# Denoising Autoencoder-Based Defensive Distillation as an Adversarial Robustness Algorithm Against Data Poisoning Attacks

**Article** *in* ACM SIGAda Ada Letters · June 2024

DOI: 10.1145/3672359.3672362



**3 authors**, including:

Bakary Badjie
University of Lisbon
**4** PUBLICATIONS **46** CITATIONS

SEE PROFILE

Antonio Casimiro
University of Lisbon
**128** PUBLICATIONS **1,273** CITATIONS

SEE PROFILE

# Denoising Autoencoder-Based Defensive Distillation as an Adversarial Robustness Algorithm Against Data Poisoning Attacks

***Bakary Badjie, José Cecílio, António Casimiro***
*LASIGE, Departamento de Informática, Faculdade de Ciências da Universidade Lisboa, Lisboa; email: {bbadjie, jmcecilio, casim}@ciencias.ulisboa.pt*

## Abstract

*Deep neural networks (DNNs) have demonstrated promising performances in handling complex real-world scenarios, surpassing human intelligence. Despite their exciting performances, $DNNs$ are not robust against adversarial attacks. They are specifically vulnerable to data poisoning attacks where attackers meddle with the initial training data, despite the multiple defensive methods available, such as defensive distillation. However, defensive distillation has shown promising results in robustifying image classification deep learning (DL) models against adversarial attacks at the inference level, but they remain vulnerable to data poisoning attacks. This work incorporates a data denoising and reconstruction framework with a defensive distillation methodology to defend against such attacks. We leverage a denoising autoencoder (DAE) to develop a data reconstruction and filtering pipeline with a well-designed reconstruction threshold. We added carefully created adversarial examples to the initial training data to assess the proposed method's performance. Our experimental findings demonstrate that the proposed methodology significantly reduced the vulnerability of the defensive distillation framework to a data poison attack.*

*Keywords: Deep Neural Network, Denoising Autoencoder, Defensive Distillation, Adversarial attacks and Robustness, Data Poisoning*

## 1 Introduction

The necessity of $DNNs'$ resilience against adversarial attacks has shown a growing concern as their use in real-world safety-critical systems has increased exponentially. Adversarial attacks [1] are attempts to trick $DNNs$ by making subtle alterations to the input data $x$. The alterations are usually designed to be imperceptible to humans but can detrimentally impact the $DNN's$ ability to make accurate decisions. Several defense strategies have been proposed to overcome this problem, including defensive distillation, which has successfully defended $DNNs$ from input perturbations $\epsilon$ in run-time settings [2]. Nevertheless, one of the drawbacks of the defensive distillation framework is that it remains susceptible to data poisoning attacks, in which adversaries aim to impair the model's performance by either maliciously altering the existing training data or inserting erroneous data entries into it. These harmful data entries could be carefully crafted to be close to the model's decision boundary, bypassing the countermeasures offered by defensive distillation. This study presents a novel method for robustifying a distilled network against data poisoning attacks by integrating a denoising autoencoder (DAE) [3] in the defensive distillation pipeline. Defensive distillation involves training two $DNNs$, the instructor model and the student model, such that the knowledge of the former is transferred into the latter, making it robust to adversarial examples and previously unseen input [1, 4]. A $DAE$ is a type of $DNN$ trained in an unsupervised setting to learn a latent representation of input data $x$, enabling it to reconstruct distorted data back to its original form. In the training phase, it is exposed to a perturbed version of $x$ and learns to reconstruct it while filtering out noise. The aim is minimizing the reconstruction loss between the $x$ and reconstructed input $x_r$ as much as possible [3].

This paper is motivated by the fact that the instructor model is not immune to data poisoning adversarial attacks. However, the student model has more latitude to reject variations in the input space $X$ because it leverages the "distilled" version of the training data, where training data is labeled with the soft labels or probability vector $F(X)$ obtained by the increase in the softmax temperature parameter $T$ of the instructor model [2]. Thus, minimizing the instructor model's susceptibility to data poisoning attacks is pivotal for developing a reliable and robust distilled $DNN$. To achieve this, we designed a $DAE$ to remove noise and reconstruct poisonous samples within the training dataset. The defensive distillation method already offers a strong resilience foundation in the test phase; combined with a $DAE$, the resilience against data poisoning adversarial attacks is significantly strengthened in the training phase. Moreover, our strategy considers several adversarial attack aspects, such as the attacker's access to the training data and trial-and-error techniques to access the model gradient, making it a more effective defense mechanism against such attacks.

We used the fast gradient sign method (FGSM) [5] and the iterative-fast gradient sign method (I-FGSM) [6] to evaluate the effectiveness of the proposed algorithm. We used these adversarial examples generation algorithms to perturb a proportion of the training data.

The results in Figure 2 show that the proposed approach enables the detection and reconstruction of the majority of poisonous inputs in the training data and significantly improves the robustness of the instructor network against data poisoning attacks. The perturbations used by the $FGSM$ and $I-FGSM$ algorithms are so small that the resulting poisonous data within the training set is imperceptible to humans. The proposed approach offers a more potent protection mechanism against adversarial poisoning attacks by combining defensive distillation with a $DAE$. This enables the distillation algorithm to significantly mask or lower the gradient around the training data and widen the search space that attackers need to explore in order to craft adversarial examples in the input space $X$. The proposed approach significantly reduces the limitations and susceptibility of the defensive distillation algorithm to data poisoning attacks.

Knowledge distillation was first proposed in [4]. The author aimed to lower the computational resources required to deploy large-scale $DNNs$ on resource-constrained devices such as smartphones. Therefore, they extrapolate the probability vector or class knowledge produced by the instructor network and use it to train small networks, reducing the network's scalability without compromising accuracy and allowing deployment on resource-constrained devices.

Statistically and experimentally, Papernot et al. [2] further explore this idea as a preventive measure against adversarial inputs. Using class knowledge from the instructor network and distilling it to the student network, the authors minimize the amplitude of the student network's gradients that attackers required access to generate adversarial examples in the input space $X$. The authors do not necessarily transfer knowledge from large-scale networks to smaller ones; they instead use two networks of the same architecture. Moreover, they demonstrate that models developed via defensive distillation are less susceptible to adversarial inputs in runtime settings. Additionally, according to Goldblum et al., [7], while effective neural networks (NN) may be developed by transferring information from the teacher model to the student model, they may still be exposed to strong adversarial attacks in run-time scenarios. To address this limitation, they developed an Adversarially Robust Distillation (ARD), which involves creating small $NNs$ and distilling their robustness in a larger network. The authors say this strategy performs better than the conventional defensive distillation benchmark. Previous works where the defensive distillation technique was leveraged include [8, 9, 10]. Although these strategies successfully prevent evasion attacks (run-time attacks), none of the previous authors have considered addressing adversarial poison attacks in the context of knowledge distillation. The end goal of this paper is motivated by this limitation, which makes the defensive distillation algorithm vulnerable to data poisoning attacks. As a result, we used the $DAE$ as a filter in the knowledge distillation pipeline. Yue et al. [11] demonstrated that $DAEs$ are highly helpful in spotting and reconstructing contaminated images in the training data manifold. They apply this approach to identify and mitigate adversarial poison attacks in the federated learning setting. Other works in which $DAE$ is used as a filter against poison data include [12, 13].

# 2 Mitigation of adversarial poison attacks

## 2.1 Defensive Distillation

As mentioned in the previous section, the technique used in defensive distillation involves training two $DNNs$ that are similar in structure; the instructor network $F\phi_1$ and the student network $F\phi_2$. In this work, a standard $NN$ training procedure is used to train $F\phi_1$ using the original dataset, with an increase in the softmax temperature $T$ to 5 degrees.

The softmax layer normalizes an output vector or logits $Z(X)$ of $F\phi_1$ final hidden layer into a probability vector $F(X)$ more closely aligned with the data manifold's uniform statistical distribution. This assigns a probability value to each class of the dataset for each input $x$. A specific neuron in the softmax layer that corresponds to a class indexed by $i \in 0..N-1$, where $(N = numberofclasses)$ calculates element $i$ of the probability vector given by;

$$F(X) = \left[ \frac{e^{Z_i \frac{(X)}{T}}}{\sum_{l=0}^{N-1} e^{Z_l \frac{(X)}{T}}} \right]_{i \in 0..N-1} \tag{1}$$

The $F(X)$ is used as a label for the inputs in the training data to train $F\phi_2$ with the same temperature value as used in $F\phi_2$. $T$ is usually reset to its default value of 1 during testing so that the distilled network can produce more discrete output. This also discourages overly confidence in the distilled network's output and improves its generalization to new inputs.

The motivation for knowledge distillation stems from the fact that the semantic characteristics of the data that $F\phi_1$ learn are encoded in both its class probability vectors and learned weight parameters. In addition to only transferring a sample's correct class, extrapolating class knowledge from these vectors and utilizing it to label inputs in the training data for training $F\phi_2$ with the same or different architecture will supplement it with extra information about each class. Another advantage of using the class probability vector as a label to train $F\phi_2$ is that it multiplies its features that must be altered to create a successful attack. Attackers would need to modify many features in the input space $X$ to be able to drive network $F\phi_2$ into incorrect predictions, making it considerably more challenging to construct adversarial inputs. Also, it reduces variations in $X$ and masks or lessens the model's gradient, which an attacker is required to exploit in order to find adversarial examples around $x$. However, it is worth noting that standard defensive distillation is still vulnerable to data poison attacks, which motivates us to

incorporate it with an $DAE$, and the distilled network mostly losses precision.

### 2.2 Denoising Autoencoder

The goal of the $DAE$ is to learn a compressed representation of $x$, correct any abnormalities in the data, and reconstruct it back to its original, undistorted state with the help of the latent vector $h$. The design comprises an encoder network $f_e$ and a decoder network $f_d$, represented as a composition of $i-th$ encoding layer $f_e^{(i)}$ and decoding layer $f_d^{(i)}$, respectively. The first layer of $f_e$ receives the erroneous form of data $x^*$ from the input space $X$ and translates it to a lower-dimensional latent vector $h^{(i)}$. The subsequent layer maps $h^{(i)}$ from the previous layer to generate more compressed latent features. This process continues until the last layer, which represents the final lower-dimensional latent representation of the input $h$. Conversely, the last layer of $f_d$ takes in $h$ and maps it to the next layer in a backward pass direction until the first layer, which maps the reconstructed features to the original input space, producing the reconstructed output image $x_r$.

During training, the $DAE$ learns to minimize the reconstruction error between the initial unperturbed input data $x$ and output reconstructed data $x_r$. This is accomplished by the network's ability to learn the mapping function resilient to adversarial perturbation $\epsilon$ and rebuild the $x^*$ to their original form without compromising the most significant intrinsic statistical properties of $x_r$. The latent representation of the $i-th$ layer can be expressed as;

$$h^{(i)} = f_e^{(i)}(h^{(i-1)}) \wedge h^{(0)} = x \tag{2}$$

where;

$h^{(i)}$ = latent representation at the $i-th$ encoder layer.
$i$ = the layer index of the encoder
$f_e^{(i)}$ = composition of individual encoder functions

While the final lower-dimensional latent representation of the input is expressed as;

$$h = f_{e1}(f_{e1}(f_{ei}(x_i))) \tag{3}$$

where;

$h$ = latent representation vector
$f_{e1}(f_{e1}(f_{ei}$ = first decoder layer, second up to the last layer.
$x$ = input of the encoder

The reconstructed output at each $i-th$ layer is expressed as;

$$x_r^{(i)} = f_d^{(i)}(x_r^{(i-1)}) \tag{4}$$

where;

$x_r^{(i)}$ = the reconstructed output at the $i-th$ decoder layer
$f_e^{(i)}$ = composition of individual decoder functions

The final reconstructed output of the decoder network is computed as;

$$x_r = f_{di}(f_{d2}(f_{d1}(h_i))) \tag{5}$$

where;

$h$ = latent representation vector
$f_{di}(f_{e2}(f_{e3}$ = last decoder layer, up to the first layer.
$x$ = input of the encoder

### 2.3 Adversarial Attacks and Robustness

Adversarial attacks are types of malicious data modification that seek to deceive $ML$ models in their decision-making [1]. A wide range of $ML$-based applications are susceptible to adversarial attacks. In contrast, adversarial robustness describes the model's capacity to maintain its expected performance in the presence of malicious interruptions or adversarial attacks [14]. A model is deemed resilient if it correctly classifies $x \in X$ within or outside a range of perturbation sets defined in $X$. Robustness is a crucial characteristic for $ML$ models since it guarantees their dependability in real-world applications where the $x$ may be noisy or purposefully altered to cause misclassification [15]. In our experiment, we used the following parameters to generate adversarial perturbations for $FGSM$ and $IFGSM$; $epsilon_fgsm = 0.01, epsilon_ifgsm = 0.01, alpha = 0.01, num_iterations = 10$. The epsilon $\epsilon$ determines the magnitude of the perturbation needed to trigger the model's sensitivity to changes in the image's pixel values. The alpha $\alpha$ parameter controls the step size of the perturbations in each iteration of the $IFGSM$ attack. $num_iterations = 10$ indicates that the $IFGSM$ will be applied at every 10 iteration.

## 3 Autoencoder-based Defensive Distillation Approach

Following the methodology used in [10], the student model in our study is built to be uncertain about its prediction when a new input $x_n$ is statistically different from the training set (i.e., discouraging overconfidence in its classification). We used the Kullback–Leibler (KL) divergence [16] to quantify the statistical differences between the $x_n$ and the ground truth. If their statistical difference is more significant than the $P-value$, the network becomes uncertain about its prediction and returns a null value. This ambiguity is evaluated during "dropout inference," a technique in which the entire network is used for prediction while turning off the dropout layer. This approach allowed the model to offer uncertainty estimates for the predictions and estimate Bayesian inference. Analyzing the instructor model's predictions yields the uncertainty measurements needed to train the student model.

As done in [2], the $F(X)$ generated by the teacher model is not always sufficient to reduce the gradient surrounding the training data and handle variation in $X$, which would prevent sophisticated attacks from successfully compromising the student model. Relying only on the probability vector will make

the student model classify inputs based on traditional softmax probability estimation without considering the uncertainty of its prediction.

While developing the $DAE$ model, we used the "$OPTUNA$ library," a Pytorch-based hyperparameter optimization toolkit, to select the appropriate hyperparameters. The $OPTUNA$ uses clever methods like Bayesian optimization (BO) to search the hyperparameter dimension effectively. With a few trials, it selects the best set of hyperparameters suitable to improve the model's performance. We generate adversarial inputs from a portion of the Germany Traffic Sign Recognition Benchmark (GTSRB) dataset while the remaining portion is used as clean data. Mean squared error (MSE) loss is used to evaluate the reconstruction error between $X$ and $x$, where the goal is to minimize the statistical difference between the $x^*$ and $x$. Furthermore, we use the random weight initialization strategy to avoid issues with vanishing or exploding gradients and promote quicker and more efficient convergence during training. In the training phase, both $X$ and $x^*$ are fed to the $DAE$ as inputs to enable it to learn the statistical correlations between them and also to learn the noisy pattern in the input. During backpropagation, the loss function is minimized while the weights are updated using a stochastic gradient descent (SGD) optimization algorithm. The effectiveness of the trained $DAE$ is evaluated on a different test dataset containing both clean and distorted images. A specified reconstruction threshold is set, which serves as a yardstick or decision line to verify the integrity of each data point after reconstruction. The images whose reconstruction error is above the threshold are termed adversarial inputs and are subsequently discarded before reaching the teacher model.

## 4 Results

Many works in the literature that use $DAE$ randomly select the reconstruction threshold, which can lead to inconsistencies in the evaluation process and introduce additional constraints in interpreting the level of similarity between the input and reconstructed image, making it difficult to understand the behavior of the network. There is also a risk of selecting a sub-optimal threshold value that does not optimize the network's performance.

To avoid this scenario, in this work, an initial reconstruction threshold of $0.015$ was chosen randomly at the beginning of the $DAE$ network's training process. This made updating the reconstruction threshold to $0.003$ easier during the evaluation phase. We used the "$inferthreshold$" function as a decision threshold to update the initial threshold. The "$inferthreshold$" function calculates the values inferred from the percentage of instances identified as adversarial in the evaluation dataset. This function is to be found in the "$alibi-detect$" outlier/adversarial detection library. Figures 1 and 2 show how the $DAE$ network performed on the evaluation datasets, which comprised instances of only adversarial inputs and a combination of both adversarial and clean inputs, respectively.

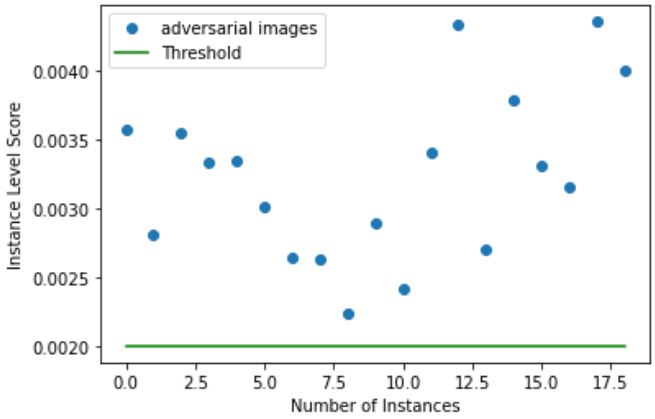


**Figure 1: Illustration of the DAE's performance on only adversarial inputs during reconstruction**

Upon reconstruction, images with reconstruction errors greater than the updated threshold value $(0.003)$ are labeled as adversarial inputs since the $DAE$ network cannot restore them to their original state. Reconstructed-clean images are those with reconstruction errors below the updated threshold value. The identified adversarial inputs are subsequently eliminated before passing the dataset to the instructor model in the distillation stage. As shown in Figure 2, the green dotted circles indicate the reconstructed clean images, whereas the red dotted ones represent the adversarial images that the $DAE$ network could not reconstruct. According to our experimental results, the average reconstruction error produced by the $DAE$ network using the evaluation dataset was $0.008190$, indicating that our designed $DAE$ network functioned satisfactorily. Figure 3 shows the average reconstruction and validation loss at each epoch.

The result also demonstrates that $IFGSM$ devises more robust adversarial examples than $FGSM$ because the latter only takes into account the gradient of the loss function with respect to the target $x \in X$ only once and then makes a single step along the path of the gradient around $x$ to create an adversarial example, whereas the former takes into account the gradient of the loss function with respect to the target $x \in X$ at every iteration and keeps updating the perturbation in $X$ in the direction of the gradient's sign around the target $x \in X$ which results in generating more robust adversarial examples.

The reconstructed images are sent to the instructor model, trained with a softmax temperature of $T = 5$. This produces soft labels to annotate the new dataset, which we then use to train the student model (distilled network). The instructor model's total accuracy throughout training is $99.89\%$, with an average loss of $0.11\%$. The student model also performed well by correctly classifying adversarial inputs in the test dataset with an accuracy rate of $76.09\%$. Although this precision might not be particularly excellent in other attack scenarios, the design is incredibly robust to $IFGSM$ and $FGSM$ adversarial attacks.

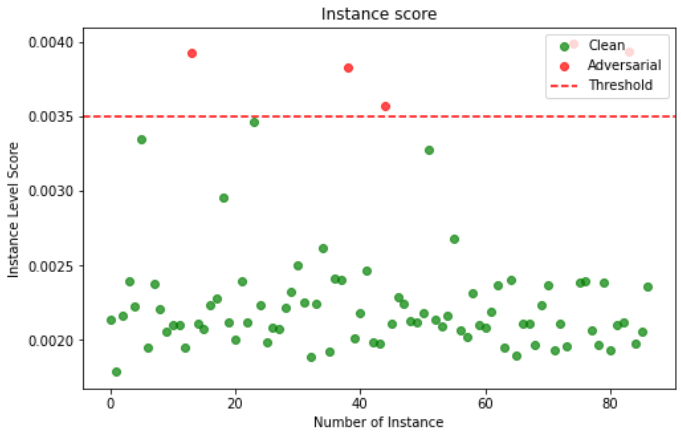


**Figure 2: Illustration of the DAE's performance on the evaluation dataset during reconstruction.**

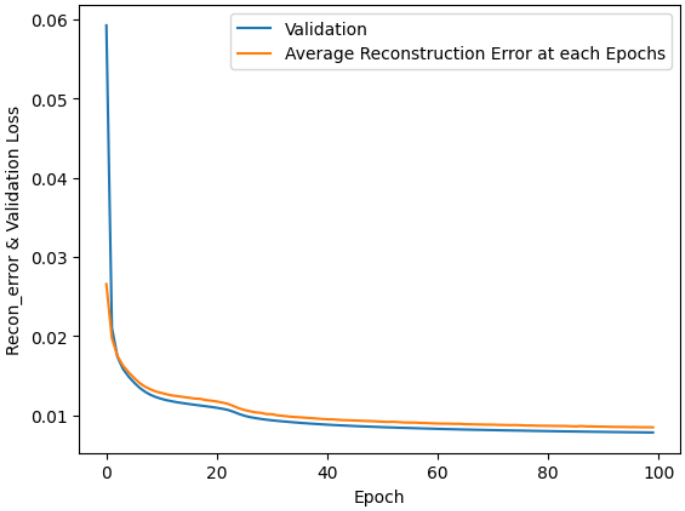


**Figure 3: Illustration of the DAE's performance on the evaluation dataset during reconstruction.**

## 5 Conclusion

The use of a $DAE$ network in combination with defensive distillation has proven to be an effective method in enhancing the robustness of a distilled network against both poisoning and evasion adversarial attacks. It was also shown that the "alibi-detect" library helped the $DAE$ network detect and get rid of adversarial inputs in the training dataset. By updating the reconstruction threshold during the evaluation phase, the $DAE$ network was able to accurately identify inputs as either adversarial or clean. The use of $IFGSM$ resulted generating in stronger adversarial examples than $FGSM$, highlighting the importance of considering the gradient of the loss function with respect to the $x \in X$ at every iteration. The student model, trained with soft labels produced by the teacher model, demonstrated excellent robustness against adversarial attacks. Based on the results, the proposed approach could be a valuable addition to the arsenal of techniques available to improve the reliability in $ML$-driven systems.

## 6 Acknowledgments

This work was supported by the LASIGE Research Unit (ref. UIDB/00408/2020 and ref. UIDP/00408/2020), and by the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197 (VEDLIoT project).

## References

[1] Y. Chen, M. Zhang, J. Li, and X. Kuang, "Adversarial attacks and defenses in image classification: A practical perspective," in *2022 7th International Conference on Image, Vision and Computing (ICIVC)*, pp. 424–430, IEEE, 2022.

[2] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE symposium on security and privacy (SP)*, pp. 582–597, IEEE, 2016.

[3] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020.

[4] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *stat*, vol. 1050, p. 9, 2015.

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

[6] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[7] M. Goldblum, L. Fowl, S. Feizi, and T. Goldstein, "Adversarially robust distillation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 3996–4003, 2020.

[8] M. Kuzlu, F. O. Catak, U. Cali, E. Catak, and O. Guler, "Adversarial security mitigations of mmwave beamforming prediction models using defensive distillation and adversarial retraining," *International Journal of Information Security*, pp. 1–14, 2022.

[9] E.-C. Chen and C.-R. Lee, "Ltd: Low temperature distillation for robust adversarial training," *arXiv preprint arXiv:2111.02331*, 2021.

[10] N. Papernot and P. McDaniel, "Extending defensive distillation," *arXiv preprint arXiv:1705.05264*, 2017.

[11] C. Yue, X. Zhu, Z. Liu, X. He, Z. Zhang, and W. Zhao, "A denoising autoencoder approach for poisoning attack detection in federated learning," *IEEE Access*, vol. 9, pp. 43027–43036, 2021.

[12] A. Kascenas, N. Pugeault, and A. Q. O'Neil, "Denoising autoencoders for unsupervised anomaly detection in brain mri," in *International Conference on Medical Imaging with Deep Learning*, pp. 653–664, PMLR, 2022.

[13] M. Tripathi, "Facial image denoising using autoencoder and unet," *Heritage and Sustainable Development*, vol. 3, no. 2, p. 89, 2021.

[14] Y. Deng, X. Zheng, T. Zhang, C. Chen, G. Lou, and M. Kim, "An analysis of adversarial attacks and defenses on autonomous driving models," in *2020 IEEE international conference on pervasive computing and communications (PerCom)*, pp. 1–10, IEEE, 2020.

[15] F. Wang, C. Zhang, P. Xu, and W. Ruan, "Deep learning and its adversarial robustness: A brief introduction," in *HANDBOOK ON COMPUTER LEARNING AND INTELLIGENCE: Volume 2: Deep Learning, Intelligent Control and Evolutionary Computation*, pp. 547–584, World Scientific, 2022.

[16] F. Raiber and O. Kurland, "Kullback-leibler divergence revisited," in *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, pp. 117–124, 2017.