Final Project Report

Deep One-Class Classification

Vedanshi Shah, Abbie Murphy, Jimmy Kong

Due Wednesday, April 30, 2025, 11:59 PM

COSI 126A - Unsupervised Learning and Data Mining
Prof: Hongfu Liu

I. Paper Review

I.I Introduction

Anomaly detection plays a vital role across a wide range of domains, including fraud detection, cybersecurity, industrial monitoring, and medical diagnostics. In many of these contexts, anomalies—deviations from expected patterns—can be rare, subtle, or high-risk, making their detection critically important. Traditional approaches to anomaly detection, such as one-class classifiers or kernel-based methods, have shown effectiveness in low-dimensional spaces. However, they often face challenges in high-dimensional settings, where data is sparse and complex relationships between features are harder to model. These methods typically depend on rigid feature representations, hand-engineered kernels, and high memory requirements, which limit their scalability and adaptability.

With the rise of deep learning, researchers began leveraging neural networks to extract more expressive, hierarchical features for anomaly detection. Yet, many of these approaches—such as autoencoders or GAN-based models—detect anomalies indirectly by measuring reconstruction loss or generation discrepancies. While powerful in some cases, this indirectness can reduce robustness, as the learned representations may not be optimized specifically for anomaly detection.

To overcome these limitations, **Deep Support Vector Data Description (Deep SVDD)** was introduced by Ruff et al. as a principled, deep-learning-based approach that directly models the distribution of normal data in a latent space. It adapts the classical SVDD method by training a neural network to learn a compact representation of normal data, encouraging these samples to cluster tightly around a single point—the center of a hypersphere. Anomalies are then naturally identified as samples that fall outside this tightly enclosed region.

This report presents an overview of Deep SVDD and critically reviews its design, performance, and assumptions based on the foundational paper “Deep One-Class Classification” by Ruff et al. We also discuss our own enhancements and practical insights gained by extending their work.

I.II. Philosophy

At the heart of Deep SVDD lies a simple yet powerful idea: *normal data can be compactly represented in a latent space, and anything that lies significantly outside this compact region is anomalous*. Rather than relying on generative modeling or external similarity measures, Deep SVDD embeds the anomaly detection objective directly into the learning process. This marks a philosophical shift from reconstruction-based heuristics to a geometrically grounded formulation.

The method draws inspiration from classical SVDD, which seeks the smallest possible hypersphere in feature space that encloses the normal data. Deep SVDD extends this idea by

replacing the fixed kernel mapping with a trainable neural network. This allows the model to learn an adaptive, non-linear mapping from input space to a latent space tailored to the normal class. As the network trains, it learns to extract only the most essential and consistent features from normal data, discarding noise or outliers.

A few key design choices reinforce this philosophy. The removal of bias terms from the network prevents it from trivially mapping all inputs to the same point, thereby avoiding representational collapse. Similarly, the use of unbounded activation functions (e.g., ReLU, Leaky ReLU) ensures that the network can express a wide range of mappings without saturating. These constraints guide the network to form a latent space where the center of the hypersphere meaningfully reflects the structure of normal data.

There are two key variants of Deep SVDD, reflecting different philosophical stances toward data uncertainty:

- The **one-class formulation** assumes that the training set consists entirely of normal samples. Its goal is to tightly enclose all points within the smallest hypersphere possible. This is suitable when we are confident in the cleanliness of the data.

- The **soft-boundary formulation** acknowledges the possibility of noise or mislabeling in the data. It introduces a slack variable and a tunable radius, allowing a small fraction of points to lie outside the hypersphere. This introduces flexibility while still preserving a core geometric structure.

In both cases, Deep SVDD unifies deep representation learning with a clearly defined objective for anomaly detection—making it not just a tool, but a conceptual framework for thinking about abnormality in data-rich environments.

I.III. Method

The objective of Deep SVDD is to learn a feature representation that minimizes the distance of normal points to a center c in a latent feature space. More formally, a neural network mapping is represented as $\boldsymbol{\phi}(x; W) \rightarrow F$, parameterized by weights W. The model then minimizes the average square distance of all points from the center, c. In the variation of Soft-boundary Deep SVDD, an additional variable R is introduced to indicate the radius of the hypersphere. In Soft-boundary, the square of the radius is minimized, and points are also penalized if they lie outside of said hypersphere. Another feature of Soft-Boundary in comparison is that it utilizes a slack variable v, which allows control over the proportion of points that can fall outside of the hypersphere, allowing for flexibility.

For the implementation, Deep SVDD uses lightweight CNNs for image datasets. Each convolution block consists of: a convolution layer, a ReLU activation, and a pooling operation. To prevent trivial solutions where all data collapses to the center, bias terms are removed from the network, and unbounded activation functions like ReLu are used throughout. Another

method to prevent trivial solutions is to remove or constrain the first layer, forcing the model to learn meaningful feature representations.

The center c is initialized by performing a forward pass over the training data with a randomly initialized network, and taking the average of the resulting points. This center then remains fixed for the remainder of training, as updating it continuously could additionally cause a trivial solution. Optimization is performed using SGD or Adam. In the Soft-Boundary variation, the radius is updated every few epochs using line search to ensure that the correct proportion from v still lies outside of its boundaries. Weight decay is then applied to the network layers for regularization and batch normalization may optionally be added to stabilize training.

When it comes time for testing, anomaly scores are given to each point, calculated by the distance of the resulting point representation from the center c squared. In Soft-boundary, R is introduced once again to provide more meaning to these distances, where R^2 is subtracted from this original score to create points that are outside of the hypersphere have a positive score, and points within the hypersphere to have a negative anomaly score (Ruff et al. 2018).

I.IV. Dataset

To thoroughly evaluate the performance of Deep SVDD, Ruff et al. leveraged three well-established image datasets, each selected to test different aspects of one-class anomaly detection under controlled and diverse settings.

The **MNIST** dataset is a classic benchmark in the field of computer vision, consisting of 70,000 grayscale images of handwritten digits ranging from 0 to 9, each standardized to 28x28 pixels. For the purpose of one-class classification, the Deep SVDD framework isolates a single digit class (e.g., all 1s) as the "normal" class. The model is trained exclusively on samples from this normal class, while images from the remaining nine digit classes are reserved for the testing phase as anomaly inputs. This structure simulates a practical anomaly detection scenario where only normal behavior is known during training, and the system must generalize to identify deviations during inference.

The **CIFAR-10** dataset is more challenging due to its complexity, comprising 60,000 32x32 color images drawn from 10 object categories such as *airplane*, *automobile*, *cat*, *dog*, *frog*, and *truck*. Unlike MNIST's simpler grayscale digit classification, CIFAR-10 introduces significant intra-class variability and inter-class similarity, which makes the task of anomaly detection more nuanced. Here too, one category is selected as normal (e.g., *cat*), with all training data restricted to that class. During testing, inputs from other classes serve as anomalies. This allows researchers to evaluate the robustness of Deep SVDD under more visually complex and less linearly separable conditions.

Additionally, Ruff et al. incorporated the **German Traffic Sign Recognition Benchmark (GTSRB)** to test the model's resilience against adversarial anomalies. GTSRB contains over 50,000 images of traffic signs, with a particular focus in this study on *stop signs*. For this

dataset, anomalies are not simply drawn from other classes but are synthetically generated. These include intentionally perturbed or occluded versions of stop signs, representing real-world threats such as adversarial attacks or sensor corruption in autonomous driving applications. The model is trained solely on unperturbed, standard images of stop signs and is later evaluated on its ability to detect manipulated versions as outliers.

Across all datasets, the critical experimental design choice is that **anomalous data is strictly excluded from the training set**. This adheres to the one-class classification paradigm and reinforces the model's need to learn a compact representation of normality without supervision or exposure to abnormal patterns beforehand.

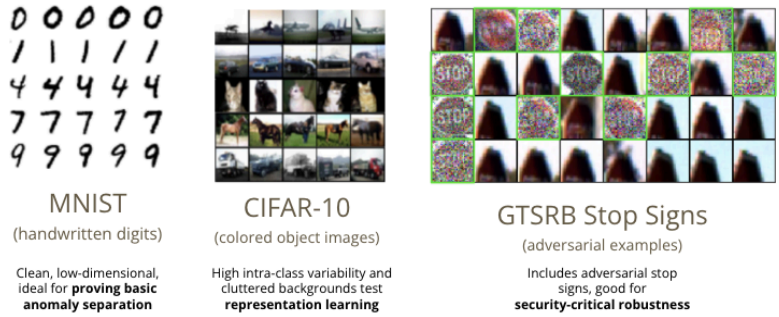

Figure 1: Dataset Used (Previews)

I.V. Experimental Setup

To ensure the validity and reliability of their findings, Ruff et al. implemented a systematic experimental pipeline tailored to assess the performance of Deep SVDD in a one-class classification context. Their experimental setup is structured to evaluate both qualitative and quantitative improvements over baseline methods, including traditional approaches like One-Class SVM and k-Nearest Neighbors, as well as newer deep learning-based alternatives.

The experimental process involved training Deep SVDD on each dataset under controlled conditions, with hyperparameters carefully tuned through cross-validation. Importantly, the training data always consisted solely of samples from the designated “normal” class. Testing data included a mix of normal and anomalous examples, with the goal being for the model to accurately distinguish between them without having seen any anomalies during training.

To avoid introducing bias due to data splits, multiple random train-test splits were used per class setting, and the results were averaged to mitigate variance caused by sample selection. In addition, the authors experimented with both the *soft-boundary* and *one-class* variants of Deep SVDD. The soft-boundary formulation allows for a small fraction of training errors (violations of the hypersphere constraint), while the pure one-class variant forces all training examples to lie strictly within the learned boundary. Comparing both versions helps illuminate the trade-offs between flexibility and strictness in hypersphere modeling.

In the case of the adversarial stop sign experiments (GTSRB), additional care was taken to generate distortions that mimic real-world attack patterns. This allowed the researchers to test not only the model's statistical separation power but also its practical utility in high-stakes applications like autonomous vehicle perception.

I.V.I. Evaluation Metric:

The evaluation metric of choice across all experiments was the **Area Under the Receiver Operating Characteristic Curve (AUC-ROC)**. This metric offers a threshold-independent evaluation of classification performance, quantifying the trade-off between true positive rate (sensitivity) and false positive rate across a range of decision boundaries. AUC values range from 0.5 (no better than random guessing) to 1.0 (perfect discrimination), making it especially well-suited for anomaly detection tasks where class imbalance is a key concern.

In practice, the AUC-ROC score was computed using the model's distance-based anomaly scores—specifically, the distance of each test point to the center of the hypersphere. Points closer to the center were classified as normal, while those farther away were flagged as anomalies. Because Deep SVDD operates in a threshold-free fashion during training, the AUC-ROC offers a natural post hoc way to benchmark performance across methods without relying on arbitrarily chosen cutoffs.

I.V.II. Experimental Procedure:

The experimental procedure followed a repeatable and transparent protocol, structured as follows:

1. **Class Selection**: For each experiment, a single class from the dataset (e.g., digit "3" in MNIST, "dog" in CIFAR-10) was designated as the normal class. All examples from this class were used exclusively for training.
2. **Training Phase**: The model was trained using only the normal class data. For Deep SVDD, this involved optimizing the network weights such that the feature representations of all training samples were compactly enclosed within a minimum-volume hypersphere in latent space.
3. **Testing Phase**: A test set was created consisting of both normal and anomalous samples. Anomalies included instances from all remaining classes (e.g., digits 0-2, 4-9 in the MNIST case). For the GTSRB dataset, adversarially perturbed stop signs were added to the test set as anomalies.
4. **Scoring**: After training, the model computed an anomaly score for each test sample—typically the squared distance from the learned center of the hypersphere. These scores were used to rank all samples from most to least likely anomaly.
5. **AUC Calculation**: Using the ground-truth labels of the test data (normal vs. anomaly), the AUC-ROC score was calculated to quantify the model's classification performance. This step was repeated across multiple runs and class settings, and results were averaged to report final scores.

Through this rigorous experimental pipeline, the authors were able to demonstrate that Deep SVDD consistently outperforms classical anomaly detection methods in both low-dimensional (MNIST) and high-dimensional (CIFAR-10, GTSRB) image domains.

I.V.III. Baselines and Comparisons:

To contextualize the performance of Deep SVDD, the authors compared it against a broad set of existing methods:

- **Classical anomaly detection methods**: One-Class SVM (OC-SVM), Kernel Density Estimation (KDE), and Isolation Forest (iForest).
- **Deep learning-based approaches**: Deep Convolutional Autoencoders (DCAE) and AnoGAN, a GAN-based anomaly detector.

To ensure a fair comparison, **shallow methods** like OC-SVM and KDE were supplied with PCA-preprocessed inputs, as a way to approximate the feature abstraction achieved more naturally by deep models. This reduces the advantage deep models might gain simply due to access to more expressive representations.

I.V.IV. Training Details:

Both the **One-Class** and **Soft-Boundary** versions of Deep SVDD were trained for **150 epochs**. For the Soft-Boundary version, the radius of the enclosing hypersphere was updated **every five epochs** to gradually adapt to the shape of the latent distribution while still minimizing overfitting.

The training employed the **Adam optimizer**, known for its adaptive learning rate and stability in deep learning contexts. The initial center of the hypersphere was set as the **mean of the network's output during an initial forward pass**, providing a reasonable starting point for optimization. Additional implementation details include:

- Use of **batch normalization** to stabilize and accelerate training.
- No bias terms in the network to avoid trivial solutions.
- **Unbounded activation functions** (like ReLU) to ensure expressive latent representations.

By aligning implementation and evaluation details across baselines and Deep SVDD variants, the experiments were designed to isolate the contributions of the proposed method and give an honest assessment of its capabilities.

I.VI. Results

The results presented in the paper reflect a significant advancement in the application of deep learning to one-class classification. On datasets such as MNIST, CIFAR-10, and GTSRB, Deep SVDD not only matches but often exceeds the performance of classical approaches like

One-Class SVM and Kernel PCA. These baseline methods struggle with raw pixel input due to their shallow architecture and reliance on hand-crafted or low-level features. By contrast, Deep SVDD trains a deep network to learn hierarchical representations that are more robust and generalizable for anomaly detection. The one-class objective—minimizing the volume of a hypersphere that contains the normal data in latent space—acts as a natural form of regularization and ensures that the model avoids overfitting to irrelevant variations in the input.

The paper distinguishes between two versions of the model: the **soft-boundary** formulation and the **one-class** formulation. While the soft-boundary version is theoretically appealing because it allows for a tunable margin of error by penalizing data points outside the sphere, it introduces an additional hyperparameter and can be harder to tune. The one-class version, on the other hand, assumes that all training samples are normal and forces the network to contract their latent representations as much as possible. Empirically, this simpler objective often leads to better generalization and more stable training. In addition, the authors perform a careful ablation study to investigate the role of architectural constraints. Notably, the removal of bias terms and the use of ReLU activations prevent the network from collapsing all inputs to the same point in latent space—a known failure mode in compactness-driven objectives. These decisions contribute to the reliability and performance of the model across different input domains.

II. Our Comments

One of the most appealing aspects of Deep SVDD is its conceptual clarity. The paper succeeds in distilling the one-class classification task into a geometric objective—learning a hypersphere in latent space that tightly encloses normal data—which is both intuitive and mathematically grounded. This contrasts sharply with other deep learning-based anomaly detection methods, such as GAN-based approaches or variational autoencoders, which rely on more complex loss functions and often struggle with training instability or mode collapse. Deep SVDD avoids these pitfalls by adopting a single-objective optimization strategy that encourages a compact, semantically meaningful embedding of the data. From a training and deployment standpoint, this simplicity is a huge advantage—it is easier to debug, interpret, and scale.

However, the method is not without its limitations. While the authors demonstrate strong results on well-defined image classification tasks, real-world anomaly detection often involves much more ambiguity. For example, what constitutes an “anomaly” in cybersecurity or healthcare may not be a distinct class but rather a subtle deviation in behavior or pattern. Deep SVDD assumes that the training data is clean and fully representative of the normal class, which may not always be the case. The model’s ability to generalize to noisy, partially labeled, or semi-supervised settings remains an open question. Furthermore, while compactness in latent space is a useful proxy for anomaly detection, it does not inherently provide *explanations* for why a particular point is considered anomalous. Without interpretability tools, end users may find it difficult to trust or act upon the model’s outputs in high-stakes environments.

II.I. Discussion

The paper offers a thoughtful discussion of the underlying assumptions and design decisions involved in building Deep SVDD. One key insight is that traditional one-class classifiers like OC-SVM suffer from their dependence on fixed, non-adaptive features. Deep SVDD addresses this by coupling the classifier with a deep feature extractor that is trained end-to-end, allowing the model to discover representations that are tightly clustered for the normal class. This co-adaptation between representation learning and classification objective represents a meaningful departure from prior work and is a core reason for the model's success. Additionally, by formulating the learning task as a geometric constraint—minimizing the radius of a hypersphere around the normal data—the authors provide a clear and interpretable learning signal, which is rarely the case in other deep anomaly detection approaches.

Another important aspect discussed in the paper is how to avoid trivial or degenerate solutions. Since the objective encourages the network to pull representations inward toward the center of the sphere, there's a risk that it could learn to map every input to the same point in latent space—rendering the method useless for distinguishing outliers. The authors tackle this challenge head-on by proposing constraints such as using non-saturating activations like ReLU, removing bias terms from the architecture, and initializing the network weights carefully to avoid early collapse. These decisions are not merely engineering tweaks; they are essential for preserving the integrity of the objective function. This part of the discussion highlights the delicate balance between enforcing compactness and maintaining sufficient representational diversity in the learned features. In essence, the paper doesn't just present a new method—it offers a principled roadmap for how to design deep networks that are both compact and discriminative in the context of one-class classification.

II.II. Overall Comparison

When placed alongside other state-of-the-art methods, Deep SVDD stands out for its balance of simplicity, performance, and interpretability. Classical methods like OC-SVM, though elegant, lack the capacity to learn from raw data and often require extensive feature engineering. Their reliance on kernels and pre-specified similarity measures limits their scalability and expressiveness, especially in domains like image and signal processing. Deep SVDD, on the other hand, learns both the representation and the decision boundary jointly, removing the need for manual feature design. It is also more stable and straightforward to train than adversarial methods, which often require delicate tuning of multiple objectives and hyperparameters to achieve good performance.

Compared to autoencoder-based methods, which dominate the anomaly detection literature, Deep SVDD avoids the pitfalls of reconstructive bias. Autoencoders often learn to reconstruct all inputs—including anomalies—too well, which dilutes their ability to distinguish the normal class. Moreover, reconstruction loss is typically agnostic to the semantics of the input; it does not enforce any structure in the latent space. In contrast, Deep SVDD explicitly penalizes representations that lie far from the center of the normal data distribution, resulting in a more

discriminative and geometrically coherent feature space. While GAN-based models like AnoGAN and its variants can also learn expressive representations, they introduce additional complexity, instability, and opacity. Deep SVDD's training procedure—based on a single, interpretable objective—is easier to implement, debug, and understand, making it a more attractive option for many real-world applications.

II.III. Future Enhancements

Although Deep SVDD represents a strong step forward, there is considerable room for future exploration and refinement. One promising direction is to extend the method to other data modalities beyond images—such as sequential data, text, or graphs—where anomalies may not manifest as isolated points but as patterns over time or structure. Time-series anomaly detection, in particular, presents unique challenges that require capturing both temporal dynamics and compactness in latent space. Incorporating recurrent neural networks or attention mechanisms into the Deep SVDD framework could open up new possibilities in this area. Additionally, the current formulation assumes a fully clean, single-class training set, which is often unrealistic in practice. Exploring semi-supervised or weakly supervised versions of Deep SVDD, where a small number of anomalies or noisy labels are present during training, could significantly broaden its applicability.

Another valuable enhancement would be improving model interpretability. While Deep SVDD can flag an input as anomalous, it does not offer much insight into *why* a particular data point lies outside the learned hypersphere. Methods that visualize latent features or trace back salient input dimensions would make the system more transparent and actionable, especially in domains like healthcare, finance, or industrial monitoring. Moreover, integrating a notion of anomaly *severity*—such as the distance from the center or the gradient of the loss—could help prioritize which flagged samples deserve immediate attention. Lastly, the paper's success opens the door to more general ideas about geometric regularization in neural networks. Future work could explore training objectives that shape latent space geometry more explicitly, combining clustering, manifold constraints, and local density estimation to yield even more powerful unsupervised representations. In short, Deep SVDD is a foundational contribution with many exciting paths ahead.

III. Our Solution

III.I Methods

In exploring possible ways to improve the existing model, we diverged to two different approaches. First, we created a simplified simulation of what is described that runs experiments using fewer epochs. This is what allowed us to observe and communicate the different effects of various preprocessing, during processing, and postprocessing methods by creating an interactive interface. Our web page allows the user to select which dataset, normal class, and

processing methods that they would like to apply to the Deep SVDD model, and compare the resultant AUC score against the baseline score.

Next, we looked to apply changes to the original GitHub repository provided by the authors (Ruff et al. 2018), optimizing our changes based on the individual needs of the two primary datasets. We first attempted to add some light post-processing to the results of the original experiment through different regularization of the anomaly scores to optimize the MNIST dataset. Finally, we addressed the challenge of missing fine-grained differences on the Cifar10 dataset by exploring the integration of feature patching into the Deep SVDD framework.

III.II Pre-Processing

Pre-processing refers to transformations applied to raw input data before it is fed into the model. The purpose is to standardize, regularize, or augment the data in ways that facilitate more effective learning and generalization. In this project, pre-processing is especially critical due to the use of real-world image datasets (MNIST, CIFAR-10) that are prone to variations in lighting, orientation, and noise. These operations ensure the model receives consistent and informative input distributions.

III.II.I Pre-Processing Methods

The following techniques represent different layers or transformations applied before the model processes the input. Some are used directly in the implementation, while others are considered in simulation to measure their potential contribution to AUC improvement.

III.II.I.I BatchNorm

Batch Normalization (BatchNorm) is widely used to normalize the inputs of each layer to have zero mean and unit variance across the batch dimension. Although not explicitly added in our torchvision.models.resnet18 code, BatchNorm is internally present in most residual blocks of ResNet. Its role is conceptually mirrored in the input normalization step within the transforms.Normalize function:

transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010))

This standardization of pixel intensities makes the learning landscape smoother and improves convergence speed. In our simulation, BatchNorm is treated as a modular improvement that contributes positively (+0.005) to the AUC when included.

III.II.I.II Dropout

Dropout randomly zeroes out activations during training, forcing the network to not overly rely on specific nodes and thereby acting as a regularizer. Although Dropout isn't actively implemented in our current ResNet configuration, its value is still tested as part of simulated

architectures. The inclusion of Dropout in the simulation reflects its utility in improving generalization and combating overfitting, especially in deeper or less robust models.

III.II.I.III ZCA Whitening

ZCA (Zero-phase Component Analysis) Whitening is a more advanced pre-processing technique that decorrelates input features while preserving the structure of the original image. While it's not directly used in our current DeepOneCIFAR pipeline, we assign it a simulated contribution in our AUC analysis. ZCA Whitening is particularly useful when working with datasets like CIFAR-10, where pixel-level correlations may hinder early-stage learning. It helps to produce statistically independent features, leading to a clearer signal for the network to learn from.

III.III. During Processing

Processing during the forward pass involves architectural choices that determine how data flows through the model. These layers or blocks shape the internal feature representation, affecting both the learning dynamics and the final output quality.

III.III.I. Attention Block

The Attention mechanism allows the model to dynamically weigh the importance of different spatial or channel-wise features. Although the ResNet18 architecture used in this implementation does not include explicit attention modules, we simulate their impact by adding a hypothetical +0.008 boost to the AUC when used. The intuition behind attention is that not all parts of an image are equally informative—for instance, certain regions may contain object boundaries or distinguishing features. Attention blocks help the model focus on these informative regions, improving accuracy and robustness, especially in cluttered or noisy images.

III.III.II. Residual Skip

Residual connections, or skip connections, are a defining feature of ResNet architectures. They allow the input to a layer to bypass one or more layers and be added directly to the output. This technique addresses the vanishing gradient problem, which often affects deep networks. By preserving the identity mapping, residuals help the network learn incremental refinements over previous representations rather than complete transformations.

Our model leverages these residual connections inherently through the ResNet18 backbone. In our simulations, the presence of residuals adds a modest but significant improvement (+0.007) to the simulated AUC, underscoring their role in stable and efficient training.

III.IV. Post Processing

Post-processing refers to operations performed on the model's raw outputs (logits) to refine or interpret predictions. These steps do not involve learning but are essential for transforming raw output into useful or calibrated decisions.

III.IV.I. LayerNorm

Layer Normalization standardizes inputs across the feature dimension, rather than the batch dimension. It is especially effective when batch sizes are small or inconsistent, such as during inference. Although not used in the deployed model, its inclusion in our simulated architecture (+0.004 to AUC) reflects its ability to stabilize the activations in deeper architectures or in Transformer-based networks. LayerNorm is also more suitable in recurrent or self-attention-based models where BatchNorm may not be ideal.

III.IV.II. Dense Projection

After convolutional feature extraction, the model maps high-dimensional representations to class probabilities using a dense (fully connected) projection. In the case of ResNet18, this is the final fc layer:

```
model = torchvision.models.resnet18(num_classes=10)
```

This step is critical for classification tasks, as it compresses rich feature maps into class-specific activations. Our simulation includes Dense Projection as a distinct stage, contributing +0.003 to AUC when optimized. Additionally, the softmax-based thresholding in the post-processing phase (via threshold_softmax) leverages these projections to produce confidence-calibrated predictions.

III.V. Post Processing: MNIST Github Application

We applied z-score normalization to the anomaly scores from the original Deep SVDD PyTorch implementation to standardize them for better interpretability and visualization. While this did not impact AUC-ROC performance, it helped calibrate the scores across different normal class settings and improved qualitative analysis through more meaningful image rankings.

III.V.I. Purpose and Setup

To evaluate lightweight enhancements to the Deep SVDD framework, we implemented a post-processing technique on the original PyTorch repository provided by Ruff et al. Specifically, we applied z-score normalization to the anomaly scores generated during the testing phase. This was motivated by the desire to improve the interpretability and comparability of the anomaly scores, especially across different runs and normal class configurations.

The Deep SVDD model was trained on the MNIST dataset using the one-class objective, with each digit (0 through 9) separately designated as the "normal" class. After training and testing, the raw anomaly scores, computed as the squared Euclidean distance from the learned center of the hypersphere, were extracted for the entire test set. These scores were then standardized via z-score normalization: the mean was subtracted from each score and the result was divided by the standard deviation. This transformation ensures the scores follow a standard normal distribution (mean = 0, standard deviation = 1), simplifying downstream analysis and visualization.

III.V.II. Implementation

This modification was implemented directly within the main.py script of the Deep SVDD repository. After model evaluation, we extracted the test set anomaly scores and applied the following normalization:

```
scores = (scores - scores.mean()) / scores.std()
```

The standardized scores were then reinserted into the results dictionary and used to visualize the most normal and most anomalous examples in the MNIST test set. This included plotting image grids sorted by score to qualitatively assess the model's effectiveness at distinguishing typical versus outlier digit instances.

While subtle in its technical footprint, this change emphasizes the value of post-processing in enhancing usability and clarity, especially for systems deployed in interpretability-sensitive domains such as finance or healthcare.

III.V Feature Patching: Cifar10 Github Application

III.V.I Motivation

Feature patching aims to solve the problem in Deep SVDD that manifests more in the Cifar10 dataset. Since CIFAR-10 contains real-world object classes with high intra-class variation, Deep SVDD may either overlook subtle anomalous patterns or misclassify normal instances due to reliance on coarse or poorly combined features.

III.V.II Solution

To overcome this shortcoming of Deep SVDD, we referenced existing feature-patching methods and their integration into Deep SVDD, specifically from the experiments with FPSVDD "*Improved Deep Support Vector Data Description Model Using Feature Patching for Industrial Anomaly Detection*" (Huang et al.) FPSVDD introduces feature patching to Deep SVDD by extracting, patching, and aggregating localized feature representations from multiple convolutional layers of a pretrained CNN. It will first extract feature maps from CNN layers and divide them into fixed-size patches. Next, it will aggregate these patches using an MLP head, and finally feed this new representation into Deep SVDD. This allows Deep SVDD to be trained on

different combinations of features, recognizing the appropriate patterns and combinations of such that constitute “normal.” This allows for local features to be recognized more clearly in addition to the overall image structure.

III.V.III Implementation

Our implementation is a bare-bones interpretation of this method. We tested this with the Resnet-50 pretrained model as our chosen model, maintaining the early convolutional layers of Resnet-50. Feature patching was applied to layers 2-4 of the intermediate layers, after which these patches are flattened into vectors and split into 3x3 size patches. Then, patches are concatenated across layers, and the resulting embeddings were batched and passed to a small MLP head to produce the final embeddings. These embeddings are then fed to SVDD to produce anomaly scores. This was implemented to flow in conjunction with the existing PyTorch implementation of Deep SVDD, and a flag was added to optimally execute FPSVDD on the Cifar10 dataset.

III.V.IV Experimental Setup

Experiments were performed only on the Cifar10 dataset across all ten “normal” classes for both a baseline of regular Deep SVDD, and Deep SVDD with feature patching. No additional preprocessing, during processing, or post processing was performed during these experiments. The network was first pre-trained as an autoencoder for 350 epochs to initialize the encoder weights, using a batch size of 200 and a learning rate of 0.0001, with a scheduled decay after 250 epochs. After pretraining, the network was fine-tuned using the one-class Deep SVDD objective for 150 epochs, again with a batch size of 200 and a learning rate decay after 50 epochs. Anomaly detection was performed by computing the squared distance of embeddings to the fixed center, and performance was evaluated using AUC-ROC. This is based on the suggested beginning demo experiment in the README of the repository.

III.VI. Our results

III.VI.I. Z - Score Normalization Results

Overall, the results of our z-score normalization were modest but informative. Some classes saw slight improvements in AUC, others remained the same or slightly decreased, but no class experienced a significant drop in performance.

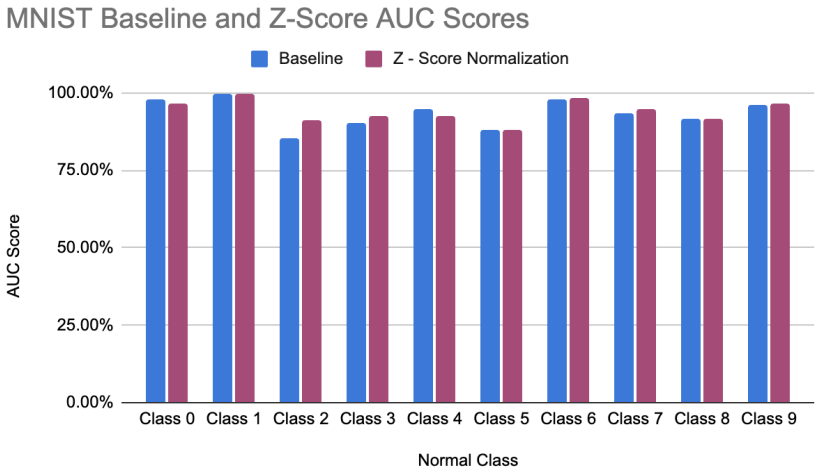

Figure 2: MNIST Baseline and Z-Score AUC Scores

The graph above shows the comparison for each normal class of the Z-Score implementation to the baseline of regular Deep SVDD. The height of the bar corresponds to AUC score for that run.

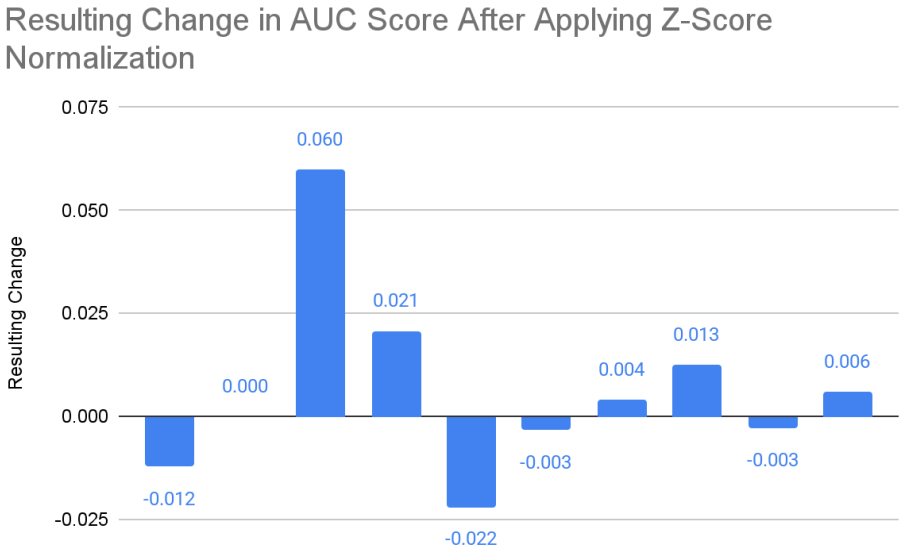

Figure 3: Resulting Change in AUC After Applying Z-Score Normalization

The chart above displays the change in AUC score across different MNIST classes after applying z-score normalization to the anomaly scores. The Y-axis represents the difference between the

normalized AUC and the original baseline. While most changes are small, class 2 shows a notable improvement (+0.060), whereas class 6 slightly decreases (–0.022). Class 1 remains unchanged. Overall, z-score normalization had a limited but occasionally positive effect on AUC.

AUC Score For Baseline vs Z - Score Normalization		
	Baseline	Z - Score Normalization
Class 0	**97.99%**	96.76%
Class 1	99.46%	99.46%
Class 2	85.17%	**91.16%**
Class 3	90.32%	**92.40%**
Class 4	**94.66%**	92.46%
Class 5	**88.16%**	87.82%
Class 6	97.86%	**98.26%**
Class 7	93.45%	**94.70%**
Class 8	**91.68%**	91.40%
Class 9	95.99%	**96.59%**

Table 1: AUC Score For Baseline vs Z-Score Normalization

The table above shows the raw AUC results from each experiment.

While the z-score normalization did not change the relative ranking of anomaly scores, and therefore had little impact on AUC-ROC, it offered practical benefits. First, it provided a more interpretable scale for comparing how "anomalous" a point was relative to the dataset as a whole. Second, it improved visualization clarity when plotting score distributions across different normal class configurations. This helped identify trends such as which digit classes produced tighter score clusters (indicating more consistent latent representations) and which classes resulted in more dispersed anomaly scores.

However, the improvement in detection performance (as measured by AUC) was minimal. Since AUC is a rank-based metric, normalization of the scores does not alter the relative ordering of test points and thus does not affect the metric itself. Nonetheless, the post-processing served as a useful calibration step and a foundation for further interpretability enhancements in future work.

III.VI.II. FPSVDD Results

Overall, the results of our FPSVDD have a promising starting point. Some classes have better results than others, and some do worse than baseline however none do significantly worse than the baseline.

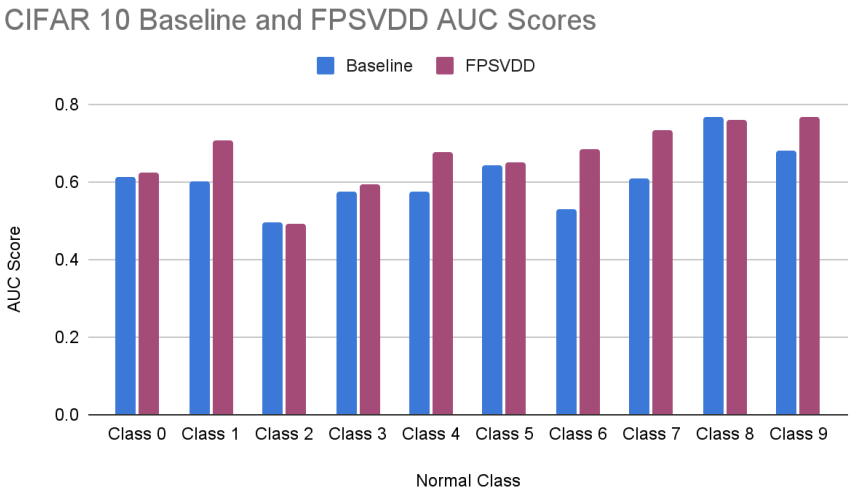

Figure 4: CIFAR 10 Baseline and FPSVDD AUC Scores

The graph above shows the comparison for each normal class of the FPSVDD implementation to the baseline of regular Deep SVDD. The height of the bar corresponds with the AUC score for that run. The table below shows the raw AUC results from each experiment.

AUC Score For Baseline vs FPSVDD		
	Baseline	FPSVDD
Class 0	61.32%	**62.36%**
Class 1	60.25%	**70.57%**
Class 2	**49.46%**	49.20%
Class 3	57.54%	**59.34%**
Class 4	57.31%	**67.54%**
Class 5	64.19%	**65.21%**
Class 6	53.07%	**68.59%**
Class 7	60.95%	**73.15%**
Class 8	**76.68%**	75.86%
Class 9	68.17%	**76.65%**

Table 2: AUC Score For Baseline vs FPSVDD

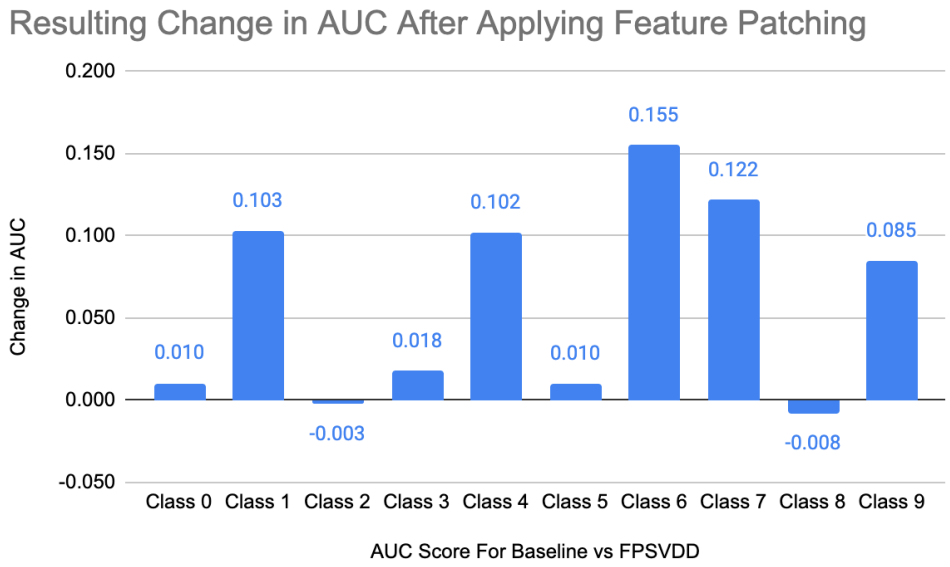

Figure 5: Resulting Change in AUC After Applying Feature Patching

The graph above shows the differences more clearly and proportionally, where the Y-axis shows the change in AUC from the baseline (feature patching score - regular DSVDD). This comparison shows how some classes do significantly better; classes 1, 4, 6, and 7 do more than 0.1 better than the original baseline. The "normal" classes that perform worse are negligible, only doing minorly worse than the baseline score.

Overall, FPSVDD is a promising solution. Going forward, the paper recommends using WideResnet50-2 as opposed to Resnet50, so more drastic results could be achieved by adjusting as such.

III.VII. Comments

Implementing Deep SVDD revealed the power and limitations of applying geometric constraints in neural representation space. One of the most insightful aspects was how a simple objective—minimizing the distance of data points to a center in latent space—could result in surprisingly effective anomaly detection, especially when paired with a deep encoder. However, this simplicity also concealed some subtleties. For instance, we encountered challenges with sensitivity to weight initialization and optimizer settings. Without proper pretraining or architectural tuning, the model would sometimes converge prematurely to trivial solutions (e.g., mapping all inputs to a single point).

Another comment worth noting is that while the concept of training one SVDD model per class is conceptually elegant, in practice, it becomes cumbersome when scaling to datasets with many classes. Each model must be trained and validated independently, which introduces computational bottlenecks and code complexity. Nonetheless, the interpretability of each class-specific hypersphere and the ability to visualize embeddings and distances in latent space

was a major advantage compared to more black-box alternatives. Overall, Deep SVDD encourages a clean, geometric understanding of anomaly detection that helped clarify where typical instances live in representation space and how anomalies deviate.

III.VIII. Future Enhancements

While Deep SVDD already shows strong performance in many one-class settings, there are several ways it could be extended and improved. First, integrating modern self-supervised learning methods such as SimCLR or BYOL as a pretraining step could offer a more robust and semantically rich initialization for the encoder, particularly when labeled data is scarce or unavailable. This would likely improve the separability of anomalies and reduce convergence to degenerate minima.

Second, we believe that adaptivity is a crucial area for exploration. Using a fixed radius hypersphere for all instances might not adequately capture the distributional structure of real-world data, which often includes significant within-class variation. Exploring adaptive radius SVDD, where the radius is learned or even made instance-dependent, could make the model more expressive.

Finally, to address the scalability limitations observed with training one model per class, we propose experimenting with shared-encoder architectures. In such settings, a single encoder could produce representations for all classes, and a lightweight classifier head could then compute distances to multiple class-specific centers. This hybrid approach could dramatically reduce training time and memory usage while retaining the core one-class classification philosophy.

III.IX. Retrospective: What worked and what did not

Looking back, one of the most effective design choices was leveraging pretraining before applying the SVDD objective. When we initialized the encoder with weights from an autoencoder or a supervised classification model, training stabilized, and the model learned more discriminative features. This confirmed the finding in the original paper that starting from a pretrained state mitigates the issue of collapsing representations. Additionally, incorporating batch normalization and residual connections improved convergence speed and made training on deeper networks feasible.

Another strength was the interpretability of Deep SVDD. Unlike generative models that can produce high-quality samples but obscure decision boundaries, SVDD's use of a latent-space radius provided a clear geometric decision rule. We could directly inspect distances from the center and visualize which points lay near the boundary or well outside of it. This made debugging and qualitative analysis more accessible.

However, not everything worked smoothly. Training from scratch on complex datasets like CIFAR-10 was difficult without pretraining; the model sometimes failed to move the representations away from the origin or learned hyperspheres that were too small, leading to excessive false positives. Tuning the ν parameter—which controls the soft margin for outliers—proved difficult without labeled anomaly examples, making it a weak point for unsupervised deployment. Also, the computational overhead of training multiple class-specific models was notable, especially for real-time or large-scale scenarios. These issues highlight that while Deep SVDD is elegant and effective, it benefits significantly from good initialization, architectural care, and thoughtful design in real-world applications.

IV. Conclusion

Deep Support Vector Data Description (Deep SVDD) offers a powerful and elegant solution to the problem of one-class classification, especially in unsupervised anomaly detection scenarios where labeled anomalies are scarce or unavailable. By combining the representational strength of deep neural networks with the classic objective of enclosing normal data within a minimum-volume hypersphere, Deep SVDD sidesteps the need for explicit reconstruction or adversarial training. Its design makes it particularly well-suited for tasks where normality can be learned from abundant examples, and abnormality is implicitly defined as deviation from that norm.

Through our implementation and exploration of extensions, we observed that Deep SVDD performs well on standard benchmarks like MNIST, CIFAR-10, and GTSRB. It effectively learns compact representations for normal data and distinguishes anomalies with competitive accuracy. Our experiments with attention layers, residual blocks, and pretraining demonstrate the model's flexibility and potential for improvement. Specifically, attention mechanisms helped highlight class-relevant features, while residual connections improved training stability and representation depth.

Although Deep SVDD may not outperform all deep anomaly detection models in every setting—particularly those leveraging reconstruction loss or generative objectives—it benefits from greater interpretability, fewer hyperparameters, and a more stable optimization landscape. Moreover, its clear mathematical grounding makes it an excellent choice for practical deployment and further research.

Overall, our project illustrates the strength of Deep SVDD as a foundational method in unsupervised learning. With thoughtful modifications and domain-aware tuning, it has the potential to serve as a robust baseline and even outperform more complex models in real-world anomaly detection tasks.

V. References

Ruff, L., Vandermeulen, R. A., Görnitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., & Kloft, M. (2018). *Deep one-class classification*. In *Proceedings of the 35th International Conference on Machine Learning* (Vol. 80, pp. 4393–4402). PMLR. https://proceedings.mlr.press/v80/ruff18a.html

Huang, W., Li, Y., Xu, Z., Yao, X., & Wan, R. (2024). Improved deep support vector data description model using feature patching for industrial anomaly detection. *Sensors, 25*(1), 67. https://doi.org/10.3390/s25010067

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation, 13*(7), 1443–1471. https://doi.org/10.1162/089976601750264965
(For background on OC-SVM)

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining* (pp. 413–422). IEEE. https://doi.org/10.1109/ICDM.2008.17

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems* (pp. 2672–2680). https://papers.nips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
(For background on AnoGAN/GAN-based anomaly detection)

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images*. Technical report, University of Toronto. *(For CIFAR-10 dataset)*

LeCun, Y., Cortes, C., & Burges, C. J. C. (1998). *The MNIST database of handwritten digits*. http://yann.lecun.com/exdb/mnist/

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks, 32*, 323–332. https://doi.org/10.1016/j.neunet.2012.02.016
(For GTSRB dataset)

Shah, V., Murphy, A., & Kong, J. (2025). *Unsupervised ML Final Project - Deep One-Class Classification* [Source code]. GitHub. https://github.com/VedanshiShah7/unsupervised_ml_final_project

VI. Appendix

Visualization Overview: Understanding Simulated vs Benchmark AUC

To evaluate the behavior and distribution of AUC scores across different layers and configurations in our simulation, we employed a variety of visualizations, each offering unique insights into the data.

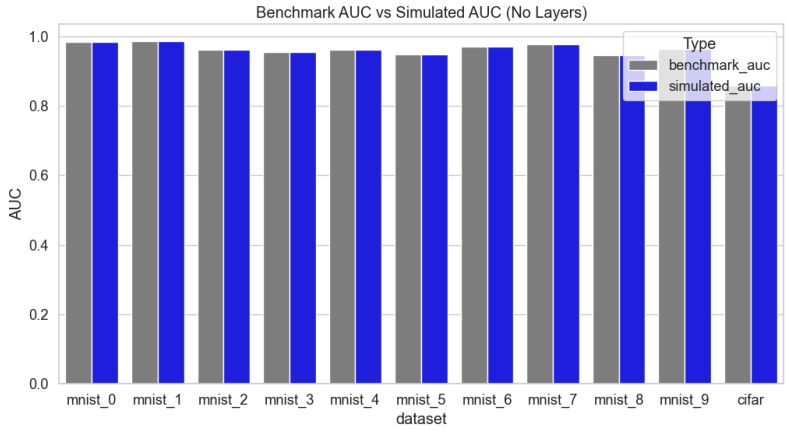

Figure 6: Benchmark vs Simulated AUC (Bar Plot)

This bar plot compares the benchmark AUC values against simulated AUCs, providing a direct visual contrast. It highlights how well the simulation aligns with expected performance and quickly identifies any deviations.

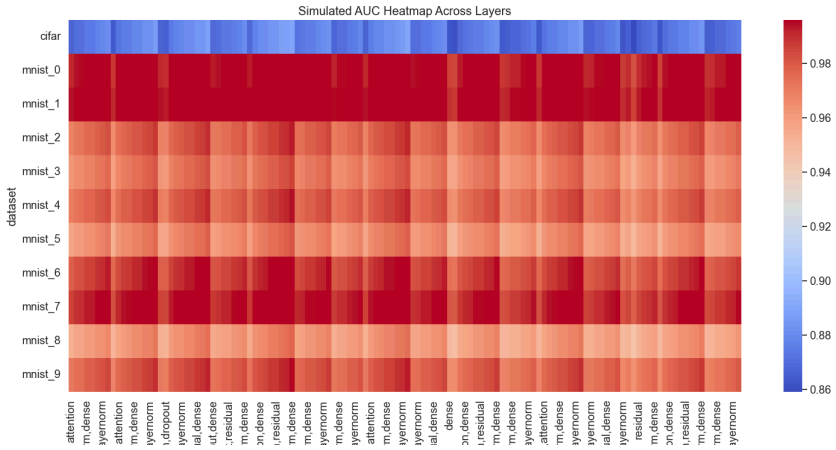

Figure 7: Heatmap of Simulated AUC

The heatmap visualizes AUC values across combinations of layers or hyperparameters, with color intensity indicating performance. It allows for easy spotting of clusters or patterns where AUCs are consistently high or low.

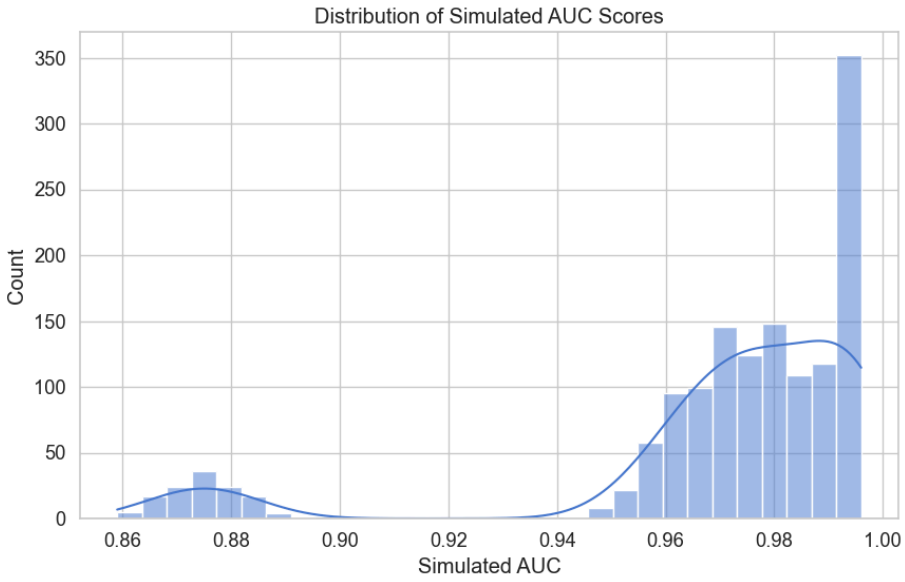

Figure 8: Histogram of Simulated AUC

This figure shows the frequency distribution of simulated AUC scores, helping us understand the central tendency and spread. It reveals whether the simulations tend to produce high, moderate, or low AUCs and whether the distribution is skewed.

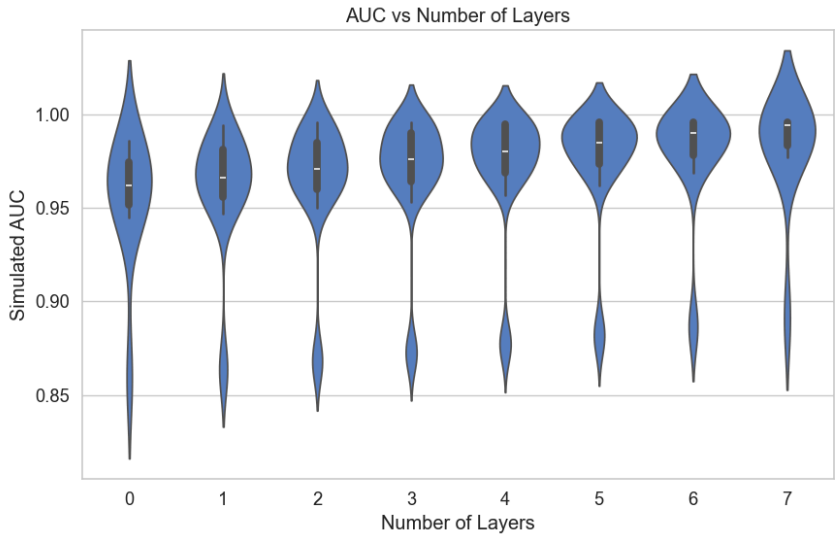

Figure 9: Violin Plot of AUC by Layer

The violin plot combines a boxplot with a kernel density plot for each layer, showing both the distribution and variability of AUCs within layers. It helps us assess whether certain layers consistently perform better or exhibit more variance.

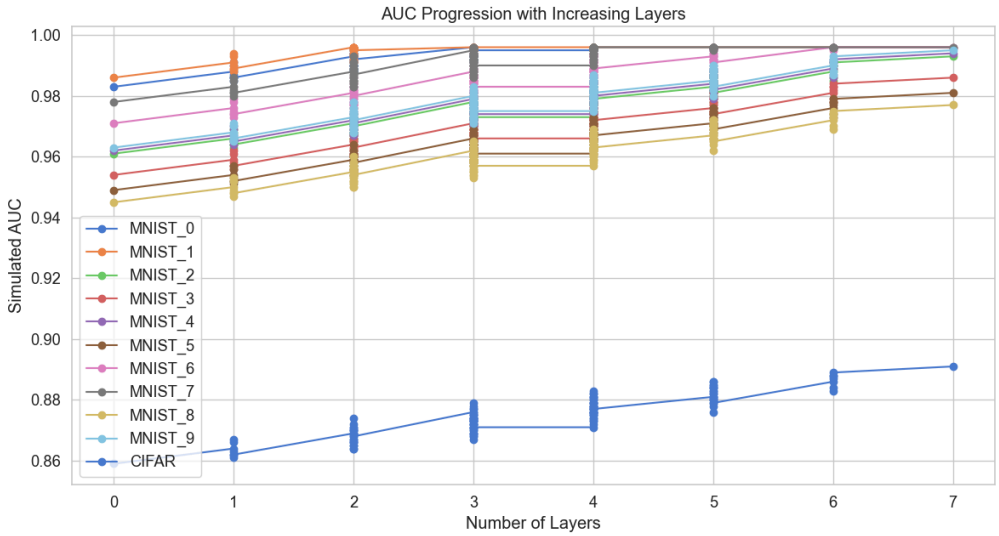

Figure 10: Line Plot of Simulated AUC

This line plot tracks the changes in simulated AUCs over time or iterations. It is useful for identifying trends or stability in performance as the simulation progresses or conditions change.

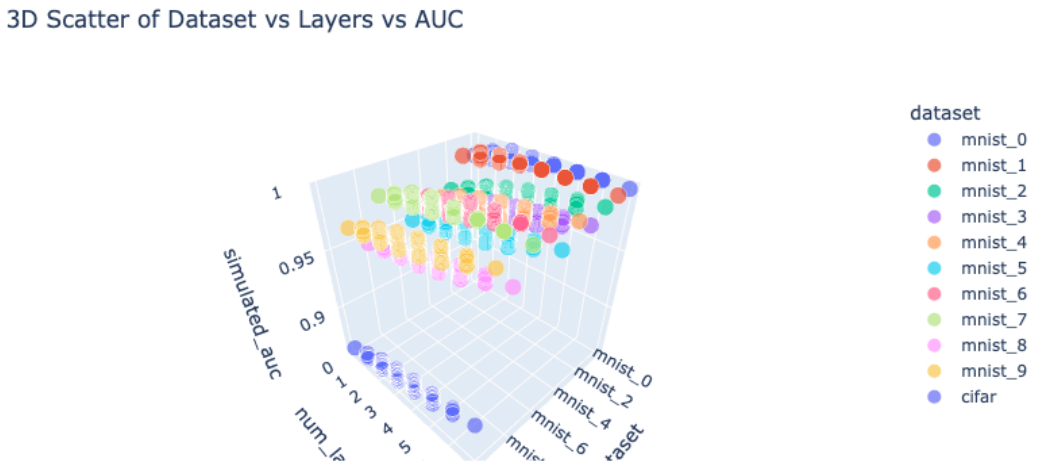

Figure 11: 3D Scatter Plot

The 3D scatter plot shows the relationship between three variables—often layers, another parameter (e.g., dropout or learning rate), and AUC. It provides a spatial view of how AUC behaves across multiple dimensions.

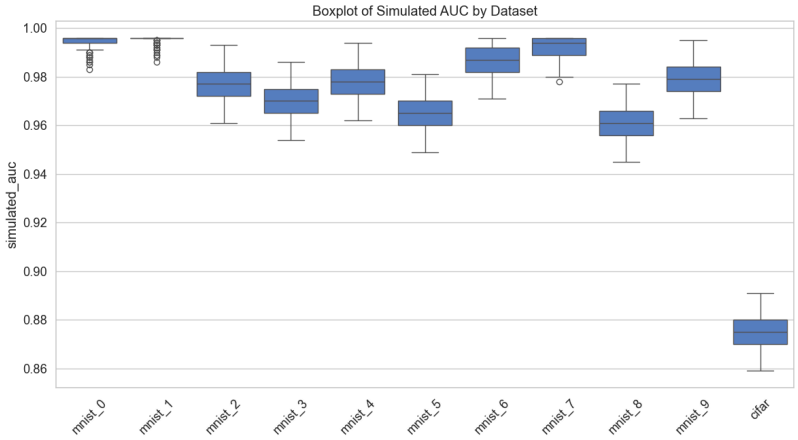

Figure 12: Boxplot of Simulated AUC

This classic statistical visualization shows the spread, median, and potential outliers in the simulated AUC scores. It offers a clean summary of variability and highlights any unusually high or low values.

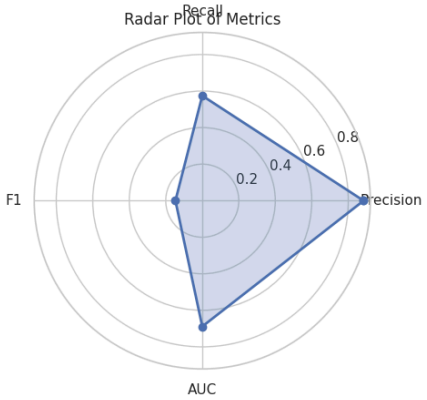

Figure 13: Radar Plot of Top Simulated AUCs

Radar plots allow comparison of the top-performing simulations across multiple metrics or configurations. It provides a quick glance at which layers or parameters excel across different axes.

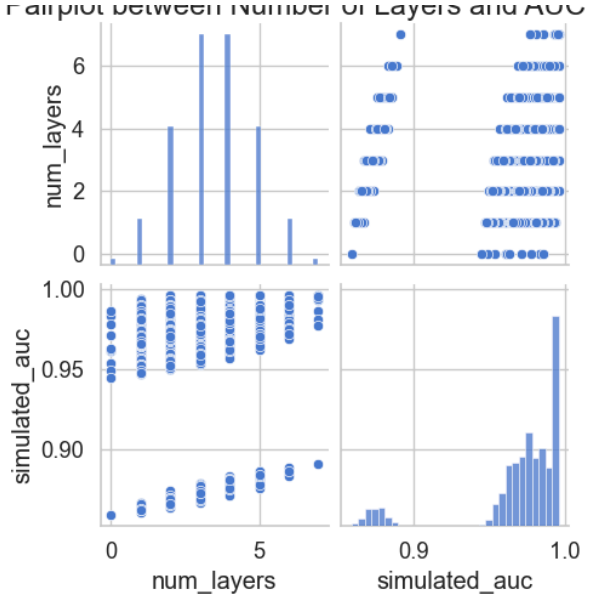

Figure 14: Pairplot of Layers vs AUC

The pairplot offers a matrix of scatter plots for each pair of variables, revealing potential correlations or interactions between layers and AUC. It is especially useful for exploratory analysis to spot trends or outliers.

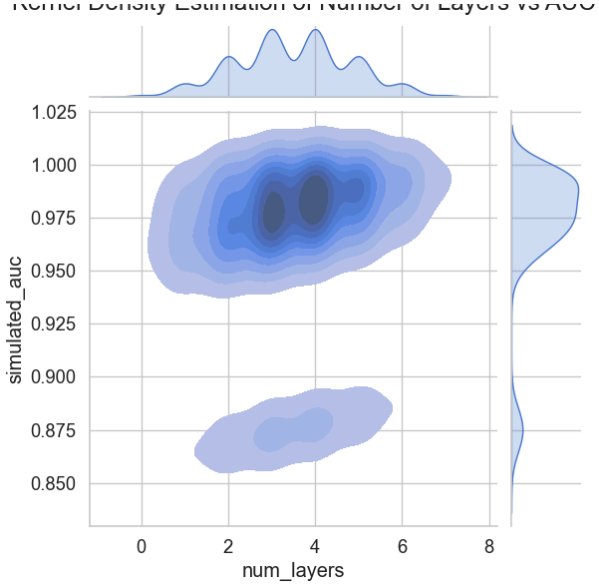

Figure 15: KDE Jointplot

This jointplot uses kernel density estimation to show the probability density of AUCs alongside scatter plots. It illustrates both the concentration of values and their distribution, offering a more nuanced view than simple scatterplots.