & services

Note that the above list is not comprehensive. Digital forensics, in general, is a domain requiring a lot of innovation and where new techniques continuously arise. The overall goal of using these techniques in AI model forensics are to distinguish factors, such as author identification, author characterization, and author intent, if possible. The exact methods for utilizing these techniques and answering the aforementioned investigative questions need to be established. Storing and arbitrating AI models using Blockchain helps preemptively solve a lot of issues for AI model forensics. We discuss this context in the next section.

5 Blockchain and AI Model Forensics

Blockchain systems provide properties of transparency, visibility, immutability, traceability, nonrepudiation, and smart contracts. These properties reveal important information for AI model forensics; it makes it straightforward to answer questions about who is responsible for the inquired model behaviour.

All the transactions taking place within a Blockchain are stored in a public, decentralized, and append-only ledger. The ledger is accessible to anyone with access to the Blockchain and is secure and tamper-proof. Each transaction in the Blockchain is cryptographically signed by all involved parties. The signed transactions must be verified by a majority of the Blockchain users before being officially added to the ledger. The Blockchain provides an ecosystem where AI developers, utility providers, deployment engineers, and other participants can interact. This allows AI models to be tracked back to their origins. Such an ecosystem supporting both AI and XAI is described in [27].

With Blockchain support, the causes of misconduct of an AI model and the liability for bad behaviour become easily identified. During investigation, the AI model left over is compared to the one stored in the Blockchain. If there is no

match, the model has been modified without the consent of the participants. If there is a match, the investigation may trace the source of the buggy behavior causing the deficit. Examples include a library update, a previously committed buggy version, a shortage in training data, or a misconfiguration in the number of training epochs. Using Blockchain in this way can offer potential solutions to author identification, which can further lead to author characterization. That said, we cannot expect that a wide adoption of Blockchain for arbitrating AI models is occurring anytime soon. AI model forensics must not solely rely on the existence of cryptographic ledgers.

Intellectual Property or copyright can be managed through transactions and smart contracts. Privacy of the entities involves the life-cycle of the AI model, which can be achieved by using public keys instead of real identities. For instance, identities such as the forensic investigator, a witness, or an AI expert can be masked using public keys. Of course, the topic of blockchain for AI forensics can be further developed and detailed in a future research. Blockchain as a solution is promising but needs more investigation in future work.

6 AI Model Ballistics

The first documented use of ballistics in forensics occurred in 1835 [39]. Matching the unique bullet mold to one in the possession of the suspect led to a confession and eventual conviction. Forensic ballistics has evolved and now the microscopic indentations on the bullet and cartridge case after a gun is fired can be used to create a ballistic fingerprint [28]. When the term ballistics is used in relation to forensics, conventionally, it implies firearm analysis. However, in recent years some areas have adapted this term, in particular, camera ballistics. Camera ballistics uses the camera sensor to match a photo to a camera, like a bullet to a gun [24]. In a similar fashion, AI model ballistics is needed to match a creator to an AI model.

The motivation behind AI model ballistics is to identify and manipulate AI model artifacts in a crime scene investigation in a forensically sound manner. The goal of AI model ballistics is to identify relevant information for the investigation, such as: the type and creator of an AI model, the intention of an AI model, the compliance of the model's behaviour with the creator's intended behaviour, the framework, tools and datasets used in model creation, and more. Fingerprinting a model is another way to view AI model ballistics. To acquire the *fingerprint* of a model, an investigator would need to interrogate a model, such as passing various inputs and analyzing the outputs.

For this reason, we tentatively define AI model ballistics as the subfield of AI model forensics pertaining to the processing and analysis of AI model artifacts left behind. These artifacts could potentially lead to the identification of the model's creation framework, creator's intent (benign or malicious), and any other relevant identifiers or information that can be extracted.

7 Challenges of AI Model Forensics

While incorporating methods from cloud, software, and network forensics, AI model forensics inherit their challenges and manifest unprecedented ones. For example, an AI model stored in the cloud inherits a major limitation of cloud forensics, which is the inability to physically acquire and seize evidence. In standard digital forensic investigations, the digital device in question is collected for analysis; however, in cloud forensics the evidence cannot be physically collected and the data can be dispersed through numerous servers and architectures. The AI model service may be dependent on other web micro services or federated learning. Investigation relies on help from cloud service providers, which makes the chain of custody and the validity of the evidence less credible.

Other challenges include legal issues, such as jurisdictional boundaries and obtaining a search warrant. AI models are generated by AI code that is linked to different developers, researchers, and providers. AI models are also typically located in shared storage and use shared computational resources. User privacy in these settings is an additional challenge. It is difficult investigating one user's data without accessing and violating the privacy of other users.

Applying methods of software forensics are not directly applicable to AI models and source code. The swift development in AI frameworks and deployment models makes the search space even larger. AI model forensics must be able to determine as much information as possible about the development frameworks and libraries, authorship attribution, training and testing history or logs, adversarial behaviour, or intentionally designed malicious backdoors.

In addition, adversarial attacks and anti-forensics are always a concern in any form of digital forensics. We expect adversaries to perform anti-forensics in many different ways and hinder the investigation.

8 Conclusion

In this paper, we sketched the motivation and the description for a sub-domain of AI forensics that we dubbed AI model forensics. We elaborated on the background and nature of AI model development and deployment, and highlighted the fact that these models can be replaced, trojanized, gradually poisoned, or fooled by adversarial input. We suggested an extended definition of AI model forensics, namely examining left-behind AI models, and their surrounding artifacts, such as logs, traces, service code, source code, datasets, and input samples with the goal of helping an investigation reach correct conclusions about a security incident. In particular, we focus on the identity of the model creator and the intention of the model, which is our motivator for the new subfield AI model ballistics.

We explored the relationships and the dependencies of the newly arising sub-domain with the literature work from software forensics, highlighting the importance of author identification, characterisation, and intent and cloud and network forensics, focusing on the challenges faced in the field. The use-case

showed the peculiarities of AI model forensics in a proper context. Blockchain is a possible solution to support AI model forensics, but we cannot assume that AI models are tracked and managed by a Blockchain in all cases. AI model ballistics is a new topic as it relates to identifying the type, owner, and intent of a AI system. Finally, we approached the challenges of this novel field. Our work is a step towards further exploring and defining AI model forensics and model ballistics as an interesting and fascinating subfield under the umbrella of AI forensics.

Future work should explore applying and creating scenarios to show how to apply a sound methodology for AI model ballistics in end-to-end investigation use-cases and exploring their artifacts. Each scenario should address both malicious and benign incidents of model files to find an appropriate solution. For instance, each hypothesis presented in the case-scenario will be further explored to find resolutions. Future work should also explore the contents of a model file in depth for a better understanding of how software forensic techniques can be applied.

Acknowledgements. This material is based upon work supported by the National Science Foundation under Grant Number 1921813. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Alsulami, B., Dauber, E., Harang, R., Mancoridis, S., Greenstadt, R.: Source code authorship attribution using long short-term memory based networks. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) ESORICS 2017. LNCS, vol. 10492, pp. 65–82. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66402-6_6
2. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., Mané, D.: Concrete Problems in AI Safety (2016)
3. Behzadan, V., Baggili, I.M.: Founding the domain of AI forensics. In: SafeAI@ AAAI, pp. 31–35 (2020)
4. Behzadan, V., Hsu, W.: Sequential triggers for watermarking of deep reinforcement learning policies. arXiv preprint arXiv:1906.01126 (2019)
5. Binkley, D.: Source code analysis: a road map. In: Future of Software Engineering (FOSE 2007), pp. 104–119 (2007). https://doi.org/10.1109/FOSE.2007.27
6. Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017)
7. Datt, S.: Learning Network Forensics. Community Experience Distilled. Packt Publishing, Birmingham (2016)
8. Digital Forensics Market: Market Research Firm (2018). https://www.marketsandmarkets.com/Market-Reports/digital-forensics-market-230663168.html
9. Frantzeskou, G., MacDonell, S., Stamatatos, E.: Source code authorship analysis for supporting the cybercrime investigation process. In: Proceedings of the 1st International Conference on E-Business and Telecommunications Networks, pp. 85–92 (2004)

10. Grispos, G., Storer, T., Glisson, W.B.: Calm before the storm: the challenges of cloud computing in digital forensics. Int. J. Digit. Crime Forensics (IJDCF) **4**(2), 28–48 (2012)
11. Herman, M., et al.: NIST cloud computing forensic science challenges. Technical report, National Institute of Standards and Technology (2020)
12. Jeong, D.: Artificial intelligence security threat, crime, and forensics: taxonomy and open issues. IEEE Access **8**, 184560–184574 (2020)
13. Jiang, F., et al.: Artificial intelligence in healthcare: past, present and future. Stroke Vasc. Neurol. **2**(4), 230–243 (2017). https://doi.org/10.1136/svn-2017-000101
14. Karpisek, F., Baggili, I., Breitinger, F.: Whatsapp network forensics: decrypting and understanding the whatsapp call signaling messages. Digit. Investig. **15**, 110–118 (2015). https://doi.org/10.1016/j.diin.2015.09.002. https://www.sciencedirect.com/science/article/pii/S1742287615000985
15. Kent, K., Chevalier, S., Grance, T., Dang, H.: Guide to integrating forensic techniques into incident response. NIST Special Publication 800–86 10(14) (2006)
16. Konečný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: strategies for improving communication efficiency. CoRR abs/1610.05492 (2016). http://arxiv.org/abs/1610.05492
17. Kurtukova, A., Romanov, A., Shelupanov, A.: Source code authorship identification using deep neural networks. Symmetry **12**(12) (2020). https://doi.org/10.3390/sym12122044. https://www.mdpi.com/2073-8994/12/12/2044
18. Levinson, J., et al.: Towards fully autonomous driving: systems and algorithms. In: 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 163–168 (2011). https://doi.org/10.1109/IVS.2011.5940562
19. Li, J.: Cyber security meets artificial intelligence: a survey. Front. Inf. Technol. Electron. Eng. **19**(12), 1462–1474 (2018). https://doi.org/10.1631/FITEE.1800573
20. Li, Z., Hu, C., Zhang, Y., Guo, S.: How to prove your model belongs to you. In: Proceedings of the 35th Annual Computer Security Applications Conference (2019). https://doi.org/10.1145/3359789.3359801
21. MacDonell, S.G., Buckingham, D., Gray, A.R., Sallis, P.J.: Software forensics: extending authorship analysis techniques to computer programs. JL Inf. Sci. **13**, 34–69 (2002)
22. Mell, P., Grance, T., et al.: The NIST definition of cloud computing. NIST Special Publication 800–145 (2011)
23. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**, 529–33 (2015). https://doi.org/10.1038/nature14236
24. MOBILedit: Camera Ballistics. https://www.mobiledit.com/camera-ballistics
25. Mukkamala, S., Sung, A.H.: Identifying significant features for network forensic analysis using artificial intelligent techniques. Int. J. Digit. Evid. **1**, 1–17 (2003)
26. Nassar, M., Itani, A., Karout, M., El Baba, M., Kaakaji, O.A.S.: Shoplifting smart stores using adversarial machine learning. In: 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), pp. 1–6. IEEE (2019)
27. Nassar, M., Salah, K., ur Rehman, M.H., Svetinovic, D.: Blockchain for explainable and trustworthy artificial intelligence. Wiley Interdiscip. Rev. Data Mining Knowl. Discov. **10**(1), e1340 (2020)
28. NIST: Ballistics (2021). https://www.nist.gov/ballistics
29. Palmer, G.: A road map for digital forensic research. Technical report. DFRWS (DTRT0010-01) (2001)

30. PyTorch: PyTorch tutorials: saving and loading models (2017). https://pytorch.org/tutorials/beginner/saving_loading_models.html#saving-loading-model-for-inference
31. Ruan, K., Carthy, J., Kechadi, M.T., Baggili, I.: Cloud forensics definitions and critical criteria for cloud forensic capability: an overview of survey results. Digit. Investig. **10**, 34–43 (2013)
32. Ruan, K., Carthy, J., Kechadi, T., Crosbie, M.: Cloud forensics. In: Peterson, G., Shenoi, S. (eds.) Advances in Digital Forensics VII (2011). https://doi.org/10.1007/978-3-642-24212-0_3
33. Sallis, P., Aakjaer, A., MacDonell, S.: Software forensics: old methods for a new science. In: Proceedings 1996 International Conference Software Engineering: Education and Practice, pp. 481–485. IEEE (1996)
34. Schneider, J., Breitinger, F.: AI forensics: did the artificial intelligence system do it? Why? (2020)
35. Shah, J.J., Malik, L.G.: Cloud forensics: issues and challenges. In: 6th International Conference on Emerging Trends in Engineering and Technology, pp. 138–139 (2013). https://doi.org/10.1109/ICETET.2013.44
36. Sikos, L.F.: Packet analysis for network forensics: a comprehensive survey. Forensic Sci. Int. Digit. Investig. **32**, 200892 (2020). https://doi.org/10.1016/j.fsidi.2019.200892. https://www.sciencedirect.com/science/article/pii/S1742287619302002
37. Spafford, E.H., Weeber, S.A.: Software forensics: can we track code to its authors? Comput. Secur. **12**(6), 585–595 (1993)
38. TensorFlow: TensorFlow core: save and load models (2021). https://www.tensorflow.org/tutorials/keras/save_and_load#save_the_entire_model
39. Tilstone, W., Tilstone, W., Savage, K., Clark, L.: Forensic Science: An Encyclopedia of History, Methods, and Techniques. ABC-CLIO (2006). https://books.google.com/books?id=zIRQOssWbaoC
40. Wang, Z., Liu, C., Cui, X.: Evilmodel: hiding malware inside of neural network models. arXiv preprint arXiv:2107.08590 (2021)
41. Zhang, J., et al.: Protecting intellectual property of deep neural networks with watermarking. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security, ASIACCS 2018, pp. 159–172. Association for Computing Machinery (2018). https://doi.org/10.1145/3196494.3196550