Detecting Poisoning Attacks with DynaDetect

Sabrina Perry[1], Yili Jiang[1](✉), Fangtian Zhong[2], and Chong Yu[3]

[1] University of Mississippi, University, MS 38677, USA
yjiang7@olemiss.edu
[2] Montana State University, Bozeman, MT 59717, USA
[3] University of Cincinnati, Cincinnati, OH 45221, USA

Abstract. In the age of increasing reliance on machine learning (ML) in various environments, ensuring the security and reliability of ML models is essential. Data poisoning attacks pose a significant threat to ML models, compromising their reliability. Although the traditional K-Nearest Neighbor (KNN) algorithm can offer potential defense mechanisms due to its adaptability and capability to detect poisoned data, its static nature limits its effectiveness against dynamic malicious challenges. To this end, this work proposes DynaDetect, a dynamic KNN-based algorithm designed to detect data poisoning attacks. Our methodology adapts the traditional KNN model to a dynamic framework, allowing to adjust its parameters, such as the number of neighbors considered, based on the characteristics of the data. The experimental results indicate a marked improvement in the detection accuracy of poisoned data, enhancing the reliability of ML models.

Keywords: machine learning security · dynamic KNN · data poisoning attacks

1 Introduction

Machine learning (ML) has gained significant attention in various industries, including healthcare and autonomous vehicles, due to its ability to learn and make predictions guided by patterns extracted from data [2]. As ML models evolve, they increasingly utilize sophisticated data-driven techniques to enhance their decision-making abilities. Although this allows for improving accuracy in predictions, it simultaneously makes these models vulnerable to malicious attacks, compromising their security, reliability, and overall performance [16]. Data poisoning attacks have emerged as significant threats, impacting various applications, ranging from autonomous vehicles and spam filters to healthcare [5]. In these attacks, attackers can manipulate training data, causing models to make incorrect predictions or classifications, leading to degradation of a model's performance [1,9,13].

As we refine ML models for better decision-making in various applications like healthcare and autonomous driving, their susceptibility to data poisoning increases. This attack manipulates training data, leading directly to incorrect model outputs. For example, in healthcare, corrupted or poisoned medical images can lead to misdiagnoses, potentially endangering patients. In a digital environment, a poisoned spam filter might fail to catch phishing attempts, increasing the risk of data breaches. Likewise, in autonomous vehicles, misinterpretations of traffic signs can lead to potential accidents on roads and highways.

Given the increasing threats from poisoning attacks in machine learning, various detection and mitigation techniques have been developed. The K-Nearest Neighbors (KNN) algorithm stands out due to its straightforward approach and adaptability. KNN's principle, based on the proximity of data points in a feature space, makes it a practical choice for classification tasks. This method is particularly effective because it does not presume any specific data distribution, enhancing its versatility across different datasets. In addition to KNN, other significant methods have been explored. For instance, supervised learning techniques have shown promise, as evidenced by Ning *et al.* [8] in their work using a Resnet18 classifier trained on a mix of poisoned and clean images. Moreover, heuristic approaches have been investigated for online learning contexts. Zhang *et al.* [18] have made notable contributions by employing model predictive control and deep reinforcement learning to counteract data poisoning.

However, the KNN algorithm still faces significant barriers when addressing sophisticated poisoning attacks and unpredictable changes in real-world data. Firstly, the algorithm's static nature leads to high computational overhead in large datasets where computing distances among all data points is necessary. Secondly, optimizing the value of k is crucial, as it determines the number of nearest neighbors to be considered and, therefore, affects the algorithm's performance in terms of accuracy. Thirdly, in high-dimensional spaces, KNN suffers from high computational costs due to the complexity of calculating distance measurements. In addition, the traditional method of assigning equal weights to all neighbors does not guarantee the most accurate results, highlighting the need for more advanced detection methods to maintain the reliability and efficiency of machine learning models amidst dynamic challenges [10,12,14,17].

To address the above challenges, we are motivated to design a defense framework that can detect data poisoning attacks in a dynamic manner. Particularly, in this work, we consider Poison Brew [5], a typical data poisoning attack that employs a gradient matching technique. To efficiently detect Poison Brew, we propose DynaDetect: a dynamic KNN-based detection algorithm. Key highlights of our work contributions are:

– **Dynamic Parameter Tuning:** Unlike the traditional KNN, which uses a fixed k value and distance metric, our proposed DynaDetect adaptively adjusts these parameters. It accesses the data, adjusts k, and chooses the most appropriate distance metric based on current data. This enhances its resilience to subtle Poison Brew [5] manipulations.
– **Gradient Analysis for Neighbor Assessment:** In our approach for detecting the Poison Brew [5] attack, where data gradients are intentionally manipulated, our work adopts a strategic method. It does not treat all neigh-

boring data points equally. Instead, the algorithm examines both the gradient directions, indicating how data points are changing and their distances from one another. This approach is built on the assumption that attacks display two principal behaviors: the uniformity in the direction of data changes among close neighbors and the intentional grouping of altered data points to create densely packed areas of concern within the data space.

- **Data Understanding:** Our model employs an evolving approach to develop a profound understanding of the data it processes. Especially over time, it becomes adept at discerning clean data points from potentially poisoned data created through Poison Brew [5] gradient matching.
- **Improved Detection Performance:** Through experiments, we demonstrate that DynaDetect outperforms the traditional KNN model in accurately identifying poisoning attacks. Our results significantly improve detection accuracy, highlighting the practical effectiveness of our dynamic KNN-based approach in real-world scenarios.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 describes Poison Brew, a data poisoning attack conducted by Geiping *et al.* [5], leveraging their methodology and findings as a framework in our work to develop a detection algorithm for poisoning attacks. Section 4 delves deeply into our proposed DynaDetect, a dynamic KNN-based methodology, emphasizing its differences from the traditional KNN approach. Following this, Sect. 5 outlines our experimental design, encompassing datasets and configurations. Section 6 unveils the experimental results of the proposed DynaDetect in identifying poison images. Moving forward, Sect. 7 scrutinizes DynaDetect's performance and highlights its strengths. Section 8 draws conclusions.

2 Related Work

The field of secure machine learning has witnessed a growing interest on researching data poisoning attacks targeting machine learning algorithms. In these data poisoning attacks, attackers intentionally manipulate specific instances in the training data to compromise the performance of the machine learning system.

In further exploration of these threats, Aryal *et al.* [2] investigated the resilience of ML-based malware detectors to data poisoning attacks. Their research assessed eight widely used machine learning models in the realm of malware detection by introducing 10 percent and 20 percent poisoned data into the training datasets. All models experienced a decline in performance, exposing the crucial vulnerabilities of malware detection when confronted with intentional poisoning attacks. Unexpectedly, certain models, such as the SVM, exhibited better performance with 20 percent poisoned data than 10 percent. This surprising outcome was attributed to the unconstrained approach to data poisoning that was employed.

Seetharaman *et al.* [11] introduced an influenced-based defense mechanism to counter data poisoning attacks in online learning. They addressed the deceptive nature of such attacks, which carefully corrupt training data to damage learning models. Their strategy employed influence functions coupled with Slab data sanitization method, focusing on identifying and reducing the impact of questionable data points.

Paudice *et al.* [9] proposed a defense mechanism utilizing the KNN analogy in response to the increasing threats of poison attacks. Their approach aimed to restore the label integrity by enforming homogeneity among close instances, especially those distant from the decision boundary. By examining each instance's k nearest neighbor in the feature space, the algorithm determined if this local area's dominant label aligned with the instance's label. If a significant misalignment is detected, determined by a threshold parameter n, the instance label was adjusted to match the local consensus. This method counters the attack by relabeling suspicious instances, potentially neutralizing the adversarial effect of data poisoning. Additionally, Taher *et al.* [13] introduced the Label-based Semi-Supervised Defense (LSD) approach, designed to counter data poisoning attacks with partially labeled data. The LSD algorithm ranked predicted labels based on validation data and utilized techniques like Label Propagation (LP) and Label Spreading (LS) to mitigate labeling noise. It aims to create a two-stage framework for learning flipped labels and employs voting to determine final labels for training samples.

While the aforementioned works have significantly advanced our understanding of machine learning attacks, they have limitations. Our work aims explicitly to enhance the detection of poisoned data in machine learning, addressing these gaps. Aryal *et al.* [2] insightful work, for instance, primarily focuses on fixed percentages of poisoned data and might not fully capture subtler attack patterns. Seetharaman *et al.* [11] method, while effective in online learning environments, may have limited applicability in other contexts. Paudice *et al.* [9] KNN-based approach might only identify some poisoned data, particularly in cases of sophisticated poisoning techniques. Likewise, Taher *et al.* [13] strategy for handling partially labeled data may need help with complex labeling noise. DynaDetect advances the field by overcoming the limitations of current models, serving as an adaptable tool for detecting data poisoning threats in machine learning systems.

3 Poison Brew

Geiping *et al.* [5] revealed a detailed data poisoning attack, Poison Brew, which highlights subtle vulnerabilities in machine learning models. Poison Brew employs a gradient matching technique to subtly inject poisoned data into the dataset, unnoticeable yet effectively influencing model training. This clever technique of embedding poisoned data enables evasion of traditional detection mechanisms.

In detail, by using gradient matching techniques, Poison Brew [5] can subtly guide the model's learning path by introducing poisoned data points that ingeniously mimicked the gradients of target data. This methodology is expressed in Eq. 1

$$B(\Delta,\theta) = 1 - \frac{\left\langle \nabla_\theta \mathcal{L}(F(x^t,\theta), y^{adv}), \sum_{i=1}^{P} \nabla_\theta \mathcal{L}(F(x_i+\Delta_i,\theta), y_i) \right\rangle}{\|\nabla_\theta \mathcal{L}(F(x^t,\theta), y^{adv})\| \cdot \left\| \sum_{i=1}^{P} \nabla_\theta \mathcal{L}(F(x_i+\Delta_i,\theta), y_i) \right\|}. \quad (1)$$

1. $B(\Delta,\theta)$: Quantifies the alignment of the gradients between the poisoned and target data, a measure necessary for the success of the attack. The closer this value is to 1, the more influential the poisoning.
2. The gradient, represented by $\nabla_\theta L$, shows how the loss or error changes as the model parameters, θ, are tweaked.
3. $F(x_t,\theta)$: This is what the model predicts for an input, x_t, with current parameters.
4. y_{adv}: The desired (often malicious) outcome for the target data.
5. $x_i + \Delta_i$: These are the real data points that have been slightly altered.
6. y_i: The actual labels for those data points.

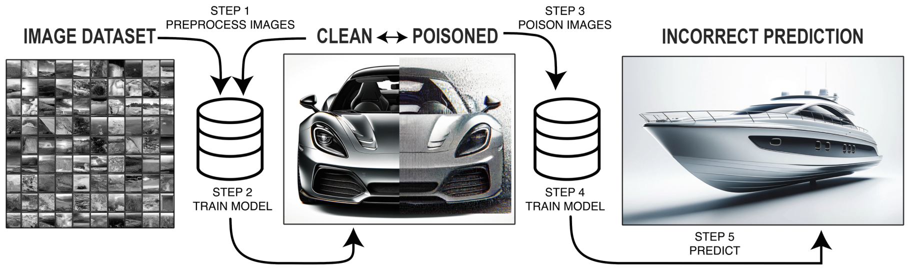

Fig. 1. Poison Brew [5]: a data poisoning attack on image classification using gradient matching techniques.

Figure 1 depicts the stages of the poisoning attack on a machine learning model. The attack begins with a dataset of clean, unaltered images. Before the model starts training, selected images are altered with subtle gradient changes that were difficult for humans to spot but significant enough to deceive the machine learning algorithm. These altered images, now poisoned, are reintroduced into the model's training data. During training, the model learns from both the clean and the poisoned images, which unknowingly misleads to incorrect predictions. Therefore, the model displays inaccuracies in classifying new images.

The enlightening outcomes of their research demonstrates the effectiveness of the Poison Brew [5] attack. As Poison Brew introduces nuanced vulnerabilities expertly concealed, it bypasses standard detection safeguards without difficulty, effectively but noticeably misleading predictions. Therefore, it is necessary to design defense mechanisms against such data poisoning attacks.

Building on this groundbreaking research, our work proposes a dynamic KNN-based algorithm to detect Poison Brew. It leverages the principles of gradient alignment, dynamically adjusting the weighting and assessment of neighboring data points. Our algorithm operates in real-time, constantly analyzing data and adapting to new threats as they emerge. It provides continuous detection against advanced threats, guaranteeing machine learning models' integrity.

4 Methodology

4.1 Dynamic KNN Overivew

The KNN algorithm is an important classification algorithm that needs several parameters to work properly. Its main goal is to predict the label of a given data point. Features are attributes that help to differentiate data points from each other. In the traditional KNN method, the number of nearest neighbors considered for labeling is determined by the k parameter. However, our work introduces a new variable radius parameter called b that is used to establish a flexible radius around a data point for neighborhood determination. This method offers more flexibility than the fixed k parameter and improves the algorithm's resilience by allowing adjustments to different data densities. As a result, it enhances the detection of abnormalities that could indicate data poisoning.

4.2 Proposed DynaDetect Detection Algorithm Using Dynamic KNN

We propose DynaDetect, a dynamic KNN-based detection algorithm, to detect poisoned data in Poison Brew. The dynamic KNN approach with our methodology is simplified through the KNNModelPersisted class. This class incorporates a training methodology that handles parameters such as *features*, *labels*, k, b, and the type of methods. When set to adaptive, the RadiusNeighborsClassifier is employed, which uses a specified radius b for neighbor identification. The trained classifier predicts labels using the radius b. Each model possesses a unique identifier derived from attributes such as method type, k value, and b value, simplifying the model storage and retrieval process.

As outlined in Algorithm 1, our methodology is central to applying dynamic KNN to differentiate between clean images (free from malicious poisoning) and poisoned images (manipulated) within diverse datasets, inspired by the findings of Geiping *et al.* [5]. Our method involves training a classifier on a dataset containing clean and poisoned images, with the primary goal of accurately identifying the poisoned images. This enhances the model's defense against malicious manipulations. The approach involves dynamically adjusting the number of neighbors in the KNN algorithm, allowing it to effectively adapt to different data densities.

4.3 Adaptive Parameterization in KNN-Based Detection

In the development of our DynaDetect algorithm, we initially set specific values for two key parameters: the threshold (T) and radius (b), based on our analysis of the test datasets. However, it's important to note that these values are not fixed constants but rather starting points for parameter tuning. The selection of these values was influenced by the characteristics of the datasets used in our initial experiments, and they may vary in different contexts or with diverse types of poisoning attacks.

Our algorithm assesses the likelihood of each data point being manipulated within the dataset. It computes a 'poisoned score' for every data point based on certain characteristics identified as indicators of data poisoning, such as the alignment with its neighbors. Higher scores indicate a greater probability of manipulation. In our preliminary analysis, we observed a consistent pattern where data points with poisoned scores above the threshold of 5 were generally associated with poisoning. Therefore, we initially classify data points with scores exceeding this threshold as poisoned, indicating a high probability of manipulation. In contrast, we consider those below the threshold as clean, suggesting a lower likelihood of poisoning.

However, the adaptability of the threshold T and the radius b is a key feature of our methodology, allowing for the algorithm's effective application across various datasets and attack scenarios. This flexibility is crucial, as the optimal threshold and radius may differ based on specific dataset characteristics and the nature of the poisoning attack. We emphasize the need to adjust these parameters to ensure the robustness and accuracy of the algorithm in identifying poisoned data in diverse situations.

Algorithm 1. DynaDetect: Dynamic KNN-based Detection

```
1: Input: A set of mixed training dataset (both clean and poisoned data), dataset of
   poisoned images $X_p$, radius $b$, threshold $T = 5$
2: Initialize a classifier $\mathcal{C}$
3: Train $\mathcal{C}$ on mixed dataset using RadiusNeighborsClassifier with radius $b$
4: Initialize predictions list $P$ to store status of each image $x_i$
5: for each image $x_i$ in $X_p$ do
6:    Classify $x_i$ using $\mathcal{C}$
7:    if classifier identifies $x_i$ as an anomaly and the anomaly score is above threshold
      $T$ then
8:       Append "poison" to $P$
9:    else
10:      Append "clean" to $P$
11:   end if
12: end for
13: return $P$
```

4.4 Distance Metrics

KNN algorithm distance metrics play an important role. These metrics measure the proximity between two data points and help to determine whether the data points are considered neighbors. The choice of distance metrics can significantly influence the performance of the KNN algorithm [6] [3]. In this work, we employ one primary distance metric: Mean Squared Error (MSE).

MSE: To understand the difference between two images, MSE is utilized, as shown in Eq. 2. It calculates the average squared difference between the corresponding pixels of the two images

$$\mathrm{MSE}(I_1, I_2) = \frac{1}{N}\sum_{i=1}^{N}(I_{1i} - I_{2i})^2, \tag{2}$$

where I_1 represents the first image and I_2 the second, and N is the number of pixels. The importance of utilizing MSE in our work is its ability to quantify the similarities between images. In Algorithm 1, MSE accesses the similarities between a test image x_i and its neighbors within the specified radius b. If the MSE between x_i and its neighbors falls below a certain threshold, it indicates that these neighbors are very similar to x_i, which is valuable for classifying test images.

Neighborhood Consideration in Dynamic KNN: Unlike the traditional KNN, which classifies a sample based on the top k closest training samples, the dynamic radius-based KNN used in this work employs a different approach for defining neighbors. In this method, neighbors are identified as all training samples that fall within a predefined radius b

$$\mathrm{Neighbors}(x) = x', |, \mathrm{MSE}(x, x') < b, \tag{3}$$

where x is the test sample, and y represents samples from the training data. In our work, a radius of 0.50 was utilized to define the neighborhood. This radius was determined through a series of experiments with various radius values to find the one that best balances the accurate detection of poisoned data and high classification performance. A 0.50 radius ensures that only samples within this distance are considered neighbors, effectively allowing our algorithm to adapt to the local data density. Such neighborhood consideration is crucial for our work as it determines which training samples are relevant for classification, enhancing the algorithm's ability to detect poisoned data in diverse datasets by making decisions based on nearby data points.

5 Implementation

5.1 Hardware Configuration

Our experiments are conducted on a Linux server equipped with 4 NVIDIA GPUs, ensuring efficient computation and parallel processing capabilities.

5.2 Datasets

Our experiments utilize diverse datasets to evaluate the dynamic KNN-based detection mechanism. These datasets include:

- **CIFAR-10** [7]: This dataset contains 50,000 images with dimensions of 32×32 pixels in RGB format, distributed across 100 classes, each having 500 images. The CIFAR-10 is a well-established dataset for assessing image classification tasks. For our training, we used a subset of this dataset, consisting of 48,500 poisoned images and an equal number of clean images.
- **Fashion MNIST (F-MNIST)** [15]: This dataset comprises 60,000 images of 28×28 pixels in grayscale format, divided into 10 classes, each containing 6,000 images. Out of these, 50,000 images are designated for training and 10,000 for testing. The images depict various clothing items. Given their lower resolution, distinguishing between some images can be challenging, which adds complexity to the classification task. Our training employed a balanced selection of 33,500 poisoned and 33,500 clean images.
- **ImageNet** [4]: This dataset includes an extensive collection of over 1.2 million images, but for our work, we specifically utilized a subset of 40,000 images. These consist of 20,000 images with dimensions of 224×224 pixels in RGB format, chosen from 1,000 classes, each class contributing 200 images. The diversity of the dataset, characterized by varying object scales, occlusions, and contexts, presents a significant challenge in identification tasks. For training our model, we selected a balanced set comprising 20,000 poisoned and 20,000 clean images from this subset, providing comprehensive and high-dimensional data to rigorously test and validate our model's performance.

After training our model with the specified datasets, we conduct a validation phase to assess the model's ability to accurately identify poisoned data. For this purpose, we use a separate set of 1,000 known clean images and 1,000 known poisoned images that the model has never encountered before. This validation set is specifically chosen to ensure a thorough test of the model's ability to generalize to new data.

The inclusion of this unseen data is crucial in our experimental setup, as it provides a measure of the model's effectiveness in real-world conditions, where it encounters data that is not part of its training environment. By using this approach, we can evaluate our model's accuracy and robustness in detecting poisoned data under conditions that closely mimic actual deployment scenarios.

5.3 Attack Intensity

In evaluating our approach for detecting data poisoning, we quantify the strength of the poisoning attack using an epsilon (ϵ) value, which we set to 8. This ϵ value dictates the maximum change allowed for each pixel in an image. Essentially, it caps how much we can adjust the brightness or color of each pixel. With ϵ set to 8, we ensure that the alterations to each pixel are moderate but not extreme. This constraint on pixel intensity alteration, defined by ϵ is expressed in Eq. 4:

$$x' = x + \delta, \|\delta\|_\infty \leq \epsilon, \tag{4}$$

where x is the original pixel value, x' is the altered value, and δ is the change.

6 Data Preprocessing

We commence our with the dataset initially employed by Geiping *et al.* [5], specifically by introducing poisoned data to enhance the resilience of our analysis. A meticulous preprocessing phase was conducted to guarantee the cleanliness of the dataset and ensure accurate labeling. In the initial phase, images from the entire dataset were loaded and subjected to preprocessing steps to optimize the data for consistency and effectiveness. Each image was resized to a uniform dimension of 224×224 pixels, ensuring consistent compatibility with our model.

Afterward, the images were converted to grayscale, which allows the model to focus on the structure of an image rather than the color information. This is beneficial so that the model can focus on essential features. Normalization was applied to the preprocessed images, ensuring that the dataset values were within a suitable scale. Normalization ensures that all pixel values are within a similar scale, which is essential for the model to learn effectively during the training process. This crucial normalization process enhanced the model's stability and performance effectiveness. Utilizing the powerful capabilities of the Python NumPy library, each image was flattened by a process that transformed them into one-dimensional arrays, which further optimized them for efficient and precise analysis by our algorithm.

To enhance our work, we extended our dataset beyond the initial dataset used by Geiping *et al.* [5]. This extension involved incorporating a range of diverse datasets, each carefully preprocessed and organized to maintain the highest data integrity and reliability in our analysis. Such rigor ensures that our findings provide a robust examination of our algorithm's performance in detecting poisoned data across various datasets.

7 Performance Evaluation

Our evaluation compares traditional KNN [14] and our proposed DynaDetect. Traditional KNN, with its static selection of k nearest neighbors, may be vulnerable to poisoned images due to its fixed parameters. In contrast, DynaDetect dynamically adjusts k, offering enhanced stability against poisoned images. We assess the performance of both algorithms across various datasets-CIFAR-10, F-MNIST, and ImageNet-to ensure the reliability of predictions in the presence of poisoning attacks, as shown in Figs. 2, 3 and 4.

In our performance evaluations, we observe consistent improvements with DynaDetect over traditional KNN across all datasets. For example, using the CIFAR-10 dataset, DynaDetect shows a 4–5% higher detection accuracy across different k values compared to traditional KNN. Similarly, in the F-MNIST and ImageNet datasets, DynaDetect maintains a higher accuracy, reinforcing its adaptability and robustness in diverse image classification tasks. These results illustrate DynaDetect's ability to handle complex patterns in the data, providing more accurate and reliable classifications.

Across all tested datasets, the trend remains clear: DynaDetect consistently outperforms traditional KNN, especially as the k value increases. This consistent performance across different types of datasets from grayscale images in F-MNIST to the varied and complex images in ImageNet-demonstrates the versatility and effectiveness of DynaDetect in varied image recognition scenarios.

In conclusion, our evaluation reveals the comparative performance of the traditional KNN and our proposed DynaDetect KNN algorithm in detecting poisoned images across various datasets. The traditional KNN, with its static k, shows limitations in adaptability and performance across diverse datasets, requiring a careful initial k selection. In contrast, our work displays improved performance and adaptability with its dynamic k adjustment. This finding highlights the DynaDetect KNN's resilience and reliability, making it a more helpful choice for handling dynamic datasets and ensuring possible predictions against poisoning attacks.

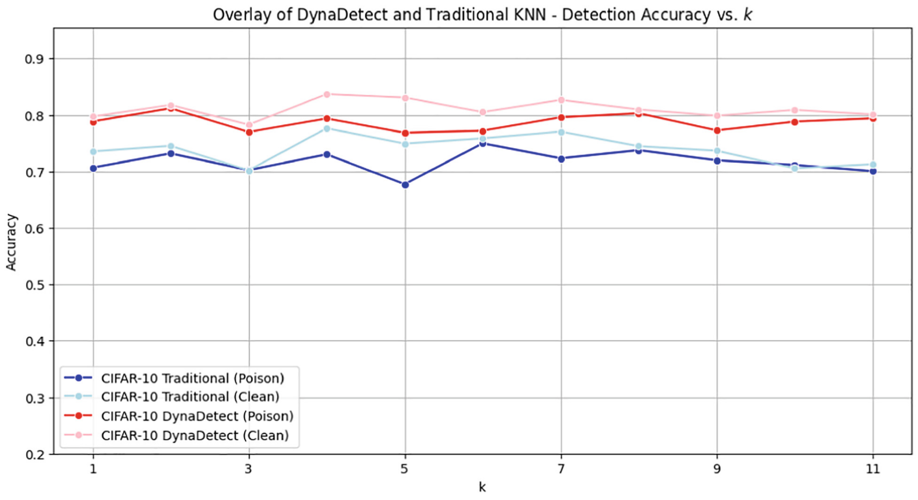

Fig. 2. Performance evaluation of Traditional KNN and DynaDetect on CIFAR-10 Dataset.

8 Discussion

This work introduces DynaDetect, a dynamic KNN-based detection algorithm, advancing the detection of poisoned images in machine learning datasets. We

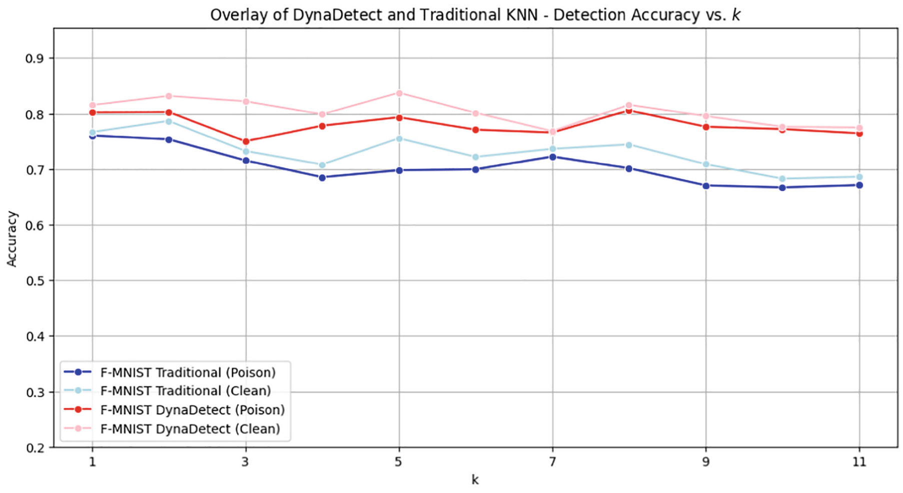

Fig. 3. Performance evaluation of Traditional KNN and DynaDetect on FMNIST Dataset.

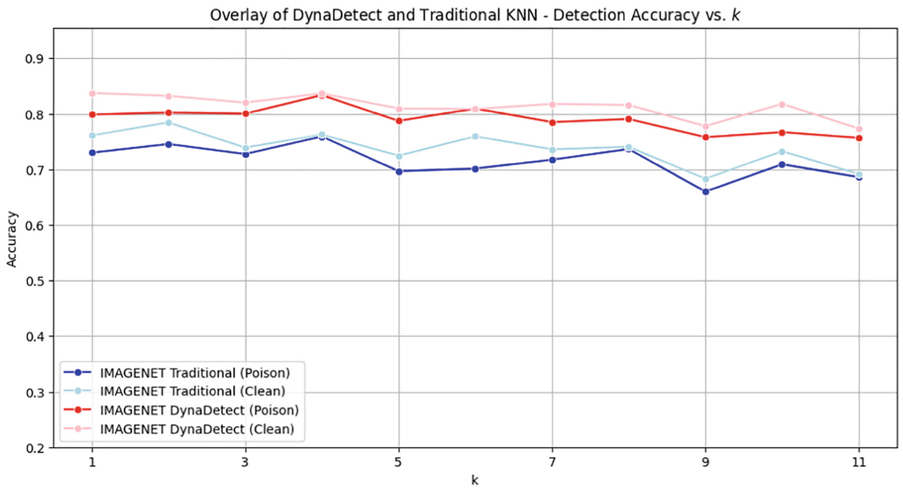

Fig. 4. Performance evaluation of Traditional KNN and DynaDetect on ImageNet Dataset.

compare its performance with traditional KNN methods across diverse datasets such as CIFAR-10, F-MNIST, and ImageNet. These comparisons demonstrate DynaDetect's superior adaptability and accuracy.

Traditional KNN, with its static approach to selecting the k nearest neighbors, exhibits vulnerabilities, especially when dealing with poisoned images. DynaDetect addresses this by dynamically adjusting k, leading to more stable performance in the presence of poisoned data. Our results show that DynaDetect consistently outperforms traditional KNN in detection accuracy, with improvements ranging from 4% to 5%, depending on the dataset and k value.

The capabilities of the dynamic KNN-based detection algorithm indicate its potential usefulness in various fields. For instance, in autonomous vehicles, where decision-making relies heavily on data integrity, a dynamic KNN-based detection algorithm could be key in distinguishing between clean images and those compromised by poison attacks. In healthcare, the algorithm's precision in identifying clean data could be instrumental in ensuring accurate diagnoses, free from the influence of manipulated information. Similarly, dynamic KNN could enhance the accuracy of spam filters in digital communication by more effectively separating authentic emails from spam. These examples highlight the possible benefits of applying our findings to practical, real-world challenges where clean data is crucial.

9 Conclusion

In response to the evolving field of image detection and the increasing threat of poisoning attacks, we have proposed DynaDetect, a dynamic KNN-based detection algorithm designed to detect Poison Brew attacks. Our algorithm improves the detection accuracy of data poisoning attacks by addressing the drawbacks of traditional KNN models. Our experimental findings have highlighted the following advantages of the proposed DynaDetect algorithm:

- Our dynamic KNN-based detection algorithm has consistently outperformed traditional KNN models through testing across multiple datasets and varying k values. The significant improvement in detection accuracy as the k values increase highlights the benefits of our dynamic parameter tuning and gradient analysis for neighbor assessment.
- Furthermore, our model identifies poisoned data instances, showcasing its stability in the face of growing poisoning attacks. These results emphasize the real-world impact of our dynamic parameter tuning, gradient analysis for neighbor assessment, and data understanding contributions.

Our future work will focus on enhancing the robustness of DynaDetect and its applicability in real-time scenarios. We aim to optimize its performance for practical applications such as image recognition in autonomous vehicles. Moreover, we see potential in integrating DynaDetect with other machine learning models, which could lead to significant advancements in data security and model reliability.

Acknowledgements. This work is partially supported by IDS Data Science and AI Fellowship at the University of Mississippi.

References

1. 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), pp. 159–178. IEEE (2021)
2. Aryal, K., Gupta, M., Abdelsalam, M.: Analysis of label-flip poisoning attack on machine learning based malware detector. In: 2022 IEEE International Conference on Big Data (Big Data), pp. 4236–4245. IEEE (2022)
3. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR09 (2009)
5. Geiping, J., et al.: Witches' brew: industrial scale data poisoning via gradient matching. In: International Conference on Learning Representations (2021). https://openreview.net/forum?id=01olnfLIbD
6. Jain, A.K., Duin, R.P.W., Mao, J.: Statistical pattern recognition: a review. IEEE Trans. Pattern Anal. Mach. Intell. **22**(1), 4–37 (2000)
7. Krizhevsky, A., Hinton, G.E.: Learning multiple layers of features from tiny images. University of Toronto, Tech. rep. (2009)
8. Ning, R., Li, J., Xin, C., Wu, H.: Invisible poison: a blackbox clean label backdoor attack to deep neural networks. In: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, pp. 1–10 (2021)
9. Paudice, A., Muñoz-González, L., Lupu, E.C.: Label sanitization against label flipping poisoning attacks. In: Alzate, C., et al. (eds.) ECML PKDD 2018. LNCS (LNAI), vol. 11329, pp. 5–15. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-13453-2_1
10. Ray, S.: A quick review of machine learning algorithms. In: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), pp. 35–39. IEEE (2019)
11. Seetharaman, S., Malaviya, S., Vasu, R., Shukla, M., Lodha, S.: Influence based defense against data poisoning attacks in online learning. In: 2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS), pp. 1–6. IEEE (2022)
12. Sun, S., Huang, R.: An adaptive k-nearest neighbor algorithm. In: 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery, vol. 1, pp. 91–94. IEEE (2010)
13. Taheri, R., Javidan, R., Shojafar, M., Pooranian, Z., Miri, A., Conti, M.: On defending against label flipping attacks on malware detection systems. Neural Comput. Appl. **32**, 14781–14800 (2020)
14. Taunk, K., De, S., Verma, S., Swetapadma, A.: A brief review of nearest neighbor algorithm for learning and classification. In: 2019 International Conference on Intelligent Computing and Control Systems (ICCS), pp. 1255–1260. IEEE (2019)
15. Zalando, S.E.: Fashion MNIST (2023). https://github.com/zalandoresearch/fashion-mnist
16. Zhang, J., et al.: Poison ink: robust and invisible backdoor attack. IEEE Trans. Image Process. **31**, 5691–5705 (2022)

17. Zhang, S., Li, X., Zong, M., Zhu, X., Wang, R.: Efficient kNN classification with different numbers of nearest neighbors. IEEE transactions on neural networks and learning systems **29**(5), 1774–1785 (2017)
18. Zhang, X., Zhu, X., Lessard, L.: Online data poisoning attacks. In: Proceedings of the Learning for Dynamics and Control Conference, pp. 201–210. PMLR (2020)