Understanding the Effect of Data Augmentation in Self-supervised Anomaly Detection

Jaemin Yoo¹, Tiancheng Zhao², Leman Akoglu¹

¹ Heinz College of Information Systems and Public Policy, Carnegie Mellon University
² School of Architecture, Carnegie Mellon University
jaeminyoo@cmu.edu, tianchen@andrew.cmu.edu, lakoglu@andrew.cmu.edu

Abstract
Self-supervised learning (SSL) has emerged as a promising alternative to create supervisory signals to real-world tasks, avoiding extensive cost of careful labeling. SSL is particularly attractive for unsupervised problems such as anomaly detection (AD), where labeled anomalies are costly to secure, difficult to simulate, or even nonexistent. A large catalog of augmentation functions have been used for SSL-based AD (SSAD), and recent works have observed that the type of augmentation has a significant impact on performance. Motivated by those, this work sets out to put SSAD under a larger lens and carefully investigate the role of data augmentation in AD through extensive experiments on many testbeds. Our main finding is that self-supervision acts as a yet-another model hyperparameter, and should be chosen carefully in regards to the nature of true anomalies in the data. That is, the alignment between the augmentation and the underlying anomaly-generating mechanism is the key for the success of SSAD, and in the lack thereof, SSL can even impair (!) detection performance. Moving beyond proposing another SSAD method, our study contributes to the better understanding of this growing area and lays out new directions for future research.

1 Introduction

Machine learning as a field has made tremendous progress in creating models that can learn from carefully labeled data. However, the cost of high-quality labeled data is a major bottleneck for the future of this supervised learning paradigm. Most recently, self-supervised learning (SSL) has emerged as a promising alternative to internally generate some kind of supervisory signal to solve a task. In essence, the unsupervised task is transformed into a supervised problem by auto-generating the labels. This new paradigm has had great success in advancing NLP (e.g., BERT [Devlin et al. 2019], XLM-R [Conneau et al. 2020], etc.) and has helped excel at various computer vision tasks (Goyal et al. 2021). Today, SSL is arguably the key toward “unlocking the dark matter of intelligence” (LeCun and Misra 2021).

SSL is particularly attractive for unsupervised learning problems such as anomaly detection (AD), as labeled data is (i) already rare/scarce or altogether nonexistent, (ii) costly to obtain by simulation (e.g. sandbox simulations in cybersecurity), or (iii) nontrivial to simulate in the face of unknown anomalies. As a result, the literature has seen a recent surge of SSL-based AD (SSAD) techniques (Golan and El-Yaniv 2018; Golan and El-Yaniv 2019; Bergman and Hoshen 2020; Li et al. 2021; Qiu et al. 2021; Cheng et al. 2021; Shenkar and Wolf 2022). While these employ various different formulations (e.g., augmentation prediction, contrastive learning, etc.; see Sec. 2), as a common thread, they aim to learn how to distinguish normal data (or inliers) from self/auto-generated anomalies, the latter being input data (typically inlier-only) transformed via a what-is-called augmentation function.

Interestingly, a large catalog of different augmentation functions have been used in the SSAD literature – including rotation, flipping, cut-and-paste, blurring, masking, color jittering, etc. as well as “cocktail” augmentations as in GEOM (Golan and El-Yaniv 2018) and SimCLR (Chen et al. 2020) (See Sec. 3) – without any clear guidance on which one(s) to use on a new AD task. Alarmingly, there exist evidences in the literature that the choice of augmentation has significant impact on the outcome. As a motivating example, Li et al. (2021) have shown that on industrial defect detection (e.g. scratch on wood), a local augmentation such as CutPaste...
works significantly better than a global one such as random rotation (90.9 vs. 73.1 AUC on avg., see their Table 1). In contrast, on semantic AD tasks where inliers and outliers are different object images from CIFAR-10 (Krizhevsky, Hinton et al. 2009), the reverse holds true (69.4 vs. 91.3 AUC, see their Sec. 5.4). They conclude that different type of AD tasks needs different augmentation designs.

Motivated by the above, this paper sets out to put SSAD under a larger lens and investigate the following questions:

1. **Success and failure (Sec. 4):** Does augmentation improve the vanilla model in all cases? Which augmentation functions perform best on certain types of anomalies?

2. **How augmentation works (Sec. 5):** How does augmentation function? What changes does it incur on error distributions and data representations of a detector $f$?

Through extensive experiments on controlled testbeds as well as in-the-wild datasets, we find that self-supervision is not a “magic stick” for unsupervised AD, and rather remains to be a (yet-another) model hyperparameter—the choice of which heavily influences performance. Specifically, we observe that the alignment between the augmentation function $aug$ and anomaly-generating function $gen$ is essential to the success of SSAD, and it can even impair performance if the alignment is poor (depicted in Fig. 1). The alignment is determined not only by the type of $aug$, but also by its continuous parameters that affect the amount of augmentation.

Our observations are supported by various previous works on SSAD (details in Sec. 6). For example, Golan and El-Yaniv (2018) showed that geometric transformations such as rotation or flipping are suitable for detecting semantic class anomalies in image data, while Li et al. (2021) designed new augmentation functions to detect local defects in images of various objects. These works present partial evidence to our observed phenomena, while our work is the first formal comprehensive study to provide comprehensive evidence.

Our work aims to better understand existing approaches in a growing research field instead of proposing a new one. Being a systematic measurement study, this work is similar to those in other domains (Erhan et al. 2010; Mesquita, Jr., and Kaski 2020; Ruff et al. 2020) that also aim to critically review or understand growing areas. Three notable works significantly better than a global one such as random rotation (90.9 vs. 73.1 AUC on avg., see their Table 1). In contrast, on semantic AD tasks where inliers and outliers are different object images from CIFAR-10 (Krizhevsky, Hinton et al. 2009), the reverse holds true (69.4 vs. 91.3 AUC, see their Sec. 5.4). They conclude that different type of AD tasks needs different augmentation designs.

Motivated by the above, this paper sets out to put SSAD under a larger lens and investigate the following questions:

1. **Success and failure (Sec. 4):** Does augmentation improve the vanilla model in all cases? Which augmentation functions perform best on certain types of anomalies?

2. **How augmentation works (Sec. 5):** How does augmentation function? What changes does it incur on error distributions and data representations of a detector $f$?

Through extensive experiments on controlled testbeds as well as in-the-wild datasets, we find that self-supervision is not a “magic stick” for unsupervised AD, and rather remains to be a (yet-another) model hyperparameter—the choice of which heavily influences performance. Specifically, we observe that the alignment between the augmentation function $aug$ and anomaly-generating function $gen$ is essential to the success of SSAD, and it can even impair performance if the alignment is poor (depicted in Fig. 1). The alignment is determined not only by the type of $aug$, but also by its continuous parameters that affect the amount of augmentation.

Our observations are supported by various previous works on SSAD (details in Sec. 6). For example, Golan and El-Yaniv (2018) showed that geometric transformations such as rotation or flipping are suitable for detecting semantic class anomalies in image data, while Li et al. (2021) designed new augmentation functions to detect local defects in images of various objects. These works present partial evidence to our observed phenomena, while our work is the first formal comprehensive study to provide comprehensive evidence.

Our work aims to better understand existing approaches in a growing research field instead of proposing a new one. Being a systematic measurement study, this work is similar to those in other domains (Erhan et al. 2010; Mesquita, Jr., and Kaski 2020; Ruff et al. 2020) that also aim to critically review or understand growing areas. Three notable advantages of such studies include: (i) put novel approaches in a growing area under a critical lens, (ii) shed light onto their working assumptions via carefully designed experiments, and (iii) pinpoint potential loopholes and misleading beliefs. We expect that our analyses will provide a better understanding of the role of SSL for AD and help steer the future directions of research on the topic.

2 **Notations and Related Works**

We first introduce notation and present recent approaches on self-supervised learning for anomaly detection (SSAD).

2.1 **Notations**

We define the problem of anomaly detection as follows. Let $gen : \mathbb{R}^d \rightarrow \mathbb{R}^d$ be an anomaly-generating function, and $p(x)$ and $p_{gen}(x)$ be the distributions of normal (or inlier) and anomalous (or outlier) data, respectively, where $x \in \mathbb{R}^d$. We define the problem of anomaly detection as follows. Let $gen(x) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ be an anomaly-generating function, and $p(x)$ and $p_{gen}(x)$ be the distributions of normal (or inlier) and anomalous (or outlier) data, respectively, where $x \in \mathbb{R}^d$. Given a set $\{x \mid x \sim p(x)\}$ of normal training data, the goal is to learn a score function $s : \mathbb{R}^d \rightarrow \mathbb{R}$ such that $s(x') > s(x)$ if $x'$ is anomalous and $x$ is normal. We denote by $D_{gen}$ a dataset whose anomalies are generated by $gen$.

We also define self-supervised anomaly detection (SSAD) as a training scheme of $s$ that involves a set $\mathcal{U}$ of augmentation functions. Each augmentation function $aug : \mathbb{R}^d \rightarrow \mathbb{R}^d$ modifies an example $x$ while preserving its semantic information, such as by adding Gaussian noise to each element of $x$. In most cases, the target of training is typically a neural network $f$, and the score function $s$ is defined on top of the data representations learned by $f$. We denote a network trained with $aug$ by $f_{aug}$. Fig. 2 illustrates the overall process of SSAD to train a detector network $f_{aug}$.

The main focus of our study is the alignment $d(aug, gen)$ between functions $aug$ and $gen$. However, it is not straightforward to formally define $d(\cdot, \cdot)$ in image data, where pixel-level distances are not matched with semantic or visual differences. Thus, we say that $aug$ and $gen$ are aligned if they create visually similar outputs or work in a similar fashion, as given by our categorization of augmentation functions in Sec. 5 and they are perfectly aligned if $aug = gen$.

2.2 **Generative Models**

Generative models learn the distribution of normal data and predict an unseen example as an anomaly if it does not follow the learned distribution. Traditional autoencoder models (Zhou and Paffenroth 2017; Chen and Konukoglu 2018; Zong et al. 2018), generative adversarial networks (Schlegl et al. 2017; Zenati et al. 2018ab) (Akçay, Abarghoei, and Breckon 2018), and flow-based models (Yu et al. 2021; Gudovskiy, Ishizaka, and Kozuka 2022; Rudolph et al. 2022) are popular categories of generative models.

Recent works (Cheng et al. 2021; Ye et al. 2022) propose generative models for SSAD in the framework of denoising autoencoders (DAE), which involve an augmentation function $aug$ to add noise to each input sample. Although traditional works (Vincent et al. 2008; 2010) used elementwise Gaussian noise or Bernoulli masking as $aug$, recent works have shown that more complex domain-specific augmentation such as random rotation in images is helpful to learn the
semantic information of data, achieving higher accuracy on anomaly detection (Cheng et al. 2021, Ye et al. 2022).

2.3 Augmentation Prediction

One popular approach for SSAD is to create samples with pseudo labels via multiple augmentation functions to learn a classifier. Given \( U = \{ \text{identity}, \text{aug}_1, \cdots, \text{aug}_N \} \), where identity \((x) = x\), the task is to conduct \((N+1)\)-way classification for predicting the function \( \text{aug} \in U \) given \( \text{aug}(x) \). In this way, a classifier \( f \) is trained to differentiate different augmentation functions and then used to generate better representations of \( x \) in one of its intermediate layers.

Many augmentation functions were proposed for anomaly detection in this sense, including geometric transformation (Golan and El-Yaniv 2018), patch rearranging (Wang et al. 2019), random affine transformation (Bergman and Hoshen 2020), local image transformation (Li et al. 2021), and learnable neural network-based transformation (Qu et al. 2021). Some of them adopt existing augmentation functions to the task of anomaly detection, while some propose new ones.

2.4 Contrastive Learning

Contrastive learning is similar to augmentation prediction in that multiple augmentation functions are used to compose an objective function. The main difference is that contrastive learning utilizes pairs of augmentation functions and learns an estimator to produce similar outputs for the positive pairs, while augmentation prediction considers each function \( \text{aug} \) as an independent label to predict as classification.

Many previous works (Tack et al. 2020, Sehwag, Chiang, and Mittal 2021) based on contrastive learning use SimCLR (Chen et al. 2020) to create a set of positive samples. That is, the objective function is to learn similar representations for raw data and those transformed by SimCLR. The difference between various works is how they compose the set of negative samples, whose representations are learned to be further from those of the positive ones. There is also a recent work that proposes a new augmentation function for contrastive learning on tabular data (Shenkar and Wolf 2022).

3 Experimental Setup

Model Among various models for anomaly detection, we adopt a denoising autoencoder (DAE) (Vincent et al. 2008, 2010) as the target model of experiments due to the following reasons. First, DAEs have been used widely in anomaly detection due to their robustness and generalizability (Cheng et al. 2021, Ye et al. 2022). Second, the training of DAEs is done simply by minimizing the reconstruction error without requiring additional regularizer terms. Third, DAEs support the vanilla version that works without augmentation, which is considered as the no-SSL base model for comparison. Finally, as DAEs learn to reconstruct the original sample from its corrupted (in our case, augmented) version, in effect they produce the counterfactual of an anomaly at the output, helping with the interpretation of anomalies at test/decision time. Detailed information on the model is in Appendix A.

Training and Inference The training of a DAE \( f \) is done by minimizing the following objective function:

\[
l(\theta) = \mathbb{E}_{x' \sim p_{\text{aug}}(x)} \left[ \| f(x'; \theta) - x \|^2_2 + \lambda \| \theta \|_2^2 \right],
\]

where \( \theta \) is the set of all learnable parameters of \( f \), \( \text{aug} \) is an augmentation function (or a noise function in the context of traditional works), \( p_{\text{aug}}(x) \) is a distribution of samples made by \( \text{aug} \), and \( \lambda \) is a strength regularization. The training of a vanilla (i.e. no-SSL) AE is done by employing the identity augmentation function \( \text{aug}(x) = x \) to Eq. (1).

After training a DAE, we use the mean squared error \( r(x) \) between an input \( x \) and the reconstructed one \( \hat{x} \) to compute an anomaly score. Specifically, given a test set \( \mathcal{T} \) containing both normal data and anomalies, we define the score \( s(x) \) of \( x \) as the percentile of \( r(x) \) in \( \mathcal{T} \). That is, \( x \) is predicted as an anomaly if \( r(x) \) is relatively high. We evaluate the DAE by measuring the area under the ROC curve (AUC) of scores.

Datasets Our experiments are conducted on two kinds of testbeds. The first is in-the-wild testbed, which is to select a single semantic class as normal and another class as anomalous. We include four different image datasets in this testbed: MNIST (LeCun et al. 1998), FashionMNIST (Xiao, Rasul, and Vollgraf 2017), SVHN (Netzer et al. 2011), and CIFAR-10 (Krizhevsky, Hinton et al. 2009). Since all datasets have 10 different classes, we have 90 pairs of normal and anomalous classes for each dataset. The second is a controlled testbed, where we adopt a known augmentation function as the anomaly-generating function \( \text{gen} \) to have the full control of anomalies. Given SVHN and CIFAR-10, we apply three \( \text{aug} \) functions with limited randomness and different characteristics from each other: CutOut, Flip, and Invert. We denote these datasets by SVHN-C and CIFAR-10C, respectively.

Augmentation Functions We study various types of augmentation functions, which are categorized into five groups. The bullet colors are the same as in Figs. [1] and [4].

- Geometric: Crop (Chen et al. 2020), Rotate, Flip, and GEOM (Golan and El-Yaniv 2018).
- Local: CutOut (Devries and Taylor 2017), Zhong et al. (2017), CutPaste and CutPaste-scar (Li et al. 2021).
- Elementwise: Blur (Chen et al. 2020), Noise, and Mask (Vincent et al. 2010).
- Color-based: Invert and Jitter (Chen et al. 2020).
- Mixed: SimCLR (Chen et al. 2020).

Geometric augmentations make global geometric changes to images such as by rotation or flipping. Local augmentations, in contrast, modify only a part of an image such as by erasing a small patch. Elementwise augmentations modify every pixel locally or individually. Color-based augmentations change color information, while mixed augmentations combine multiple categories of augmentation functions. Detailed information is given in Appendix B. Detailed information is given in Appendix B.

4 Success and Failure of Augmentation

First, we investigate when augmentation succeeds or otherwise fails, by comparing various DAEs (with augmentation) with the vanilla (no-SSL) AEs on our testbeds.
Wilcoxon Tests  We conduct each experiment for all possible pairs of classes and then run the paired Wilcoxon signed-rank test (Groggel 2000) between a DAE and the vanilla AE. For example, Fig. 1 shows the (pseudo)medians, 95% confidence intervals, and p-values as a result. The x-axis depicts the relative AUC compared with that of the vanilla AE. We consider that aug is beneficial (or disadvantageous) if the p-value is smaller than 0.05 (or larger than 0.95). We present only the main results, leaving details to Appendix C.

4.1 Controlled Testbed

First, we run experiments on the controlled testbed in which anomalies are generated by known augmentation functions. Figs. 1 and 3 show the results on gen:=CutOut. Each row summarizes the AUC values on 20 tasks across two datasets (CIFAR-10C and SVHN-C) and ten classes each. We have the following observations from the results.

Observation 1 Given $D_{\text{gen}}$, DAE $f_{\text{aug}}^{}$ outperforms $f_{\text{aug}}'$ if aug is better aligned with gen than aug' is.

Observation 2 DAE $f_{\text{aug}}^{}$ impairs the accuracy of vanilla $f$ if the alignment between aug and gen is poor in $D_{\text{gen}}$.

We show in Fig. 1 the relative AUC of DAE $f_{\text{aug}}$ with various choices of aug. The local augmentation functions that exhibit the best alignment with gen:=CutOut significantly improve the vanilla AE. On the other hand, the remaining functions make negligible changes or even cause significant decrease in AUC. This demonstrates the importance of the alignment between aug and gen, and shows that DAEs can perform worse than the baseline under poor alignment. Our observations are consistent with other choices of gen, specifically Flip and Invert, as shown in Appendix C.1.

It is notable that the augmentations of the same category (i.e. the same color) are located together in the figure, implying that similar augmentation functions have similar characteristics even though their specifics are different. For example, CutOut works best because it is the perfect match with gen, while CutPaste and CutPaste-scar still perform well since they also augment a small image patch.

Figure 3: Relative AUC on the controlled testbed where the value of $c \in [0.1, 0.9]$ in aug:=CutOut-c depicts the size of erased patches. The anomaly function gen:=CutOut has an average size of patches $\approx 0.2$. Detection works better when $c$ in aug is better aligned with that of gen. See Obs. 3.

Figure 4: (best in color) Relative AUC on the in-the-wild datasets. Color in augmentation names represents their categories. The (blue) geometric and (red) local augmentations improve the no-SSL baseline. See Obs. 4 and 5.

Observation 3 The alignment between aug and gen in DAE $f_{\text{aug}}$ is determined not only by the type of the function aug, but also by the amount of augmentation made by aug.

Fig. 3 compares multiple variants of CutOut-c, changing the width c of erased patches in augmented images. For example, CutOut-0.2 means that the width of erased patches is 20% of that of each image, making their area to 4%. Note that the original CutOut used as gen selects the patch width randomly in (0.02, 0.33), making an average of 0.19.

The figure shows that CutOut-c is better with smaller c, and it starts to decrease the base AUC when $c \geq 0.6$. This is because CutOut-c with small c achieves the best alignment with gen, which also removes small patches of average size $\approx 0.2$. Even though the type of aug is the same as gen, the value of c determines whether $f_{\text{aug}}$ succeeds or not.

4.2 In-the-Wild Testbed

Fig. 4 summarizes the results on 360 tasks across 4 datasets and 90 class pairs each, whose anomalies represent different semantic classes. Unlike in the controlled testbed, here the alignment between aug and gen is not known a priori. We have the following observations from the results, which are consistent also in individual datasets (see Appendix C.2).

Observation 4 Geometric aug performs best for datasets in which anomalies consist of different semantic classes.

Observation 5 SimCLR does not provide a meaningful improvement over the vanilla AE, although it is one of the most popular augmentation functions in the literature.

We observe from Fig. 4 that Rotate and other geometric augmentations perform well in semantic anomaly detection tasks. This implies that simple rotation achieves the best alignment between aug and gen, replicating the observation of a previous work (Li et al. 2021) introduced in Sec. 1. A plausible explanation is that many classes in those datasets, such as dogs and cats in CIFAR-10, are rotation-sensitive. Thus, the rotation by aug creates plausible samples outside the normal data distribution, giving $f_{\text{aug}}$ an ability to differentiate anomalies that may look like rotated normal images.

We support this claim by detailed analysis in Sec. 5.
Figure 5: Images from CIFAR-10C and SVHN, where the three rows represent original images and those reconstructed by the vanilla AE and DAE, respectively. (a, b) \( \text{aug}:=\text{CutOut} \) and \( \text{gen}:=\text{CutOut} \). (c, d) \( \text{aug}:=\text{Rotate} \), and the digits of 6 and 9 are normal and anomalous classes, respectively. The success and failure represent whether the images are assigned accurately to their true classes (normal vs. anomaly). DAEs preserve the original images (on the left), while applying the inverse augmentation \( \text{aug}^{-1} \) to anomalies (on the right), making them resemble normal ones with high reconstruction errors. See Obs. 6 and 7.

Local augmentations also work well, since changes in image patches make objects presented in different ways, affecting the semantic class information. Nevertheless, the amount of improvement is not as large as in geometric ones.

A notable observation is that SimCLR, which is one of the most popular augmentation functions in the literature, does not introduce a significant improvement to the base model. This is because SimCLR is designed to preserve the semantic information of images while making pixel-level changes, while what DAEs require for \( \text{aug} \) is to change the semantic information for better alignment with \( \text{gen} \).

5 How Augmentation Works

Next, we analyze how and why augmentation helps DAEs achieve better performance than the vanilla AE.

5.1 Case Studies

We visually inspect individual samples to observe what images DAEs reconstruct for normal versus anomalous inputs. Fig. 5 shows the results on CIFAR-10C and SVHN with different \( \text{aug} \). We have the following observations.

**Observation 6** Given an anomaly \( \text{gen}(x) \), a DAE \( f_{\text{aug}} \) approximates the inverse augmentation \( \text{aug}^{-1} \) if \( \text{gen} \) and \( \text{aug} \) are aligned well, satisfying \( f(\text{gen}(x)) \approx x \).

**Observation 7** Given a normal sample \( x \sim p(x) \), a DAE \( f \) approximates the identity function, satisfying \( f(x) \approx x \).

In Fig. 5 we see that given normal images, both models reconstruct similar images giving low reconstruction errors, but the vanilla AEs fail to predict them as normal since the errors are low also for the anomalous images. Given anomalies, the DAEs recover their counterfactual versions by applying the inverse augmentation function \( \text{aug}^{-1} \), increasing their reconstruction errors to become higher than those from the normal images. This makes DAEs achieve higher AUC than vanilla AEs. We derive the same observations from different \( \text{gen} \) functions on both testbeds (see Appendix D).

It is notable that the task of detecting digits 9 as anomalous from digits 6 is naturally aligned with the Rotate augmentation, since in effect, the DAE learns the images of rotated 6 to be anomalies. This alignment is an essential factor for the success of SSAD as we claim in Obs. 1 and 2.

5.2 Error Histograms

Next we study the effect of augmentation functions on the reconstruction error distributions of the normal samples and anomalies. Fig. 6 shows the histograms of errors on CIFAR-10C. We make the following observations from the results, which are consistent with different \( \text{gen} \) functions in the controlled and in-the-wild testbeds (see Appendix E).

**Observation 8** The overall reconstruction errors are higher in DAEs than in the vanilla AEs.

**Observation 9** The reconstruction errors increase in overall with the degree of change that augmentation employs.

We compare DAEs with three different \( \text{aug} \) functions and the vanilla AE in Fig. 6. The DAEs show higher reconstruction errors in general than the vanilla AE, providing more
right-shifted error distributions. This is because the objective function of DAEs involves a denoising term that increases the reconstruction errors for augmented data. The improved AUC of DAEs is the result of increasing the reconstruction errors for anomalies than those for normal data.

In Fig. 6 the amount of change incurred by an augmentation $aug$ increases from (6a) to (6d). The error distributions are shifted to the right accordingly, as augmented data become more distant from the normal data. Notably, the distribution for the true anomalies is more right-shifted in Fig. 6c than in Fig. 6d showing the higher AUC of 0.978, thanks to the better alignment between $aug$ and $gen$.

### 5.3 Embedding Visualization

Figs. 7 and 8 show embeddings on CIFAR-10C and CIFAR-10, respectively, learned by various augmentation functions. We have the following observations from the results.

**Observation 10** Embeddings from a DAE $f_{aug}$ for normal and anomalous data are separated when $aug$ makes global changes, whereas mixed when $aug$ makes local changes.

The first two and the last two plots in Fig. 7 exhibit different scenarios. In Figs. 7a and 7b there are separate clusters: one for normal data, and another for augmented and anomalous data. In Figs. 7c and 7d all data points compose a single cluster without a clear separation between the different categories. The difference between the two scenarios is driven by the characteristic of $aug$: Flip makes global changes in images, while CutOut affects only a part of each image.

**Observation 11** Augmentation can work even under imperfect alignment if the anomalies lie between the normal and augmented data in the embedding space.

Fig. 8 visualizes the result on CIFAR-10 with semantic anomalies and $aug:=Rotate$. The alignment between $aug$ and $gen$ is not perfect, which is shown by the separation between the augmented data and the anomalies in the embedding space; the augmented points make four separate clusters based on the four degree options of Rotate, which are $\{0, 90, 180, 270\}$. Nevertheless, the augmentation improves the vanilla AE, since the anomalies lie between normal and augmented data in the embedding space.

We also study the cases of different $gen$ functions in Appendix F, including both success and failure scenarios. The failure case of SimCLR creates an isolated cluster of augmented samples, which misguides the training of $f$.

### 6 Discussion

We present further discussion on our experiments and observations, in the context of prior related works.
Table 1: AUC of the GEOM model on CIFAR-10. The first four rows use the original GEOM augmentation, while the last four use CutPaste. The columns represent classes treated as normal at each experiment. Semantic means the semantic anomalies of different classes in the one-vs-rest scheme, while CutOut, Invert, and Flip mean anomalies constructed via augmentation as in the other experiments. GEOM model works better with a better alignment, consistent with our main Obs. 1 and 2.

| Augment | Anomaly | Air. | Auto. | Bird | Cat | Deer | Dog | Frog | Horse | Ship | Truck |
|---------|---------|-----|------|------|-----|------|-----|------|-------|------|-------|
| GEOM    | Semantic| 0.737| 0.958| 0.782| 0.731| 0.875| 0.875| 0.854| 0.956  | 0.940| 0.914 |
| GEOM    | CutOut  | 0.841| 0.910| 0.866| 0.704| 0.797| 0.797| 0.831| 0.820  | 0.812| 0.745 |
| GEOM    | Invert  | 0.820| 0.961| 0.794| 0.730| 0.925| 0.880| 0.810| 0.956  | 0.925| 0.982 |
| GEOM    | Flip    | 0.724| 0.998| 0.948| 0.873| 0.988| 0.968| 0.959| 0.999  | 0.988| 0.995 |
| CutPaste| Semantic| 0.523| 0.533| 0.561| 0.558| 0.543| 0.593| 0.617| 0.566  | 0.481| 0.507 |
| CutPaste| CutOut  | 0.979| 0.999| 1.000| 0.999| 1.000| 0.998| 1.000| 0.999  | 1.000| 0.997 |
| CutPaste| Invert  | 0.436| 0.625| 0.629| 0.619| 0.718| 0.540| 0.803| 0.638  | 0.493| 0.605 |
| CutPaste| Flip    | 0.500| 0.491| 0.475| 0.489| 0.487| 0.473| 0.471| 0.482  | 0.512| 0.493 |

6.1 Generalization to Other Models

This work studied denoising autoencoders (DAE) as the target model, yet our observations continue to hold with other SSAD models. To that end, we also conduct experiments on the GEOM model that uses augmentation prediction (Golan and El-Yaniv [2018]) as the objective function (see Sec. 2.3).

Table 1 presents the AUC of the GEOM model on CIFAR-10 with various types of anomalies: different object classes and anomalies constructed from known augmentation functions. With the original GEOM augmentation as aug, the model achieves the highest AUC when gen:=Flip. This is because Flip is a special case of the GEOM augmentation. With the CutPaste augmentation, on the other hand, the model works best when gen:=CutOut that aligns relatively well with aug, yet significantly poorly in other cases.

6.2 Insights from Previous Works

We introduce previous works for SSAD that present insights consistent with our observations about the importance of the alignment between aug and the nature of true anomalies. Golan and El-Yaniv [2018] observed that geometric transformations perform better than non-geometric ones such as Gaussian blurring or image sharpening on detecting semantic anomalies. Hendrycks et al. [2019] found that rotation prediction helps detect harder anomalies located near the decision boundary between normal and anomalous data.

et al. [2021] showed that local augmentation functions that focus on small perturbations of images better capture defects and flaws appearing in anomalous objects, and work significantly superior to global augmentations such as rotation. Ye, Chen, and Zheng [2021] identified that any supervisory signal given to the training of an anomaly detector induces a bias, which distorts the score distribution of the detector toward the given signals ignoring the relationships between classes. Since data augmentation in SSAD works as a generator of supervisory signals, the alignment between aug and gen determines the possible direction of bias. Tack et al. [2020] and Sohn et al. [2021] noticed that hard augmentations such as random rotation can shift the sample distribution and harm the performance of contrastive learning for anomaly detection. Their observations imply that the optimal choice of augmentation depends not only on the nature of anomalies, but also the objective function used in the training as it introduces a different requirement for aug.

Qiu et al. [2021] proposed to learn an augmentation function aug as a neural network jointly with an anomaly detector f to automatically adapt to given datasets. Nevertheless, the training is done only for the differentiable parameters in aug, however the function family of parameterized augmentations (i.e., its network structure and hyperparameters) is left to be pre-specified manually.

7 Conclusion

In this work, we studied the role of self-supervised learning (SSL) in unsupervised anomaly detection (AD). Through carefully set-up experiments on numerous datasets and extensive analyses, we showed that the alignment between augmentation and the anomaly-generating mechanism plays an essential role in the success of SSL; and importantly, SSL may even hurt detection performance under poor alignment. As such, SSL for AD emerges as a data-specific solution, rather than a general cure-all panacea, effectively rendering it a model hyperparameter—nontrivial to specify in fully unsupervised settings. Our study is motivated (and findings are also supported) by various partial evidences reported in previous works, and serves as the first systematic large-scale study to provide comprehensive evidence. We expect our work will trigger future research on the better understanding of this growing area, in addition to principled SSL for AD techniques that can tackle the outlined challenges.
References

Akçay, S.; Abarghouei, A. A.; and Breckon, T. P. 2018. GANomaly: Semi-supervised Anomaly Detection via Adversarial Training. In ACCV.

Bergman, L.; and Hoshen, Y. 2020. Classification-Based Anomaly Detection for General Data. In ICLR.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. E. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In ICML.

Chen, X.; and Konukoglu, E. 2018. Unsupervised Detection of Lesions in Brain MRI using constrained adversarial autoencoders. CoRR, abs/1806.04972.

Cheng, Z.; Zhu, E.; Wang, S.; Zhang, P.; and Li, W. 2021. Unsupervised Outlier Detection via Transformation Invariant Autoencoder. IEEE Access, 9: 43991–44002.

Conneau, A.; Khandelwal, K.; Goyal, N.; Chaudhary, V.; Wenzek, G.; Guzmán, F.; Grave, E.; Ott, M.; Zettlemoyer, L.; and Stoyanov, V. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In ACL.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In NAACL-HLT.

Devries, T.; and Taylor, G. W. 2017. Improved Regularization of Convolutional Neural Networks with Cutout. CoRR, abs/1708.04552.

Erhan, D.; Bengio, Y.; Courville, A. C.; Manzagol, P.; Vincent, P.; and Bengio, S. 2010. Why Does Unsupervised Pretraining Help Deep Learning? J. Mach. Learn. Res., 11: 625–660.

Golan, I.; and El-Yaniv, R. 2018. Deep Anomaly Detection Using Geometric Transformations. In NeurIPS.

Goyal, P.; Caron, M.; Lefauveaux, B.; Xu, M.; Wang, P.; Pai, V.; Singh, M.; Liptchinsky, V.; Misra, I.; Joulin, A.; and Bojanowski, P. 2021. Self-supervised Pretraining of Visual Features in the Wild. CoRR, abs/2103.01988.

Groggel, D. J. 2000. Practical Nonparametric Statistics. Technometrics, 42(3): 317–318.

Gudovskiy, D. A.; Ishizaka, S.; and Kozuka, K. 2022. CFLOW-AD: Real-Time Unsupervised Anomaly Detection with Localization via Conditional Normalizing Flows. In WACV.

Hendrycks, D.; Mazeika, M.; Kadavath, S.; and Song, D. 2019. Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty. In NeurIPS.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. Proc. IEEE, 86(11): 2278–2324.

LeCun, Y.; and Misra, I. 2021. Self-supervised learning: The dark matter of intelligence.

Li, C.; Sohn, K.; Yoon, J.; and Pfister, T. 2021. CutPaste: Self-Supervised Learning for Anomaly Detection and Localization. In CVPR.

Mesquita, D. P. P.; Jr., A. H. S.; and Kaski, S. 2020. Re-thinking pooling in graph neural networks. In NeurIPS.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning.

Qiu, C.; Pfommer, T.; Kloft, M.; Mandt, S.; and Rudolph, M. 2021. Neural Transformation Learning for Deep Anomaly Detection Beyond Images. In ICMIL.

Ruff, L.; Vandermeulen, R. A.; Franks, B. J.; Müller, K.; and Kloft, M. 2020. Rethinking Assumptions in Deep Anomaly Detection. CoRR, abs/2006.00339.

Schlegl, T.; Seeböck, P.; Waldstein, S. M.; Schmidt-Erfurth, U.; and Langs, G. 2017. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. In IPMI.

Sehwag, V.; Chiang, M.; and Mittal, P. 2021. SSD: A Unified Framework for Self-Supervised Outlier Detection. In ICLR.

Shenkar, T.; and Wolf, L. 2022. Anomaly Detection for Tabular Data with Internal Contrastive Learning. In ICLR.

Shenkar, T.; Li, C.; Yoon, J.; Jin, M.; and Pfister, T. 2021. Learning and Evaluating Representations for Deep One-Class Classification. In ICLR.

Tack, J.; Mo, S.; Jeong, J.; and Shin, J. 2020. CSI: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances. In NeurIPS.

Vincent, P.; Larochelle, H.; Bengio, Y.; and Manzagol, P. 2008. Extracting and composing robust features with denoising autoencoders. In ICML.

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; and Manzagol, P. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. J. Mach. Learn. Res., 11: 3371–3408.

Wang, S.; Zeng, Y.; Liu, X.; Zhu, E.; Yin, J.; Xu, C.; and Kloft, M. 2019. Effective End-to-end Unsupervised Outlier Detection via Inlier Priority of Discriminative Network. In NeurIPS.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. CoRR, abs/1708.07747.

Ye, F.; Huang, C.; Cao, J.; Li, M.; Zhang, Y.; and Lu, C. 2022. Attribute Restoration Framework for Anomaly Detection. IEEE Trans. Multim., 24: 116–127.

Ye, Z.; Chen, Y.; and Zheng, H. 2021. Understanding the Effect of Bias in Deep Anomaly Detection. In IJCAI.

Yu, J.; Zheng, Y.; Wang, X.; Li, W.; Wu, Y.; Zhao, R.; and Wu, L. 2021. FastFlow: Unsupervised Anomaly Detection and Localization via 2D Normalizing Flows. CoRR, abs/2111.07677.

Zenati, H.; Foo, C. S.; Lecouat, B.; Manek, G.; and Chandrasekhar, V. R. 2018a. Efficient GAN-Based Anomaly Detection. CoRR, abs/1802.06222.
Zenati, H.; Romain, M.; Foo, C.; Lecouat, B.; and Chandrasekhar, V. 2018b. Adversarially Learned Anomaly Detection. In ICDM.

Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; and Yang, Y. 2017. Random Erasing Data Augmentation. CoRR, abs/1708.04896.

Zhou, C.; and Paffenroth, R. C. 2017. Anomaly Detection with Robust Deep Autoencoders. In KDD.

Zong, B.; Song, Q.; Min, M. R.; Cheng, W.; Lumezanu, C.; Cho, D.; and Chen, H. 2018. Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. In ICLR.
A Structure of Autoencoder Networks

In our experiments, we use the autoencoder (AE) model used in a previous work (Golan and El-Yaniv 2018). The encoder and decoder networks consist of four encoder and decoder blocks, respectively. Each encoder block has a convolution layer of $3 \times 3$ kernels, batch normalization, and a ReLU activation function. A decoder block is similar to the encoder block, except that the convolution is replaced with a transposed convolution layer having the same kernel size.

We normalize the pixel values of images into the range of $[0, 1]$ and use the tanh activation function after the last layer to limit the range of output. We use the objective function given in Eq. (1) in the training of the model. The number of epochs and the size of hidden features are set to 256, and the number of convolution features is $(64, 128, 256, 512)$ for the four layers of the encoder block, respectively. We set the batch size largely to 1024 to stabilize the training.

We also show that the main results of our experiments are robust to the choice of hyperparameters of AE by conducting additional experiments. Details are given in Appendix C.3.

B Augmentation Functions

We provide detailed information on augmentation functions that we study in our experiments. We use the official PyTorch implementations of augmentation functions and their default hyperparameters when available.

Geometric augmentations modify images with geometric functions without removing existing information.

- **Rotate** (random rotation) makes a random rotation of an image with a degree in $(0, 90, 180, 270)$.
- **Crop** (cropping) randomly selects a small patch from an image with relative size in $(0.08, 1.0)$, resizes it, and uses it instead of the original image.
- **Flip** (vertical flipping) vertically flips an image. We use it only in the experiments of synthetic data, since it is not appropriate as a general augmentation function since it is a deterministic function without randomness.
- **GEOM** (Golan and El-Yaniv 2018) conducts three augmentation functions at the same time: Rotate, Flip, and Translate, where Translate means a random horizontal or a vertical translation by 8 pixels.

Local augmentations change only a small subset of image pixels without affecting the rest.

- **CutOut** (or random erasing) (Devries and Taylor 2017; Zhong et al. 2017) replaces a small patch in an image as zero, i.e., black pixels. The patch size is chosen randomly from $(0.02, 0.33)$.
- **CutPaste** (Li et al. 2021) copies a small patch from an image and pastes it into another location in the same image. The difference from CutOut is that CutPaste has no black pixels in resulting images, making them more plausible. The patch size is chosen from $(0.02, 0.15)$.
- **CutPaste-Scar** (Li et al. 2021) is a variant of CutPaste, which augments thin scar-like patches instead of rectangular patches. The width and height of patches is chosen from $(10, 25)$ and $(2, 16)$, respectively, in pixels. The selected patches are rotated with a degree in $(-45, 45)$.

Elementwise augmentations make a change in the value of each image pixel individually (or locally).

- **Noise** (addition of Gaussian noise) (Vincent et al. 2010) is a traditional augmentation function used in early works on denoising autoencoders. It adds a Gaussian noise with the standard deviation of 0.1 to each image pixel.
- **Mask** (Bernoulli masking) (Vincent et al. 2010) conducts a random trial to each pixel whether to change the value to zero or not. The masking probability is 0.2.
- **Blur** (Gaussian blurring) (Chen et al. 2020) smoothes an image by applying a Gaussian filter whose kernel size is 0.1 of the image. The $\sigma$ of the filter is chosen randomly from $(0.1, 2.0)$ as done in the SimCLR function.

Color augmentations change the color information of an image without changing actual objects.

- **Jitter** (Color jittering) (Chen et al. 2020) makes random changes in the global color information of an image with the brightness, contrast, saturation, and hue. The amount of changes is the same as in the SimCLR function.
- **Invert** (Color inversion) inverts the color of an image. In numerical implementation, it returns $1 - x$ for each pixel $x \in [0, 1]$. It is used only in the experiments on synthetic data, like Flip, since it is a deterministic function.

Mixed augmentations combine augmentation functions of multiple categories, making unified changes.

- **SimCLR** (Chen et al. 2020) has been used widely in literature as a general augmentation function. It conducts the following at the same time: cropping, horizontal flipping, color jittering, gray scaling, and Gaussian blurring.

C More Results of Relative AUC

We present more results of relative AUC on both controlled and in-the-wild testbeds, supporting Obs. 1, 2, 4, and 5 with different types of $\text{aug}$ and $\text{gen}$ functions. We present informal descriptions of observations as follows:

- Obs. 1: DAE performs better under better alignment between $\text{aug}$ and $\text{gen}$ functions.
- Obs. 2: DAE fails under poor alignment.
- Obs. 4: Geometric aug functions perform well in detecting semantic anomalies.
- Obs. 5: SimCLR fails at detecting semantic anomalies.

C.1 Controlled Testbed

Fig. 9 shows the result on the controlled testbed across two datasets CIFAR-10C and SVHN-C and ten classes each. The $\text{aug}$ functions that align well with $\text{gen}$ work best, consistent with Fig. 1, supporting our main findings (Obs. 1 and 2).

It is notable that SimCLR works well on Fig. 9a, although it fails in most other cases including Fig. 9c and 9d. This is because SimCLR is a “cocktail” augmentation that contains color jittering, which perturbs the color of images, and hence it exhibits a natural alignment with $\text{gen}=\text{Invert}$. This is similar to the success of GEOM in Figure 9b, which is also a “cocktail” augmentation that includes Flip.
Figure 9: (best in color) Relative AUC with respect to the no-SSL baseline on the controlled testbed with anomaly-generating functions (a) gen:=Invert and (b) gen:=Flip. Color in augmentation names denotes their categories. Augmentation functions that are aligned with the anomaly-generating functions perform well: (a) Color-based augmentations (in green) and (b) geometric augmentations (in blue). They support our main findings (Obs. 1 and 2).

Figure 10: (best in color) Relative AUC with respect to the no-SSL baseline on the in-the-wild datasets: (a) MNIST, (b) FashionMNIST, (c) CIFAR-10, and (d) SVHN. Color in augmentation names denotes their categories. Geometric augmentations (in blue) work best, and local augmentations (in red) work second-best in all four cases only with slight changes. This shows that the effectiveness of an augmentation function depends on the nature of anomalies, rather than the characteristic of normal data; these four datasets have different types of images but have the same category of anomalies: namely, different semantic classes.
Figure 11: (best in color) Relative AUC with respect to the no-SSL baseline on the in-the-wild datasets: (a) MNIST, (b) FashionMNIST, (c) CIFAR-10, and (d) SVHN. We summarize 16 different settings of hyperparameters of the DAE and AE models; each row is generated by running the Wilcoxon test on 1440 AUC values (90 tasks $\times$ 16 models). Color in augmentation names denotes their categories. The trends are the same as in Fig. 10: geometric (in blue) and local (in blue) augmentations work best. This supports the reliability of our results, which are consistent with different settings of hyperparameters.

### C.2 In-the-Wild Testbed

Fig. 10 shows the result of the Wilcoxon test on each of the four datasets in the in-the-wild testbed. Each row is thus a summary of 90 different tasks for all possible pairs of normal and anomalous classes in each dataset. Note that Fig. 11 is a summary of the four datasets in Fig. 10.

The main observation is that local augmentations (colored in blue) perform best in detecting semantic anomalies in all four datasets, even though the images in those datasets have very different characteristics. This implies that the effectiveness of an augmentation function is determined mainly by the nature of anomalies, rather than the characteristic of normal data. This supports our main finding that the alignment between $\text{aug}$ and $\text{gen}$ is the key factor to the success of self-supervised learning on anomaly detection. It also shows that Obs. 4 and 5 are consistent in individual datasets.

### C.3 Different Hyperparameters

Our study aims at making observations that are generalizable across multiple settings of models and hyperparameters. To that end, we show that our experimental results are consistent with respect to varying choices of hyperparameters of the AE model.

First, we run experiments on the following choices of hyperparameters and summarize the results with the Wilcoxon test in Fig. 11. Each row includes the results from 1440 tasks (90 different class pairs $\times$ 16 hyperparameter settings).

- Number of epochs: $\{64, 128\}$
- Number of features: $\{128, 256\}$
- Weight decay: $\{10^{-5}, 5 \times 10^{-5}\}$
- Hidden size: $\{64, 128\}$

Then, we compare the result with our main experiment in Fig. 10 which is related to Fig. 11. Both results show almost identical patterns and trends of augmentation functions in all four datasets in the in-the-wild testbed, only with slight differences in numbers. This shows that although the absolute values of AUC can be affected by hyperparameters, the relative AUC with respect to the vanilla AE is stable since we use the same setting for DAEs and AEs.
A notable difference is that the 95% confidence intervals are smaller in Fig. 11 than in Fig. 10. This is because of the increased number of trials from 90 to 1440.

D More Results of Case Studies

We present more detailed results of case studies, which support Obs. 6 and 7 under different types of gen functions. We present informal descriptions of observations as follows:

- Obs. 6 DAE applies \( \text{aug}^{-1} \) to anomalous data.
- Obs. 7 DAE makes no notable changes on normal data.

Controlled Testbed Fig. 12 shows the images of CIFAR-10C when (top) \( \text{gen}:=\text{Flip} \) and (bottom) \( \text{gen}:=\text{Invert} \). The DAEs apply the inverse augmentation function \( \text{aug}^{-1} \) to the anomalies, reconstructing normal-like ones; in Fig. 12b, the given images are rotated in the reconstructed ones when \( \text{aug}:=\text{Flip} \), while in Fig. 12d, the color of given anomalous images is inverted back in the reconstructed ones when \( \text{aug}:=\text{Invert} \). In contrast, vanilla AEs reconstruct both normal and anomalous images close to the input image.

In-the-Wild Testbed Fig. 13 shows the images of SVHN, in which anomalies are images associated with different numbers. We study four different \( \text{aug} \) functions (from top to bottom): Flip, CutOut, CutPaste, and SimCLR.

4 vs. 7 When \( \text{aug}:=\text{Rotate} \), and the task is 4 (normal) vs. 7 (anomalous), the DAE successfully generates 4-like images from 7 by applying the inverse of Rotate, which is also a rotation operation. This is because the digit 7 can be considered or somewhat resembles a rotated 4 as shown in Fig. 13b, as in the case of 6 vs. 9 in Figs. 5c and 5d.

0 vs. 3 Next, we study the task of 0 (normal) vs. 3 (anomalous) with the remaining \( \text{aug} \) functions. CutOut and CutPaste work well, because the scarred images of 0 can look like the digit 3 by chance. The inverse of CutOut is to transform a small black patch into another form observed during training. Thus, CutOut performs well with anomalous images of black background, which allows the inverse \( \text{aug}^{-1} \) to be applied easily. In other words, if an image has white background, it is difficult to find a black erased patch to revert by \( \text{aug}^{-1} \). CutPaste is a similar local augmentation, but it does not require anomalous images to have background of specific color, since it copies and pastes an existing patch instead of erasing image pixels.

The SimCLR augmentation in Figs. 13g and 13h, unlike the other \( \text{aug} \) functions, change the information of color and shape at the same time. This is because SimCLR is a “cocktail” augmentation that makes a large degree of change. The degree of change is significant even with normal images in Fig. 13g which makes it perform poorly in terms of anomaly.
Figure 13: Images from SVHN, where the three rows represent original images and those reconstructed by the vanilla AE and DAE, respectively. (a, b) Rotate works in the same way as in Figs. 5c and 5d, since the digit 7 resembles the rotated digit 4. Given the anomalous images of 7, the DAE recovers the 4-like images by rotation. (c, d, e, f) DAEs with CutOut and CutPaste work well in the task 0 (normal) vs. 3 (anomalous), because the images of 0 having erased patches can mimic those of 3. (g, h) SimCLR changes not only the shapes of digits, but also the color information due to the high degree of augmentation.
Figure 14: (best in color) Reconstruction error on CIFAR-10C with Automobile as the normal class and gen := CutOut (in the top row) and gen := Invert (in the bottom row). The distributions gradually shift to the right as the augmentation $aug$ changes the input images more and more: (a) Identity, (b) CutOut (local), (c) GEOM (“cocktail” augmentation), and (d) Invert (color-based), which inverts the values of all pixels. The distributions of augmented samples and anomalies are matched the most in (b) and (h) when $aug = gen$, which is the case of perfect alignment, consistent with Fig. 6 (see Obs. 8 and 9).

Figure 15: (best in color) Reconstruction error on SVHN for the task of 6 (normal) vs. 9 (anomalous). Similar observations are derived as in Fig. 6 since the gen function here can be thought of approximate Rotate. Still, the distribution for anomalies is not as clearly separated as in Fig. 6, because the alignment between $aug$ and $gen$ is not perfect (see Obs. 8 and 9).

E More Results of Error Histograms

We present more detailed results on error histograms to support Obs. 8 and 9 under different types of gen functions. We present informal descriptions of observations as follows:

- Obs. 8: Overall reconstruction errors are higher in DAEs than in the vanilla AEs.
- Obs. 9: The reconstruction errors are proportional to the degree of augmentation.

Controlled Testbed Fig. 14 shows the error histograms on CIFAR-10C with two types of gen: CutOut and Invert. The figure supports Obs. 8 and 9 by presenting similar patterns as in Fig. 6 even with different gen functions. A notable observation is that the distribution of augmented data is more right-shifted in Invert than in GEOM, even though GEOM is a “cocktail” approach that combines multiple augmentations. This is because Invert changes the value of every pixel simultaneously to invert the color of an image, which results in higher reconstruction errors.
in a dramatic change with respect to pixel values.

**In-the-Wild Testbed** Fig. 15 shows the reconstruction errors on the SVHN dataset with different \(\text{aug}\) functions. The task is 6 (normal) vs. 9 (anomaly), where \(\text{gen}\) can be thought of approximate rotation. \text{Rotate} works best among the four \(\text{aug}\) options thanks to the best alignment with \(\text{gen}\). \text{GEOM} makes the most right-shifted distribution of augmented data, as in Fig. 6, due to its “coctail” nature. The result on SVHN shows that Obs. 8 and 9 are valid not only for the controlled testbed, but also for the in-the-wild testbed.

**F More Results of Embedding Visualization**

We present more results on embedding visualization on both controlled and in-the-wild testbeds, supporting Obs. 3, 10, and 11 with different \(\text{aug}\) and \(\text{gen}\) functions. We present informal descriptions of observations as follows:

- **Obs. 3**: The alignment between \(\text{aug}\) and \(\text{gen}\) is affected by the degree of augmentation.

- **Obs. 10**: Embeddings for normal and anomalous data are separated with global \(\text{aug}\), and mixed with local \(\text{aug}\).

- **Obs. 11**: \(\text{aug}\) works well if anomalies lie between normal and augmented data.

**Controlled Testbed** Fig. 16 shows the results on CIFAR-10C with \(\text{gen}:=\text{Invert}\) and two different \(\text{aug}\) functions. In Figs. 16a and 16b when \(\text{aug}:=\text{Invert}\) which makes the perfect alignment with \(\text{gen}\) and makes global changes in the image pixels through augmentation, the points make separate clusters supporting Obs. 10 and result in AUC of 0.990. In Figs. 16c and 16d when \(\text{aug}:=\text{Flip}\) whose alignment with \(\text{gen}\) is imperfect, it still achieves high AUC of 0.889, since it succeeds in putting the anomalies in between normal and augmented data in the embedding space. This allows \(f\) to effectively detect the anomalies, as claimed in Obs. 11.

Fig. 17 shows the embeddings of \(\text{aug}:=\text{CutOut-c}\) with different values of \(c\) when \(\text{gen}:=\text{CutOut}\). In Fig. 17a the
distributions of augmented data and anomalies are matched well in the embedding space, due to the high alignment between $\text{aug}$ and $\text{gen}$. However, the distribution of augmented samples starts to be separated from that of normal data and anomalies from Fig. 17a to 17d, since the alignment with $\text{gen}$ becomes weaker as the value of $c$ increases. This supports Obs. 3 with respect to embedding visualization.

**In-the-Wild Testbed**  
Fig. 18 compares the vanilla AE and DAE with $\text{aug}:=$GEOM on CIFAR-10. The vanilla AE in Figs. 18a and 18b make a single cluster containing both normal data and anomalies. It achieves AUC of 0.494, which is not better than a random guess. On the other hand, the DAE in Figs. 18c and 18d makes separate clusters for normal data and anomalies as a result of augmentation. The augmented data reside in a cluster that is separate from both normal data and the test anomalies, since the alignment between $\text{aug}$ and $\text{gen}$ is unclear/unknown. Still, the DAE achieves high AUC of 0.701, since it puts the anomalies in between normal and augmented data. This supports Obs. 11.

---

of $c$ randomly from a specified range, making an average of 0.19. Refer to Section 4.1 and Appendix B for detailed information.