UNICAD: A Unified Approach for Attack Detection, Noise Reduction and Novel Class Identification

Alvaro Lopez Pellcier, Kittipos Giatgong, Yi Li, Neeraj Suri, Plamen Angelov
Lancaster University, UK
{a.lopezpellicer, giatgong, y.li154, p.angelov, neeraj.suri}@lancaster.ac.uk

***Abstract*—As the use of Deep Neural Networks (DNNs) becomes pervasive, their vulnerability to adversarial attacks and limitations in handling unseen classes poses significant challenges. The state-of-the-art offers discrete solutions aimed to tackle individual issues covering specific adversarial attack scenarios, classification or evolving learning. However, real-world systems need to be able to detect and recover from a wide range of adversarial attacks without sacrificing classification accuracy and to flexibly act in unseen scenarios. In this paper, UNICAD, is proposed as a novel framework that integrates a variety of techniques to provide an adaptive solution.**

For the targeted image classification, UNICAD is able to provide accurate image classification while still handling unseen scenarios by detecting unseen classes and detecting and recovering adversarially attacked inputs. This has been achieved by leveraging Prototype and Similarity-based DNNs, along with denoising autoencoders. Our experiments performed on the CIFAR-10 dataset highlight UNICAD's effectiveness in adversarial mitigation and unseen class classification, outperforming traditional models.

***Index Terms*—Adversarial attacks detection, prototype-based deep neural networks, open set classification**

I. Introduction

Machine learning and deep neural networks (DNNs) are seeing increasing usage across a diversity of applications such as transportation, manufacturing, agriculture, and healthcare, offering enhanced efficiency, safety, and decision-making capabilities [1], [2]. Although highly efficient in tasks such as image classification [3]–[5] object detection [6]–[8], and natural language processing [9]–[11] this technology is exposed to a number of security risks that can be exploited by malicious entities, potentially resulting in harmful outcomes.

Recent research [12]–[16] has shown that deep neural network (DNN) models, crucial for computer vision applications amongst others, are vulnerable to adversarial attacks. These consist on inputs also called adversarial examples specially created to lead models to give incorrect results. Moreover, many machine learning models are inflexible to adapt to new, unseen data categories without the need of retraining and supervision.

Adversarial examples were first introduced by [12] involve small image alterations that deceive deep learning models. They are, in many cases, imperceptible to the human eye but can still lead to incorrect model outputs. First examples include Evasion Attack and Box-Constrained L-BFGS [17]. The transferability of these adversarial examples across different models was later explored in [13], [15], [16] and underlines further security risks.

Current approaches [12], [18], [19] have aimed to address challenges concerning adversarial attacks and flexibility on a discrete scenario-by-scenario basis. Systems that are robust to adversarial attacks often do so at the expense of reduced classification accuracy and fail to handle the unknown scenarios. While the opposite happens for systems designed for open set classification. However in a world where AI is expected to handle such complex tasks such as fully autonomous driving, a framework that provides a unified solution to maintain a robust performance against attacks, standard classification and the unanticipated scenarios is increasingly becoming a necessity.

In this paper, the UNICAD framework (Unified Framework for Adversarial Attack Detection, Noise Reduction, and Novel Class Identification) is proposed as the first integrated approach to these challenges. UNICAD offers a detection system based on similarity, leveraging DNNs feature extraction capabilities to compare incoming data against known prototypes capable of image classification and identifying any possible adversarial alterations or previously unseen (untrained on) classes leveraging drops in similarity. This is paired with a state-of-the-art denoiser layer, trained to restore data altered by adversarial interference while retaining its essential characteristics without the need for retraining.

The main contributions include:

- Novel attack detection supporting robustness to adversarial attacks for prototype and similarity based DNNs explained over Sections III/IV.
- Noise reduction via a novel denoising autoencoder schema depicted in Section III-D and Section IV.
- Novel class identification through drops in similarity, with the functionality explained over Sections III-E and Section IV.
- The unification of state-of-the-art techniques in similarity-based DNNs for attack detection [20], interpretable open set classifiers [21], and the proposed denoiser autoencoder into a novel architecture, UNICAD. This unified framework is depicted in Section III, with its performance demonstrated in Section IV.

This work is supported by the Engineering and Physical Sciences Research Council [Grant number:EP/V026763/1]

- Performance analysis of UNICAD with different backbone networks, including VGG-16 [22], DINOv2 [23] against current state-of-the-art models, in Section IV.

II. Related Work

Adversarial attacks can be created in different forms, including text [9]–[11], audio [24]–[26], and images [13], [15], [16]. This paper, is focused on image domain attacks, which can be digital (adding adversarial noise to images) or physical (alterations in the real world) [27]–[29]. The most common digital attacks include FGSM [13], PGD [30], DDN [31], and CARLINI & WAGNER [16].

A. Adversarial Attacks in Image Domain

For this study, digital attacks in the image domain, specifically FGSM, PGD, and CARLINI & WAGNER, are the primary focus due to their prevalence and can be described as follows:

- Fast Gradient Sign Method (FGSM): FGSM employs the $L\infty$ metric to create image perturbations as $\hat{x} = x - \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$, leveraging $\nabla_x J$ as the cost function gradient and ϵ as a limiting factor [13].
- Projected Gradient Descent (PGD): Extending FGSM, PGD aims to maximize loss in DNNs, employing L_2 and $L\infty$ norms. It is formulated as $\min \rho(\theta)$, where $\rho(\theta) = E_{(x,y)\sim D}[\max L(\theta, x+\delta, y)]$ [30].
- Carlini & Wagner Attack: The approach seeks to find the minimum perturbation size which still produces missclasification. It uses $\min ||\rho||_p + c \cdot f(x+\rho)$, s.t. $x + \rho \in [0,1]^m$ [16].

For illustrative purposes, an example of FGSM attack is provided by Fig. 1 where the attack can be easily seen by the human eye due to the large perturbation size. However effective attacks can easily be crafted to be almost imperceptible.

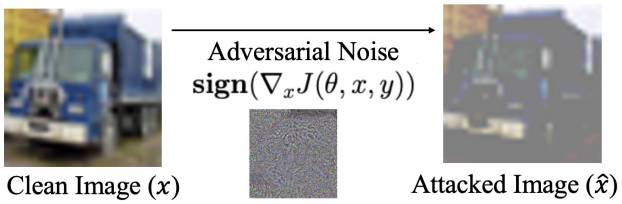

Fig. 1. Example of FGSM attack on CIFAR-10 data

B. Defences Against Adversarial Attacks

Defences against Adversarial Attacks include a variety of strategies categorized as preventive methods also known as proactive, which includes Adversarial Training [13], Input Data Pre-Processing [18], Model Ensemble [32], Model Regularization [12], Model Distillation [12], Probable Defences [33], Certification and Verification [34]. And reactive defences such as Detection Mechanisms [35], and Denoising/Reconstruction Defences [36]. Both proactive and reactive defenses often address only specific aspects of adversarial attacks and lack a holistic approach. Moreover Proactive defense often add too much computational overhead. Reactive defences offer a way to detect [35] or recover attacked images through denoising.

Denoising defences are generally implemented using Denoising Autoencoders (DAEs) [37], [38], which are essential in countering adversarial attacks due to their image recovery advantage. However, they are often proposed as a preprocessing step, which adds computational overhead and frequently hinders overall classification accuracy, especially when clean images are processed. This highlights the need for more integrated solutions that can offer a balanced functionality with consistent robustness across various scenarios, without imposing significant computational costs.

C. Novel Class Identification Techniques

While techniques like xClass [21], Deep SVDD [39], and the Adaptive Class Augmented Prototype Network [40] offer effective solutions in their respective areas, they often operate in isolation. This singular focus on specific aspects of the broader challenge limits their practical applicability in dynamic and unpredictable real-world environments. The discrete nature of these solutions underscores the demand for a more holistic approach. A unified framework, such as UNICAD, that seamlessly integrates these different functionalities, could provide a comprehensive solution, ensuring both the reliability and adaptability required for practical applications in diverse settings.

Overall, current state-of-the art offers discrete solutions for image classification, adversarial attack prevention or recovery and novel class classification. However as shown, the non unified use of these techniques bring significant drawbacks that hinders its reliability for practical applications. This calls for unified solutions that can provide a balanced functionality with consistent robustness.

III. Proposed UNICAD Architecture

Given the intricate and evolving nature of adversarial threats, along with the limitations in existing defense mechanisms, UNICAD framework is proposed. UNICAD distinguishes itself by not only assimilating various state-of-the-art techniques but by integrating them into a cohesive, innovative, and efficient novel architecture. This framework represents a paradigm shift from traditional approaches, maintaining together the strengths of similarity-based attack detection from [20], advanced noise reduction via a novel DAE in layer D, and novel class identification inspired by [21].

UNICAD's design goes beyond just merging existing methods. It improves these strategies, adding unique elements like the Denoising Layer (Layer D) and the Attack Decision Layer (Layer E). These new components enable UNICAD to classify images—whether they're clean, non-novel, or not attacked—more efficiently than conventional systems that depend heavily on pre-processing defenses. This approach not only saves computational resources but also makes UNICAD more responsive to adversarial challenges and new class types.

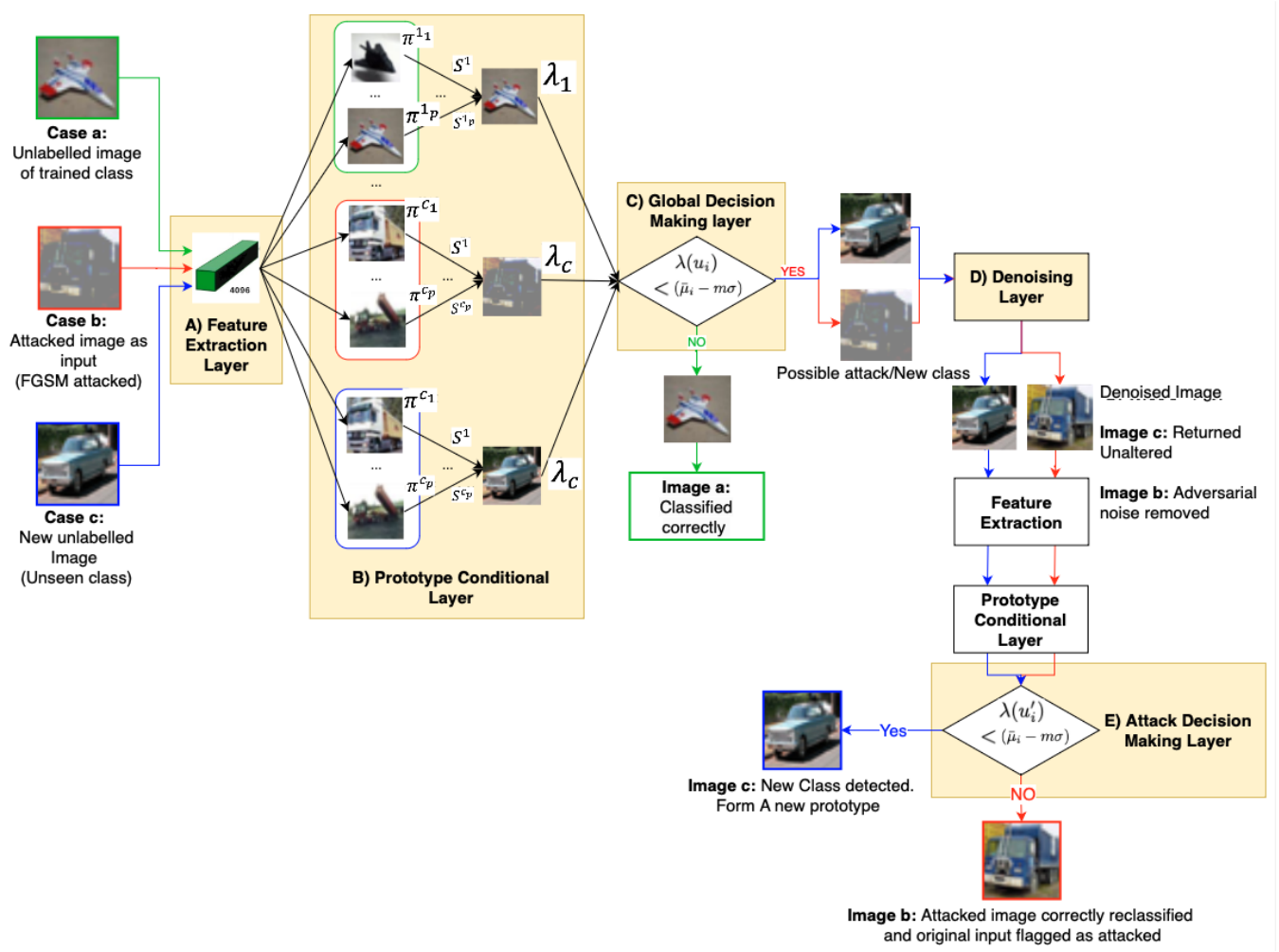

Fig. 2. UNICAD overall architecture. Layers A,B,C,D and E are represented. The 3 scenarios. (a) Image of a trained class, (b), attacked image, and (c) unseen class are represented. Each would be imputed separately and pass through the different layers as illustrated

At its heart, UNICAD is built for strong adversarial attack detection, effective noise reduction with denoising autoencoders, and precise recognition of novel classes using similarity-based neural networks. The framework uses a variation of the Sim-DNN approach [20] for detecting attacks, employing prototype comparison to spot adversarial intrusions. UNICAD enhances this capability by adding mechanisms to respond to detected attacks and broadening its range to include the identification and handling of unknown classes.

A crucial feature of UNICAD is its ability to detect shifts in data trends or concept drifts. It does this by closely monitoring changes in similarity scores between new data and preestablished prototypes from the training phase. If the similarity score drops below a certain threshold, a possible deviation from the norm is signaled, which could be an adversarial attack or a new class.

Upon detecting such anomalies, UNICAD directs the data to its Denoising Layer, where any adversarial noise is removed. The cleaned images are then reassessed within the framework. If a denoised image is then consistent with an existing class, indicating no further similarity drop, the original image is flagged as attacked but correctly classified, effectively removing the initial adversarial threat. In cases where the similarity drop persists, UNICAD interprets this as an indication of a new class and accordingly creates a new prototype for this unidentified category. This tailored, class-specific approach underlines UNICAD's adaptability, novelty and wide-ranging applicability

The overall architecture of UNICAD is illustrated in Fig. 2 as a layered design comprising of:

A) The Feature Extraction Layer
B) The Prototype Conditional Probability Layer
C) The Global Decision Making Layer
D) The Denoising Layer
E) The Attack Decision Making Layer

A. *Feature Extraction Layer*

The UNICAD's feature extraction layer drawn from [20] is tasked with determining the necessary feature space (dimensionality) for the algorithm. Due to the adaptable design of

UNICAD, features from various sources as network inputs can be incorporated. Notable feature sources for UNICAD include: i) convolutional neural networks [41], ii) residual neural networks like Resnet [42], iii) Transformers-based models [43], and iv) combinations of multiple sources (ensemble) [44]. In this study, two different methods are explored: VGG–16 [22] due to its high efficiency in this domain [45] and for direct comparison with related work [20], [21], and DINOv2 [23] due to its outstanding performance in classification tasks and grow in popularity.

The training dataset for UNICAD is denoted as $x = \{x_1, \ldots, x_N\} \in \mathbb{R}^n$, with corresponding class labels $y_1, \ldots, y_C \in \{1, \ldots, C\}$. Here, $N = N_1 + \ldots + N_C$ represents the total number of training samples, n is the feature count (dimensionality), and C is the number of classes in the dataset. In this paper, $n = 4096$ when VGG-16 is used and $n = 1536$ when DINOv2 is used for feature extraction. The most representative samples in the dataset are selected as class-specific prototypes $\pi \in P \subset X$. These prototypes facilitate a reasoning process based on the similarity (proximity in feature space) between a data sample and a prototype [46]. In this case, prototypes are identified as the local density peaks [47], essentially the most representative samples from the training set. Therefore, only a select few samples from the entire training set are chosen as prototypes, ensuring the system's efficiency and compatibility with a broad range of devices.

Consequently, M_j represents the total number of prototypes for class j, with $M_j = |P_j|$ and the total M being the sum of M_j across all classes: $M = \sum_{j=1}^{C} M_j$. In this approach, more than one prototype is considered for each class, so $M_j > 1$ for all j. Any new data sample, $x \in \mathbb{R}^n$, is associated with the nearest prototype from the sets $P_1, P_2, \ldots, P_C$; collectively $P = P_1 \cup P_2 \cup \ldots \cup P_C$. The label, L, is assigned as follows:

$$L(x) = \underset{x \in X, \pi \in P}{\arg\min}\, \mathrm{d}(x, \pi). \tag{1}$$

B. Prototype Conditional Probability Layer

This layer is primarily responsible for assessing similarity to established prototypes as well as the formation of prototypes in training and estimating conditional probabilities based on this similarity. It sets the stage for classification by identifying how closely new data aligns with known data patterns drawing from the previously extracted features.

Neurons are characterized by a similarity measure based on data density, denoted as S as outlined in [48]. This similarity measure is designed to assess the relative or exchangeable closeness of data points within the data space. The measure is articulated through a specific Cauchy equation, as presented in Eq. 2. Theoretically, this similarity metric, which employs either Euclidean or Mahalanobis distance, takes the form of a Cauchy function, as established in the theoretical framework of [48]. This is quantified by the following Cauchy equation:

$$S(x) = \frac{1}{1 + \frac{||x-\mu||^2}{||\sigma||^2}} \tag{2}$$

where x represents a data point, with μ and σ denoting the global mean and variance, respectively. This step evaluates the proximity of data samples within the feature space, utilizing either Euclidean or Mahalanobis distance as metrics.

An empirical estimation of conditional probabilities is then performed to estimate the probability of the input belonging to existing prototypes based on the distance measure. A multi-modal probability density function is employed where preset assumptions about the data distribution are avoided. The class probability given a data sample, x, is computed as:

$$p(y|x) = \frac{\sum_{i=1}^{M} N_i S(x)}{\sum_{i=1}^{M} N_i \int_{-\infty}^{\infty} S(x)dx} \tag{3}$$

integrating the similarity measure over the entire data space for a refined class probability estimation.

Lastly, prototype association is performed. Class-specific prototypes, distinguished by their peak similarity values, are independently identified. The association of data samples to these prototypes is determined by the equation:

$$j^* = \underset{i=1..N; j=1..M}{\arg\min} ||x_i - \pi_j||^2 \tag{4}$$

A sophisticated assessment of each input is ensured by thus methodology reliant on the proximity of data points to the strategically determined prototypes.

C. Global Decision Making Layer

This is where the actual classification takes place. Based on the information processed in the Prototype Conditional Probability Layer, this layer makes the final decision on whether an input belongs to an existing class, is an instance of a new class, or is an adversarial attack. This is achieved by employing the recursive mean $\bar{\mu_i}$ of a parameter λ, formulated as follows:

$$\bar{\mu_i} = \frac{i-1}{i}\bar{\mu_i} - 1 + \frac{1}{i}\lambda_i\bar{\mu_1} \quad \text{where} \quad \lambda_1 = \bar{\mu_1}. \tag{5}$$

where λ is the degree of similarity between a new data sample and the nearest prototype as described in [20].

The m-σ rule is applied to detect potential attacks. This rule can be depicted through an inequality condition:

$$\begin{aligned} \text{IF} \quad & \lambda(u_i) < (\bar{\mu}_i - m\sigma) \\ \text{THEN} \quad & (\mathrm{u}_i \in \text{ Potential new attack or rew class and} \\ & \mathrm{u}_i \text{ forwarded to denoising layer}) \\ \text{ELSE} \quad & (\mathrm{u}_i \in \text{ Classification Label}) \end{aligned} \tag{6}$$

where u_i is the unlabeled input.

If this condition is met, it suggests that the system has recognized a divergence or a new data concept, distinct from the established data patterns used to generate the prototypes. If not, it indicates no significant change in the data concept, allowing the algorithm to continue with its standard classification process. This mechanism enables UNICAD to adaptively respond to new data and effectively identify potential adversarial attacks or unseen scenarios.

D. Denoising layer

The Denoising layer is comprised of a denoising autoencoder trained to remove adversarial noise and at the same time leave clean input images unaltered. Inputs are only passed to this layer if eq. 6 is met The key distinguishing elements of this autoencoder are its novel loss function, the training setup that is demonstrated to work not only on clean images and images attacked using the kind of attack it has been trained on but also work on images attacked with different attack algorithms as well, and its location within the framework. Although similar papers using denoising autoencoders for adversarial attacks may suggest they should be used as a preprocessing technique [19], by placing this layer between the global decision making layer and the attack decision layer in a prototype based system, UNICAD is able to flag images as attacked, which could be leveraged to learn from the different existing and new types of attack in a real world scenario with the interpretability advantages that it provides.

The proposed denoising autoencoder is trained on the original clean images and images attacked using FGSM with $\epsilon = 0.3$ and $\epsilon = 0.03$, the choice of FGSM under different configurations has been chosen to improve trasnferability across different untrained types of attacks.

Its loss function, described in eq. (10) is a linear combination of mean squared error (MSE) loss defined as:

$$\mathcal{L}_{\text{MSE}}(y, y') = \frac{1}{M}\sum_{j=1}^{M}(y_j - y'_j)^2 \tag{7}$$

where y is the vector of original pixel values, y' is the vector of reconstructed pixel values by the autoencoder, M is the total number of pixels in the image, and y_j and y'_j are the individual pixel values of the original and reconstructed images, respectively.

The Structural Similarity Index Measure (SSIM) loss defined as:

$$\mathcal{L}_{\text{SSIM}}(x, x') = 1 - \frac{(2\mu_x\mu_{x'} + C_1)(2\sigma_{xx'} + C_2)}{(\mu_x^2 + \mu_{x'}^2 + C_1)(\sigma_x^2 + \sigma_{x'}^2 + C_2)} \tag{8}$$

where μ_x and $\mu_{x'}$ are the average values of the original and reconstructed images, σ_x^2 and $\sigma_{x'}^2$ are the variances, $\sigma_{xx'}$ is the covariance, and C_1, C_2 are constants used to stabilize the division with weak denominator. The SSIM index is a value between -1 and 1, where 1 indicates perfect similarity.

And Feature-based Loss defined as:

$$\mathcal{L}_{\text{Feat}}(f_\theta(x), f_\theta(x')) = 1 - \frac{\phi(f_\theta(x)) \cdot \phi(f_\theta(x'))}{\|\phi(f_\theta(x))\| \cdot \|\phi(f_\theta(x'))\|} \tag{9}$$

where f_θ is the autoencoder with parameters θ, ϕ represents the feature extractor, x is the input image and x' is the reconstructed image. The loss is calculated as one minus the cosine similarity between the feature representations of x and x', promoting alignment in the feature space.

Consequently, the combined loss function of the autoencoder, given a set of training samples, is defined as:

$$\begin{aligned}\mathcal{L}_{\text{comb}}(x, x') = &\omega_{\text{MSE}} \cdot \mathcal{L}_{\text{MSE}}(x, x') \\ &+ \omega_{\text{SSIM}} \cdot \mathcal{L}_{\text{SSIM}}(x, x') \\ &+ \omega_{\text{Feature}} \cdot \mathcal{L}_{\text{Feat}}(\psi(x), \psi(x'))\end{aligned} \tag{10}$$

where $\psi(x)$ represents the feature vector obtained from the feature extractor (VGG-16 or DINOv2 in the context of this paper), and $\omega_{\text{MSE}}, \omega_{\text{SSIM}}, \omega_{\text{Feature}}$ are the weights corresponding to the Mean Squared Error, Structural Similarity Index Measure, and Feature-Based loss components, respectively.

Hence, the training formulation for the autoencoder can be defineed as follows:

$$\theta^* = \arg\min_\theta \left[\frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{L}_{\text{comb}}(x_i, x'_i) + \mathcal{L}_{\text{comb}}(\hat{x}_i, x'_i)\right)\right] \tag{11}$$

where θ^* represents the optimal parameters for the autoencoder. $\hat{x}_i$ indicates the i-th adversarial image. $\mathcal{L}_{\text{comb}}$ is the combined loss function. N is the total number of training samples.

This training approach is crucial for enhancing the model's capability to generalize across both clean and adversarially perturbed inputs. By simultaneously optimizing for clean and attacked images, the autoencoder learns a more robust representation, improving its efficacy in real-world scenarios where input integrity cannot be guaranteed. This methodology ensures not only the fidelity of image reconstruction. It allows for images of unknown classes to be returned unedited both visually and in their latent space representation which is crucial for them to be classified as new class in the attack decision making layer and for attacked images to be reclassified correctly, therefore neutralizing the attack.

The autoencoder's architecture depicted in Fig. 3 is meticulously crafted, featuring an encoder that incrementally compresses the input using a sequence of convolutions, batch normalization, and pooling layers, interspaced with residual blocks. These blocks are instrumental in retaining information across the network's depth. The decoder is tasked with upscaling the condensed representation to reconstruct the original image. Through a series of transposed convolutions and the application of non-linear activation, such as ReLU and Sigmoid, the decoder learns to reverse the encoding effectively.

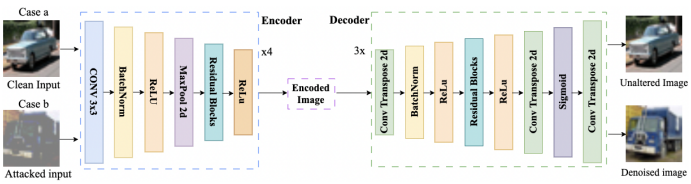

Fig. 3. Proposed Denoising Autoencoder Architecture. Case a, clean image, which is outputted unaltered. Case b attacked image, which is outputted without adversarial noise

This autoencoder has shown state-of-the-art results which have been displayed in Section IV. A visual example of its

performance both in attacked images and clean images is also shown in Fig. 3.

E. Attack Decision Making Layer

Reconstructed images are forwarded to this last layer. Here the reconstructed input is fed forward into the framework once more. After the reconstructed input is processed by layers A, and B, a decision about the suspected input is taken as follows:

$$\begin{aligned} \text{IF} \quad & \lambda(u_i') < (\bar{\mu}_i - m\sigma) \\ \text{THEN} \quad & (\text{u}_i \in \text{ New class.} \\ & \text{Form a new prototype}) \\ \text{ELSE} \quad & (\text{u}_i \in \text{Correct Classification Label and} \\ & \text{original input flagged as attack}) \end{aligned} \tag{12}$$

where u_i' the reconstructed input.

It is important to note that inputs flagged as attacked could in the future be used as prototypes of adversarial attacks to develop evolving attack detectors and attack classifiers with visual representations for human interpretability.

IV. Results and discussion

A. Dataset

A variety of attack scenarios have been performed to validate the robustness of the UNICAD framework. Experiments have been performed using the standard CIFAR-10 [49] datasets given their popularity as a benchmark and their complexity in matching real-world situations. For training, the CIFAR-10 training dataset was used, which has 50000 training images, 5000 images per class. The overall training formulation has been performed as follows:

B. Experimental Configuration

For layer, A, when VGG-16 has been chosen as Feature Extractor, a VGG-16 pretrained on ImageNet has been Fine-Tuned for 10 epochs into CIFAR-10, and last FC layer has been used as Feature Extractor. When DINOv2 was used, no Fine-Tuning was performed and the last FC was chosen as FE.

The proposed DAE in layer D was trained for 20 epochs as described in Section III-D. The feature extractor chosen for layer A was chosen for Feature-Based loss in the autoencoder (eq. 9).

Layer B was trained to find prototypes with CIFAR-10, taking an average of 0.17% of the images to form the necessary prototypes and an average training time of 27.09 seconds. Layers C and E do not require training. This highlights the training efficiency and adaptability of the core framework.

The CIFAR-10 validation dataset was later used for validation which cointains 10000 different images, 1000 images per class.

The experimental setup for unseen class detection is as follows: UNICAD and methods used for comparison were trained in CIFAR-10 classes 0-8. Hence class 9 was left unseen. The training steps are the same as described in the previous paragraph, but with one class dropped. This means, 45000 images were used in training, 5000 images per class still. For validation, all images from class 9 were used (6000).

It is important to note that class 9 correspond to truck images, and class 0 are automobile, so some error is expected as prototypes from truck images and vehicle images are close both visually and on the latent space. Further investigation on the similarity of different classes in the latent space and its relevance is suggested as future work.

The metric considered to evaluate unseen class detection rate is as follows:

$$\text{Detection}(\%) = \frac{TP + TN}{TP + FP + TN + FN} \times 100, \tag{13}$$

C. Results

The ablation study is out of scope of this paper. However, to confirm the effectiveness of each contribution, Table I, shows the difference in capabilities between UNICAD, and the main competitor models: XClass [21], SimDNN [20] and DDSA [19]

TABLE I
COMPARISON OF DIFFERENT METHODS' CAPABILITIES

Method	Unseen Class Detection	Attack Detection	Attack Recovery
XClass [21]	✓	×	×
simDNN [20]	×	✓	×
DDSA [19]	×	×	✓
UNICAD (ours)	✓	✓	✓

Table II shows the performance of the denoising autoencoder part of the denoising layer using VGG-16 and DINOv2 as the backbone for Feature-based loss (eq. 9) in a clean image setting and against FGSM, PGD and C&W attacks at different settings. The performance of VGG-16 and DINOv2 under this attacks with no defence has been provided along with results of current state-of-the-art DAEs, namely DDSA [19]. Accuracy has been evaluated on CIFAR-10 testing dataset for each scenario. Accuracy values displayed are measured in percentage, and it is measured from classification of the reconstructed input images in each scenario by the chosen classifiers, VGG-16 or DINOv2 , depending on the feature backbone of the network. Results demonstrate that the proposed denoiser autoencoder works better than current state of the art, with DINOv2 performing best as the backbone for Feature-based loss as expected due to its standalone performance. It is also very remarkable that Clean accuracy after reconstruction drops by less than 1% in all cases compared to current state-of-the-art denoisers where that drop is much more pronounced. This is due to the novel combined loss function that preserves both latent space and visual integrity of images. Robustness against no defence is also stated with consistently over 80% results in the defence approach compared to the performance of classifiers where no defence is implemented with attacks managing to drop their accuracy to as low as 0% in the worst case scenarios.

TABLE II
COMPARISON OF DIFFERENT MAINSTREAM IMAGE LASSIFIERS AND DENOISERS AGAINST OUR PROPOSED DAE FOR LAYER D IN DIFFERENT SCENARIOS

Model	Clean	PGD $\boldsymbol{\varepsilon = 0.01}$	PGD $\boldsymbol{\varepsilon = 0.3}$	FGSM $\boldsymbol{\varepsilon = 0.01}$	FGM $\boldsymbol{\varepsilon = 0.03}$	FGM $\boldsymbol{\varepsilon = 0.3}$	C&W L_2 norm
Ours (VGG-16 for Feat loss)	93.10	83.58	85.82	82.67	78.10	83.26	83.2
Ours (VGG19 for Feat loss)	93.10	83.58	85.82	82.67	78.10	83.26	83.4
Ours (DINOv2 for Feat loss)	97.24	**90.55**	**92.71**	**90.3**	**88.87**	**92.24**	**93.7**
DDSA [19]	81.4	60.00	61.10	63.00	61.00	57.10	36.70
VGG-16 No defence	92.0	0.05	32.12	31.06	22.56	0.11	0.00
DINOv2 No defence	**97.63**	56.8	0.7	58.9	17.3	12.4	0.8

TABLE III
COMPARISON OF DIFFERENT SCENARIOS CONSIDERING UNICAD FRAMEWORK AND MAINSTREAM APPROACHES

Scenario	UNICAD (VGG-16 FE)	UNICAD (DINOv2 FE)	xClass (VGG-16 FE)	DDSA [19]	VGG-16 No defence	DINOv2 No defence
Clean	80.86	92.93	80.86	81.4	92.0	**97.63**
PGD $\boldsymbol{\varepsilon = 0.01}$	74.81	**77.77**	49.3	60.00	0.05	56.8
PGD $\boldsymbol{\varepsilon = 0.3}$	72.63	**82.29**	14.2	61.10	32.12	0.7
FGSM $\boldsymbol{\varepsilon = 0.01}$	70.01	**77.37**	49.0	63.00	31.06	58.9
FGM $\boldsymbol{\varepsilon = 0.03}$	64.9	**76.02**	47.1	61.00	22.56	17.3
FGM $\boldsymbol{\varepsilon = 0.3}$	73.09	**81.10**	15.6	57.10	0.11	12.4
C&W L_2 norm	73.2	**79.33**	0.6	36.70	0.00	0.8
Unseen Class detection	62.30	**83.38**	62.30	0.00	0.00	0.00

Table III shows the performance of the overall framework with VGG-16 and DINOv2 as the backbone feature extractors for the Feature extraction layer. In this table classification accuracy of attacked images using FGSM, PGD and C&W have been included as well as clean image classification and unseen class classification along with resutls of current state-of-the-art DAEs, namely DDSA [19], and xClass [21]. Accuracy is presented in percentage and accuracy for the UNICAD framework has been calculated at the global decision making layer for clean images and at the attack decision making layer for attacked and unseen images.

Accuracy is notably lower than that of the standalone denoising autoencoder, attributed to the new configuration's ability to detect unseen classes and provide interpretability via prototype-based classification. Despite this, the results remain robust, with DINOv2's classification accuracy dropping $<$ 5% compared to its undefended setup. In clean accuracy, the performance surpasses DDSA when using VGG-16, with DINOv2 yielding the best outcomes, particularly in clean accuracy, thanks to its superior feature extraction capabilities.

xClass [21] has been included for comparison due to its unseen class detection capability, despite not being robust against adversarial attacks. It is matched by UNICAD in classification and new class identification. UNICAD excels with consistant $>$ 70% accuracy in the presence of adversarial attacks, outperforming mainstream methods and affirming its theoretical robustness against such attacks and effectiveness in detecting new classes.

V. CONCLUSION AND FUTURE WORK

Overall, UNICAD, a novel framework designed to increase the robustness of deep neural networks against adversarial attacks and enable the detection of novel, unseen classes has been presented. Different experiments using CIFAR-10 have been shown which demonstrated not only UNICAD's capability to withstand various adversarial attacks but also its ability to identify and classify new, unseen classes with high accuracy. The integration of a novel state-of-the-art denoising autoencoder within the denoising layer significantly increased the framework's robustness to adversarial attacks. Furthermore the limitations of the current state-of-the-art and the impending need for a unified framework has been demonstrated throughout the paper.

Another key element of UNICAD is its prototype-based architecture, which has contributed to both the interpretability of the system and its ability to adapt to new data scenarios without extensive retraining. This feature is especially crucial in dynamic environments where models frequently encounter data outside their initial training distribution such as Autonomous Systems.

While UNICAD has marked a significant advancement in defensive strategies against adversarial attacks and unseen class detection, the ongoing arms race in adversarial machine learning is acknowledged. Future work will be aimed to further enhance the scalability and efficiency of UNICAD, particularly focusing on optimizing the denoising layer to provide better adaptability in a landscape of changing and evolving adversarial attacks.

REFERENCES

[1] T. B. Sheridan, "Human–robot interaction: status and challenges," *Human Factors*, vol. 58, no. 4, pp. 525–532, 2016.

[2] B. Siciliano and O. Khatib, Eds., *Springer handbook of robotics*, 2nd ed. Springer Cham, jul 2016.

[3] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, Y. Liu, H. Pham, X. Dong, T. Luong, C.-J. Hsieh, Y. Lu, and Q. V. Le, "Symbolic discovery of optimization algorithms," *arXiv preprint arXiv: 2302.06675*, 2023.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv: 1409.1556*, 2014.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, may 2017.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893 vol. 1.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *arXiv preprint arXiv: 1506.02640*, 2016.

[8] R. Girshick, "Fast r-cnn," *arXiv preprint arXiv: 1504.08083*, 2015.

[9] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," *arXiv preprint arXiv: 1804.07998*, 2018.

[10] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv: 1712.06751*, 2018.

[11] M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer, "Adversarial example generation with syntactically controlled paraphrase networks," *arXiv preprint arXiv: 1804.06059*, 2018.

[12] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv: 1312.6199*, 2013.

[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv: 1412.6572*, 2014.

[14] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroSP)*, 2016, pp. 372–387.

[15] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," 2016, pp. 2574–2582.

[16] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," pp. 39–57, 2017.

[17] R. Fletcher, *Practical Methods of Optimization*, 2nd ed., . John Wiley Sons, Ed. Hoboken, NJ, USA: Wiley, 2013.

[18] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1378–1387.

[19] Y. Bakhti, S. A. Fezza, W. Hamidouche, and O. Déforges, "Ddsa: A defense against adversarial attacks using deep denoising sparse autoencoder," *IEEE Access*, vol. 7, pp. 160 397–160 407, 2019.

[20] E. Soares, P. Angelov, and N. Suri, "Similarity-based deep neural network to detect imperceptible adversarial attacks," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2022, pp. 1028–1035.

[21] P. Angelov and E. Soares, "Detecting and learning from unknown by extremely weak supervision: exploratory classifier (xclass)," *Neural Comput Applic*, vol. 33, pp. 15 145–15 157, 2021.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[23] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: learning robust visual features without supervision," *arXiv preprint arXiv: 2304.07193*, 2023.

[24] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 1–7.

[25] X. Yuan, Y. Chen, Y. Zhao, Y. Long, X. Liu, K. Chen, S. Zhang, H. Huang, X. Wang, and C. A. Gunter, "Commandersong: A systematic approach for practical adversarial voice recognition," *arXiv preprint arXiv: 1801.08535*, 2018.

[26] L. Schönherr, K. Kohls, S. Zeiler, T. Holz, and D. Kolossa, "Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding," *arXiv preprint arXiv: 1808.05665*, 2018.

[27] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv: 1607.02533*, 2016.

[28] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial example," *arXiv preprint arXiv: 1707.07397*, 2017.

[29] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1528–1540.

[30] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv: 1706.06083*, 2017.

[31] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, "Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4317–4325.

[32] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Mach. Learn.*, vol. 51, no. 2, p. 181–207, may 2003.

[33] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," *arXiv preprint arXiv: 1705.01320*, 2017.

[34] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," *arXiv preprint arXiv: 1810.12715*, 2018.

[35] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 506–519.

[36] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, 2005, pp. 60–65 vol. 2.

[37] D. Meng and H. Chen, "Magnet: A two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 135–147.

[38] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 1096–1103.

[39] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4393–4402.

[40] R. Li, J. Zhong, W. Hu, Q. Dai, C. Wang, W. Wang, and X. Li, "Adaptive class augmented prototype network for few-shot relation extraction," *Neural Networks*, vol. 169, pp. 134–142, 2024.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[43] I. O. Sigirci, H. Ozgur, and G. Bilgin, "Feature extraction with bidirectional encoder representations from transformers in hyperspectral images," in *2020 28th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2020, pp. 1–4.

[44] C. Jia and W. He, "Enhancerpred: a predictor for discovering enhancers based on the combination and selection of multiple features," *Scientific reports*, vol. 6, no. 1, pp. 1–7, 2016.

[45] D. Theckedath and R. Sedamkar, "Detecting affect states using vgg16, resnet50 and se-resnet50 networks," *SN Computer Science*, vol. 1, no. 2, pp. 1–7, 2020.

[46] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," pp. 39–57, 2017.

[47] P. Angelov and E. Soares, "Towards explainable deep neural networks (xdnn)," *Neural Networks*, vol. 130, pp. 185–194, 2020.

[48] P. Angelov and X. Gu, *Empirical Approach to Machine Learning*. Springer, 2019.

[49] A. Krizhevsky, "Learning multiple layers of features from tiny images," pp. 32–33, 2009.