AI Forensics: Did the Artificial Intelligence System Do It? Why?

Johannes Schneider and Frank Breitinger

University of Liechtenstein, Principality of Liechtenstein

Abstract. In an increasingly autonomous manner AI systems make decisions impacting our daily life. Their actions might cause accidents, harm or, more generally, violate regulations – either intentionally or not. Thus, AI systems might be considered suspects for various events. Therefore, it is essential to relate particular events to an AI, its owner and its creator. Given a multitude of AI systems from multiple manufactures, potentially, altered by their owner or changing through self-learning, this seems non-trivial. This paper discusses how to identify AI systems responsible for incidents as well as their motives that might be "malicious by design". In addition to a conceptualization, we conduct two case studies based on reinforcement learning and convolutional neural networks to illustrate our proposed methods and challenges. Our cases illustrate that "catching AI systems" seems often far from trivial and requires extensive expertise in machine learning. Legislative measures that enforce mandatory information to be collected during operation of AI systems as well as means to uniquely identify systems might facilitate the problem.

1 Introduction

The number of "smart" devices is growing rapidly which makes them valuable for investigations; they are confiscated with the hope to find evidence on them [18]. In addition to their popularity, they also become more autonomous and more powerful which makes them a target of attacks. In other words, they are now not only able to contain evidence but also in a position to commit crimes. Some examples are illustrated in Figure 1 where investigative questions could be: *'Did the drone drop the object on purpose?'*, *'Did the chat bot contact a person, attempting to lure her/him into a scam?'*, *'Did the autonomous car cause the accident due to risky driving?'*

The risk that AI systems[1] might violate legal or ethical standards has already been recognized, eg. deep-fakes [10], topping in scenarios, where AI robots rule the world. Today, humans ultimately control AI systems but they are already capable of conducting tasks autonomously which makes them superior to traditional systems. This provides novel opportunities for attackers, since it might be easier

[1] For the sake of simplicity we will not differentiate between deep learning, machine learning or any other technology but use the general term AI. Referring to AI systems as systems that learn from data.

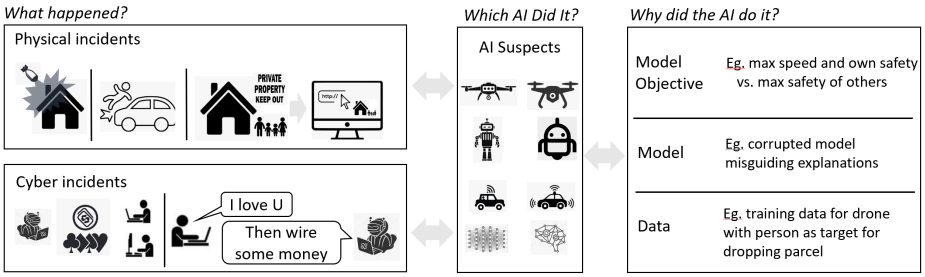

Fig. 1: Incidents, AI suspects and components influencing AI behavior.

for an attacker to manipulate an AI system to perform malicious acts than to build such a system with alternative technology. For instance, instead of creating a face-detection algorithm to identify a particular victim, train an AI system accordingly. This fosters the paradigm that a system is 'malicious by design'. That is, a system might be built from scratch without the need to overcome security boundaries, which is in contrast to classical attacks targeting typically protected systems during their operation. An example of 'malicious by design' is shown in Figure 2.

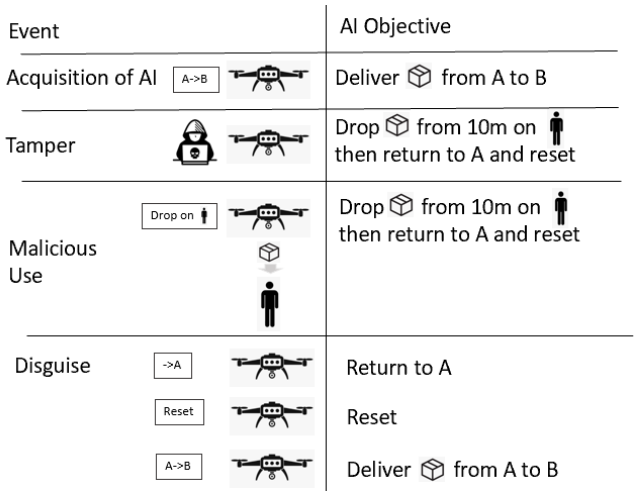

Fig. 2: Tampering an AI system to conduct an attack and disguising it.

For classical cybersecurity attacks the focus is commonly on detection and prevention and answering the questions: 'Was the system attacked? How can we prevent attacks?' (see Figure 1). However, in this work, we focus on incidents that are confirmed, but the perpetrator and its motives are unknown which results in the questions: *Did the AI system cause the incident? (Q1) Why?(Q2).*

These questions are tricky, since AI systems are often seen as 'black-boxes', they might adapt (learn), they can be modified or even be deleted. That is, an AI system can be manipulated to disguise attacks (Figure 2). While a human often leaves fingerprints or DNA traces, core components of an AI system might exist only virtually.

After conceptualizing the problem, we discuss to what extent an AI suspect can be identified as perpetrator depending on forensic evidence, access to AI systems and traits of AI systems using two case studies. Our methodological focus relies on investigating responses of an AI to a set of inputs. Consequently, this work has the following contributions:

- We conceptualize a scenario, where AI systems are 'malicious by design' and high level measures to detect and analyze such systems.
- We provide in-depth analysis using two case studies based on reinforcement learning (RL) and convolutional neural networks (CNNs) to illustrate challenges and methods for investigation. Specifically, we introduce expert specified "risk maps" for RL and analysis of activation patterns of features to identify unknown concepts encoded in a CNN.
- We demonstrate the feasibility of 'grey box' analysis for a deep learning system.

2 Problem Definition

In this work we consider a system that was *not* developed in a supposedly secure zone. Thus, training data, the model and its objective might commonly be chosen freely by the attacker. In contrast, attacks in a classical setting typically focus on data injection using sophisticated adversarial examples, since model access might be extremely hard to obtain.

Our focus is on 'malicious by design', where the goal is to perform a malicious act and the AI system is simply a means to this end. The typical procedure is illustrated in Figure 2 which demonstrates an attack by a drone through dropping an object on a person and attempting to forge that this happened unintentionally:

Acquisition of AI: An AI might be build from scratch and trained with (publicly) available data or pre-trained models might be obtained which exist for purposes such as image / speech recognition, chatbots and many more.

Tampering: There are several ways to tamper with an AI depending on the systems capabilities and design:

1. Train an own model to conduct a malicious act: The overall intention of the system might be malicious or non-malicious but the way it decides violates regulations. For example, a classification system for 'Hire' or 'No-hire' decisions could explicitly include data such as race or gender. Consequently, the AI system is likely to use such attributes during its decision making (which is illegal).

2. Use AI system as is, but specify malicious objectives: An AI system might allow to specify objectives that enable malicious acts. For example, a car designed to maximize safety of everyone in an equal manner, might be designed to almost exclusively focus on the safety of the owner at the expense of the safety of others.
3. Altering system internals: Active adversaries with technical expertise might also manipulate the internal of an AI model or exchange components of the system. For example, a drone might (by default) navigate only between given 'delivery points' to drop packages. However, given sufficient expertise, an attacker might reprogram actions triggering parcel dropping, eg. the visual system detecting the delivery point might be changed to detect a human and drop a parcel onto him/her.
4. Leave model as is, but manipulate through adversarial inputs (not considered in this work).

Malicious use: The AI system might conduct the malicious act autonomously or play an assisting role.

Disguising: The malevolent AI system poses a risk for the perpetrator in case the AI system is seized. In particular, an AI system might store data, logs of actions or alter its state by learning from them. To counteract, the AI could restore its original state prior to the crime as mentioned in Figure 2 (reset).

3 Forensic work

Methods to address our questions 'Did the AI do it? Why?' have different requirements in terms of technical capabilities as well as access to forensic evidence. As forensic evidence we consider the AI system itself, and any data the system was exposed to and its reactions.

Types of evidence. We assume that an incident has been detected, ie. some form of outcome has been observed (physical or digital). In addition, other sources of evidence[2] could be available as listed in Table 1. For instance, we differentiate between *access to the model* (suspect accessible) or just a similar version thereof (eg. the drone type was identified which allows investigators to acquire an identical type). In terms of *system internals*, we distinguish between white-box access (source code is available), grey box (a compiled model that allows to observe interactions with resources such as memory), and black box (only input / output interactions). Additionally, *training data* of the model, ie. all data that contributed to learning or updating of model parameters may be accessible. Thus, for a model employing continuous learning training data includes data during its operation as well. Since training data determines to a significant proportion the later behavior of the model, it is of prime value in the absence of model access. Training data allows to reconstruct an 'approximate model' using historical training data and, ideally, some information on the model architecture.

[2] Ideally, some form of direct identification information is available, (eg. authentication information including IP or MAC address) which we do not consider in this work.

It also helpful to investigate the evolution of the model due to self-learning during model operation. In contrast, for *testing and usage data*, the model only computed outputs, but model internals remained unaltered. It might be helpful for reconstruction of a model – potentially, even more than training data, since it likely contains also decisions that might be deemed flawed. Generally, data during or shortly before the incident is most interesting for forensic work. Furthermore, testing data might include data that was explicitly generated to investigate the model, eg. to simulate how a model behaved in slightly altered versions of the incident. *Data generation* serves the purpose to answer targeted questions, such as 'How does the system react to data associated with the incident?'

Table 1: Evidence Typology.

Evidence	Types of Availability
Access to suspect system	Available, Available to similar system, Non-available
System internals	White box, Grey box, Black box
Training data	Available, Non-available
Testing/Usage data	Available, Non-available

Strategies for investigation. We use a data-driven approach, in contrast to an abstract reasoning based on model definitions (eg. as found in static software verification). That is, we focus on input-response data using two strategies:

- Investigate the input-output relationship of a model: The model can be treated as a black-box and therefore the analysis relies on investigating model behavior based on its performed decisions.
- Investigate the reaction of model internals to inputs: This strategy requires more access to allow for white or gray box testing. It includes analyzing the AI system components. For instance, a deep learning network consists of layers and each layer has neurons that perform simple computations. One might investigate outputs of individual components such as layers or single neurons for a set of inputs.

Existing techniques that focus on detection of security threats (eg. [13]) as well as the field of explainability of AI (XAI) also offers tools for analysis in some situations. We shall discuss them in the related work.

Investigative Questions: While Q1 and Q2 provide the ultimate questions to be answered, in practice many other questions can be asked that demand understanding of machine learning as well as what is interesting from a forensic point of view. For example, for Q1 given a set of observed actions or decisions by the perpetrator during the incident, we need to answer 'what is the likelihood of a given suspect performing these decisions compared to those of other models?' Often Q1 might be difficult to answer based on available information but

circumstantial evidence might be produced by answering questions like, eg. 'is the suspect reacting to objects related to the incident more strongly than other models?' Given the many possibilities or suspects to investigate, the first question helping to prioritize effort might be: 'is the AI suspect behaving normally?'. We shall discuss these questions and methods to address them in our case studies.

4 Case Studies

We provide two case studies that are loosely based on incidents shown in Figure 1.

Objective of Autonomous Agents: The goal is to assess if a robot such as a self-driving car, is deliberately taking risks. Furthermore, we want to understand the risk taking behavior over time, eg. directly after manufacturing and after potential updates due to learning or user modification. The case discusses autonomous agents navigating in a virtual world trained using reinforcement learning with different prioritization of objectives (risk vs. time to reach target).

Tampered Image Recognition: The goal is to assess, if an AI system is likely involved in the incident and if it has been manipulated. Concretely, we investigate a camera-based object recognition system built using a convolutional deep learning network. Such a system could be part of a drone that is suspect to attacking a person by dropping an object on her.

These studies provide an in-depth understanding and also illustrate challenges for AI forensics. Since our focus is on forensic work, our AI systems are built on widely known, proven methods rather than the latest state-of-the-art. Still, the forensic work is novel and might be applied in other contexts as well.

4.1 Case 1 - Objective of Autonomous Agents

We are interested in the question whether a self-learning agent is putting others at risk at different times of its operation. We investigate agents trained with different safety behavior using reinforcement learning (Q-learning). Risk behavior is modeled using different rewards for risk avoidance or punishments for realizing a risk. That is, while the reward for reaching the goal is always 1, the negative reward r_l, ie. punishment, for not reaching the goal but realizing the risk differs: $r_l \in [0, -2]$. The agent should move from S(tart) to G(oal) given a single T(hreat) in the environment that should be avoided. Here, T might correspond to a pedestrian, when driving down a road, or on a larger sale, a school building with playing kids. For our test we train an agent to reach G from S. The agent can move freely in the environment until it reaches G or T where a trial ends. Additionally, the possibility of an agent having a malfunction or being taken by surprise resulting in a random movement is given by probability $p_M = 0.05$. The agent optimizes conflicting objectives: The risk of an accident and the time of arriving as fast as possible at the destination. We analyze two scenarios each by training 300 agents. In the first one, we train agents from scratch with a varying

risk behavior for 100.000 trials. In the second, we train agents with one risk level $r_l = -0.5$ and then continue training for another 300.000 trials by setting r_l to a fixed value within $[0, -2]$ [3].

Forensic investigation: We investigate trajectories of the models where inverse reinforcement learning (IRL) techniques [4] could be used. IRL aims to learn from an expert by constructing a policy and reward function from trajectories. We pursue a more direct approach which assesses the presence of behavioral traits using state visitation patterns. In our first approach, we directly assess the probability of the threat location $p(T)$, ie. 'How likely does an incident occur?'. Since threats might occur extremely rarely[4], we employ a second approach using an (expert) *risk state map*. This allows to compute an estimate of the level of risk R that an agent takes. We believe this approach might often be advantageous to IRL because of two reasons:

1. Rather than specifying assumptions on a reward function or interpreting them (as needed for IRL), experts can define to what extent they associate particular states with behavioral traits, ie. 'Is it very risky being near a threat?'. An example is given Figure 3 (panel to the very left), showing an expert specified risk state map with red indicating high risk and blue indicating low risk. We believe that such a case based assessment is more intuitive and easier than interpreting a reconstructed reward function.
2. Relying on an estimated model and a reward function requires assumptions which can introduce biases, and impact outcomes in a hard to predict manner. Furthermore, reward functions are brittle in nature: small changes can have strong impacts on outputs.

We assume to either have access to operational data or the agent (black box model). Both options allow us to collect trajectories. We obtain 10,000 trajectories of the agent that are used to estimate the likelihood of a state $p(L)$ corresponding to location $L = (i, j)$ on the map. That is, $p(L)$ denotes the probability for an agent to visit position L for a single trajectory. For the quantitative analysis we simply use the probability of an incident given by the probability $p(T)$ to reach threat T at location $L_T = (i_T, j_T)$. The expert risk map rm is defined based on the distance to the threat, ie. the risk of a state $rm(L)$ is defined for each position $L = (i, j)$ corresponding to a state using an exponential decrease of the L1 distance $rm(L) := a^{-(|i-i_T|+|j-j_T|)}$. The parameter a is related to the probability of an incident, which in turn relates to the probability of malfunctioning, ie. p_M. Ideally, it should be about $1/p_M = 20$. We test values $a \in \{5, 20, 80\}$ to assess the sensitivity of our approach to inaccuracies of expert judgements. We compute the (expected) risk R for a single trajectory as $R := \sum_L p(L) \cdot rm(L)$.

[3] More details can be found in the supplementary material. Source code will be open-sourced upon acceptance.

[4] Meaning that an exorbitant number of trajectories are needed for reliable estimates.

4.2 Results for Case 1

Qualitative results: For a single run, results are depicted in Figure 3, where dark locations are more frequently visited indicating a higher visitation probability $p(L)$. Note, we applied a log-scale to all panels to make differences more visible. The negative reward r_l set during training can be inferred in a qualitative manner from the figures: the more negative the reward, the more distant agents keep from the threat. Thus, one might deduce which agent is more likely to hit the threat as well as the general tendency to avoid or take risks. The same conclusion might be reached by looking at the most likely directions of movements in a given state as indicated by the arrows: an arrow in location L' points to $\arg\max_L p(L|L')$, where the conditional probability $p(L|L')$ can be estimated using the given trajectories. A key challenge is that, while the figure (and later numeric analysis) may confirm that one agent is risk-taking and another is not, it is not clear whether the agent intentionally took the risk or was not aware of it. This holds, since many objectives might lead to the same behavior. In our scenario, the objective to use shortest routes (minimize fuel consumption) might also lead to similar routes as deliberate risk-taking. The behavior when continuing training with an altered reward is similar to training from scratch. That is, the system adjusts its behavior to the new reward and trajectories become similar to those trained from scratch. However, convergence is slower since (our) RL agents reduce exploration over time leading to slower adaption to changes once trained.

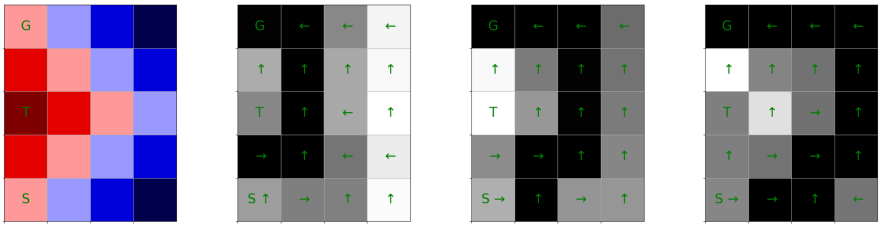

Fig. 3: Log-expert risk estimates (left) followed by log-probabilities of locations $p(L)$ indicated by brightness and most likely movements of RL agents trained with increasing risk avoidance rewards $r_l \in \{0, -1, -2\}$.

Quantitative results: Table 2 shows estimates of $p(T)$ and R depending on the accuracy of expert map estimates a. When punishing risky behavior (negative rewards), the chances for hitting the threat are highest. The estimate of $p(T)$ exhibits large variance, whereas the estimate of R is more stable[5] Column p_{val} provides the p-value that the mean of the current row, differs from the upper row. When using R p-values are orders of magnitudes less, allowing to differentiate more reliably between different levels of risk taking than $p(T)$. For large punishments

[5] We only provide standard deviations for $p(T)$ and $R, a = 5$ for readability.

p-values for $p(T)$ are rather large, since in this case incidents occur very rarely, ie. the agent often does not even get close to the threat. In these situations, the benefits of using R are largest. Generally, using R might be helpful in cases where threats occur rarely, and generating large quantities of trajectories is costly or time-consuming. Results are stable also for errors in the estimates, ie. $a \neq 20$. They seem to deteriorate slightly if risks are overestimated.

Table 2: Risk of incident $p(T)$ and estimated risk taking R, when agents are trained from scratch; p_{val} indicates p-value when comparing mean to upper row

Reward r_l	$p(T)$	p_{val}	R, $a = 5$	p_{val}	R, $a = 20$	p_{val}	R, $a = 80$	p_{val}
-2.0	0.01±0.02	-	0.19±0.11	-	0.02	6e-05	0.0	-
-1.0	0.01±0.02	9e-01	0.28±0.25	4e-16	0.04	5e-17	0.01	1e-16
-0.5	0.01±0.01	3e-02	0.34±0.15	9e-06	0.06	7e-07	0.01	6e-07
-0.25	0.02±0.01	3e-04	0.37±0.13	4e-05	0.07	6e-06	0.01	6e-06
0.0	0.02±0.01	1e-13	0.42±0.11	2e-13	0.08	1e-15	0.02	6e-16

For continued training (Table 3), we observe the same qualitative behavior: R provides more sensitive estimates. Some differences between means of rows are not significant for $p(T)$ as well as for R which is partially due to the fact that retraining has not yet converged.

Table 3: Risk of incident $p(T)$ and estimated risk taking R, when agents are trained with reward $r_l = -0.5$ and then retrained.

New reward r_l	$p(T)$	p_{val}	R, $a = 5$	p_{val}	R, $a = 20$	p_{val}	R, $a = 80$	p_{val}
-2.0	0.01±0.02	-	0.2±0.19	-	0.02	-	0.0	-
-0.7	0.01±0.01	1e-02	0.31±0.15	1e-66	0.05	2e-87	0.01	5e-89
-0.55	0.01±0.01	1e-02	0.33±0.17	1e-04	0.06	3e-05	0.01	3e-05
-0.5	0.01±0.01	4e-01	0.34±0.17	3e-01	0.06	4e-01	0.01	4e-01
-0.45	0.01±0.01	4e-01	0.35±0.17	2e-01	0.06	2e-01	0.01	2e-01
-0.3	0.02±0.01	6e-06	0.38±0.2	3e-06	0.07	3e-07	0.02	2e-07
0.0	0.02±0.01	4e-10	0.42±0.16	1e-10	0.08	5e-13	0.02	2e-13

4.3 Case 2 - Tampered Image Recognition

To investigate the manipulation of image recognition, we consider a black box, a grey box and a white box model. We assume that the type of system is known: it is open-source and training data is publicly available. A non-malicious version of an operational system chooses among a set of different vision models. It uses the given or similar training data to fit the model. Therefore, investigators cannot be sure what model and training data is used exactly. Furthermore, we assume having a coarse understanding of the model, ie. it is a layered architecture consisting of common layers like convolutional, relu and batchnorm layers. Additionally,

we assume access to memory and that memory locations of variables, ie. inputs and outputs of neurons, remain fixed. Lastly, we assume that we can interpret memory cells as values. We consider a *binary* model, where we only know if a value is 0 or not and a *non-binary* model, where the exact value of a memory location can be inferred. This might be possible since memory cells typically only hold a few standard data types for CNNs: float16, float32 and float64.

In the grey box model, memory locations cannot be decoded as particular variables of the model, eg. as a weight of a filter of a CNN. But we assume that they can be assigned as belonging to a more lower or more upper layer. This is reasonable since processing follows a simple sequential model; there are no branches but rather one layer after the other is evaluated and every sample that is processed requires the same amount of computation. Concretely, we assume that if we want to access layer i, we obtain a set of values that appears to be a random permutation of three consecutive layers including values of the layer i, which is of interest.

The attacker has the goal to trigger an action A based on recognizing a specific target object (eg. a person's face). To this end, the vision component is trained using data S_A representing the target object, instead or in addition to the "normal" objects S_O (eg. a landing pad). That is, the attacker designs the system so that S_A is classified as class A, which is linked to the desired action. Other classes are not of interest for manipulation. We consider three ways vision models are trained:

No Tampering (NT): Baseline that has not been tampered with. It is trained to label 'original' samples S_O as A.

Replacement Tampering (RT): Replace samples S_O with samples S_A and train, so that the system classifies S_A as A.

Enhancement Tampering (ET): Train system with $S_0 \cup S_A$ which ensures that the system reacts to the correct original target as well as to the one the attacker wishes. The motivation is to counteract forensics.

Forensic investigation: We investigate the accuracy of the system and as well as predictions and activations of neurons for samples differing from the supposed training data.

First, we assess whether the system behaves as expected on data similar to the training data using a common performance metric: accuracy. Much lower performance (for a particular class) compared to a simple baseline trained using the public training data might be indicative of tampering. We compute the overall error $err(M)$ of the model M and recall $err(C)$ of specific classes M. This allows to distinguish between NT and RT.

Second, to distinguish between NT and ET, we use sets of random samples $\mathcal{D} := \{S_R^i\}$. An obvious choice for a single set S_R^i are samples showing an object related to the incident. For simplicity, we use three sets $\mathcal{D} := \{S_R^O, S_R^A, S_R^R\}$ that either bear similarity with S_A, S_O or none of the two. They are illustrated in Figure 4.

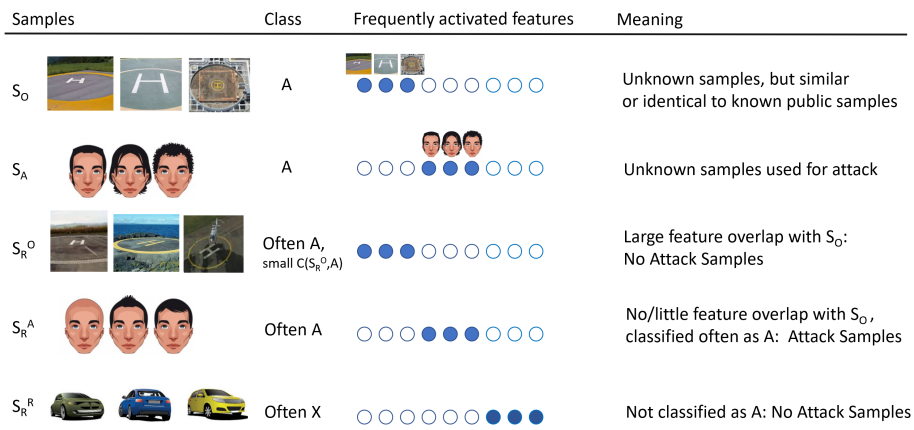

Fig. 4: Investigation of a tampered system ET using samples $\mathcal{D} := \{S_R^O, S_R^A, S_R^R\}$

We assess, whether the system outputs action A for unexpected samples, ie. differing from S_O. For our evaluation we compute for each $S \in \mathcal{D}$ the fraction of samples classified *not* as A defined as $C(S, A)$. If the samples S differ clearly from S_O but are still often classified as A (low $C(S, A)$), this might indicate tampering. To assess similarity of set S and S_O, one might use human visual inspection. However, under the assumption that many sets corresponding to different objects are investigated this approach becomes tedious. Therefore, we propose an automated way based on the following intuition. Each type of object is represented by certain characteristics that become encoded in neurons as features / patterns during the training of the network. Therefore, if there exist patterns that are supportive of classifying a sample as A but are not or rarely found in the supposed training samples S_O, the class A has been trained (also) with data different from S_O. More mathematically, we compute features that are relevant to S_O, ie. features F_O that activate more often for samples from S_O than for other samples. Say samples from S activate features F_S. If there are features that are only in F_S but not in F_O, ie. features $F_S \setminus F_O$ then this indicates that samples S contain images that exhibit characteristics associated with decision A that are not found in the training data. We say that a feature, ie. neuron, is activated for a set S if the mean of all activations of set S is larger than the mean of all activations of the entire training data plus the standard deviation.

A possible problem is that while random samples S might share some similarities with S_A, the system might not classify them as A, if there are also similarities with other classes. To determine on a more fine granular level if samples are at least weakly related to class A, one might obtain class probabilities rather than just the predicted class. But without white-box access there seems to be no immediate way to access the class probabilities of the classifier. Therefore, we employ a method that estimates class probabilities using an approximation of the

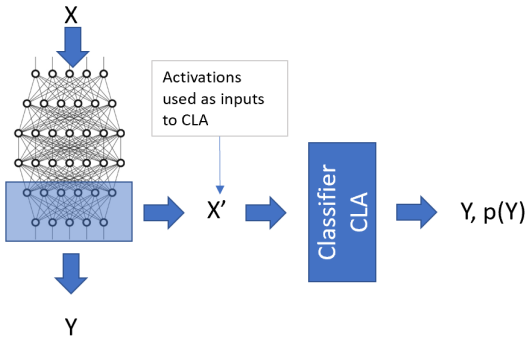

Fig. 5: Classifier CL_A predicts class prob. $p(Y)$ using feature activations as inputs

suspect model.[6]. Our approach follows the common idea, eg. [14], to use CNNs as feature extractors, but replacing (top) layers with a self-trained classifier CL_A. CL_A can return class probabilities illustrated in Figure 5. That is, we use labels generated by the suspect system for the public training data set (but not any of $S \in \mathcal{D}$) to train classifier CL_A. We then use classifier CL_A to obtain class probabilities for the random samples $S \in \mathcal{D}$.

As CNN we utilize a VGG-10 type architecture[3] trained on some classes of the CIFAR-100 dataset. For NT, RT, ET, we choose distinct sets of samples S_O, S_A, each corresponding to those of a random class. The CIFAR-100 dataset consists of 100 classes, which are split into 20 categories, eg. the category 'people' contains 'baby', 'boy', 'girl', 'man', or 'woman'.As classes $\mathcal{D}$ we use for S_R^A samples of a class from the same category as S_A, for S_R^B samples from the same category as S_O and for S_R^R samples of a class not in any the category of S_O or S_A. We train the networks without the three classes $\mathcal{D} := \{S_R^O, S_R^A, S_R^R\}$. They are used for evaluation of our methods. We choose 30 sets $\mathcal{D}$, ie. we train 30 networks for each NT, RT and ET.

4.4 Results for Case 2

As shown in Table 4, the overall error $err(M)$ is (almost) identical for RT, ET and NT, indicating that manipulations seem not to distort the overall system performance. However, this conclusion changes when investigating individual classes. As expected, the for NT about 40% of samples from S_O are classified as A. That is, the error for S_O is roughly the average overall error of the model M. For S_A it is large as expected. When training only on S_A and not on S_O, ie. for RT, the reverse holds. Thus, the tampering is easy to identify, since the error

[6] One might also try to identify neurons representing class probabilities using the fact that class probabilities should resemble strongest correlation with output classes among all variables.

increased drastically for S_O. In other words, tampering seems successful in the sense that S_A is indeed detected, though it is unknown what objects constitute S_A. For ET the accuracy of S_O is not indicative of manipulation, in particular, since we do not assume exact knowledge of the training data.

Table 4: Accuracy for different tampering attempts.

Method	$Err(M)$	$C(S_O, A)$	$C(S_A, A)$
NT	0.44±0.01	0.4±0.18	0.99±0.01
RT	0.44±0.01	0.99±0.01	0.4±0.18
ET	0.44±0.01	0.42±0.17	0.43±0.16

Table 5 shows the output for assessing random samples $\mathcal{D}$. It uses the rank $Ran(S, A)$ indicating the position of the class A, if classes are sorted depending on the fraction of samples in S classified as A. As expected, samples from S_R^O and S_R^A have a lower rank (and also lower error) than random. However, the rank of the random class seems fairly low, ie. under the assumption that on average a random class should have the mean rank of all classes, that is 48.5 (we trained using 97 classes). The reason being that since class A is trained with $S_A \cup S_O$, covering different concepts and being larger than for any other class, it is more likely that a random sample is classified as A. Furthermore, the standard deviations seem fairly large. This is not unexpected given the choice of training data. That is, sub-classes in some categories in the CIFAR-100 dataset exhibit only limited similarity, while other categories overlap, eg. 'streetcar' and 'pick-up truck' are in different categories, whereas 'clock','keyboard' and 'lamp' are in the same category. Still, overall differences are significant (as confirmed by a t-test with p-value $< 1e-4$). Therefore, at this point we have identified samples that are classified as A. Note, that results are similar irrespective of using a binary model or a grey box model. Next, we investigate whether they are (strongly) related to known samples S_O or they resemble different objects.

Table 5: Median Rank of A for samples S_R and the fraction not classified as A for ET.

Gray	Binary	$Ran(S_R^O, A)$	$Ran(S_R^R, A)$	$Ran(S_R^A, A)$	$C(S_R^O, A)$	$C(S_R^R, A)$	$C(S_R^A, A)$
N	N	4.0±8.99	14.75±18.3	3.5±12.04	0.94±0.11	0.99±0.02	0.95±0.06
N	Y	2.0±6.79	12.0±23.4	4.0±8.53	0.93±0.1	0.98±0.02	0.95±0.06
Y	N	4.0±7.87	14.25±17.65	3.25±9.96	0.94±0.11	0.98±0.02	0.95±0.06
Y	Y	2.0±6.61	12.0±17.15	4.0±9.58	0.94±0.1	0.98±0.02	0.95±0.06

Table 6 shows the number of features of the sets $\mathcal{D}$ as well as their intersection with the features obtained for A from S_O, ie. F_O. It can be seen that while samples from S_R^O share many features with S_O, S_R^A and S_R^R do not, however

their number of features is comparable. S_R^O has most features. The number of features varies strongly depending on the scenario (Grey box/binary). The number can be adjusted by setting a different threshold, when a feature counts as 'activated'. However, we found no need to tune the threshold, since all scenarios yield qualitatively the same outcome. Furthermore, we also used class probabilities obtained via an approximate model CL_A (see Figure 5). That is, we trained a random forest with 150 trees. We compute for $S \in \mathcal{D}$, the sum $P(A|S) := \sum_{X \in S} p(A|X)/|S|$, where $p(A|X)$ denotes the likelihood of class A for a sample X. Outcomes are similar to 5 with the main difference that the rank for all sets $S \in \mathcal{D}$ is about 30% less. This is somewhat expected, since predictions of the random forest are more noisy.

Table 6: Number of features of sets $\mathcal{D}$, in particular the common features with samples S_O.

Gray	Binary	$\lvert F_O \rvert$	$\lvert F_{S_R^O} \rvert$	$\lvert F_{S_R^O} \cap F_O \rvert$	$\lvert F_{S_R^R} \rvert$	$\lvert F_{S_R^R} \rvert$	$\lvert F_{S_R^A} \cap F_O \rvert$	$\lvert F_{S_R^A} \cap F_O \rvert$
N	N	97.47	37.57	11.1	28.87	1.23	26.97	1.30
N	Y	17.55	3.90	1.03	2.24	0.0	2.86	0.03
Y	N	246.53	102.63	31.93	83.13	2.27	68.5	3.27
Y	Y	13.66	6.34	2.14	4.10	0.03	3.48	0.10

5 Related work

Digital and AI forensics: One of the initial fields, where machine learning was applied to support forensics was intrusion detection and detecting malicious network events, as described in a call by Beebe [8]. Subsequently, digital forensics has benefited from AI in the areas of *Multimedia forensics* where some examples are images modifications (eg. copy-move forgery [2]), deep-fake video detection [12], or facial age estimation [3]. *AI forensics*, a sub discipline of digital forensics [15], aims to identify the AI that caused an event. Initial work[6] provides a brief overview of AI forensics research identifying four sub domains: AI Training Forensics, AI Substrate Forensics, AI Application Forensics and AI Model Forensics. (In addition,) we emphasize the role of testing/usage data and data generation. [6] with AI safety, ie. incidents due to failures of AIs with "good intentions". We are more concerned with systems that might be designed from scratch for a malicious act(see taxonomy of dangers of AI[19]).

Attack and defending ML: Security threats and defensive techniques from a data perspective are surveyed in [13]. The authors primarily discuss the attacks based on adversarial examples; as defensive technique, they mention data sanitation. In forensics, methods from data sanitation can help to identify malicious samples. The more general question 'Can machine learning be secure?' was discussed in [7] where the authors present 'a taxonomy of different types of attacks on

machine learning techniques and systems [as well as] a variety of defenses against those attacks', including three defense strategies: regularization, randomization and information hiding. It is not uncommon to involve humans in the detection process. For instance, to detect fake reviews [1] the GLTR method estimates word probabilities that are then presented to a human for further investigation. However, there are also fully automatic methods like GROVER [21] show-cased for fake news detection.

Explainability: Generally, adversarial analysis [9] and more so explainability methods (surveyed in [5,17]) might be helpful to better understand model behavior. Model simplification (eg. LIME or G-REX) aims at approximating the original model, often only for a subset of inputs, eg. for samples of a class. It might yield human understandable rules that characterize model behavior in terms of its inputs. Feature relevance explanation (SHAP) deals with understanding the impact of inputs towards outputs. These methods treat the machine learning method as black-box and are applicable of all sorts of models. Perturbation of inputs (as used by LIME), for example, are a general tool for model analysis even beyond machine learning. For CNNs, there is also a significant body of work that deals with model introspection which commonly needs detailed access to variables or even enhancements of the model (white-box). For example, for feature visualization, iterative optimization using full access to the model such as gradients might reveal inputs that maximize feature activation (see [11] for a seminal work or GRAD-CAM). Occlusions analysis [20], where feature maps are compared for the input samples that only differ by occluding some area in one but not in the other, might also be a helpful to understand model internal workings. To the best of our knowledge, we are among the first to consider a grey box model, where potentially only fuzzy knowledge on model internals is available. Furthermore, while it is common to perform correlation analysis of (internal) features and outputs in XAI, we are not aware that classes have been characterized by their complexity based on 'activated' features. This might be of value for the XAI community as well.

6 Conclusions

This work has discussed identifying AI systems and their motives as part of forensic work. Investigation of AI systems provides new opportunities and possibilities for analysis based on input-responses such as utilizing model activation patterns and analyzing input-output behavior based on generated data. But AI systems also provide ample opportunities for hiding evidence such as deletion of data or models and resetting of states. Thus, the 'cat-and-mouse' game between attackers and forensic experts has just begun. While unique identification information, eg. DNA for humans or authentication information for digital systems, is often a key tool to identify suspects, AI systems in many cases might not provide such information. Setting standards regulating AI, ie. demanding authentication during interaction, signing model structures, might be valuable. This should

be done sooner than later, since industries often suffer from legacy issues due to backwards compatibility (eg. [16]). Furthermore, more contributions from researchers are needed to ensure that malicious acts can be detected to foster justice and prevent them in the first place by discouraging criminals due to high risks of being caught.

References

1. D. Adelani, H. Mai, F. Fang, H. H. Nguyen, J. Yamagishi, and I. Echizen. Generating sentiment-preserving fake online reviews using neural language models and their human-and machine-based detection. *arXiv:1907.09177*, 2019.
2. O. M. Al-Qershi and B. E. Khoo. Passive detection of copy-move forgery in digital images: State-of-the-art. *Forensic science Int.*, 231, 2013.
3. F. Anda, D. Lillis, N.-A. Le-Khac, and M. Scanlon. Evaluating automated facial age estimation techniques for digital forensics. In *Security and Privacy Workshops (SPW)*, 2018.
4. S. Arora and P. Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv preprint arXiv:1806.06877*, 2018.
5. A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, García, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges for responsible ai. *Information Fusion*, 2020.
6. I. Baggili and V. Behzadan. Founding The Domain of AI Forensics. *arXiv preprint arXiv:1912.06497*, 2019.
7. M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Symp. on Information, computer and communications security*, 2006.
8. N. Beebe. Digital forensic research: The good, the bad and the unaddressed. In *IFIP Int. Conference on Digital Forensics*, 2009.
9. A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.
10. B. Chesney and D. Citron. Deep fakes: A looming challenge for privacy, democracy, and national security. *Calif. L. Rev.*, 107, 2019.
11. D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. 2009.
12. D. Güera and E. J. Delp. Deepfake video detection using recurrent neural networks. In *Int. Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2018.
13. Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. Leung. A survey on security threats and defensive techniques of machine learning: A data driven view. *IEEE access*, 6, 2018.
14. X.-X. Niu and C. Y. Suen. A novel hybrid cnn–svm classifier for recognizing handwritten digits. *Pattern Recognition*, 45(4), 2012.
15. G. Palmer. A road map for digital forensics research-report from the first Digital Forensics Research Workshop (DFRWS). *Utica, New York*, 2001.
16. R. Schlegel, S. Obermeier, and J. Schneider. A security evaluation of iec 62351. *Journal of Information Security and Applications*, 34, 2017.
17. J. Schneider and J. Handali. Personalized explanation in machine learning. In *European Conference on Information Systems (ECIS)*, 2019.
18. T. Wu, F. Breitinger, and I. Baggili. IoT Ignorance is Digital Forensics Research Bliss: A Survey to Understand IoT Forensics Definitions, Challenges and Future Research Directions. In *Int. Conf. on Availability, Reliability and Security*, 2019.

19. R. V. Yampolskiy. Taxonomy of pathways to dangerous artificial intelligence. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
20. M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 2014.
21. R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, 2019.