CleanSight: Detection Strategies for Label-Flipping Data Poisoning Attacks on MNIST

Devon Alexander[1,2], Eli Cook[1,3], Adam Fridley[1], Ashraf Ibrahim[1], Hunter Oakey[1,*], and Hunter Moore[4].

[*] *Corresponding Author: hho2em@virginia.edu*

[1] *Systems Engineering, School of Engineering, University of Virginia, Charlottesville, VA*

[2] *Computer Science, School of Engineering, University of Virginia, Charlottesville, VA*

[3] *Mathematics, College of Arts and Sciences, University of Virginia, Charlottesville, VA*

[4] *Hardshell Inc. Charlottesville, VA*

***Abstract*—The accelerating deployment of image recognition models in critical decision-making domains underscores the importance of ensuring reliable training data for present and future Artificial Intelligence (AI) and machine learning (ML) models. A significant threat to training data is label-flipping, in which an adversary alters classification labels in the training set to degrade model performance. In this work, we investigated four detection strategies to identify mislabeled data in the MNIST image dataset, using Gaussian Mixture Models (GMMs), k-means clustering, Mahalanobis distance, and Wasserstein distance, respectively. We developed an experimental pipeline that automatically label-flips samples to test these detection strategies. K-Means performed best with an overall average AUC of 0.9761 and recall of 0.9312, while GMM had the best overall average precision of 0.8550. This research contributes to the field of AI and data security by conducting a preliminary investigation into more comprehensive yet practical pre-training defenses, as demonstrated on MNIST. Future work would seek to improve the effectiveness of these methods by exploring their generalization onto different datasets and validating results with model training.**

***Index Terms*—Area under the ROC curve (AUC), data poisoning, Gaussian mixture models, k-means clustering, label-flipping, log-likelihood, Mahalanobis distance, precision, recall, Wasserstein distance**

I. Introduction

Hardshell Inc., a "data-centric AI security" startup based in Virginia, aims to provide tools and services for clients to secure their data [1]. Some of its latest research focuses on computer vision vulnerabilities, where image classification models can be applied to more complex systems, such as autonomous driving and surveillance. We are working with Hardshell researchers to develop "CleanSight" to identify mislabeled and anomalous image data, primarily defending against data poisoning attacks.

Data poisoning occurs when an adversary intentionally injects faulty data into a model to reduce its performance. In some cases, "injecting poisons that only account for 1% of the [total] training dataset" can invalidate models, enabling system-level failures [2]. For example, Anthropic found that only 250 poisoned documents were needed to compromise a model regardless of the size of its dataset [3]. For reference, the volume of these documents represented only 0.00016% of total training data in a 13-billion-parameter model. This shifts adversarial priorities from larger poisoning attacks to smaller, highly optimized ones [3]. Therefore, even small data poisoning attacks result in big shifts in model outputs, making it necessary to research how to mitigate data poisoning.

CleanSight addresses this problem with a pipeline to automate mislabeled image data detection, focusing on MNIST. This pipeline works by running four types of label-flipping attacks and four detection methods, controlling for variability by running these tests over thirteen random seeds. This work is similar to recent work by Abroshan [4], published in 2025. Whereas [4] only varied the detection strategy, dataset, and poisoning percentage, our approach tests along different detection strategies, various poisoning attacks, and more poisoning percentages.

While our work does not assess the impacts on model training, it supports trustworthy training data. By flagging data that is mislabeled by various types of label-flipping attacks, CleanSight motivates the automation of data validation by detecting mislabeled images. For Hardshell AI clients, future work building on CleanSight could reduce development costs and timeline constraints and free human cognitive strain associated with manual data validation efforts. For end users, this could help accelerate the production and deployment of more accurate models relevant to their everyday lives. Despite these impacts, this work is preliminary and it is still necessary to explore the generalizability of our approach.

II. Related Work

There is a lack of robust defenses against label-flipping attacks [4]. Most works are not generalizeable because they assume prior knowledge of poisoning attacks, being "attack-specific" or "environment-specific" [4]. In particular, these defenses require controlled contexts that are unlikely to appear in real applications. Further, there are no defenses that guarantee robust classification of a single point [5].

Outlier detection algorithms and label sanitization strategies are commonly used to combat data poisoning [6]. Outlier detection can "mitigat[e] the effect of optimal poisoning attacks," especially of label-flipping attacks [7] [8]. Strategies such as k-Nearest-Neighbors can identify data points that may have been mislabeled before model training [7]. For sanitization, these

detected data points can either be relabeled or removed [4] [5]. If the features used to detect outliers and sanitize labels "are not discriminative for poison vs. clean, the detection [method] is [often] crippled" [4].

Additionally, [4] proposed a two-phased pipeline to detect label-flipping poisoning attacks: a Behavior Monitoring Module (BMM) and a Detection Module (DM). The BMM dimension-reduces the activations of the penultimate layer of a neural network model. The DM explores both supervised and unsupervised methods, demonstrating highly accurate detection with 0.95 AUC on MNIST. For its Attack Simulation, [4] only poisons single classes at 10% and 20% of that class's distribution. Given this structure, [4] serves as an effective exploration of label-flipping attacks on various datasets. CleanSight builds upon [4] by attacking with three more attack types across ten total poisoning percentages. To expand on possible detection methods, our work also tests several unsupervised strategies different from [4].

III. Methods

Our approach focuses on the MNIST dataset, which is a set of 70,000 grayscale images of the digits 0 through 9, handwritten in various styles [9]. This dataset was chosen for its small number of classes, small image dimensions of 28x28, and its simple domain space of handwritten numbers. We only tested on the training set defined by PyTorch [10], which is 60,000 of the 70,000 images.

We deploy a multi-step pipeline that poisons the data, transforms them into useful features, runs detection strategies, and analyzes the results. For this, we ran a full factorial experiment on 111 poisoning attacks (binned into four different types), ten different poisoning percentages, four different detection strategies, and thirteen seeds to control for uncertainty, resulting in 57,720 unique tests.

A. Poisoning Attacks

We poisoned MNIST at ten levels of poisoning, inclusively, from 0.2% to 2% of the entire dataset, incrementing by 0.2%. We define four different poison attack methods:

- *"All-to-All"*: An untargeted attack that uniformly poisons all classes, randomly flipping labels to any other label. There is one permutation of this attack.
- *"All-to-One"*: A targeted attack that relabels images from any non-target class to the label of the target class. There are ten permutations, one per class.
- *"One-to-All"*: A targeted attack that relabels the images of a target class to the label of any other class but the target class. There are ten permutations, one per class.
- *"One-to-One"*: A targeted attack that relabels the images of one target class to the label of another class. There are ninety permutations of this attack, nine per class.

B. Preprocessing

Our pipeline performs basic preprocessing on the MNIST dataset. Every image in the dataset is resized to 224x224. Grayscale pixel values are then duplicated into three color channels to ensure compatibility with the ResNet18 model used later on in feature engineering. After that, the images are converted to tensors and normalized according to the means and standard deviations derived from ImageNet as described in [11]. We assume that normalizing images on ImageNet makes it more compatible with our ResNet18 model.

C. Feature Engineering

After the pipeline poisons and preprocesses the MNIST dataset, it transforms each image into a feature space based on trained model embeddings. For this, we used a ResNet18 model, pretrained on ImageNet, specifically using weights from PyTorch's ResNet18_Weights.ImageNet1K_V1 [12]. The pipeline fine-tunes this ResNet18 model on the poisoned data over multiple epochs to learn the feature space. The pipeline then takes the activations of the trained model's penultimate layer and reduces those feature embeddings to the most significant fifty features by principal component analysis.

The purpose of this feature engineering is to ensure that the data can be separable by class. Fig. 1 shows that training on a single epoch visibly separates the data by class. By separating classes at this step, our pipeline can more effectively detect mislabeled data in detection. As can be seen in Fig. 1, we used two training epochs for our feature embeddings because it visually separates data by class without overfitting to the poisoned data.

D. Detection Strategies

Our detection strategies are algorithms that assign a "poison score" to images, $p(x) \in [0, 1]$, where x is an image, and its poison score, $p(x)$, represents the chance of it being poisoned or mislabeled. This score is defined differently for each detection strategy, where $p(x) = 1$ corresponds to a strategy's certainty that the given image is poisoned. For their inputs, these strategies take in the image's feature embeddings and their corresponding labels after the dataset poisoning.

1) Gaussian Mixture Models (GMMs): GMMs are a method for fitting data distributions to sums of Gaussian components [13]. This strategy was implemented in Python using Scikit Learn's GMM package [14]. This type of model can handle data that is non-Gaussian by fitting data to multiple components. The detection strategy works in two steps: it determines the maximum log-likelihood that a given image

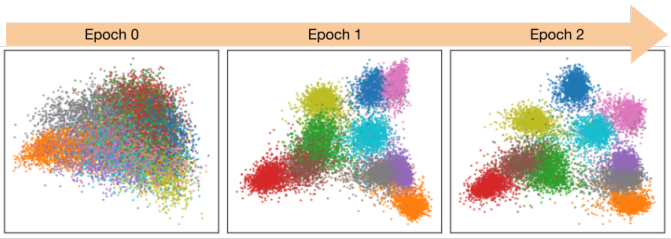

Fig. 1: Each epoch of training on MNIST further separates the data by class. In this figure, each color represents a different class, and each dot represents an image's feature embeddings, reduced to two principal components for visualization. It can be seen that each class's cluster becomes more visually distinguishable with each additional epoch of training.

is under a different class, and it determines the poison score based on that log-likelihood.

The first step is to fit each class's data to a two-component GMM model. From this, each image's features are compared to the feature distributions for every other label, calculating the out-of-class log-likelihoods and storing the maximum. This maximum represents the chance of an image being in the most similar class instead of its own, as expressed in (1) [13]:

$$\ell(\theta|X) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} w_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right) \tag{1}$$

where k is a Gaussian component from the set of K, w_k is its weight, μ_k is its mean, and Σ_k is its covariance matrix.

As exemplified by Fig. 2, out-of-class log-likelihood values follow unimodal distributions for clean classes, whereas poisoned classes tend to follow bimodal distributions. Because of this, the second step of this strategy fits another two-component GMM to the out-of-class log-likelihoods, where the component with the lower mean is assumed to be clean. From this, the poison score is calculated as the posterior probability of not fitting into the "clean" component.

2) K-Means Clustering: K-Means clustering is a method of clustering data points in such a way that the distance between the centroid and a data point in the cluster is minimized. The objective function for k-means clustering is (2) [15]:

$$\sum_{i=1}^{n} \sum_{k=1}^{c} z_{ik} \|x_i - a_k\|^2 \tag{2}$$

where $X = \{x_1, ..., x_n\}$ is a dataset in Euclidean space $\mathbb{R}^d$, and a_k and z_{ij} are defined in (3) and (4), respectively:

$$a_k = \frac{\sum_{i=1}^{n} z_{ik} x_{ij}}{\sum_{i=1}^{n} z_{ik}} \tag{3}$$

$$z_{ik} = \begin{cases} 1 & \text{if } \|x_i - a_k\|^2 = \min_{1 \le k \le c} \|x_i - a_k\|^2 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\|x_i - a_k\|$ is the Euclidean distance between the data point x_i and the cluster centroid a_k.

It is practically infeasible to minimize the objective function of the k-means algorithm with reasonable efficiency. Therefore, the k-means algorithm that we use starts with a random set of centroids and repositions them within $\mathbb{R}^d$ for a certain number of iterations. We decided to use 100 iterations because it was the default number of iterations. The k-means algorithm also requires an input of a desired number of centroids, so we used 10 to match the number of MNIST image classes. In order to calculate the poison score for each image, we first use the dictionary outputted by the k-means algorithm to identify which images were grouped into which clusters. We then compute the poison score for each image as the percentage of images in its cluster that does not match its label.

3) Mahalanobis Distance: Mahalanobis distance is a metric that measures the space between a multivariate normal distribution and a data point. To define the distribution, a mean vector and a covariance matrix must be calculated. From there, the Mahalanobis distance for each data point is calculated in (5) [16]:

$$\sqrt{(u-v)V^{-1}(u-v)^T} \tag{5}$$

Where u is the vector of the data points, v is the mean vector, and V is the covariance matrix. Mahalanobis distance assumes that the data are multivariate normal distributions [16].

We create distributions of each data point of each label, then calculate the Mahalanobis distance of each data point, flagging any data point with a higher likelihood of being poisoned the further it was from the distribution. We fit a normal curve to the distribution of Mahalanobis distances for a given label. We then calculated the poison score using the CDF of the generated distribution. A higher Mahalanobis distance means that the given data point is further from the input distribution, resulting in a higher poison score.

4) Wasserstein Distance: Wasserstein distance measures the minimum cost of transforming one distribution into another. The cost is calculated according to (6) [17]:

$$\inf_{\pi \in \Gamma(u,v)} \int_{\mathbb{R} \times \mathbb{R}} |x - y| d\pi(x, y) \tag{6}$$

where u and v are two one-dimensional probability mass functions and $\Gamma(u, v)$ is the set of all possible joint probability distributions between u and v.

Our Wasserstein distance algorithm works by first grouping all data points according to their labels. Next, for each label, the medians are calculated across each dimension of its data points. Then, the medians of the dimensions are combined into a new data point that will serve as the representative data point for a specific class. After that, the strategy computes the Wasserstein distance between a data point and the representative data point for the class corresponding to the data point's label and returns that value as the poison score. The smaller the Wasserstein distance is between two data points, the smaller the poison score. If the Wasserstein distance between two data points is 0, then the strategy treats the points as the same, and the poison score is 0.

E. Evaluation Metrics

We evaluate performance based on three metrics:

- *Area under the ROC curve (AUC)* provides a threshold-independent measure of a model's ability to distinguish clean points from poisoned ones [18].
- *Precision* measures the percentage of samples identified as poisoned that were truly poisoned [19].
- *Recall* measures the percentage of truly poisoned samples that were identified by the model [19].

Since our strategies output onto $[0, 1]$, precision and recall require a threshold, which we selected by maximizing the F1 score. F1 score is the harmonic mean of recall and precision, so it balances the tradeoffs of false positives and negatives [19].

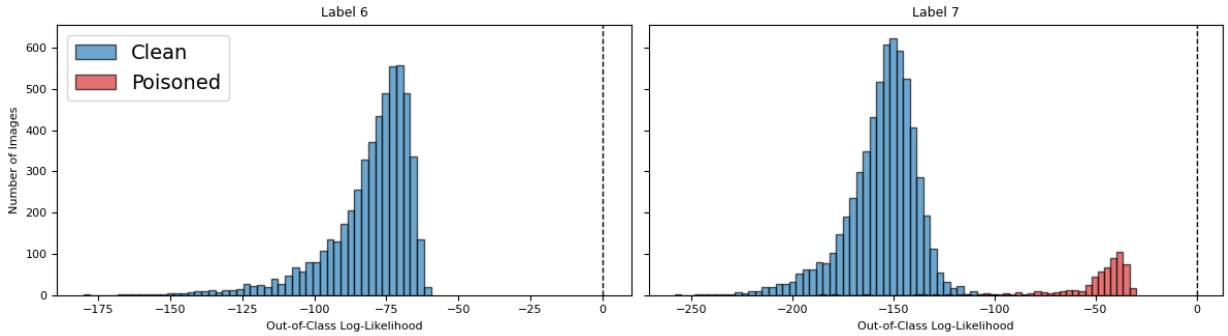

Fig. 2: This figure shows an example of a One-to-One attack where images labeled as "6" are clean and images labeled as "7" are poisoned, including images that are truly from class "6." More generally, for One-to-One attacks, the GMM strategy's log-likelihoods cluster into poisoned and clean distributions.

IV. Results

For each strategy, we aggregated results by poisoning percentage and by attack and evaluated the overall performance.

A. Strategy Performance by Poisoning Percentage

Figs. 3, 4, and 5 show the performance at different poisoning percentages, using AUC, recall, and precision, respectively. For calculating these curves, we first weighted them by attack type to equally represent each attack, since each attack has different amounts of test cases. Fig. 3 shows that k-means has an AUC close to one and that Wasserstein has an AUC of approximately 0.85, regardless of poisoning percentage. Whereas GMM's AUC increases as the poisoning percentage increases, Mahalanobis's AUC decreases. Fig. 4 shows that k-means has the highest recall, followed by GMM and Mahalobis, with Wasserstein having the lowest recall. Fig. 5 shows that GMM exhibits the best precision, followed by k-means, then Mahalanobis and Wasserstein.

B. Strategy Performance by Attack

We also analyzed AUC, recall, and precision for each strategy by finding the average over each attack type, presented in Table I. The strategies that exhibit the best performance for All-to-All and One-to-One attacks are Mahalanobis and GMM respectively. Wasserstein has low recall and precision compared to the other strategies, except on One-to-One attacks with a recall of 0.8224. Across all of Table I, the worst result was GMM's average AUC of 0.2193 for All-to-All attacks, whereas all other combinations of strategies and attacks yielded average AUCs above 0.7. Likewise, the best result in Table I was Mahalanobis's average AUC of 0.9990 on All-to-All attacks.

C. Strategy Performance Overall

As an overall performance score, we computed the arithmetic mean of the averages in Table I, as shown in the "Average" column for each metric. All strategies have an overall average AUC above 0.7. K-Means has the best overall average AUC of 0.9761 and recall of 0.9312. Wasserstein is the only strategy with a recall below 0.5. GMM has the best overall average precision of 0.8550, whereas Mahalanobis and Wasserstein have overall average precisions below 0.3.

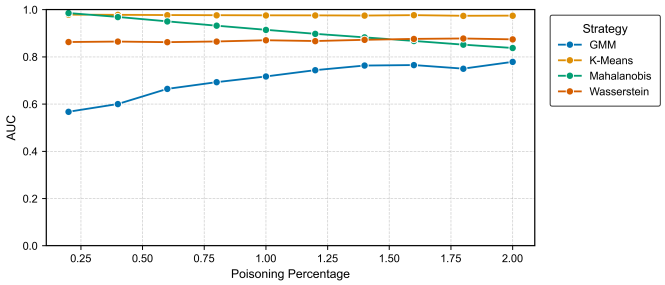

Fig. 3: Average AUC by poisoning percentage is shown above, equally weighing each attack type in this calculation. Mahalanobis's AUC decreases poisoning percentage increases, likely due a sensitive covariance matrix.

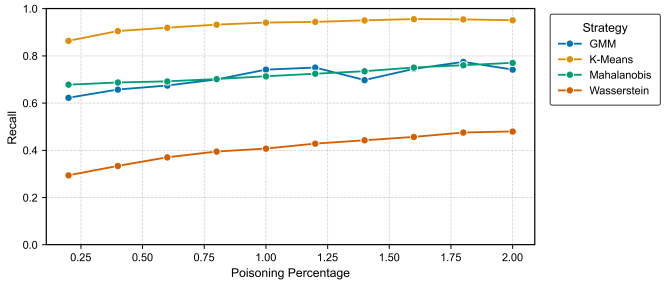

Fig. 4: Average recall by poisoning percentage is shown above, equally weighing each attack type in this calculation. K-Means achieves a recall close to one due to how it ignores labels and instead clusters by the true structure of the feature space, allowing it to be unbiased by poisoning.

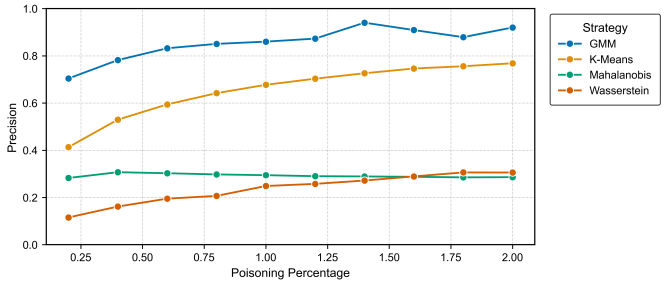

Fig. 5: Average precision by poisoning percentage is shown above, equally weighing each attack type in this calculation. GMM achieves a precision close to one because of how GMM fits to each class.

V. Discussion

A. Strengths and Weaknesses by Strategy

1) Gaussian Mixture Models: One key area of interest is GMM's low performance on All-to-All poisoning. With an

TABLE I: This table shows the performance of each strategy across attack types, reported as AUC, recall, and precision. The columns "A-A," "A-1," "1-A," and "1-1" represent All-to-All, All-to-One, One-to-All, and One-to-One, respectively. Further, the "Average" column is calculated as the arithmetic mean across these attacks in order to demonstrate each strategy's overall performance. Unlike Figs. 3, 4, and 5, this table aggregates performance by attack type, rather than by poisoning percentage. The highest values of each column are in bold. These bolded values show the best-performing strategy overall and for each attack type across each metric. For example, k-means has the best average AUC and recall of 0.9761 and 0.9312, respectively.

	AUC					**Recall**					**Precision**				
Strategy	**A-A**	**A-1**	**1-A**	**1-1**	**Average**	**A-A**	**A-1**	**1-A**	**1-1**	**Average**	**A-A**	**A-1**	**1-A**	**1-1**	**Average**
GMM	0.2193	0.8959	0.7445	**0.9568**	0.7041	0.3782	0.8374	0.6966	0.9293	0.7104	0.6179	**0.9495**	**0.9072**	**0.9455**	**0.8550**
K-Means	0.9875	**0.9881**	**0.9832**	0.9455	**0.9761**	0.8748	**0.9352**	**0.9192**	**0.9971**	**0.9312**	0.7709	0.6444	0.6624	0.5448	0.6556
Mahalanobis	**0.9990**	0.9662	0.9296	0.7401	0.9087	**0.8919**	0.7105	0.7763	0.5061	0.7212	**0.8298**	0.1873	0.1187	0.0332	0.2923
Wasserstein	0.8681	0.8731	0.8667	0.8687	0.8692	0.2195	0.2739	0.3167	0.8224	0.4081	0.1457	0.2002	0.3889	0.2083	0.2358

AUC of 0.2193, it would be better to use $1 - p(x)$. To investigate this, we looked at the maximum out-of-class log-likelihood for 7s when poisoned by an All-to-All attack rather than the One-to-One attack shown in Fig. 2. Comparing Fig. 2 and Fig. 6 reveals that the poisoned and clean distribution of data points overlap much more as a result of an All-to-All attack than a One-to-One attack on 7, with nearly the entire left half of the poisoned points distribution lying inside the clean points. This might have occurred because a GMM is fit to each class to predict how likely each image is in *another* class, rather than measuring its deviation from its true class. All-to-All attacks disrupt the likelihood of fitting into other classes aside from the most likely one, leading to more overlap in out-of-class log-likelihoods for poisoned and clean points.

Otherwise, GMM achieves the highest precision across other poisoning methods and percentages. Each metric is highest on One-to-One attacks and otherwise improves with higher poisoning percentages, indicating GMM is well suited to detecting concentrated poisonings on a single class. This method could be improved by using more than two Gaussian components to better fit the data, aggregating the likelihoods of being in other classes rather than just taking the maximum, or incorporating the likelihood of fitting into its own class as well as into other classes. In addition, better identification of poisoned points after getting the out-of-class log-likelihood distributions would improve its performance on All-to-All poisoning attacks.

2) K-Means Clustering: K-Means clustering is our most consistent detection method as it performs nearly identically across poisoning methods and proportions. This is likely due to its unsupervised technique [20] which ignores poisoned point's labels when creating the clusters, meaning the clusters are not biased by poisoned points. However, k-means clustering struggles with precision because some classes have subsets of images that are different from the rest of the class, meaning that subset might get assigned to a different cluster. A potential solution to this would be increasing the number of centroids to capture the specific subsets of each class.

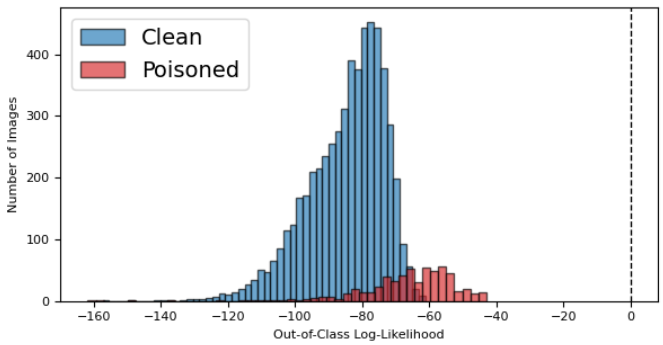

Fig. 6: The distribution of the log-likelihood of being in another class for the class of "7" when poisoned using an All-to-All poisoning, with poisoned and clean points labeled by color. In contrast to Fig. 2, the poisoned class is less distinct from the clean labels, increasing flagging difficulty.

3) Mahalanobis: Mahalanobis performs better on All-to-All poisoning techniques than One-to-One for all metrics, and its AUC and precision decrease as the poisoning percentage increases. These indicate lower effectiveness with more concentrated poisoning in a single class. This behavior is likely due to the sensitivity of the covariance matrix to outliers [21]. Thus, Mahalanobis performs best when poisoning percentages within each class are low and could be improved using bootstrapping or robust computation methods [22].

4) Wasserstein Distance: Wasserstein Distance likely only has strong recall with One-to-One attacks because of how each poisoning method affects the creation of the "representative images" that the strategy uses to detect poisoned images. In a One-to-One attack, images that initially belonged to one class become outliers in another class if their label is poisoned. Because One-to-One attacks only involve two classes of images, only two of the representative images are affected by the sudden change in their classes' images. In contrast, the other poisoning methods transfer images from either a single class to all other classes (One-to-All), images from all other classes to a single class (All-to-One), or images from every class to every other class (All-to-All). Regardless of poisoning methods used, all representative images are affected by the change in classes. Thus, the Wasserstein Distance detection strategy will perform better against poisoning methods that affect a small number of classes.

B. Ethical Considerations

As with training image recognition models, detecting poisoned images comes with many ethical considerations. For example, AI companies train image recognition models on images uploaded to the Internet by artists without their consent [23]. In the past, artists have used tools like Nightshade and Glaze to purposefully poison their images and reduce the accuracy of corporate image recognition models to deter companies from violating copyright [23]. However, if projects like CleanSight advance the field of data poisoning detection, then companies would have the freedom to train their models on online art without consent from the artists. At the same

time, CleanSight strengthens the identification of artists' digital signatures in their work.

C. Future Work

A key limitation of our work is that we do not test more complex datasets, which limits the generalizability of CleanSight. While our pipeline demonstrates effective detection on MNIST, this is likely impacted by the dataset's simple input space and high class separability. As a result, it is unclear how well these methods would perform on more complex image domains. Future work should prioritize validating this approach accordingly, which could involve developing more effective feature engineering techniques.

Further, while our detection strategies effectively flag mislabeled data, we do not examine their impacts on model training. Thus, future work should include end-to-end evaluation to demonstrate more thoroughly the feasibility and practicality of using these methods to clean data before training time. This could involve exploring how removing or correcting flagged images improves model performance.

Additionally, our methods' resulting "poison scores" are not calibrated probabilities, so they are not directly comparable. Future work could explore probability calibration to improve the consistency between and practical meaning of these scores.

VI. Conclusion

It is necessary to ensure reliable training data, especially with threats like label-flipping. Thus, we investigated four strategies to identify mislabeled data in the MNIST image dataset, using GMMs, k-means clustering, Mahalanobis distance, and Wasserstein distance, respectively. From a testing pipeline we developed, we found that k-means had an overall average AUC of 0.9761 and recall of 0.9312. GMM had the best overall average precision of 0.8550.

This preliminary data security investigation examines more comprehensive yet practical pre-training defenses, demonstrating its effectiveness on MNIST. Future work would seek to improve these methods by exploring their generalization onto different datasets, calibrating their outputs, and validating results with model training.

Acknowledgment

We thank our professors, with special thanks to our advisor, Hunter Moore, and the Systems Engineering faculty.

References

[1] "Hardshell - Securing the foundation of AI." [Online]. Available: https://www.hardshell.ai/

[2] Y. Chen, X. Zhu, X. Gong, X. Yi, and S. Li, "Data poisoning attacks in internet-of-vehicle networks: Taxonomy, state-of-the-art, and future directions," *IEEE Xplore*, vol. 19, no. 1, Jan. 2023.

[3] A. Souly, J. Rando, E. Chapman, X. Davies, B. Hasircioglu, E. Shereen, C. Mougan, V. Mavroudis, E. Jones, C. Hicks, N. Carlini, Y. Gal, and R. Kirk, "Poisoning Attacks on LLMs Require a Near-constant Number of Poison Samples," Oct. 2025, arXiv:2510.07192 [cs]. [Online]. Available: http://arxiv.org/abs/2510.07192

[4] H. Abroshan, "AI to protect AI: A modular pipeline for detecting label-flipping poisoning attacks," *Machine Learning with Applications*, vol. 22, p. 100768, Dec. 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666827025001513

[5] E. Rosenfeld, E. Winston, P. Ravikumar, and Z. Kolter, "Certified Robustness to Label-Flipping Attacks via Randomized Smoothing," *Proceedings of the 37th International Conference on Machine Learning*, vol. 119, pp. 8230–8241, Nov. 2020. [Online]. Available: https://proceedings.mlr.press/v119/rosenfeld20b/rosenfeld20b.pdf

[6] Y. R. Maramreddy and K. Muppavaram, "Detecting and Mitigating Data Poisoning Attacks in Machine Learning: A Weighted Average Approach," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 15 505–15 509, Aug. 2024. [Online]. Available: https://www.etasr.com/index.php/ETASR/article/view/7591

[7] A. Paudice, L. Muñoz-González, and E. C. Lupu, "Label Sanitization Against Label Flipping Poisoning Attacks," in *ECML PKDD 2018 Workshops*, C. Alzate, A. Monreale, H. Assem, A. Bifet, T. S. Buda, B. Caglayan, B. Drury, E. García-Martín, R. Gavaldà, I. Koprinska, S. Kramer, N. Lavesson, M. Madden, I. Molloy, M.-I. Nicolae, and M. Sinn, Eds. Cham: Springer International Publishing, 2019, pp. 5–15.

[8] A. Paudice, L. Muñoz-González, A. Gyorgy, and E. C. Lupu, "Detection of Adversarial Training Examples in Poisoning Attacks through Anomaly Detection," 2018, _eprint: 1802.03041. [Online]. Available: https://arxiv.org/abs/1802.03041

[9] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[11] PyTorch, https://docs.pytorch.org/vision/0.10/models.html.

[12] "resnet18 — Torchvision main documentation." [Online]. Available: https://docs.pytorch.org/vision/main/models/generated/torchvision.models.resnet18.html

[13] D. A. Reynolds and others, "Gaussian mixture models." *Encyclopedia of biometrics*, vol. 741, no. 659-663, p. 3, 2009.

[14] F. Pendregosa, G. Varoquax, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1056489

[16] P. C. Mahalanobis, "On the Generalized Distance in Statistics," *Sankhyā: The Indian Journal of Statistics, Series A (2008-)*, vol. 80, pp. S1–S7, 2018. [Online]. Available: https://www.jstor.org/stable/48723335

[17] L. Vaserstein, "Markov Processes over Denumerable Products of Spaces, Describing Large Systems of Automata," *Problemy Peredachi Informatsii*, vol. 5, no. 3, pp. 64–72, 1969. [Online]. Available: https://www.mathnet.ru/links/8d3b01e2837678a3fa680c2df6763ca9/ppi1811.pdf

[18] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016786550500303X

[19] K. Sujon, R. Hassan, K. Choi, and M. Samad, "Accuracy, precision, recall, f1-score, or MCC? empirical evidence from advanced statistics, ML, and XAI for evaluating business predictive models," *Journal of Big Data*, vol. 12, Dec. 2025. [Online]. Available: https://link.springer.com/article/10.1186/s40537-025-01313-4

[20] K. Sinaga and M.-S. Yang, "Unsupervised K-Means Clustering Algorithm," *IEEE Access*, vol. PP, pp. 1–1, Apr. 2020.

[21] C. Leys, O. Klein, Y. Dominicy, and C. Ley, "Detecting multivariate outliers: Use a robust variant of the Mahalanobis distance," *Journal of Experimental Social Psychology*, vol. 74, pp. 150–156, Jan. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022103117302123

[22] M. Molnar and B. Nikolic, "Multivariate Outlier Detection Using Robust Mahalanobis Distances," 2022.

[23] M. Heikkilä, "This new data poisoning tool lets artists fight back against generative AI," 2023. [Online]. Available: https://www.technologyreview.com/2023/10/23/1082189/data-poisoning-artists-fight-generative-ai/