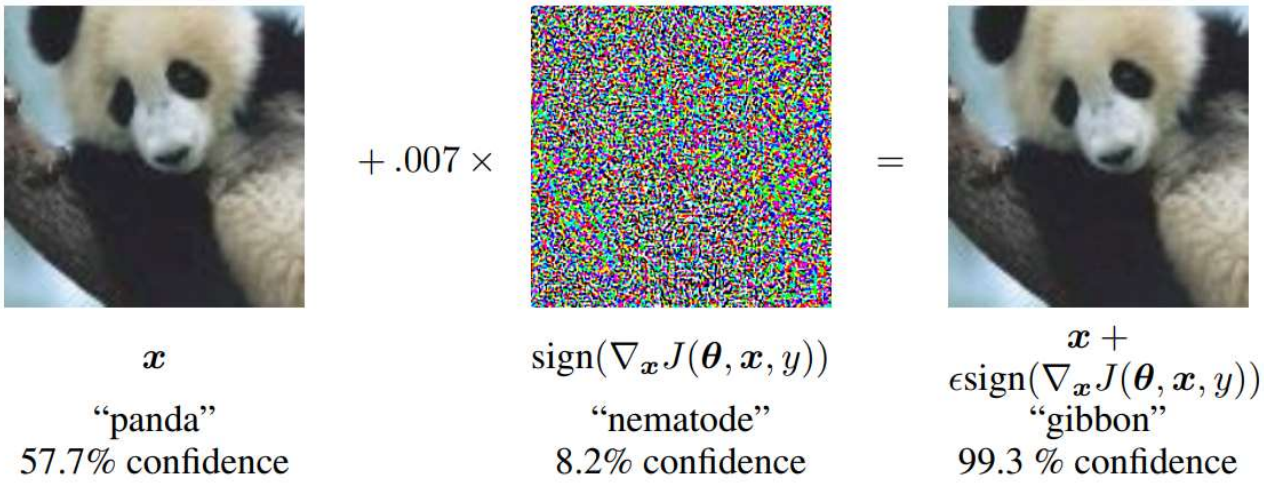
Investigating the Effects of Training Techniques on Classification Robustness

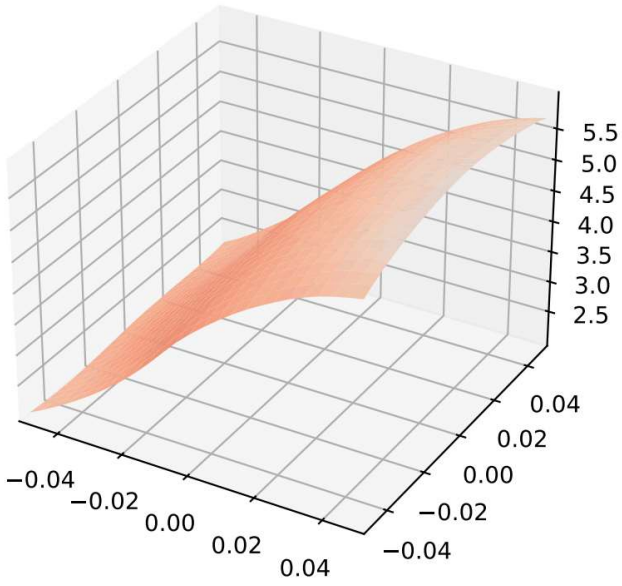
Introduction/Background

Increasing the accuracy benchmarks of classification models has always been one of the main focuses in the progress of machine learning. Recently, increasing the adversarial robustness has also become a critical goal in machine learning.

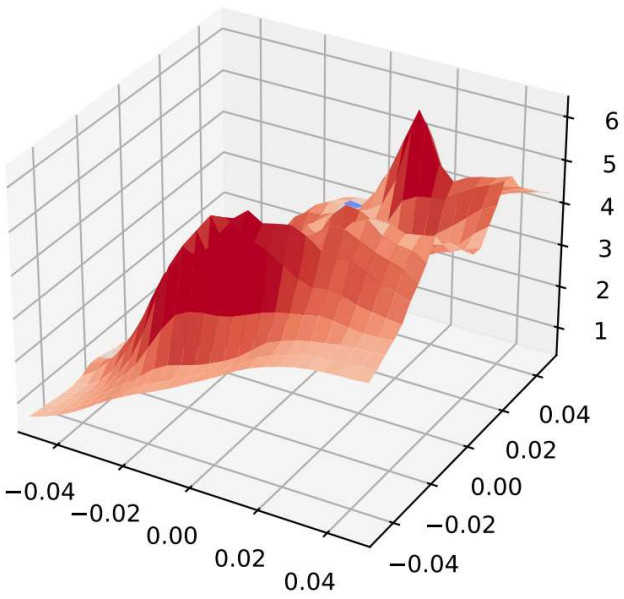
Robustness allows models to defend against adversarial attacks 1—carefully modified inputs designed to mislead classification models. On a more theoretical note, achieving robustness typically implies creating models with the desirable traits of smoother decision boundaries or loss landscapes 2. In this study we explore the effects of supervised and (especially) unsupervised training methods, such as clustering, on adversarial robustness using image classification tasks.



An example of a panda being classified incorrectly after small perturbations are added [1]



A robust model with a smooth loss landscape



A non-robust model with a rugged loss landscape

Datasets

The datasets we will work on are as follows.

- *MNIST:* Handwritten digits; Training set of 60,000 examples
- **CIFAR-10:** 10 classes with 6000 images per class.
- **Dataset Links:**
  - MNIST
  - CIFAR-10

Problem Definition

Motivation

Most research on adversarial robustness has focused on supervised learning methods. While adversarial training on supervised training has shown promising results, there is limited understanding of how unsupervised training methods impact model robustness.

Problem

We hope to understand the relationship between unsupervised methods and robustness compared to supervised methods. Specifically:

1. Investigate how unsupervised learning methods compare to supervised methods in accuracy and robustness.
2. Explain the differences or similarities from 1.
3. Experiment on adversarial training techniques applied originally on supervised algorithms and modify them to work on unsupervised algorithms.

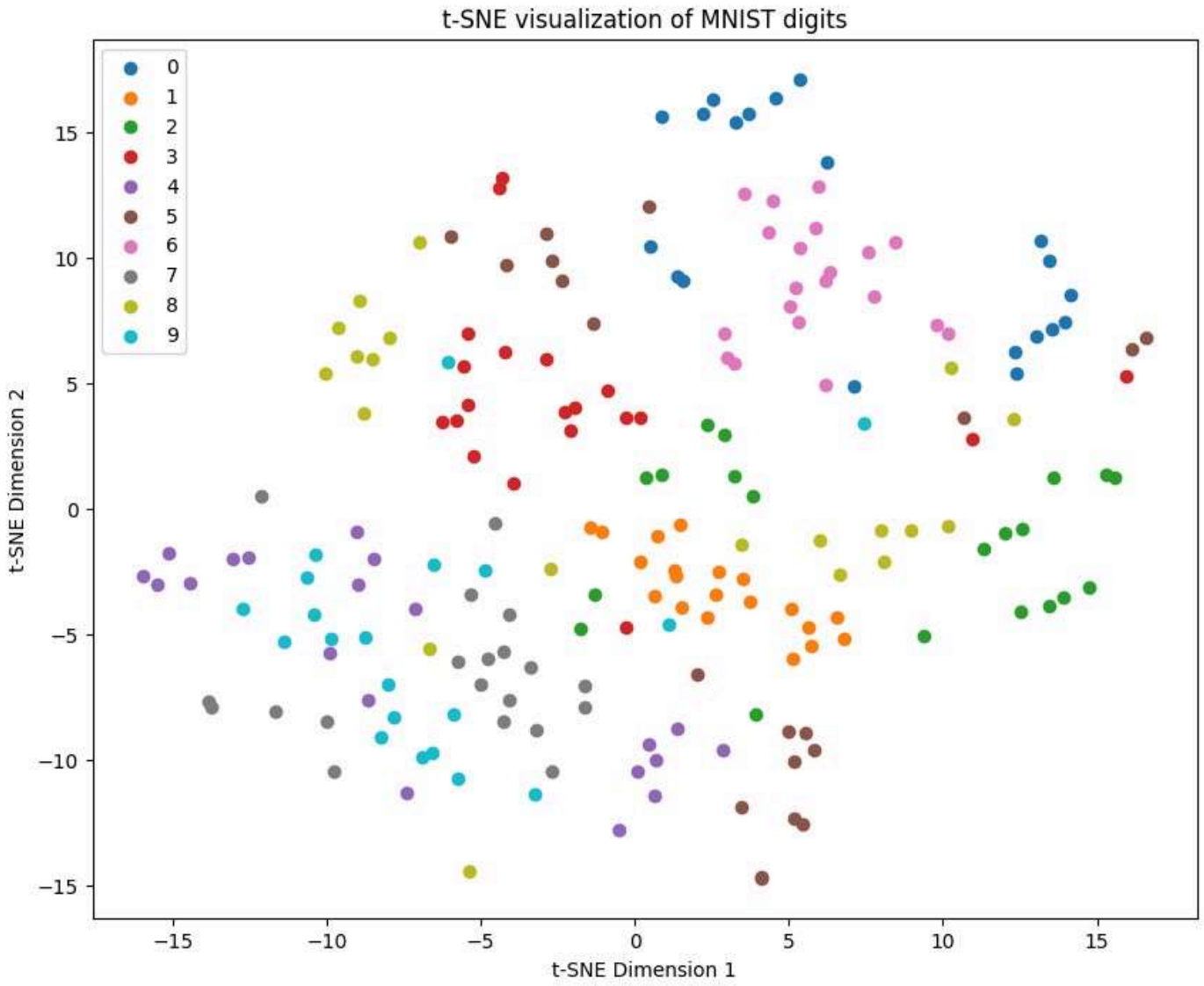
Methods

Exploratory Data Analysis

Before performing our data pre-processing steps, we performed a systematic and rigorous Exploratory Data Analysis (EDA) process on the MNIST and CIFAR-10 datasets. This enabled us to gather a deeper understanding of the underlying data, its patterns and characteristics which then motivated our data pre-processing procedures. Following were our efforts during the EDA phase. Note that the images for EDA are present in the `mlproject_EDA` Jupyter Notebook in the `preprocessing_code_directory` as they are quite involved and extensive.

- Visualizing randomly chosen datapoints initially to get an intuitive, qualitative feel for the data layout, size, quality.
- Measuring the class distribution in the training set, to determine whether class imbalances exist that could bias the results.
- Performing dimensionality reduction via PCA and visualizing the resultant Scree plots to understand the variance explained by each principal component, allowing us to identify the most informative features.
- Anomaly/Outlier detection.
- [CIFAR-10] Splitting images into their RGB color channels to observe the information each channel provides.
- [CIFAR-10] Observing variances between color channels.
- [CIFAR-10] Performing dimensionality reduction to visualize the clusters using t-SNE. It captures non-linear relationships and complex patterns in data as compared to PCA.

Following is a visualization of t-SNE applied to the CIFAR-10 dataset (first two components), with 20 samples per class. Notice that datapoints of the same class are clustered close to one another. Also notice that numbers that have a similar hand-writing to each other have their clusters with a significant level of overlap (4 and 9 is perhaps the clearest example of this).

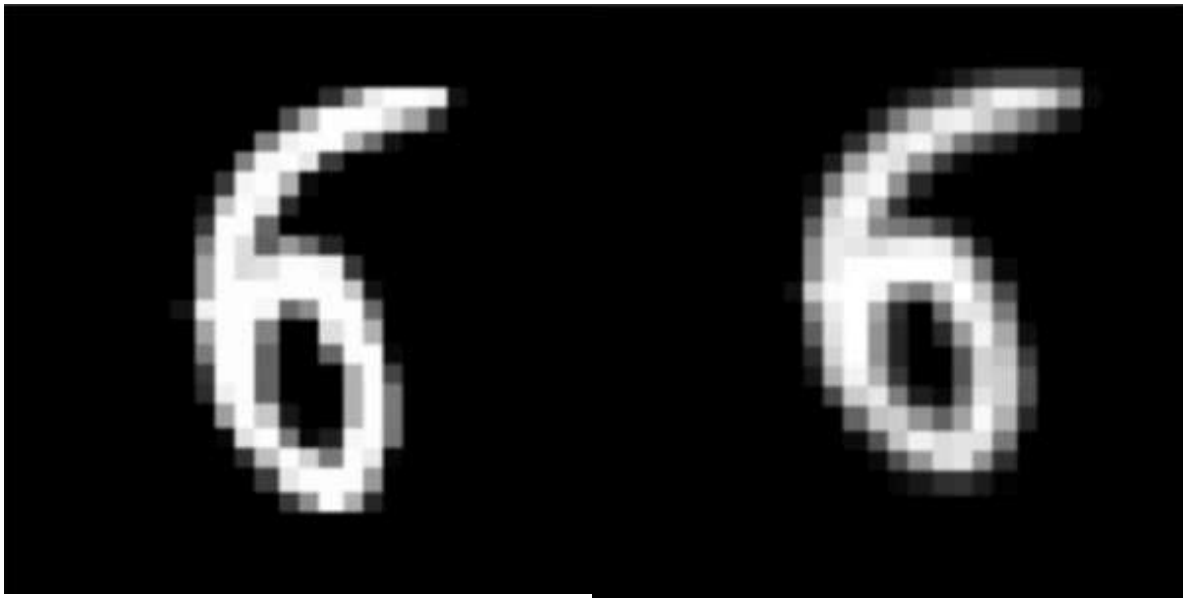


t-SNE dimensionality reduction applied to CIFAR-10

Data Preprocessing

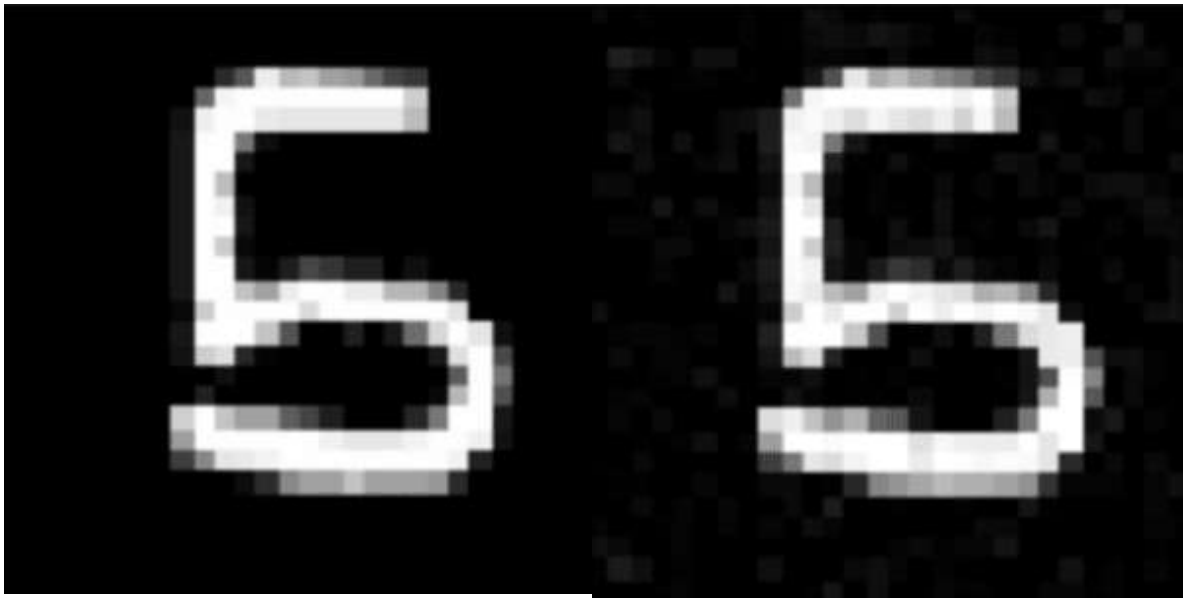
We used three preprocessing techniques on our training data in an attempt to improve the resultant clean accuracies and robustness:

- **Dataset Class Balancing:** We were surprised to learn that the training data for MNIST had a relatively uneven distribution of classes, with the smallest class (5) having only about 80 percent as many samples as the largest (1). To provide more training data for underrepresented classes, we generated additional artifical samples by applying image translations and image scaling (to reduce the chance of overfitting) to samples from each class. Samples were created such that all classes in the resulting dataset had an equal number of samples.



Original New Synthetic Sample

- **Image Augmentation:** We applied a small random crop to to the training data as well as random horizontal flips in attempt to reduce overfitting and increase our model's invariance to the spatial location of the digits/objects in the images. Horizontal flips were not applied to data for MNIST since flipped digits are not accurate samples for most numbers.
- **Gaussian Noise:** We added random gaussian noise to training images in an effort to increase robustness and reduce the likelihood of the model overfitting to small details.



Original Gaussian Noise Added

Machine Learning Techniques

We intend to apply and evaluate the following machine learning methods:

- **Baseline Supervised Learning:** Perform classical supervised training on a convolutional neural network (CNN) using batch gradient descent.
- **Adversarial Supervised Training:** Perform adversarial training using PGD-10 (Projected Gradient Descent) [3] as an inner maximization step to improve robustness.
- **Baseline Unsupervised Learning:** Use Gaussian Mixture Model as a baseline model; adversarial techniques will be applied if adaptable.

This follows from our project proposal. For our midterm checkpoint, we have implemented and evaluated both our supervised learning techniques - the baseline classical training, and adversarial training. Their implementations will be described below. For the final checkpoint, we have implemented and evaluated the baseline unsupervised learning technique. Additionally, we have successfully developed an algorithm analogous to adversarial training to improve adversarial accuracy on GMMs.

Supervised Learning: Baseline

Here, we will describe the training configuration we used for both of our supervised training techniques. Our baseline supervised learning method follows the below exactly, while our adversarial training method follows the below settings but also has its own interesting properties which will be described in the next section.

- **Datasets:** We performed experiments for both the MNIST and CIFAR-10 datasets.
- **ML Model Architecture:** We trained a convolutional neural network (CNN) with three convolutional layers, each of which is succeeded by a ReLU and Max-Pooling operation, finally concluded with two fully-connected layers. Our choice to use a CNN was based on its suitability and high performance on image data as compared to a regular multi-layered perceptron based on our exploratory experiment iterations. While our initial plan during the proposal was to use the `ResNet18` architecture, a simpler CNN performed satisfactory given the datasets we considered. It provided a better balance between model performance and training feasibility.
- **Training - Validation - Test split:** We split our data in the form of 60,000 - 8,000 - 2,000 for the respective sets.
- **Common hyper-parameters:** For MNIST and CIFAR-10, we performed 10 and 50 training epochs respectively (the RGB CIFAR-10 dataset is significantly more complex than the grayscale handwritten digits of MNIST). We set a learning rate of 0.1, momentum of 0.9 and weight decay 0.0005. For CIFAR-10, we also introduce a learning rate decay (gamma = 0.1) at epochs 25 and 40 to improve convergence and avoid overshooting minima.
- **Pocket learning criterion:** During training, we save the model with the highest validation set accuracy to use during test-time evaluation.

Supervised Learning: Adversarial Training with PGD for Robustness

We leverage the same model architecture, training set-up and hyperparameters as the baseline to facilitate reliable comparison. In addition, we set up the following properties of adversarial training:

- Inner Maximization: A Projected Gradient Descent (PGD) attack is performed on the training samples before fed to the Batch Gradient Descent training algorithm. The parameters of the PGD attacker L-infinity norm perturbation bound epsilon, perturbation step size alpha, and step count are as follows:

MNIST:

$$\epsilon = \frac{16}{255}, \quad \alpha = \frac{2}{255}, \quad \text{steps} = 20$$

CIFAR-10:

$$\epsilon = \frac{8}{255}, \quad \alpha = \frac{2}{255}, \quad \text{steps} = 10$$

- **Validation:** PGD accuracy (the accuracy of the model on a PGD-attacked dataset) is also monitored.
- **Pocket Learning Criterion:** The model with the highest PGD validation accuracy is saved for test-time evaluation.

The following formally depicts an individual 'step' of the Projected Gradient Descent attack

$$x^{t+1} = \text{Proj}_{x+\mathcal{S}}\left(x^t + \alpha\,\text{sgn}(\nabla_{x^t} L(\theta, x^t, y))\right)$$

Projected Gradient Descent[3] formula.

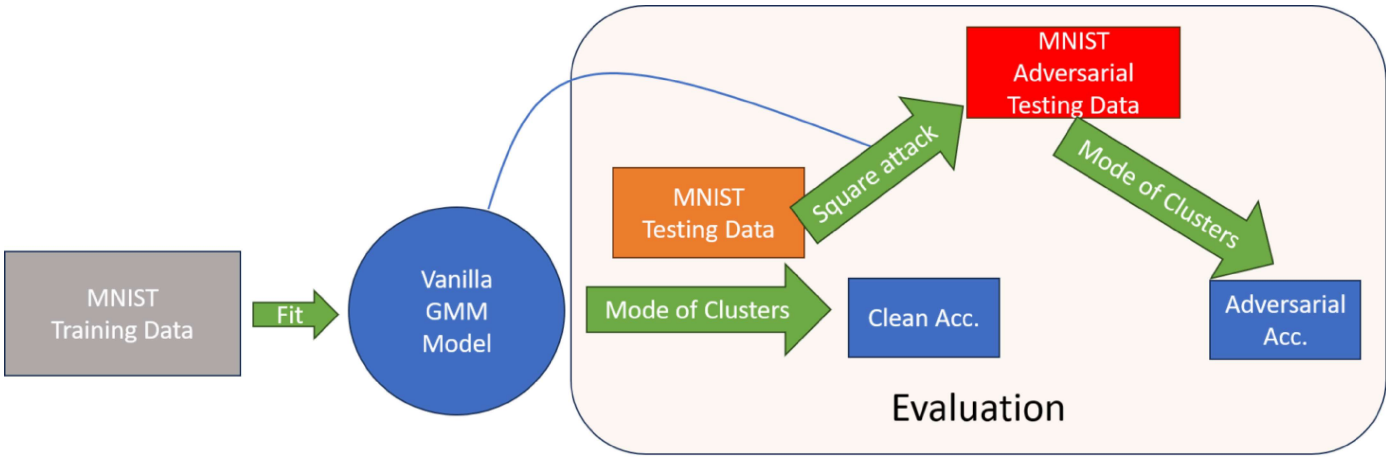
Here, the datapoint is perturbed by taking a step-size alpha in the direction of the gradient of the loss with respect to the input point - maximising the loss - and finally projecting it back onto the epsilon bound over the original input.

Unsupervised Learning: Baseline

Next, we will describe the training configuration and decisions we used for unsupervised training techniques: A baseline classically-trained Gaussian Mixture Model (GMM) which follows the below set-up exactly, as well as a novel adversarial training technique applied to a GMM, which also follows the below but has its own interesting properties which will be elaborated on in the next section.

- **Datasets:** Due to several time consuming steps when our training method is added, we performed experiments on only the MNIST dataset.
- **Model Architecture:** GMM with ten components. When predicting the class of a set of images, we cluster the set, then pick the mode label of each cluster as the predicted class.
- **Training - Test split:** We split our data in the form of 60,000 - 10,000 for the respective sets.
- **Dimensionality reduction:** Before any other aspects regarding preprocessing steps or fitting were analyzed, we first experimented on two dimensionality reducing techniques, namely PCA and UMAP. Then, considering both run time and clean accuracy when paired with the baseline GMM model, we chose the appropriate reduction technique along with target dimension count for all following experiments. Our rationale behind adopting dimensionality reduction for unsupervised learning was two-fold: Firstly, GMM training time complexity is roughly cubic to dimensionality. By theory and experimentation, we determined it to be too time consuming to train with unreduced images. Secondly, we found that lower-dimensional representation can enhance GMMs' ability to fit Gaussian components by isolating clusters.

The following figure depicts the pipeline for training and evaluation for the baseline unsupervised learning (GMM) technique.



Baseline GMM Training-Evaluation Pipeline.

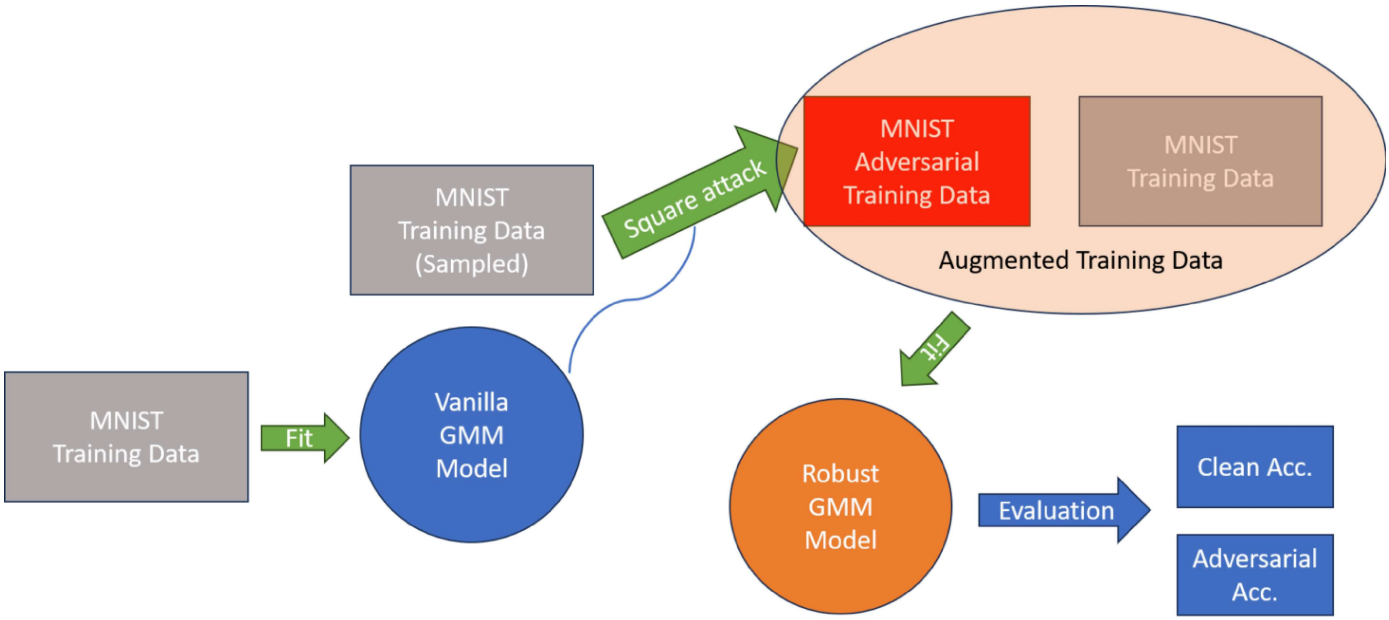
Unsupervised Learning: Adversarial Training with Square Attack for Robustness

The dataset, model architecture, train-test split and dimensionality reduction techniques follow from the baseline unsupervised learning approach highlighted above to facilitate reliable comparison. In addition, the following novel adaptations were made to the training approach to enhance model robustness to adversarial perturbations:

- Inspired by the PGD inner-maximization in classical adversarial training for neural networks, we performed a three-step training algorithm to get a robust GMM.

1. Train a normal GMM on the training set.
2. Run square attack on a randomly selected fifth of the training set, adding these perturbed images to the training set.
3. Train another GMM model on this augmented training set.

The following image depicts the training-evaluation pipeline for our novel adversarial training approach for the unsupervised learning (GMM) setting.



Robust GMM Training-Evaluation Pipeline.

Results and Discussion

Evaluation Methodology

To better understand the effects of preprocessing steps and the differences between baseline and adversarial training, we evaluate models trained with and without each preprocessing method (class balancing, image augmentation, and Gaussian noise) for both training algorithms.

Quantitative Scoring Metrics: For each model saved from the training pipeline, the following metrics are evaluated:

- **Clean Accuracy:** The classification accuracy of the model on the original (unperturbed) test images. This measure gives us an indication of how well the model has learned from its training data and generalizes to unseen data.
- **Robust Accuracy (Square Attack):** The classification accuracy of the model on test images perturbed (with epsilon as perturbation bound) by the black-box Square Attack. Black-box in the context of

adversarial machine learning refers to attacks that do not leverage internal properties/parameters of the model, essentially treating it as a black-box and learning about it via repeated querying. Measuring robust accuracy gives us an indication of how resilient a model is to adversarial perturbations crafted to attempt to fool the model, with a higher measure indicating the model is more robust. Our choice of a black-box attack here is crucial: Since it does not exploit the model's parameters or gradients, it cannot be weakened by networks intentionally performing gradient obfuscation techniques. A network obfuscating its gradients may fool a white-box attack that makes use of model gradients and depict an illusion of robustness while in reality it may not be as robust. The next section will demonstrate an example of the Square Attack in action.

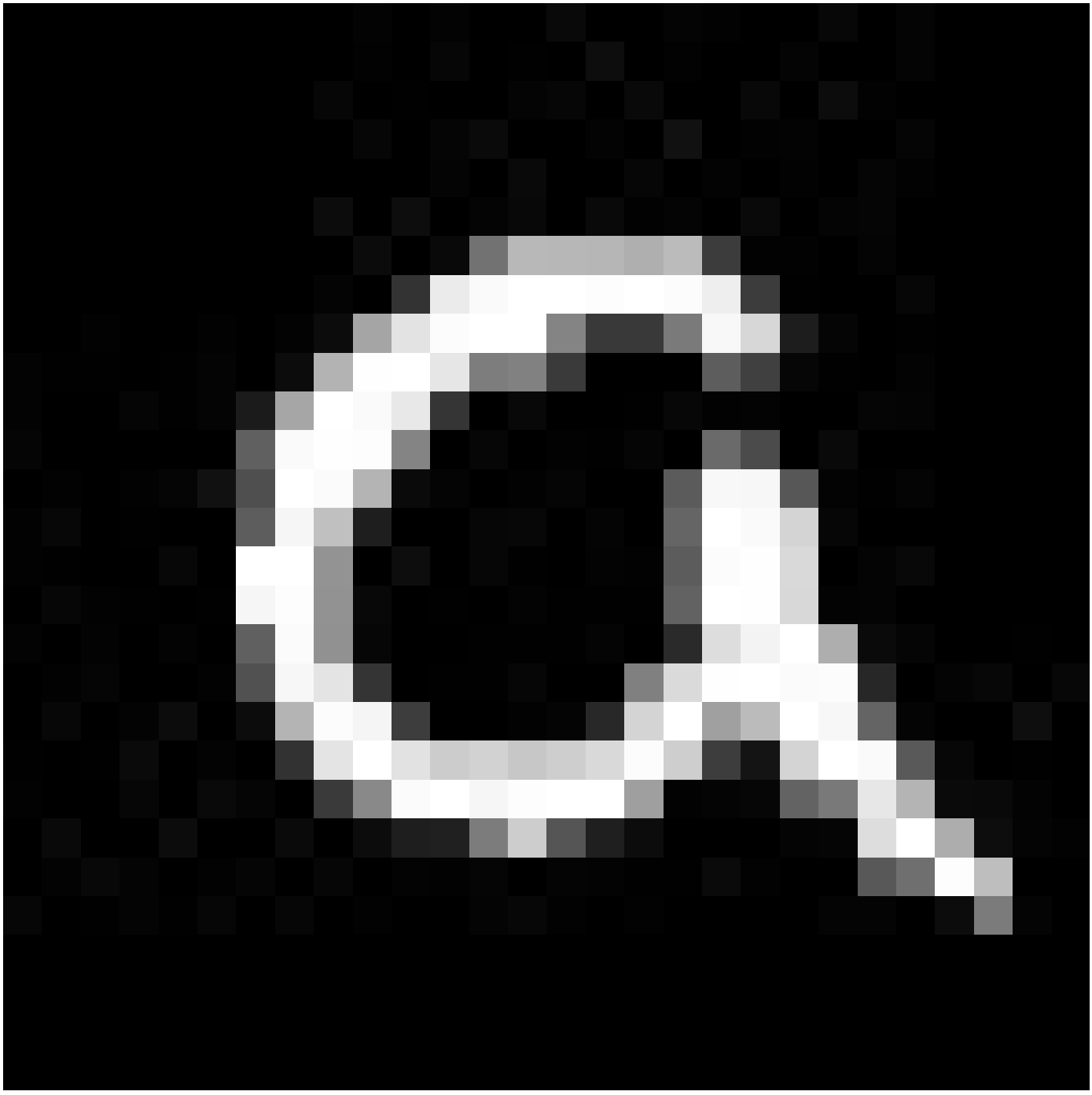
Result Visualization: We have also extracted confusion matrices for the robust accuracy evaluation of classical supervised training and adversarial training with MNIST (keeping the pre-processing configuration the same)y. A confusion matrix provides a detailed breakdown of the model's performance across classes, helping us analyse performance per class, identify error patterns and inter-class confusions by the model.

Square Attack Parameters and Example Perturbation

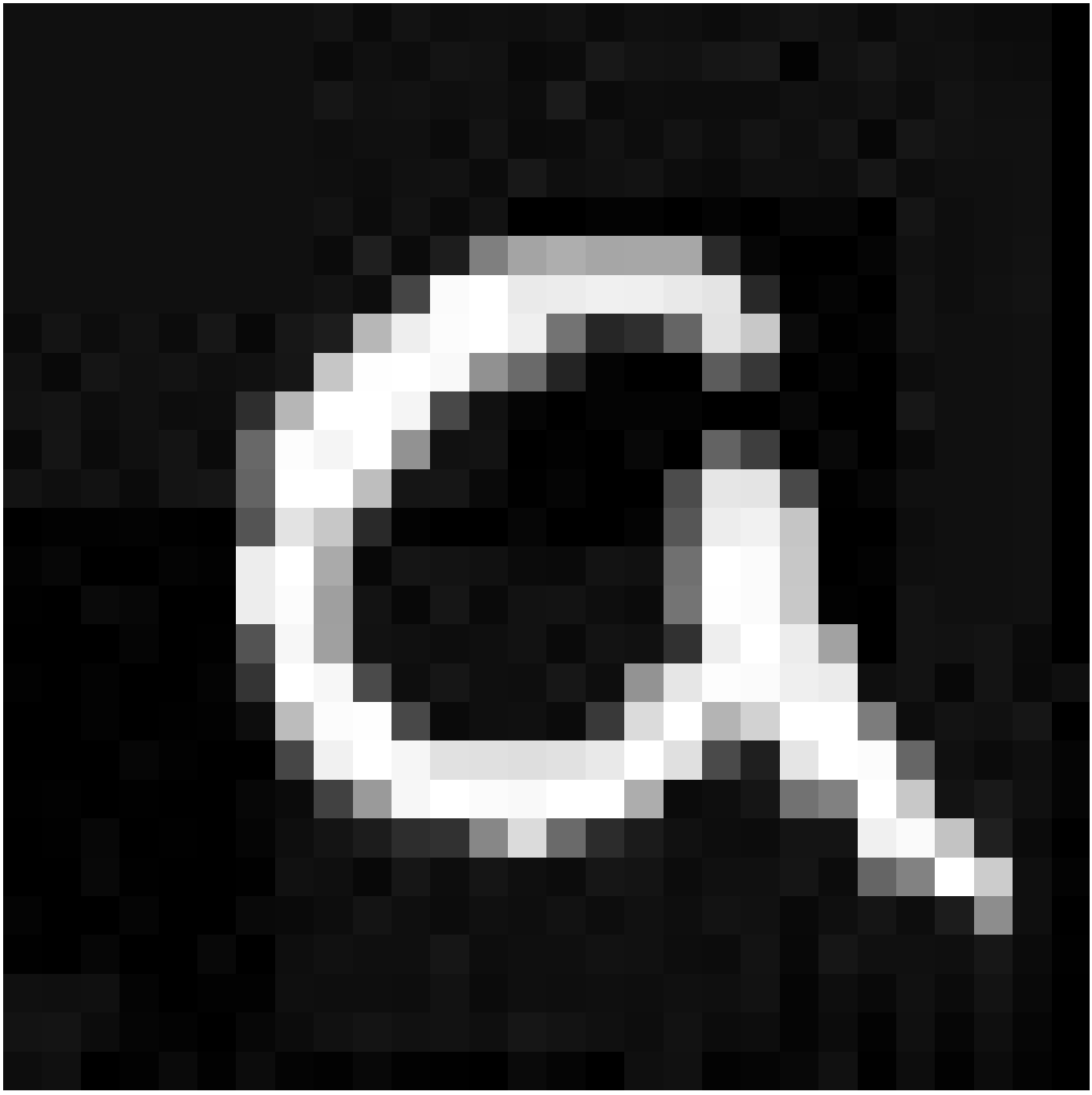
$$\text{MNIST:} \quad \epsilon = \frac{16}{255}, \quad \text{random restarts} = 1, \quad \text{number of queries} = 5000$$

$$\text{CIFAR-10:} \quad \epsilon = \frac{8}{255}, \quad \text{random restarts} = 1, \quad \text{number of queries} = 5000$$

Example Perturbation: We also display an example of a perturbed image versus the original in a successful Square Attack instance during test-time evaluation. The original image of class **9** is perturbed to be classified as class **2**. Although not extremely obvious, a faint outline of a "2-like" shape can be seen in the background.



Original Image, Class 9



Perturbed Image, Classified as 2

Evaluation Results:

Note: We do not perform data balancing on CIFAR-10, as the dataset itself is already balanced by design.

Supervised Learning Quantitative Evaluation Results

Following are clean and robust accuracy scores for all our pre-processing configurations across CNNs for MNIST and CIFAR-10, and across both our supervised learning techniques - classical Batch GD training and

adversarial training. The configuration labels (e.g., `FFF`, `TTT`) indicate whether the following preprocessing steps were applied:

- The first character: **Balancing** (`T` = Applied, `F` = Not Applied)
- The second character: **Augmentation** (`T` = Applied, `F` = Not Applied)
- The third character: **Gaussian Noise** (`T` = Applied, `F` = Not Applied)

MNIST Standard Training Results:

	Preprocessing Config	Clean Accuracy	Square Accuracy
0	FFF	98.90%	91.65%
1	FFT	98.75%	91.90%
2	FTF	99.20%	89.45%
3	FTT	98.75%	91.05%
4	TFF	99.05%	90.40%
5	TFT	98.75%	86.40%
6	TTF	98.75%	88.90%
7	TTT	98.70%	79.20%

MNIST PGD Adversarial Training Results:

	Preprocessing Config	Clean Accuracy	Square Accuracy
0	FFF	98.90%	97.55%
1	FFT	99.10%	97.60%
2	FTF	98.65%	96.40%
3	FTT	98.90%	96.70%
4	TFF	99.25%	97.80%
5	TFT	98.95%	95.85%
6	TTF	99.05%	97.85%
7	TTT	99.15%	95.60%

CIFAR-10 Standard Training Results:

	Preprocessing Config	Clean Accuracy	Square Accuracy
0	FF	72.40%	0.20%
1	FT	71.45%	0.80%
2	TF	79.25%	0.90%
3	TT	76.65%	0.95%

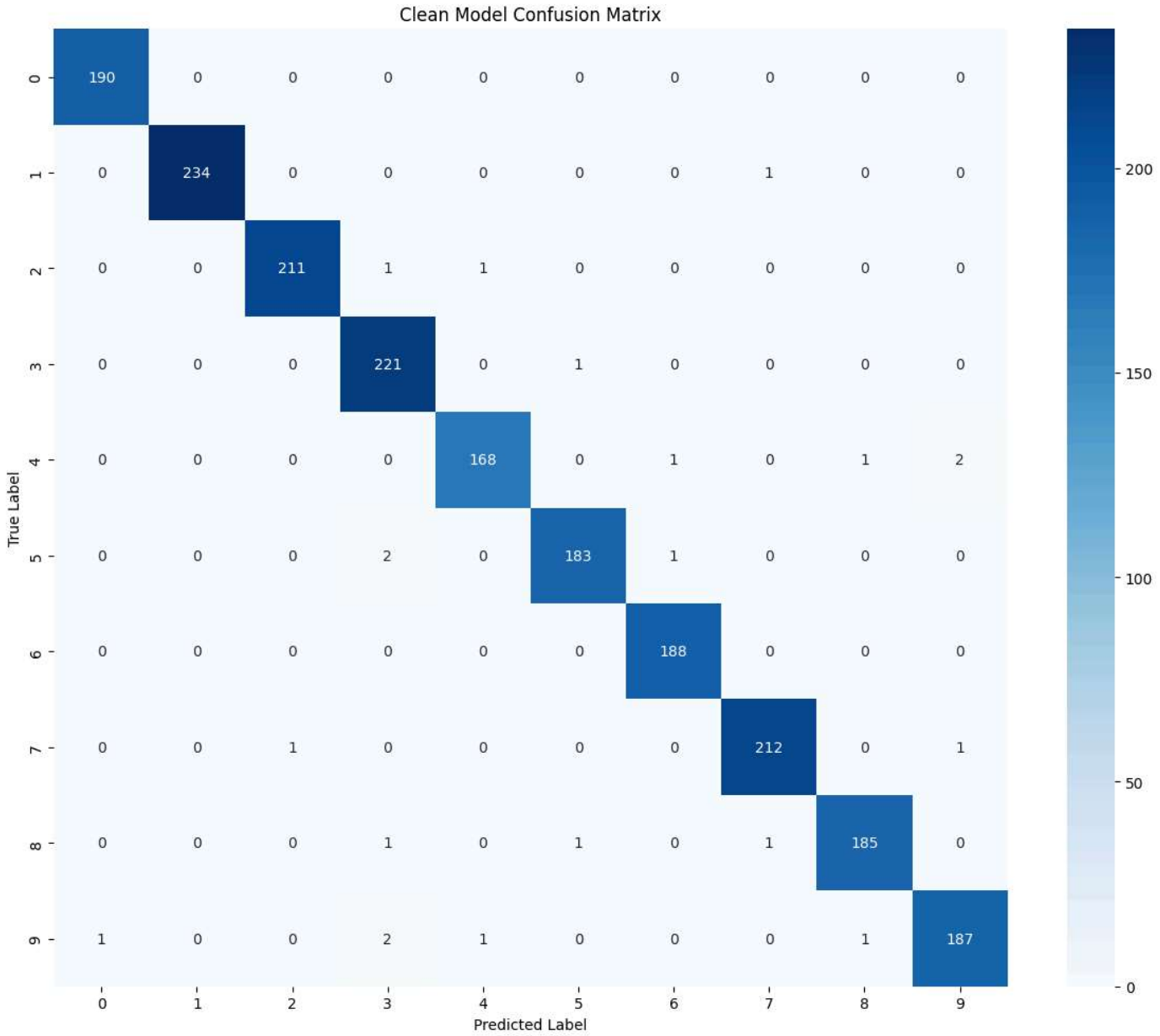
CIFAR-10 PGD Adversarial Training Results:

	Preprocessing Config	Clean Accuracy	Square Accuracy
0	FF	52.50%	28.85%
1	FT	52.35%	28.50%
2	TF	47.20%	27.25%
3	TT	46.75%	26.30%

Visualization: Confusion Matrices

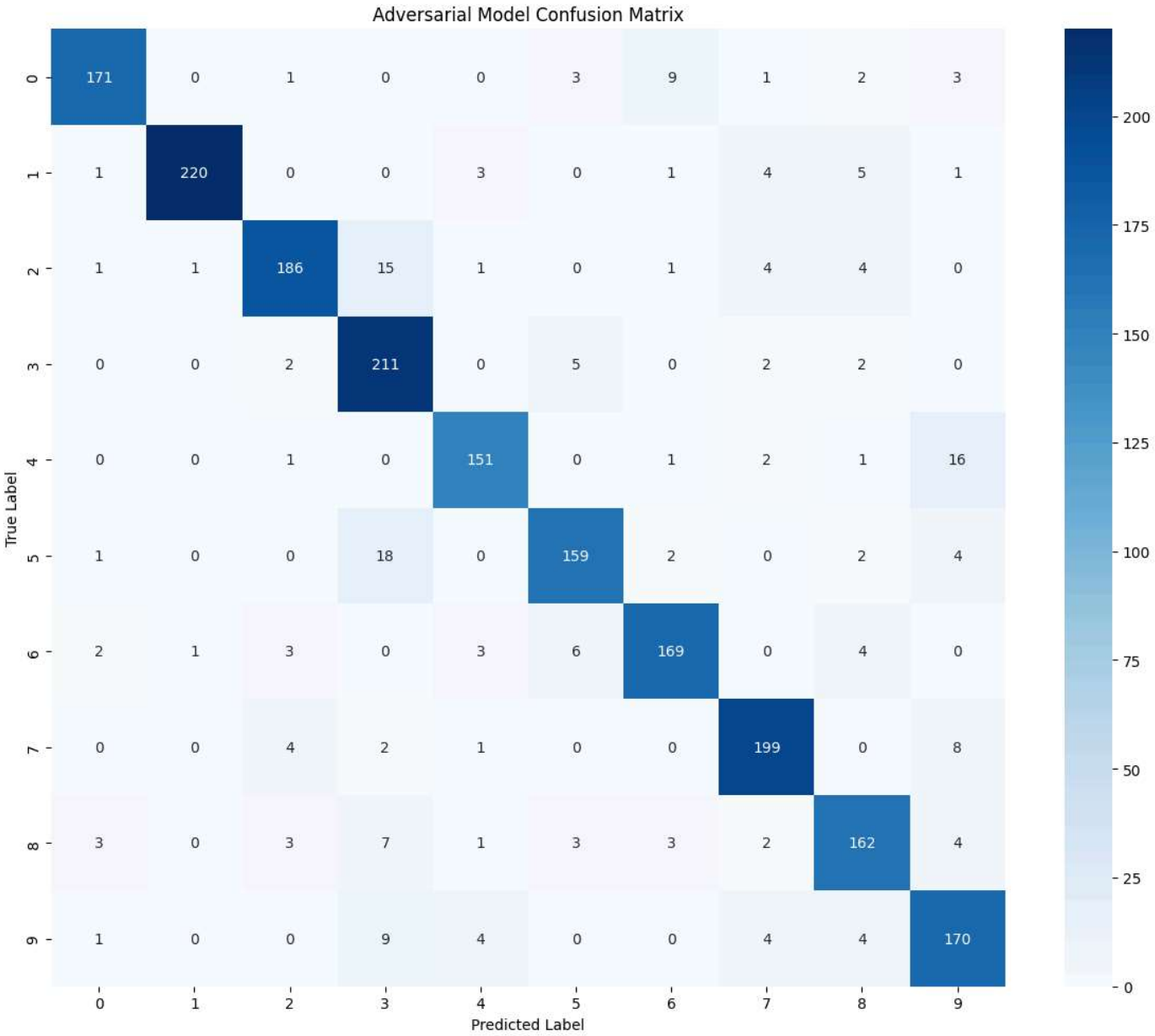
We generated confusion matrices for models trained on MNIST data - both classical and adversarial training, and for evaluation based on a clean test set and a Square-attacked test set. Following are the images we obtained:

Confusion Matrix for Clean test set of a classically-trained CNN with a TTF pre-processing configuration:



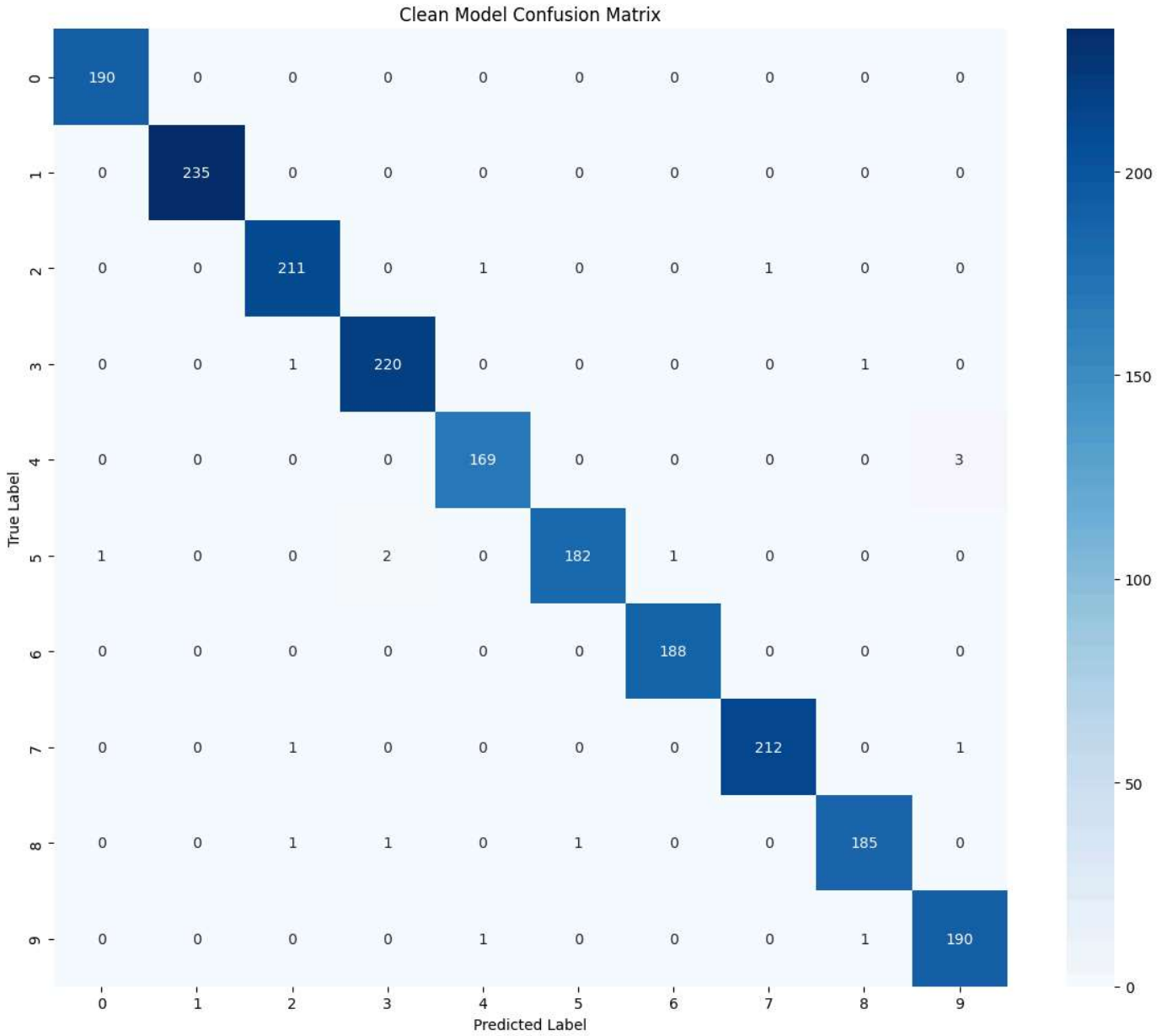
Confusion Matrix (Clean test set) for classically trained CNN

Confusion Matrix for Attacked test set of a classically trained CNN with a TTF pre-processing configuration:



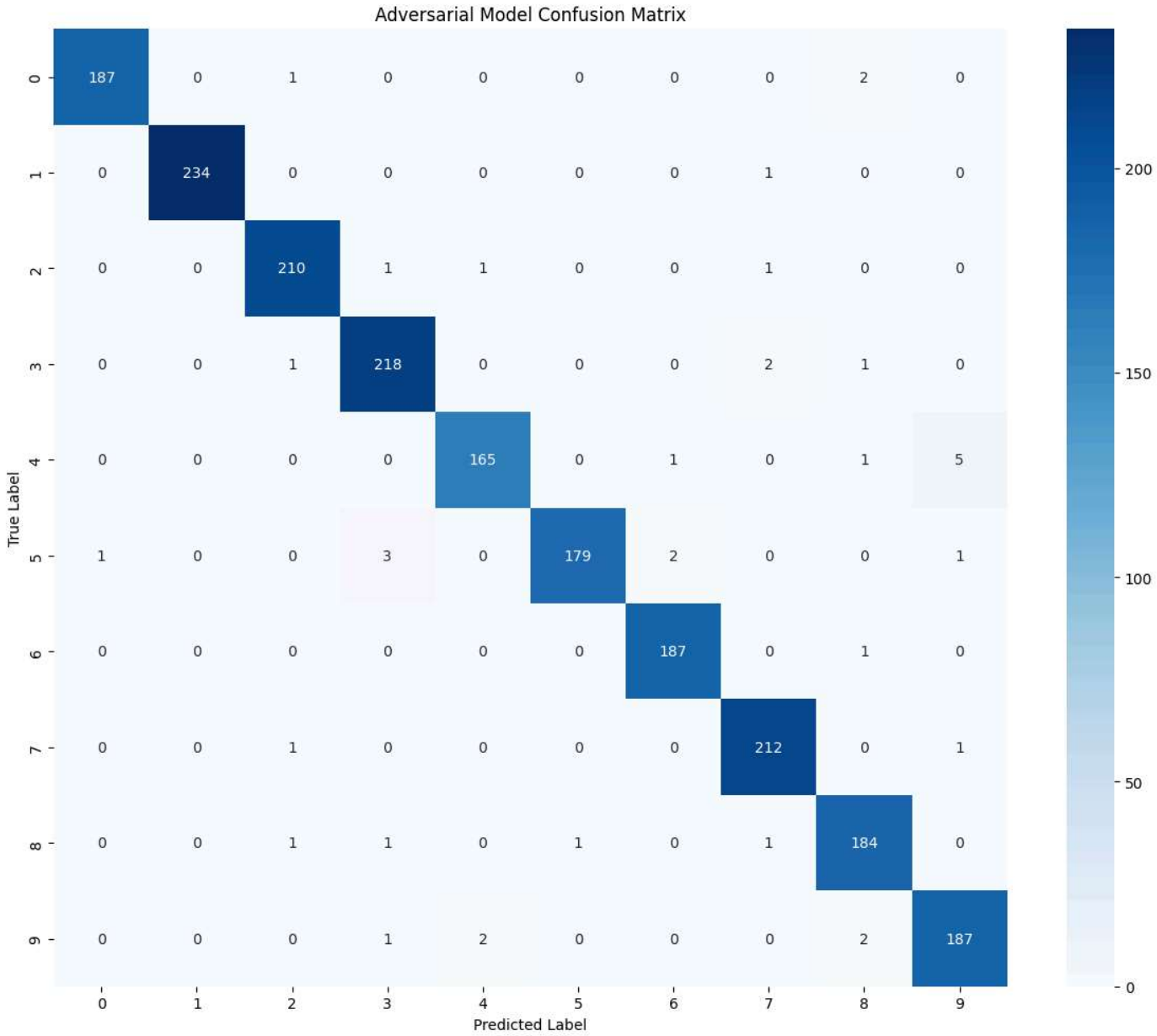
Confusion Matrix (Attacked test set) for classically trained CNN

Confusion Matrix for Clean test set of a PGD-trained CNN with a TTF pre-processing configuration:



Confusion Matrix (Clean test set) for PGD-trained CNN

Confusion Matrix for Attacked test set of a PGD-trained CNN with a TTF pre-processing configuration:



Confusion Matrix (Attacked test set) for PGD-trained CNN

Unsupervised Learning Quantitative Evaluation Results:

Dimensionality Reduction Evaluation Results:

The following are the clean accuracies of GMM classification when the dimensionality of the dataset is reduced via PCA and UMAP with different target dimensions.

PCA Results:

	n_dimensions	Clean Accuracy
0	10	63.85%
1	50	65.60%
2	100	61.22%
3	200	49.58%

UMAP Results:

	n_dimensions	Clean Accuracy
0	10	64.85%
1	50	68.30%
2	100	69.30%
3	200	70.20%

Note: Though UMAP demonstrated the best performance, due to run time concerns, we elected to use PCA with 50 dimensions for experiments in the below sections.

Baseline GMM Results:

We calculate the clean and adversarial classification accuracies of a GMM trained on the clean dataset. Cropping and chopping preprocessing steps were analyzed.

	Preprocessing Config	Clean Accuracy	Square Accuracy
0	Augment (Crop)	31.97%	12.00%
1	Augment (Rotate)	34.99%	9.60%
2	No Augmentations	65.60%	43.60%

Note: We do not analyze class balancing, since this technique is based on random rotation, and rotation has proven to be counterproductive.

Adversarial Training GMM Results:

We calculate the clean and adversarial classification accuracies of a GMM trained on the adversarially augmented dataset. Cropping and chopping preprocessing steps were analyzed.

	Preprocessing Config	Clean Accuracy	Square Accuracy
0	Augment (Crop)	33.87%	27.00%
1	Augment (Rotate)	34.99%	17.50%
2	No Augmentations	63.61%	57.90%

Result Analysis/Discussion

This section presents an analysis of the quantitative results and visualizations observed during our experimentation. Some of the key insights we observed are as follows:

Comparative Analysis of Supervised (CNN) Learning and Unsupervised (GMM) Learning Approaches

Convolutional Neural Networks (CNNs) are the go-to models for image classification due to their superior accuracy and feature extraction capabilities. In comparison, using Gaussian Mixture Models (GMMs) for image classification, although feasible, yields relatively lower performance as observed in clean accuracy. Despite its relatively poor performance, our GMM-based analysis has provided valuable insights:

1. Demonstrated that clustering-based techniques can be adapted for classification tasks.
2. Revealed that adversarial attacks can be effectively applied to unsupervised algorithms.
3. Designed and implemented a novel methodology to enhance the robustness of unsupervised algorithms against adversarial attacks.
4. Established GMM as a "proof-of-concept" model, with potential for improvement using techniques such as aggregation or hybrid approaches.

Analysis of Impact of Adversarial Training

Across both supervised and unsupervised learning techniques, one common observation was that the adversarial training techniques we introduced caused an improvement in the robust accuracy metrics, demonstrating that training machine learning models on perturbed datapoints results in the models learning more robust decision boundaries and becoming less susceptible to adversarial perturbations as a result. More specifically, we observe the following phenomena:

- **Improvement in Robust Accuracy:** Across all datasets and all the pre-processing configurations experimented with, adversarial training (with the PGD-attack for supervised CNN learning and with the Square Attack for the unsupervised GMM learning) in every case improved the robust accuracy of both the CNN and GMM. This result is in line with academic research and is an expected observation -

training on perturbed datapoints ensures that models pick up on the most relevant, robust features to differentiate between classes.

- **(Supervised) Difference between CIFAR-10 and MNIST performance:** While with MNIST, clean accuracy remained roughly the same, adversarially training the CIFAR-10 model resulted in a reduction in the clean accuracy measure. Adversarial training acts as a regularizer by forcing the model to learn more stable representations of data less sensitive to input, which as observed, overly smoothed the decision boundaries making the model less flexible. Adversarial training also alters the decision boundaries of the model to account for adversarial examples, and increases the complexity of the classification task. This phenomenon is known as the robustness-accuracy trade-off, and demonstrates that standard CNNs may not have sufficient capacity to learn both robust and highly accurate representations simultaneously. This problem was mainly observed in models trained on CIFAR-10, which is expected since its data contains a greater amount of information than grayscale MNIST digits which the model performed better on.

Analysis of Impact of Pre-processing Steps

Before starting the experimentation (training and evaluation) loop, we conducted an Exploratory Data Analysis (EDA) to identify pre-processing steps that could potentially enhance performance. Below, we analyze their actual impact on both supervised and unsupervised learning results.

Supervised Learning

- **Data Balancing (MNIST):** Data balancing had negligible impact on accuracy. This is likely due to MNIST's simplicity and its balanced class distribution, which allows standard CNNs to achieve high performance without additional balancing measures.
- **Data Augmentation (CIFAR-10):** Data augmentation significantly improved clean accuracy for CIFAR-10. Techniques like random cropping and flipping helped the model generalize better by exposing it to varied training samples.
- **Gaussian Noise:** Contrary to expectations, incorporating Gaussian noise during training had minimal impact on both standard and robust accuracy. Gaussian noise acts as a generic regularizer, but for datasets like MNIST and CIFAR-10, its isotropic nature may obscure important features rather than aiding the learning process. Furthermore, it fails to address adversarial robustness as adversarial perturbations are highly structured and targeted.

Unsupervised Learning

- **Inapplicability of pre-processing steps from supervised learning:** Preprocessing steps like data balancing, data augmentation, and Gaussian noise had negative effects on performance. Data augmentation can disrupt the natural data structure, and Gaussian noise often obscures patterns essential for clustering algorithms like GMM, reducing performance.

- **Dimensionality reduction:** Empirically, we observed that UMAP with 100 dimensions yielded us close to the best-performing model. In terms of run-time, however, PCA emerged as the faster pre-processing step, hence PCA with 50 dimensions was chosen to run our experiments.
- **Impact of number of dimensions:** Choosing the right number of dimensions is key to balancing performance and computational cost. Reducing dimensions too much can lose important information, while keeping too many can slow down processing with minimal improvement. For UMAP, 100 dimensions provided strong results, but beyond this, the performance gains were marginal (i.e. diminishing returns). PCA with 50 dimensions, on the other hand, offered a good tradeoff between speed and clustering accuracy, making it a more efficient choice.

Analysis of Visualization (Confusion Matrices)

We can gather the following insights from our plotted confusion matrices for supervised learning.

- **Classical Training:** While classical training performs well on the clean test-set as expected, some interesting inter-class confusions are observed for the Square-attacked test set. These confusions often involve visually similar digit pairs, such as mistaking 9s and 4s for one another, 0s for 6s, and 5s and 2s for 3s, as well as 7s for 9s. This suggests that the Square attack reveals more about the model's learned digit representations and how it perceives similar patterns in various hand-written styles.
- **Adversarial Training:** Adversarial training greatly reduced common inter-class confusions, showing that it strengthened the CNN's robustness against visually similar digit misclassifications. This improvement demonstrates how PGD-based adversarial training enables the model to better withstand slight variations or deceptive similarities in input images, achieving greater resilience than standard training.

Future Work

Based on the themes explored in this project as well as the results we observed, avenues for future exploration building up on our findings include:

- **Expanding the diversity in datasets:** Testing our techniques and its effects on adversarial robustness on a wider variety of datasets including those with more varied input distributions may help us generalize our results across different model architectures and application domains.
- **Incorporating eXplainable AI methods:** Developing and experimenting with techniques to integrate robustness results with model interpretability tools such as saliency maps or counterfactual explanations could give us deeper insights into the model's decision-making procedures, helping us understand what exactly drives models to make the classifications they do.
- **Formal Robustness Verification:** Currently, our measure of adversarial robustness is the accuracy of the model on an attacked test set. While this gives us a reliable heuristic measure, it does not

guarantee that the model is robust to any perturbation within the epsilon bound. Formal neural network verification techniques (such as SMT solvers) can be used to formally verify robustness of the network for a set of inputs with a perturbation bound, exhaustively.

Gantt Chart

Gantt Chart

Also viewable on [Google Sheets]

Contribution Table

Name	Project Contributions
Jonathan Weiping W Li	CNN classical and adversarial training implementation. GMM training and evaluation loop implementation. Experiment result analysis.
Richard Hayden Narey	Data Pre-processing: data augmentation, balancing and Gaussian Noise. Project video presentation slides and recording.
Rameen Gauher	Exploratory Data Analysis and Confusion Matrix setup. GMM classical training, and dimensionality reduction experimentation (PCA/UMAP).
Rithik Appachi Senthilkumar	CNN model evaluation implementation. Experiment result analysis. Organization of project report/Streamlit and code/GitHub structure.

Project Video Presentation

[Project Video Presentation]

References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, “EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES,” International Conference on Learning Representations (ICLR), 2015. Available: https://arxiv.org/abs/1412.6572

[2] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019. Available: https://arxiv.org/abs/1905.02175

[3] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," *International Conference on Learning Representations (ICLR)*, 2018. Available: https://openreview.net/forum?id=rJzIBfZAb

[4] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search," pp. 484–501, Aug. 2020. Available: https://doi.org/10.1007/978-3-030-58592-1_29

[5] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. El Ghaoui, and M. I. Jordan, "Theoretically Principled Trade-off between Robustness and Accuracy," *International Conference on Machine Learning (ICML)*, 2019, pp. 7472-7482. Available: https://arxiv.org/abs/1901.08573