

An improved real time detection of data poisoning attacks in deep learning vision systems

Vijay Raghavan¹ · Thomas Mazzuchi¹ · Shahram Sarkani¹

Received: 29 July 2022 / Accepted: 21 September 2022

Published online: 03 October 2022

© The Author(s) 2022 [OPEN](#)

Abstract

The practice of using deep learning methods in safety critical vision systems such as autonomous driving has come a long way. As vision systems supported by deep learning methods become ubiquitous, the possible security threats faced by these systems have come into greater focus. As it is with any artificial intelligence system, these deep neural vision networks are first trained on a data set of interest, once they start performing well, they are deployed to a real-world environment. In the training stage, deep learning systems are susceptible to data poisoning attacks. While deep neural networks have proved to be versatile in solving a host of challenges. These systems have complex data ecosystems especially in computer vision. In practice, the security threats when training these systems are often ignored while deploying these models in the real world. However, these threats pose significant risks to the overall reliability of the system. In this paper, we present the fundamentals of data poisoning attacks when training deep learning vision systems and discuss countermeasures against these types of attacks. In addition, we simulate the risk posed by a real-world data poisoning attack on a deep learning vision system and present a novel algorithm MOVCE—Model verification with Convolutional Neural Network and Word Embeddings which provides an effective countermeasure for maintaining the reliability of the system. The countermeasure described in this paper can be used on a wide variety of use cases where the risks posed by poisoning the training data are similar.

Keywords Deep learning · Data poisoning · Neural Network Security · Convolution neural network

1 Introduction

Deep learning systems [1] have made major breakthroughs in vision. For example, safety critical vision systems in autonomous driving have been traditionally designed with physical redundancies. These redundancies have included components such as radars. However, in the last 12 months the performance of these vision systems have improved so much that auto manufacturers such as Tesla have removed physical redundancies in their vision system. Starting in May of 2021, Tesla's popular models such as Model 3 and Y are completely dependent on neural networks for all safety critical vision capabilities such as autosteer [2]. Deep learning vision systems are also playing a key role in national security and counterterrorism by aiding situational awareness with applications such as facial recognition [3]. Recognizing faces in the real world in different poses and light settings has been a hard problem in a wide range of use cases from opening

Thomas Mazzuch, Shahram Sarkani contributed equally to this work

✉ Vijay Raghavan, vijayrag@gwu.edu | ¹Department of Engineering Management and Systems Engineering, The George Washington University, 2121 I St NW, Washington, DC 20052, USA.



smart phones [4] helping autonomous weapon systems hone in on their targets [5]. Techniques such as universal representational learning have come a long way with significant performance improvements [6]. In medicine, deep learning vision systems have made progress in detecting deadliest forms of cancer such as ovarian carcinoma. Recent studies have shown that a convolutional neural networks classifier built with AlexNet, Google Net and VGGNet can detect these types of cancers with receiver operating characteristic curve (AUCs) > 0.95 [7]. Similarly, deep learning vision systems have made inroads into solving reliability problems such as estimating the useful life of bearings [8] and assessing the reliability of structures [9].

To a large extent the effectiveness of deep learning systems is based on their ability to learn from a large number of complex data sources [10]. The complexity of the modern deep learning data ecosystem creates a large attack surface during training [11]. An attacker can use data poisoning techniques during training to attack the model. These attacks poison the training data and can lead the model to learn undesirable outcomes. These outcomes can then be exploited by an attacker when the model is moved to production to make predictions in the real world [12]. Some of these exploits can be deadly in a safety critical system [13].

1.1 Motivation

Historically, the focus on security in deep learning literature has been in developing and studying data poisoning algorithms in isolation. As if these algorithms will be deployed in a clean room environment [14]. However, in the real world the deep learning systems are made up of complex data pipelines. The actual learning takes place in sophisticated distributed systems. These systems are often connected to a number of other subsystems when the model is being trained. In general, machine learning systems are vulnerable to security attacks. Typically, a mature model system has about 5% machine learning code and the rest 95% of the code is glue code made up of libraries and data pipelines leading to a large attack surface [15].

A special challenge in vision systems is the reliance on multiple datasets from the internet which might be of varying quality. A data poison attack during training can originate from any one of the multiple data pipelines which constantly get data from several upstream systems. In addition to these, as time goes by these vision systems have to deal with changes in the distribution of data, and missing data, both of which can be a convenient vector for data poisoning attacks [16] during training.

1.2 Contributions

Given the complexities of real-world deep learning systems it is impossible to ignore a holistic systems engineering approach when considering the risks posed by data poison attacks on the overall reliability of these systems [17]. This paper has 3 specific contributions to data science and engineering literature for detecting data poisoning attacks in deep learning vision systems:

1. We explore the security threats from data poisoning from a holistic risk perspective. Our approach sees deep learning systems as complex adaptive data systems [18].
2. We explain how the complexity of the data ecosystems creates a large attack surface for data poisoning during training. We explore countermeasures in the data ecosystems to reduce the risks posed by data poisoning attacks during training.
3. In the end, we use a data poison attack which makes the model misclassify data and show the effectiveness of a robust countermeasure called MOVCE–Model verification with Convolutional Neural Network and Word Embeddings in identifying the attack.

2 Related works

Deep learning vision models make use of multimodal and transfer learning modes of training [19]. These methods need massive amounts of diverse datasets for greater accuracy. The amount and type of data in these datasets need complex engineering infrastructure and data pipelines. In addition to massive image datasets vision systems often use voice, streaming logs from other IoT devices, unstructured text on road signs, real time video and records from

spatial databases for routes. The data, pipelines and the related engineering infrastructure form the core of the data ecosystem.

Typically, during training the data lake is a central data repository of all available data [20]. Features for model development are selected from the data lake by the feature selection process. The model building process starts with building a pipeline for ingesting the data from the data lake and performing various feature engineering techniques on the data such as normalization and transformation [21]. The diversity and the high throughput of the data sources have made current deep learning models multimodal [22] and more accurate. However, on the other hand, the wide assortment of data sources and feature types ranging from logs, video, text, speech, and real time streaming have created a path for attackers with malicious intent to poison the data during training.

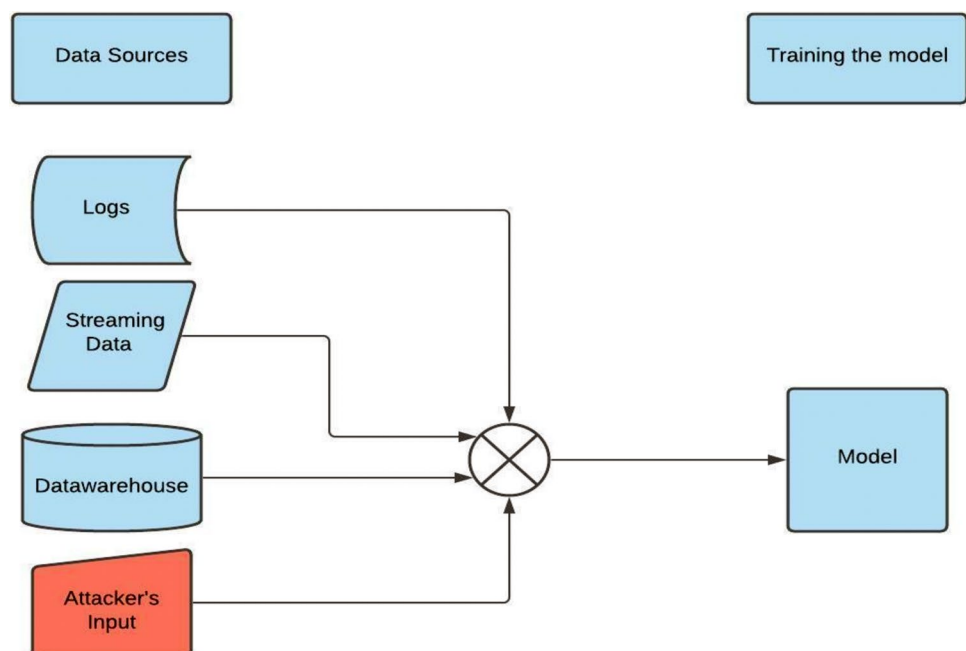
For example, many deep learning classification algorithms assume that the training dataset and the real-world data which they will be making predictions on are identically and independently distributed [23]. However, when a malicious attacker changes the distribution of data in the training set it can lead the model to learn an incorrect outcome which can later be exploited by the attacker. This is a hard problem to solve in vision systems with streaming data where labels are often non-existent. In addition to these, with the ever growing size of today's datasets it is often hard to make a judgement call on what is the right rate of data change in the long term.

2.1 Data poisoning attacks

Data poisoning attacks have been studied on classical machine learning algorithms beginning with support vector machines and Bayesian classifiers [24]. The amount, type and sources of data that is used in today's vision networks makes them a perfect candidate for data poisoning attacks during training. The classical data poisoning algorithms have the side effect of degrading the performance of the model against real world data. However, in the last two years newer data poisoning attack algorithms targeting deep neural networks have come out which are so efficient that any performance degradation is not noticeable. These algorithms hide poison in data so well that it is hard for humans to detect [25].

The objective of data poisoning attacks is to introduce specific samples into the training data and control the behavior of a deep learning system. Some of the data poisoning attacks are "clean-label" attacks. In these types of attacks there is no need for the attacker to have control over the data labeling process. These attacks are targeted and precise. For example, in autonomous driving systems which scrape the images of various features from the internet, a malicious attacker can introduce a poisoned image. A model trained on the poisoned image can then be targeted and made to misclassify a feature that was learnt on the image (see Fig. 1).

Fig. 1 Data poisoning attack during training



2.2 Data poisoning methods

2.2.1 Feature collision

In a collision attack it is assumed that an attacker has knowledge of the model and its parameters. However, there is no need for them to have any prior knowledge about the training data. The attack starts with adding small perturbations to a base dataset which is then added to the training set. The perturbations might be small enough for humans to detect. However, these perturbations can cause the classifier to misclassify by stretching or altering its typical decision boundary. Classifiers in vision systems are a perfect target for collision attacks. The classifier in a typical vision system can easily spot the difference between a deer and a car. However, if an image that looks like a deer is classified as a car during training. Then an attacker can use this specific vulnerability in real time to attack the vehicle. Let us assume that an input x is propagated through a function $f(x)$ in a deep learning network. The high-level semantic features of the input are represented in a feature space. In such a space, the probability of finding an example x that “collides” with the target (t) and is closer to the base (b) is high.

Shafahi et al. [12] represent such a poison p as follows,

$$p = \underset{\forall x}{\operatorname{argmin}} \|f(x) - f(t)\|^2 + \beta \|x - b\|^2 \quad (1)$$

Where, $f(x)$ is a function that is used to propagate the input x till the softmax layer. b is the base instance t is the target instance. β parameterizes the degree of similarity between the base and poison instance. They obtain p using an optimization algorithm that minimizes the distance between the target and base instances in a two step process.

2.2.2 Poison polytope

One of the issues with feature collision attacks is that they typically need a white box setting where the attacker has knowledge of the victim’s model. However, in the case of a polytope attack the algorithm [26] does not need information on the victim’s model or its outputs, architecture and in many cases the training data. In this form of attack, the target data is surrounded by a poisoned dataset which makes the classifier misclassify the target. The objective is to craft a convex polytope which is close to the target vector in the feature space. In an image classification setting Zhu et al. formulate the forming of a convex polytope [27] as an optimization problem which can be solved using forward-backward splitting and finding the most optimal set of poisons.

2.2.3 Bi-level optimization

Bi-level optimization is a nested optimization problem where one problem contains another optimization problem as its constraint. Since one problem has another nested inside it these problems are called upper and lower level optimization problem or a leader and a follower optimization problem. The decisions of the leader and the follower affect each other. One other issue is that in Bi-level optimization problems unless a solution is optimal for the lower level follower it cannot be feasible for the whole problem. Traditional approximation methods often cannot be used in this context as they do not guarantee an optimal solution. Bi-level optimization problems have been used in pricing, economics, and coordination. Angelo et al. [28] define Bi-level optimization problem as:

$$\begin{aligned} (L) \quad & \min_{x \in X} f_1(x, y(x)) \quad \text{subject to } g_1(x, y(x)) \leq 0 \\ (F) \quad & y(x) \in R(x) := \underset{y \in Y}{\operatorname{argmin}} f_2(x, y) \\ & \text{subject to } g_2(x, y) \leq 0, \end{aligned} \quad (2)$$

Where L is the leader and F is the follower ($f_1(x, y(x))/f_2(x, y)$), and ($g_1(x, y(x))/g_2(x, y)$) are the upper/lower level objective functions and upper/lower level constraints.

In the past, most of the data poison attack algorithms were handcrafted and it took some effort to deploy them against a target. Huang et al. have proposed a method they call MetaPoison [29]. This method poses crafting poisons

as a Bi-level optimization problem which can be solved quickly compared to the traditional collision-based methods [30, 31]. Metapoisn has a success rate of over 15% with a poison budget of 0.5% and it can be transferred as well.

2.2.4 Label flipping

In distributed training systems applications often create multiple replicas of the model and each replica is made to train on a portion of the full training dataset. The traditional forward and backward passes to compute the gradients are performed independently on multiple model replicas. Once this is done the model replicas either synchronize the gradients or they update certain parameters in the chosen algorithm. The parallelization of data and compute resources creates a larger attack surface for an entity with malicious intent to flip the labels in the training data.

2.2.5 Data drift

In addition to these, the complexity of the data ecosystem has made data, model, pipeline versioning and configuration an important part of the training process. The heterogeneity of tools, people and processes has also introduced multiple attack surfaces for a malicious attacker to poison the training data either directly or indirectly by introducing malicious drift in training data.

2.3 Countermeasures

In the literature on data poisoning attacks, countermeasures are broadly based on 3 major techniques:

1. Identifying anomalous drifts in data to detect data poisoning.
2. Using model explanation as way to detect any poisons introduced in the data.
3. Traditional data provenance based techniques.

Countermeasures against data drifts have been studied in literature both on classical machine learning algorithms and in deep learning as well. Sethi et al. introduced the MD3 algorithm to detect drifts [32]. The MD3 algorithm uses the change in margin density as a metric to identify drift and can operate with limited information. Demšar et al. propose combining a drift detector with a classification algorithm [33]. This approach uses model explanation to compute changes over time and observe their magnitude. Often data poisoning will create small drifts in the data. If these drifts are caught earlier there is a possibility to detect data poisoning. Few others have argued that since model development and training are iterative processes. Every time a model is run the provenance of its data i.e., “*where a piece of data came from and the process by which it arrived*” [34] should be traceable to avoid issues such as data poisoning. Several authors have proposed using block chain to trace the lineage [35].

Some researchers have argued for using traditional software engineering best practices such as versioning on data. When developing model data scientists often try multiple features and feature engineering methods. Versioning the raw data used to produce features helps in tracking and identifying systematic issues with data [36]. In cases where the data drifts or is not of the expected quality due to poison attacks data versioning reduces the harm to the model by helping to a different version of the source data which might be more stable.

2.3.1 Why the current set of countermeasures are ineffective for deep learning visions systems at scale?

The countermeasures described above can be effective in traditional machine learning systems. However, the sheer volume of data needed by today's deep learning vision system makes many of the above techniques ineffective. Even techniques such as versioning which can naturally scale with data cannot provide sufficient protection against bi-level optimization based data poisoning attacks. Traditional techniques for data drifts cannot handle the complexities of vision systems where a legitimate photo taken in a dark lighting or a photo done using a Instagram filter might be indistinguishable to systems looking at just data drift. Newer Blockchain based smart contracts offer a promising future. However, the compute and energy resources needed for them might make them harder to use in the near term. In addition to these in many real world scenarios it is often hard to get a balanced dataset for training vision systems. For example, to train a face recognition system it is impossible to get all the angles and lighting effects on a person's face in a dataset and then use it for training. In most computer vision problems, the samples are often unbalanced [37].

Fig. 2 Methodology workflow—data poisoning during training

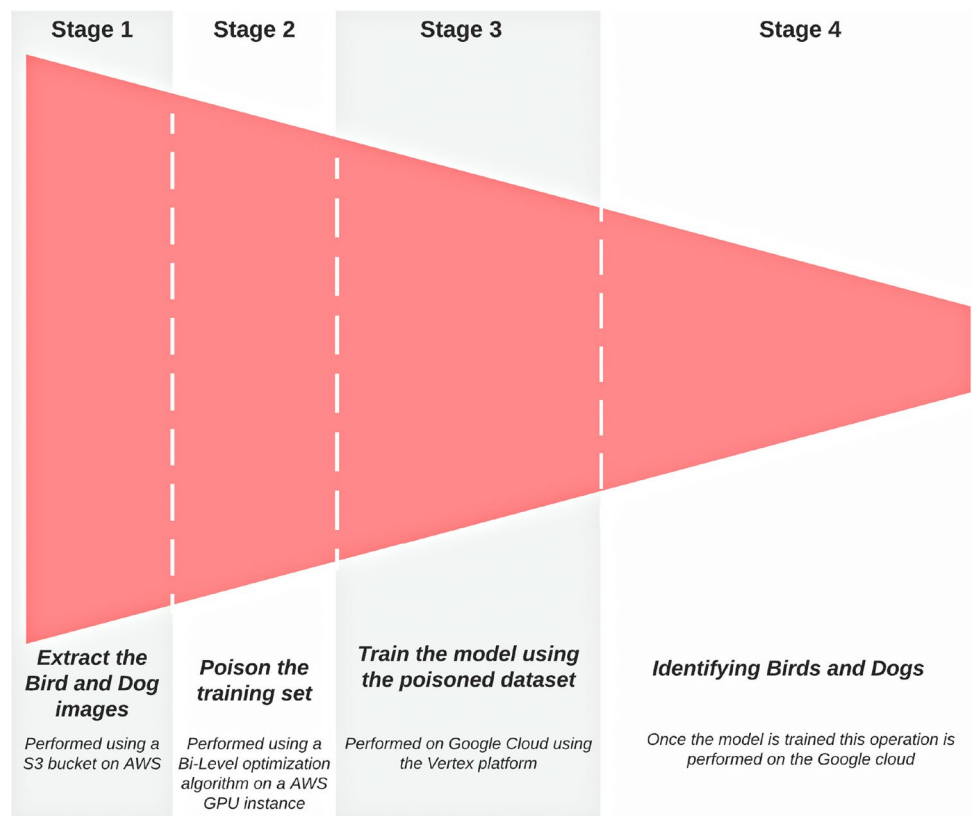


Fig. 3 CIFAR-10 Dataset sample images

All of these present unique challenges for traditional data poisoning countermeasures and make them ineffective for use in deep learning systems for computer vision. In the next section, we present a novel convolutional neural network and word embeddings based countermeasure called MOVCE—Model verification with Convolutional Neural Network and Word Embeddings. MOVCE can be integrated into any realtime machine learning pipeline, the framework is scalable, and with few modifications it can be used in a variety of use cases beyond computer vision.

3 Methods

3.1 Data

The Canadian Institute for Advanced Research (CIFAR) 10 image dataset is the one we have used for all our experiments. The CIFAR-10 image dataset [38] has been a popular image dataset for training vision models. The dataset consists of about 60,000 low resolution color images in 32×32 format. The images are divided into 10 classes ranging from an airplane to a truck with about 6000 image in each class. These images were originally assembled by teams at MIT and NYU from the web. In our experiment, we took about 30 CIFAR -10 images of dogs and birds and poisoned them using

the Metapoisson Bi-level optimization framework. The poisoned dataset was then used as a training set to build a model to classify the images of dogs and birds (see Fig. 2, 3).

3.2 Generating poisons

Bi-level optimization algorithms are an effective way to generate poisons in the training data. Huang et al. [29] have demonstrated that poisoning can be formulated as a constrained bi-level optimization problem. In this approach, the attacker adds poisons X_p to the clean training dataset X_c . The optimal number of poison images X_p^* can be expressed as the solution to the following optimization problem:

$$X_p^* = \underset{X_p}{\operatorname{argmin}} \mathcal{L}_{adv}(x_t, y_{adv}; \theta^*(X_p)) \quad (3)$$

where $\mathcal{L}_{adv}(x, y; \theta)$ is the loss function which measures the accuracy of the model in assigning label y to input x with weights θ .

$$\theta^*(X_p) = \underset{\theta}{\operatorname{argmin}} \mathcal{L}_{train}(X_c \cup X_p, Y; \theta) \quad (4)$$

The data for the model comes from the original CIFAR-10 dataset. However, as we noted in the attack plan the Metapoisson algorithm is used to create poisons in the previously clean dataset. For this experiment, the original dataset was run through the metapoisson algorithm. The algorithm was run on a multi-GPU instance on Amazon Web Services (AWS) and the entire process took more than 12 h to complete. All the experiments were tracked using Comet ML (see Fig. 4).

3.3 Image classification using AutoML

First introduced by Google in 2017 AutoML has become one of the most popular techniques for building image classifiers. In recent times AutoML frameworks have come a long way. Building deep learning networks by hand for common tasks such as image classification have been automated with AutoML frameworks [39, 40] such as Google's Vertex platform. AutoML is based on Neural Architecture search with Reinforcement Learning. In this technique, there is a Recurrent Neural network based controller which is used to train a child network with an architecture that leads to a specified accuracy. The accuracy is the reward signal and the controller is essentially trying to increase the reward by finding the right architecture that makes the child network better [41].

The job of the controller is to maximize its expected reward $J(\theta_c)$:

$$J(\theta_c) = E_{P(a_1:T; \theta_c)}[R] \quad (5)$$

Where $a_1 : T$ is the list of actions to design a child network with an accuracy R (see Fig. 5).

We used Google Compute's Vertex platform to build a model to classify the images of dogs and birds. The classifier used AutoML for building the model. Once the model was built it was tested using an image of a bird slightly perturbed to mimic a real-world setting. Instead of classifying the image as a bird the model classified it as a dog.

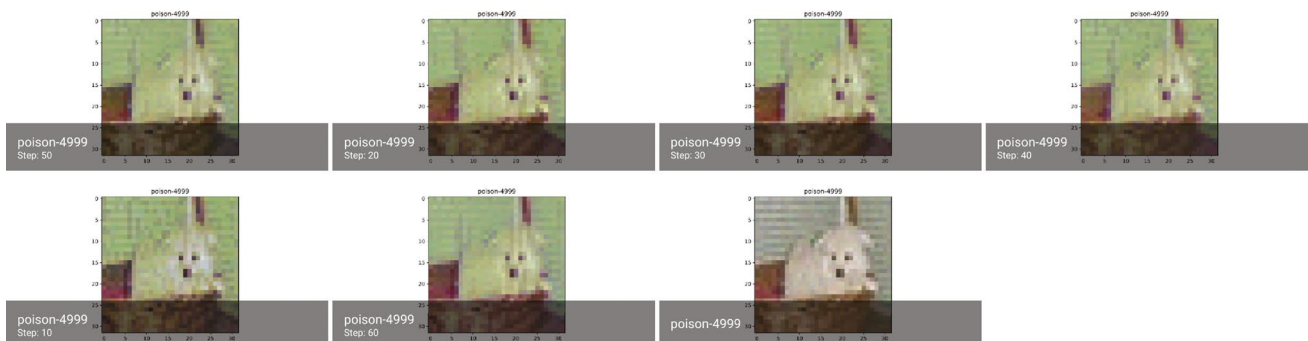


Fig. 4 Poisoned dog images sample

Fig. 5 Classifying Bird and Dog images



3.4 Model verification based countermeasure

In the above case, a poisoned dataset was used against an unsuspecting deep learning based image classifier. The nature of the dataset and the poisons made the image to be misclassified in a real world setting. Many of today's training datasets have images that were gathered from the web. Given this scenario, an attacker can plant multiple poisons across the web and these poisons can end up poisoning the data during training. In many such cases without human intervention it will be hard to detect the effects of poisons. Human intervention in many cases might not be scalable as the data and use cases grow. One way to detect if the model was attacked using poisoned data during training is to introduce a verification step after the model was trained but before it is released for real world predictions. The verification method should be amenable for automation and scalable. In the section below we introduce such a method.

3.4.1 MOVCE–Model verification with Convolutional Neural Network and Word Embeddings

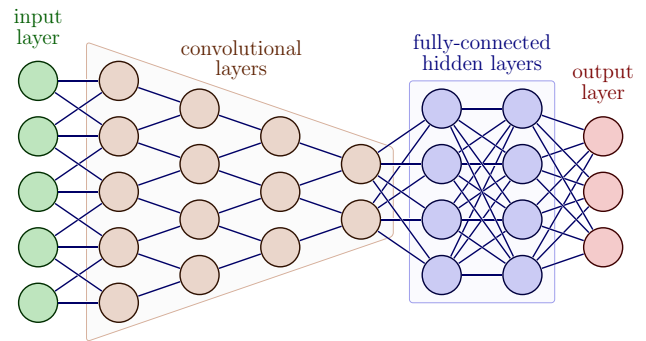
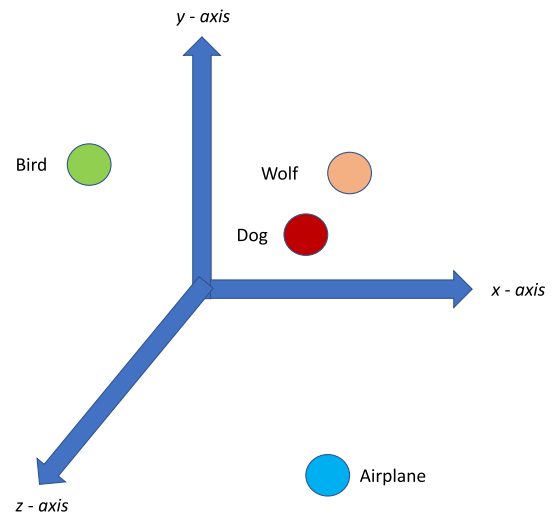
A strong countermeasure should have two objectives. The first objective of a countermeasure is to identify any security vulnerability before a model is put to use in the real world. The second objective of the countermeasure is to be an independent referee and not be influenced by any action that is internal to the model that is being verified. To achieve these two objectives we have come up with MOVCE - **Model Verification with Convolutional Neural Network and Word Embeddings**. MOVCE combines a convolutional neural network and text embeddings to verify if a model has been compromised before it is put to use in the real world. Incorporating MOVCE into the model serving ecosystem creates a holistic end to end system engineering based countermeasure against any data poisoning attack. With slight changes MOVCE can be adapted to multiple use cases from image classification to autonomous driving. MOVCE is an ensemble based approach with 2 key components:

3.4.2 Convolutional Neural Network

Convolutional Neural Network (CNN) are a type of neural network which are often used for solving computer vision problems. These networks have an input and output layer along with convolutional, pooling and hidden layers.

Convolutional layer The main goal of the convolutional layer is to extract and learn features that constitute an image [42]. The convolutional filter or the kernel passes or slides through an image based on its size or stride. The kernel is a feature detector and its sliding operation is the key for the convolutional layer to create a map of features from the image [43]. These convolutional filters have the property of translational in-variance. This makes them identify a unique feature that they learnt in the past in the future irrespective of its relative position in the image. More filters leads to learning multiple features. In the case of color RGB images they have 3 channels and a depth component as well. The Convolution layer helps the network to break down these features. As the convolutional layers go deeper, the feature detection moves from a low to a high level which helps in the identification of the image.

Pooling layer The Pooling layer acts to shrink the image stack by reducing the dimensionality of the image without losing its features. This helps to reduce overfitting. Often times the type of pooling used in CNNs is referred to as max pooling. In max pooling, the maximum value from a kernel window is extracted.

Fig. 6 MOVCE-CNN**Fig. 7** MOVCE-word embeddings

Fully Connected layer In the fully connected layer each neuron is connected with the other. This layer is responsible for the classification and prediction. These predictions are then compared with the labels and depending on the results the weights in the neural network are updated via back propagation.

Model In the model, each convolutional layer is followed by a pooling layer. These layers are then followed by two fully connected layers which map to the output predictions. In total, the model has 3 convolutional layers and 2 fully connected layers. The first convolutional layer has 3 input and 16 output channels. The second convolutional layer has 16 channels as input and the number of output channels is set at 32 and the third convolutional layer has 32 input channels and 64 output channels. The kernel size is set at 3x3. The first fully connected layer has 4*4*64 as input channels and 500 output channels. These 500 output channels are feed into the second fully connected layer with 10 output channels with each channel corresponding to a image class. Padding at 1 is added to factor in the set of pixels on the edges. The model is run on Tesla Katee GPU. To reduce overfitting we add a dropout of 0.5. The optimizer used is ADAM and the learning rate is set at 0.001. Relu is used as the activation function to avoid vanishing gradient problem (see Fig. 6, 7, 8).

Training and performance The CNN was then trained on the original CIFAR-10 dataset for 30 epochs. At the end of the 30th epoch the network had a training loss of 0.7409, training accuracy of 74.32%, validation loss 0.7476 and the final validation accuracy of 74.68% (see Fig. 9, 10).

3.4.3 Word embedding layer

If the words in a text can be converted to vectors and if these vectors are then arranged by their cosine distance then it is easy to distinguish words based on their similarity. In our approach we convert words to vectors using embedding from a skipgram based embedding technique called Fasttext. Once we have these embeddings we can use the cosine distance between the embeddings to distinguish labels such as a 'bird' from a 'dog'.

In this type of skipgram based embedding space [44] the internal structure of words are not ignored. The morphology of words is maintained by considering the sub word units as well. In this word embedding model, each word is

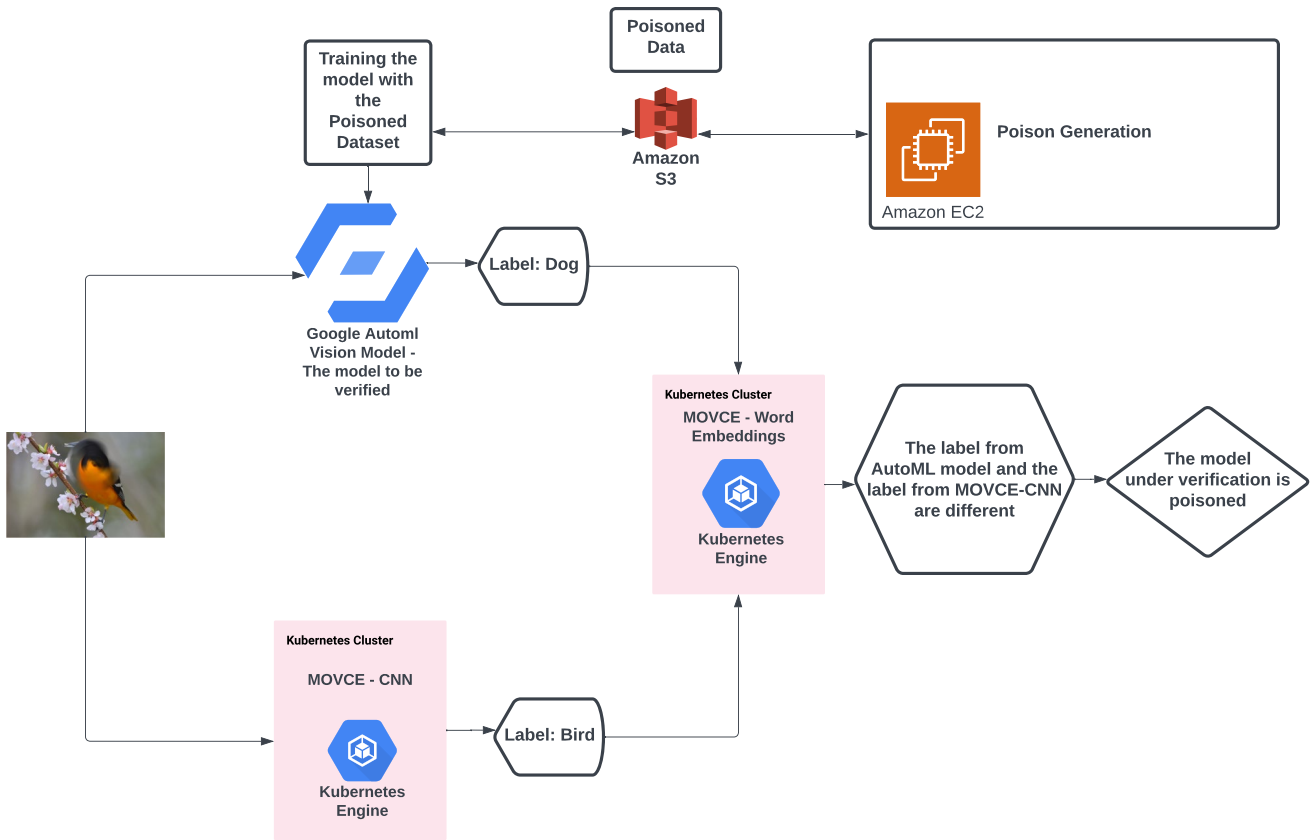


Fig. 8 Technical architecture and implementation

Fig. 9 Training-validation loss

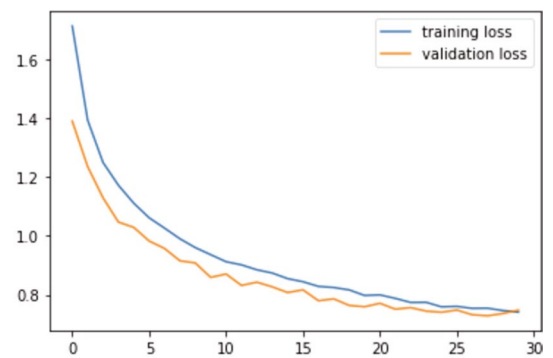
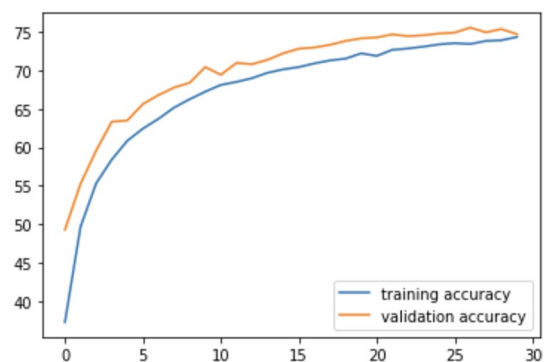


Fig. 10 Training-validation accuracy



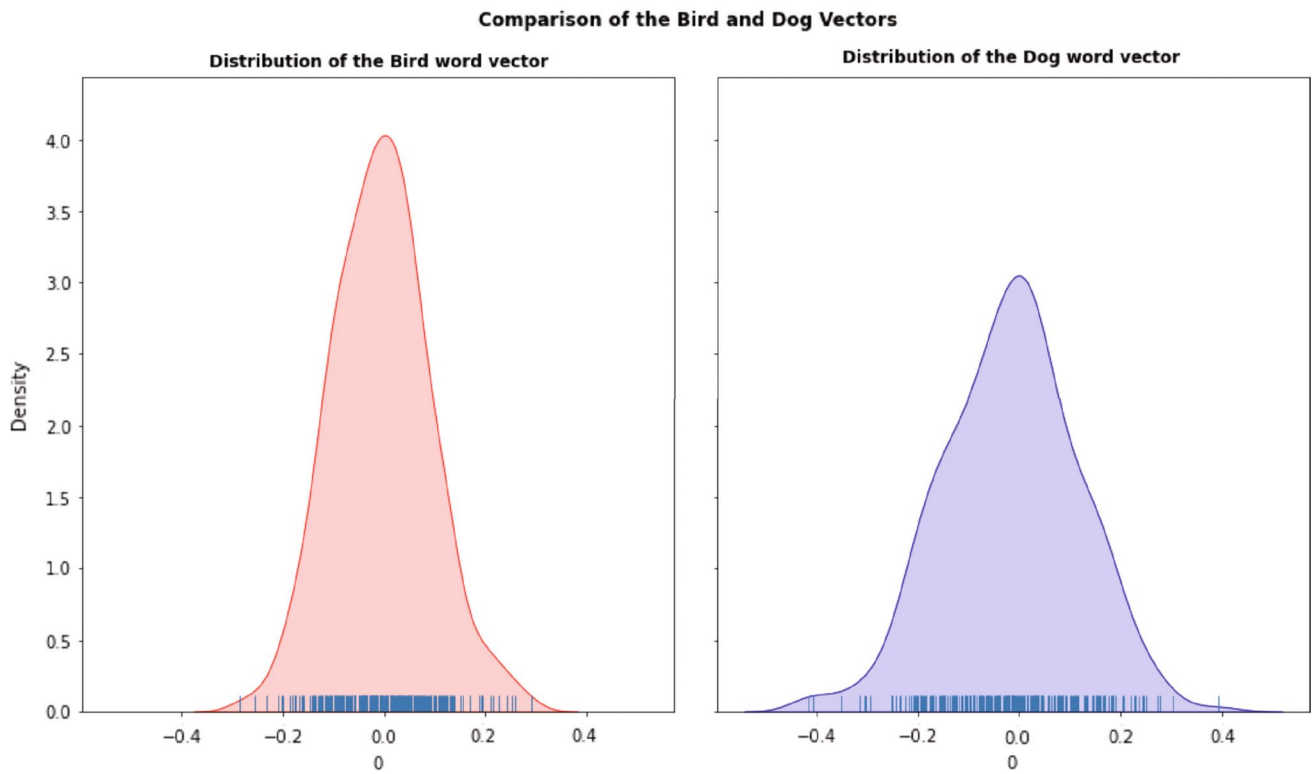


Fig. 11 Dog and bird word vectors

Table 1 Top 10 nearest word vectors to 'bird'

Similarity score	Neighbour
0.8000149726867676	'birds'
0.7060824036598206	'bird-'
0.7030669450759888	'bird.The'
0.692503035068512	'bird.I'
0.6835405826568604	'bird:'
0.6772602200508118	'non-bird'
0.677191972732544	'birdy'
0.6699777245521545	'avian'
0.6616601943969727	'sparrow'
0.6463329792022705	'raptor'

represented as a bag of character n-grams. A vector representations associates itself with each of the character n-gram. The words are represented as the sum of these vectors (see Fig. 11).

The scoring function is given by:

$$s(w, c) = \sum_{g \in G_w} \mathbf{z}_g^T \mathbf{v}_c \tag{6}$$

Where, w is a word denoted by n-grams of size G as $G_w \subset \{1, \dots, G\}$ the n-grams in w .

The word embedding layer makes MOVCE flexible to transfer learning methods in the CNN in the future. Using word embeddings also gives the framework a higher tolerance to accommodate slightly different classification labels. For example, in the table below using the word embeddings gives us the flexibility to explore the neighbourhood where

the label of interest 'bird' is located. This degree of flexibility is important to accommodate a range of transfer learning methods as some methods can label the image as 'birds' few others can label the image as 'birdy' etc (see Table 1).

3.4.4 MOVCE—workflow

The MOVCE workflow consists of 3 major parts:

1. In the first part, a test image is run though both the model which needs verification and the MOVCE-CNN.
2. In the second part, the labels of the image from both these models are then passed though the MOVCE-Word embedding layer.
3. Finally, the word vectors are compared based on their cosine similarity scores. If the words are the similar the distance between their word vectors will be smaller and the cosine similarity score will be closer to 1. In this case, if the cosine similarity score is ≥ 1 the word vector of the image is the same both in MOVCE and the model that is being tested and the model is NOT poisoned.

Algorithm 1 Calculate if the model is poisoned or NOT

Require: $image[1..maxImages]$

Ensure: $maxImages > 0$

```

1: while  $maxImages \neq 0$  do
2:    $Label_{MODEL} = MODEL_{To-Be-Verified}(image)$ 
3:    $Label_{MOVCE-CNN} = MOVCE_{CNN}(image)$ 
4:    $\vec{Vec}_{MODEL} = WordToVec(Label_{MODEL})$ 
5:    $\vec{Vec}_{MOVCE-CNN} = WordToVec(Label_{MOVCE-CNN})$ 
6:    $Cosine_{similarity} = | \vec{Vec}_{MODEL} - \vec{Vec}_{MOVCE-CNN} |$ 
7:   if  $Cosine_{similarity} \geq 1$  then
8:     NOT poisoned
9:   else
10:    Poisoned
11:   end if
12: end while

```

In the case of the perturbed bird image it is passed though both the Google AutoML Dog-Bird classifier model and the MOVCE-CNN. The result from the Google AutoML Dog-Bird classifier model and the MOVCE-CNN are captured and passed on to the MOVCE-word embedding layer. Finally, these word vectors are compared and we calculate their cosine similarity score (see Fig. 12, 13).

4 Results

To verify if a model is poisoned or not we pass the same test image to MOVCE and the AutoML model trained with the poisoned dataset. The AutoML model classifies the image as a 'dog' and the MOVCE's CNN classifies the image as a 'bird'. The labels in this case are 'dog' and 'bird' which when passed on to MOVCE's Word embedding layer give a similarity score of 0.38. Given that the similarity score is < 1 i.e. the CNN and AutoML models gave different labels to the same image. MOVCE algorithm concludes that the AutoML model was trained on a poisoned dataset and should not be moved to production i.e. put to a real world use case. To test the robustness of MOVCE we tested it further on 20 datasets and the results were consistent. In all 20 test cases, the poisoned model misclassified images of birds as dogs. However, the MOVCE's CNN rightly classified all the 20 datasets and MOVCE's Word embedding layer found low similarity scores between its label and the model's classification. In the end, MOVCE concluded that the model was poisoned in all 20 instances (Table 2).

Fig. 12 Label generation by the Poisoned model and the MOVCE-CNN

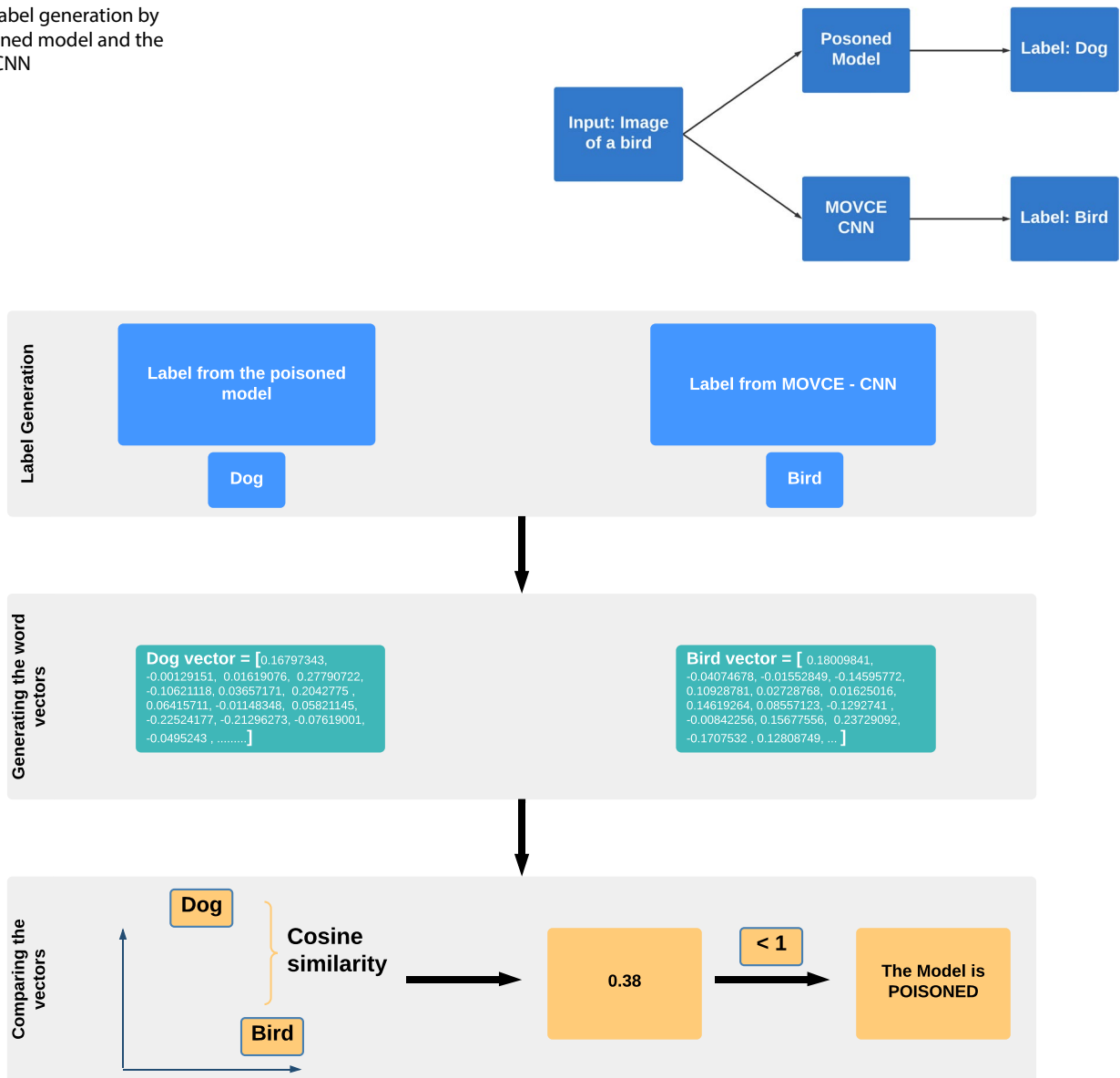


Fig. 13 Comparing the label vectors by MOVCE-WordToVec
















5 Discussion

In our use case above we reject anything less than a similarity score of 1. However, this choice may not be the best one for all the use cases. In safety critical systems such as autonomous driving the similarity score should be the highest and preferably closer to 1. The exact score that is used for cutoff and call a model poisoned depends on the precision and recall trade-off. The use of word embeddings gives sufficient flexibility for MOVCE to adapt itself to different use cases and transfer learning based approaches.

In the transfer learning approach, the features learned from a larger dataset are transferred to a model without training it from scratch. Such methods have become popular in recent times and have valuable even in the identification of eye diseases [45] (see Table 3).

To check how the model is performing we compared the model accuracy on the test dataset from CIFAR-10 and the native performance of the leading top 1% of the leading vision models on the perturbed bird image. Although, the MOVCE-CNN has an accuracy in mid 70% its performance can be improved by incorporating transfer learning based

Table 2 Results

Image	AutoML Model Classification	MOVCE Classification	MOVCE - Is the AutoML Model Poisoned?
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes
	Dog	Bird	Yes

approaches which can extend the range of models that the system can verify. In the future, our plan is to extend the breath and the depth of MOVCE using transfer learning.

Compared to other countermeasures such as MD3, Drift detection and Data versioning the main advantage of MOVCE is that it is flexible and can be integrated into production workflows in realtime. Additionally, the framework scales well with high dimension data that are needed by the deep learning vision systems of today.

One of the limitations of the MOVCE approach is that the CNN is trained only on the CIFAR-10 dataset which limits it to the images in that particular dataset. However, the MOVCE approach is flexible and can be easily extended to many other image datasets using a transfer learning approach. Another area of future work is to extend MOVCE to address issues with attacks on production systems from GANs (see Table 4).

Table 3 Comparing to other Baseline models

Model	Accuracy (%)
VGG16	23 ¹
VGG19	27 ¹
InceptionV3	11 ¹
NASNet	58 ¹
ResnetV2	43 ¹
MOVCE - CNN	74 ²

¹Native accuracy on the perturbed Bird image without any training

²Test accuracy on the CIFAR-10 dataset

Table 4 Comparing MOVCE to other alternatives

Countermeasure	Family	Advantages	Shortcomings
MD3 [32]	Drift Detection	Uses Margin density which can operate with limited information	Does not scale well with high dimension data
Drift detection with classification [33]	Model explanation	Uses Model explanation to compute changes over time	Model explanation tends to break down with neural networks
Data Versioning [36]	Provenance	Implementation ease	Cannot scale for all data
Block chain [35]	Provenance	Smart contract makes it easy to integrate during run time	Costs and compute efficiency
MOVCE	Neural Networks	Flexible and can be integrated into production workflows in realtime	Compute costs

6 Conclusion

There is no argument that deep learning has revolutionized machine learning ever since it was introduced approximately 7 years ago. The power of deep learning lies in its ability to take massive amounts of data in multiple formats and making predictions in the real world. In this paper, we have shown how the data needs of a modern vision system can also be used against it with a data poisoning attack and we have argued for a holistic system engineering based verification approach to identify data poisoning attacks in real time. In addition to these, we have introduced a holistic lightweight countermeasure called MOVCE which can be used to identify data poisoning attacks in real time. To check the robustness of the framework it was tested against 20 image datasets and it was successful in identifying poisoned model in all the 20 cases. Since the framework is neural net based it can be extended to popular transfer learning approaches and it can be deployed at scale as well.

Author contributions VR, TM, and SS contributed equally to the main manuscript. All the authors read and approved the final manuscript.

Data availability statement The datasets generated during and/or analysed during the current study are available in the GitHub repository for the project at <https://github.com/vijaygwu/DataPoison> and the entire experiment is available at <https://www.comet.com/vijayrag/craft1?shareable=eVZ5BwJAWZH2ZGAEM8ngJHtz6>

Declarations

Competing interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521(7553):436–44.
2. Tesla: Tesla transition. <https://www.tesla.com/support/transitioning-tesla-vision>. Accessed 20 Sept 2021.
3. Wang M, Deng W. Deep face recognition: a survey. *Neurocomputing*. 2021;429:215–44.
4. Dijk Tv, Croon Gd. How do neural networks see depth in single images? In: *Proceedings of the IEEE/CVF International conference on computer vision (ICCV)*. 2019.
5. Kirkpatrick DD. DroneTarget. <https://www.nytimes.com/2020/12/02/world/middleeast/iran-assassination-nuclear-scientist.html>. Accessed 20 Sept 2021.
6. Shi Y, Yu X, Sohn K, Chandraker M, Jain AK. Towards universal representation learning for deep face recognition. In: *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. 2020. p. 6816–6825. <https://doi.org/10.1109/CVPR42600.2020.00685>.
7. Yu KH, Hu V, Wang F, Matulonis UA, Mutter GL, Golden JA, Kohane IS. Deciphering serous ovarian carcinoma histopathology and platinum response by convolutional neural networks. *BMC Med*. 2020;18(1):236. <https://doi.org/10.1186/s12916-020-01684-w>.
8. Li X, Zhang W, Ding Q. Deep learning-based remaining useful life estimation of bearings using multi-scale feature extraction. *Reliab Eng Syst Saf*. 2019;182:208–18.
9. Xiang Z, Bao Y, Tang Z, Li H. Deep reinforcement learning-based sampling method for structural reliability assessment. *Reliab Eng Syst Saf*. 2020;199:106901.
10. Sun C, Shrivastava A, Singh S, Gupta A. Revisiting unreasonable effectiveness of data in deep learning era. In: *Proceedings of the IEEE International conference on computer vision*. 2017. p. 843–852. <https://doi.org/10.1109/ICCV.2017.97>.
11. Xue M, Yuan C, Wu H, Zhang Y, Liu W. Machine learning security: threats, countermeasures, and evaluations. *IEEE Access*. 2020;8:74720–42. <https://doi.org/10.1109/ACCESS.2020.2987435>.
12. Shafahi A, Huang WR, Najibi M, Suci O, Studer C, Dumitras T, Goldstein T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In: *Proceedings of the 32nd International conference on neural information processing systems*. 2018. p. 6106–6116.
13. Islam G, Storer T. A case study of agile software development for safety-critical systems projects. *Reliab Eng Syst Saf*. 2020;200:106954.
14. Liu X, Xie L, Wang Y, Zou J, Xiong J, Ying Z, Vasilakos AV. Privacy and security issues in deep learning: a survey. *IEEE Access*. 2021;9:4566–93. <https://doi.org/10.1109/ACCESS.2020.3045078>.
15. Sculley D. Hidden technical debt in Machine learning systems. In: *Proceedings of the 28th International conference on neural information processing systems*, vol. 2. Cambridge: MIT Press; 2015. p. 2503–2511.

16. Schwarzschild A, Goldblum M, Gupta A, Dickerson JP, Goldstein T. Just how toxic is data poisoning? A unified benchmark for backdoor and data poisoning attacks. In: International conference on machine learning. PMLR; 2021. p. 9389–9398.
17. Lewis GA, Ozkaya I, Xu X. Software architecture challenges for ml systems. In: 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME). 2021. p. 634–638. <https://doi.org/10.1109/ICSME52107.2021.00071>.
18. Goldblum M, Tsipras D, Xie C, Chen X, Schwarzschild A, Song D, Madry A, Li B, Goldstein T. Data security for machine learning: data poisoning, backdoor attacks, and defenses. New York: IEEE; 2020.
19. Ngiam J, Khosla A, Kim M, Nam J, Lee H, Ng AY. Multimodal deep learning. In: ICML, 2011.
20. Sawadogo P, Darmont J. On data lake architectures and metadata management. *J Intell Inform Syst.* 2021;56(1):97–120.
21. Kuhn M, Johnson K. Feature engineering and selection: a practical approach for predictive models. Boca Raton: CRC Press; 2019.
22. Gao J, Li P, Chen Z, Zhang J. A survey on deep learning for multimodal data fusion. *Neural Comput.* 2020;32(5):829–64.
23. Souza VM, dos Reis DM, Maletzke AG, Batista GE. Challenges in benchmarking stream learning algorithms with real-world data. *Data Min Knowl Discov.* 2020;34(6):1805–58.
24. Jagielski M, Oprea A, Biggio B, Liu C, Nita-Rotaru C, Li B. Manipulating machine learning: poisoning attacks and countermeasures for regression learning. In: 2018 IEEE symposium on security and privacy (SP). New York: IEEE; 2018. p. 19–35 <https://doi.org/10.1109/SP.2018.00057>.
25. Biggio B, Nelson B, Laskov P. Poisoning attacks against support vector machines. In: Proceedings of the 29th International conference on machine learning. 2012. p. 1467–1474.
26. Aghakhani H, Meng D, Wang Y-X, Kruegel C, Vigna G. Bullseye polytope: a scalable clean-label poisoning attack with improved transferability. In: 2021 IEEE European symposium on security and privacy (EuroS&P). New York: IEEE; 2021. p. 159–178.
27. Zhu C, Huang WR, Li H, Taylor G, Studer C, Goldstein T. Transferable clean-label poisoning attacks on deep neural nets. In: International conference on machine learning. 2019. p. 7614–7623.
28. Angelo JS, Barbosa HJ. A study on the use of heuristics to solve a bilevel programming problem. *Int Trans Oper Res.* 2015;22(5):861–82.
29. Huang WR, Geiping J, Fowl L, Taylor G, Goldstein T. Metapoisn: practical general-purpose clean-label data poisoning. *Adv Neural Inf Process Syst.* 2020;33:12080.
30. Bard JF. Practical bilevel optimization: algorithms and applications, vol. 30. Raleigh: Springer; 2013.
31. Colson B, Marcotte P, Savard G. An overview of bilevel optimization. *Ann Oper Res.* 2007;153(1):235–56. <https://doi.org/10.1007/s10479-007-0176-2>.
32. Sethi TS, Kantardzic M. On the reliable detection of concept drift from streaming unlabeled data. *Expert Syst Appl.* 2017;82:77–99. <https://doi.org/10.1016/j.eswa.2017.04.008>.
33. Demsar J, Bosnic Z. Detecting concept drift in data streams using model explanation. *Expert Syst Appl.* 2018;92:546–59. <https://doi.org/10.1016/j.eswa.2017.10.003>.
34. Buneman P, Khanna S, Wang-Chiew T. Why and where: a characterization of data provenance. In: Van den Bussche J, Vianu V, editors. International conference on database theory. Heidelberg: Springer; 2001. p. 316–30. <https://doi.org/10.1007/3-540-44503>.
35. Kim H, Park J, Bennis M, Kim SL. Blockchain-based on-device federated learning. *IEEE Commun Lett.* 2019;24(6):1279–83. <https://doi.org/10.1109/LCOMM.2019.2921755>.
36. Barrak A, Eghan EE, Adams B. On the co-evolution of ml pipelines and source code-empirical study of dvc projects. In: 2021 IEEE International conference on software analysis, evolution and reengineering (SANER). New York: IEEE; 2021. p. 422–433.
37. Liu H, Zhu X, Lei Z, Li SZ. Adaptiveface: adaptive margin and sampling for face recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR). 2019.
38. Krizhevsky A. Learning multiple layers of features from tiny images. Master's thesis, University of Tront. 2009.
39. He X, Zhao K, Chu X. Auttml: a survey of the state-of-the-art. *Knowl Based Syst.* 2021;212:106622.
40. Truong A, Walters A, Goodsitt J, Hines K, Bruss CB, Farivar R. Towards automated machine learning: Evaluation and comparison of auttml approaches and tools. In: 2019 IEEE 31st International conference on tools with artificial intelligence (ICTAI). 2019. p. 1471–1479. <https://doi.org/10.1109/ICTAI.2019.00209>.
41. Le Q, Zoph B. Neural architecture search with reinforcement learning. 2016. <https://arxiv.org/abs/1611.01578>.
42. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. *Adv Neural Inform Process Syst.* 2012. <https://doi.org/10.1145/3065386>.
43. LeCun Y, Huang FJ, Bottou L. Learning methods for generic object recognition with invariance to pose and lighting. In: Proceedings of the 2004 IEEE computer society conference on computer vision and pattern recognition. IEEE. 2004. p. 104.
44. Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information. *Trans Assoc Comput Linguist.* 2017;5:135–46.
45. Sarki R, Ahmed K, Wang H, Zhang Y, Ma J, Wang K. Image preprocessing in classification and identification of diabetic eye diseases. *Data Sci Eng.* 2021;6(4):455–71.