Supplementary Material

Enhancing Multiple Reliability Measures via Nuisance-extended Information Bottleneck

A. Training procedure of AENIB

Algorithm 1 Autoencoder-based nuisance-extended information bottleneck (AENIB)

Require: encoder f, decoder g, discriminators d, prior $p_0(\mathbf{z})$, $\alpha, \beta > 0$.

for # training iterations **do**
  Sample $(x_i, y_i)_{i=1}^m \sim p_d(\mathbf{x}, \mathbf{y})$
  $\mathbf{z}^{(i)}, \mathbf{z}_n^{(i)} \leftarrow f(x_i)$, and sample $z^{(i)}, z_n^{(i)} \sim \mathbf{z}^{(i)}, \mathbf{z}_n^{(i)}$
  $\hat{x}_i \leftarrow g(z^{(i)}, z_n^{(i)})$
  // UPDATE DISCRIMINATORS
  $L_{\texttt{ind}} \leftarrow \mathbb{E}_{\mathbf{z},\mathbf{z}_n \sim \mathcal{N}(0,I)}[\log d_{\mathbf{z}}(\mathbf{z}, \mathbf{z}_n)] + \frac{1}{m}\sum_i \log(1 - d_{\mathbf{z}}(z, z_n))$
  $L^D_{\texttt{nuis}} \leftarrow \frac{1}{m}\sum_i \mathbb{CE}(q_n(\mathbf{y}|z_n^{(i)}), y_i)$
  $L_D \leftarrow L^D_{\texttt{nuis}} - L_{\texttt{ind}}$
  $d_{\mathbf{z}}, q_n \leftarrow$ Update $d_{\mathbf{z}}, q_n$ to minimize L_D
  // UPDATE ENCODER AND DECODER
  $L^{\beta}_{\texttt{VIB}} \leftarrow \frac{1}{m}\sum_i [-\log q(y_i|z_i) + \beta \mathrm{KL}(p(\mathbf{z}|x_i) \| p_0(\mathbf{z}))]$
  $L_{\texttt{recon}} \leftarrow \frac{1}{m}\sum_i \frac{1}{\|x_i\|^2}\|x_i - \hat{x}_i\|_2^2$
  $L_{\texttt{nuis}} \leftarrow \frac{1}{m}\sum_i \mathbb{CE}(q_n^*(\mathbf{y}|z_n^{(i)}), \frac{1}{|\mathcal{Y}|})$
  $L_{\texttt{AENIB}} \leftarrow L^{\beta}_{\texttt{VIB}} + \alpha L_{\texttt{recon}} + L_{\texttt{nuis}} + L_{\texttt{ind}}$
  $f, g, \Pi_f \leftarrow$ Update f, g, Π_f to minimize $L_{\texttt{AENIB}}$
end for

B. Experimental details

B.1. Training details

Unless otherwise noted, we train each model for 200K updates for CIFAR-10 models, and 1M updates for ImageNet models. For training AENIB models, we use $\alpha = 10.0, \beta = 0.0001$ unless otherwise noted. We use different training configurations depending on the encoder architecture, *i.e.*, whether is it ResNet or ViT: (a) For ResNet-based models, we train the encoder part (f) via stochastic gradient descent (SGD) with batch size of 64 using Nesterov momentum of weight 0.9 without dampening. We set a weight decay of 10^{-4}, and use the cosine learning rate scheduling [78] from the initial learning rate of 0.1. For the remainder parts of our AENIB architecture, *e.g.*, the decoder g and discriminator MLPs, on the other hand, we follow the training practices of GAN instead: specifically, we use Adam [58] with $(\alpha, \beta_1, \beta_2) = (0.0002, 0.5, 0.999)$, following the hyperparameter practices explored by [68]. (b) For ViT-based models, on the other hand, we train both (transformer-based) encoder and decoder models via AdamW [79] with a weight decay of 10^{-4}, using batch size 128 and $(\alpha, \beta_1, \beta_2) = (0.0002, 0.9, 0.999)$ with the cosine learning rate scheduling [78]. We use 2K and 100K steps of a linear warm-up phase in learning rate for CIFAR and ImageNet models, respectively. Overall, we observe that a stable training of ViT (even for CIFAR-10) requires much stronger regularization compared to ResNets, otherwise they often significantly suffer from overfitting. In this respect, we apply the regularization practices those are now widely used for ViTs on ImageNet, namely mixup [127], CutMix [126], and RandAugment [18], following those established in [8].

B.2. Datasets

CIFAR-10/100 datasets [66] consist of 60,000 images of size 32×32 pixels, 50,000 for training and 10,000 for testing. Each of the images is labeled to one of 10 and 100 classes, and the number of data per class is set evenly, *i.e.*, 6,000 and 600 images per each class, respectively. By default, we use the random translation up to 4 pixels as a data pre-processing. We normalize the images in pixel-wise by the mean and the standard deviation calculated from the training set. The full dataset can be downloaded at https://www.cs.toronto.edu/~kriz/cifar.html.

CIFAR-10/100-C, and ImageNet-C datasets [35] are collections of 75 replicas of the CIFAR-10/100 test datasets (of size 10,000) and ImageNet validation dataset (of size 50,000), respectively, which consists of 15 different types of common corruptions each of which contains 5 levels of corruption severities. Specifically, the datasets includes the following corruption types: (a) *noise*: Gaussian, shot, and impulse noise; (b) *blur*: defocus, glass, motion, zoom; (c) *weather*: snow, frost, fog, bright; and (d) *digital*: contrast, elastic, pixel, JPEG compression. In our experiments, we evaluate test errors on CIFAR-10/100-C for models trained on the "clean" CIFAR-10/100 datasets, where the error values are averaged across different corruption types per severity level. For ImageNet-C, on the other hand, we compute and compare the mean Corruption Error (mCE) proposed by [35]. Specifically, mCE is the average of Corruption Error (CE) over corruption types, where CE is defined by the error rates normalized by those from AlexNet [67] to adjust varying difficulties across corruption types. Formally, for a classifier f, CE for a specific corruption type c is defined by:

$$\mathrm{CE}_c^f := \left(\sum_{s=1}^{5} \texttt{error}_{c,s}^{f}\right) \Bigg/ \left(\sum_{s=1}^{5} \texttt{error}_{c,s}^{\mathrm{AlexNet}}\right), \tag{12}$$

where s denotes the severity level ($1 \leq s \leq 5$). The full datasets, as well as the information on the pre-computed AlexNet error rates on ImageNet-C (to compute mCE), can be downloaded at https://github.com/hendrycks/robustness.

CIFAR-10.1/10.2 datasets [81, 94] are reproductions of the CIFAR-10 test set that are separately collected from Tiny Images dataset [110]. Both datasets consist 2,000 samples for testing, and designed to minimize distribution shift relative to the original CIFAR-10 dataset in their data creation pipelines. The datasets can be downloaded at https://github.com/modestyachts/CIFAR-10.1 (for CIFAR-10.1; we use the "v6" version) and https://github.com/modestyachts/cifar-10.2 (for CIFAR-10.2).

CINIC-10 dataset [21] is an extension of the CIFAR-10 dataset generated via addition of down-sampled ImageNet images. The dataset consists of 270,000 images in total of size 32×32 pixels, those are equally distributed for train, validation and test splits, *i.e.*, the test dataset (that we use for our evaluation) consists of 90,000 samples. The full datasets can be downloaded at https://github.com/BayesWatch/cinic-10.

ImageNet dataset [97], also known as ILSVRC 2012 classification dataset, consists of 1.2 million high-resolution training images and 50,000 validation images, which are labeled with 1,000 classes. As a data pre-processing step, we perform a 256×256 resized random cropping and horizontal flipping for training images. For testing images, on the other hand, we apply a 256×256 center cropping for testing images after re-scaling the images to have 256 in their shorter edges. Similar to CIFAR-10, all the images are normalized by the pre-computed mean and standard deviation. A link for downloading the full dataset can be found in http://image-net.org/download.

ImageNet-R dataset [34] consists of 30,000 images of various artistic renditions for 200 (out of 1,000) ImageNet classes: *e.g.*, art, cartoons, deviantart, graffiti, embroidery, graphics, origami, paintings, patterns, plastic objects, plush objects, sculptures, sketches, tattoos, toys, video game renditions, and so on. To perform an evaluation of ImageNet classifiers on this dataset, we apply masking on classifier logits for the 800 classes those are not in ImageNet-R. The full dataset can be downloaded at https://github.com/hendrycks/imagenet-r.

ImageNet-Sketch dataset [117] consists of 50,000 sketch-like images, 50 images for each of the 1,000 ImageNet classes. The dataset is constructed with Google image search, using queries of the form "`sketch of [CLS]`" within the "black and white" color scheme, where `[CLS]` is the placeholder for class names. The full dataset as well as the scripts to collect the dataset can be accessed at https://github.com/HaohanWang/ImageNet-Sketch.

CelebFaces Attributes (CelebA) dataset [77] consists of 202,599 face images, where each is labeled with 40 attribute annotations. We follow the standard train/validation/test splits of the dataset as provided by [77], and use the train split for training and computing FID scores following the protocol of other baselines [3, 93]. We also follow the pre-processing procedure of [77] to fit in the images into the size of 64×64: namely, we first perform a center crop into size 140×140 to the images, followed by a resizing operation into 64×64. The full dataset can be downloaded at https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html.

B.3. Computing infrastructure

Unless otherwise noted, we use a single NVIDIA Geforce RTX-2080Ti GPU to execute each of the experiments. For experiments based on StyleGAN2 architecture (reported in Table 11), we use two NVIDIA Geforce RTX-2080Ti GPUs per run. For the ImageNet experiments (reported in Table 3 in the main text), we use 8 NVIDIA Geforce RTX-3090 GPUs per run. Overall, our experiments are based on well-established encoder architectures such as ResNet and ViT, and the model size of AENIB is generally followed by these backbone models.

C. Architectural details

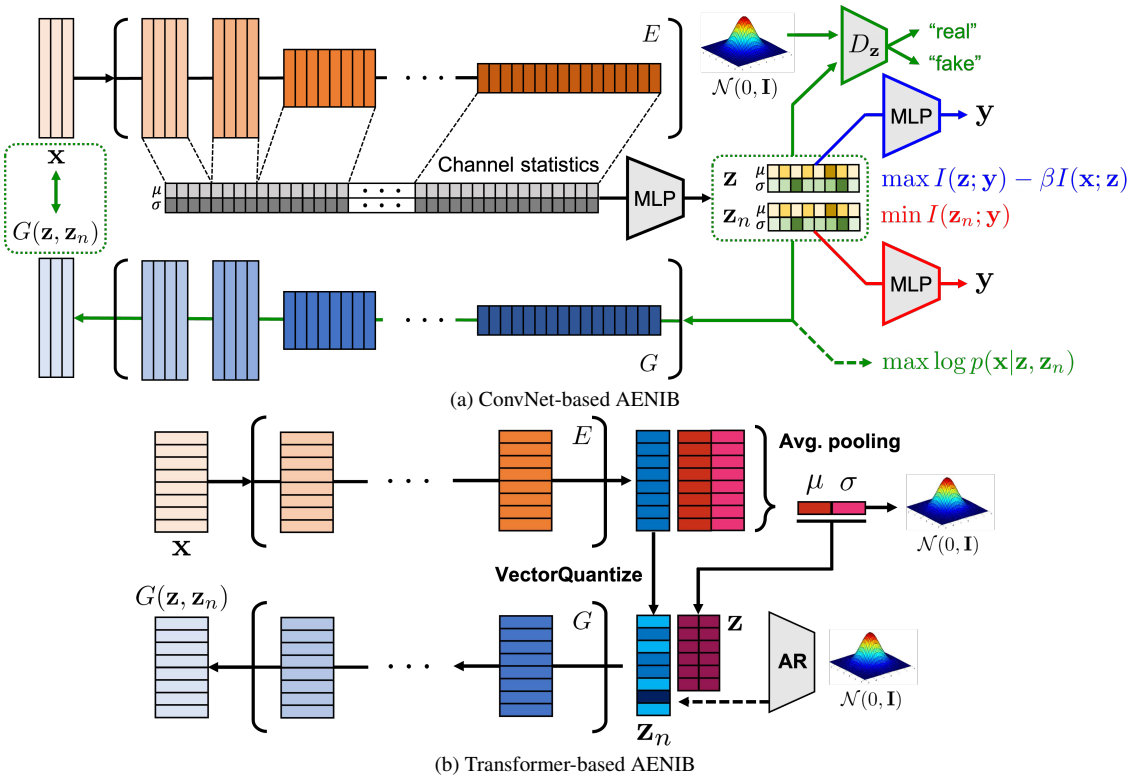


Figure 6. An overview of the proposed framework, *nuisance-extended information bottleneck* (NIB), instantiated by an autoencoder-based design with (a) ConvNet-based and (b) Transformer-based architectures.

Recall that our proposed AENIB architecture consists of (a) an encoder f, (b) a decoder g, and (c) MLP-based discriminators $d_{\mathbf{y}}$, $d_{\mathbf{z}}$, and an MLP for feature statistic projection Π_f. We set 128 as the nuisance dimension $\mathbf{z}_n$, and use hidden layer of size 1,024 for MLP-based discriminators, *e.g.*, $d_{\mathbf{y}}$, $d_{\mathbf{z}}$, and MLPs for projection Π_f.

C.1. ConvNet-based architecture

We mainly consider ResNet-18 [33] as a ConvNet-based encoder. For this encoder, we consider the generator architecture of FastGAN [75] as the decoder, but with a modification on normalization layers: specifically, we replace the standard batch normalization [45] layers in the architecture with adaptive instance normalization (AdaIN) [54] so that the affine parameters can be modulated by $\mathbf{z}$ and $\mathbf{z}_n$ as well as the decoder input: we observe a consistent gain in FID from this modification.

Adversarial similarity based guidance. We present an additional *adversarial objective* leveraging the efficiency of *feature statistics* based encoder (see Section 3.2) for ConvNet-based AENIB models to boost their generative modeling. Namely, we use the feature statistics based encoder to also provide the decoder g an extra guidance in minimizing the (pixel-level) reconstruction loss (5). We propose to additionally place a discriminator network, say $d_{\mathbf{x}} : \mathbb{R}^{|\Pi_E|} \to \mathbb{R}^e$, that computes similarity between $\Pi_f(\mathbf{x})$ and $\Pi_f(g(\mathbf{z}, \mathbf{z}_n))$ and to perform an adversarial training:

$$L_{\texttt{sim}} := \max_{d_{\mathbf{x}}} \log(1 - \sigma(\tfrac{1}{\tau} \cdot \mathrm{sim}(d_{\mathbf{x}}(\Pi_f(\mathbf{x})), d_{\mathbf{x}}(\Pi_f(g(\mathbf{z}, \mathbf{z}_n)))))), \tag{13}$$

where $\sigma(\cdot)$ is the sigmoid function and $\mathrm{sim}(\mathbf{x}, \mathbf{y}) := \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$ denotes the cosine similarity. Here, τ is a temperature hyperparameter, and we use $\tau = 0.2$ throughout our experiments. We apply this additional objective for ConvNet-based AENIB models in our experiments by default, which turns out to be helpful particularly for the generation quality of the learned autoencoders.

C.2. Transformer-based architecture

We consider ViT-S and ViT-B [24, 111] in our experiments. When Transformer-based encoder is used, we use the same Transformer architecture as the decoder model where it is preceded by linear layers that maps both $\mathbf{z}$ and $\mathbf{z}_n$ into the space of patch embedding. We assume the patch size of ViT to be 4 for CIFAR-10 and 16 for ImageNet, *i.e.*, we denote it as ViT-S/4 and ViT-S/16, respectively, so that the outputs from the models have similar numbers of patch embeddings (8×8 and 16×16, respectively) to those of ResNet-18. To model $\mathbf{z}$ and $\mathbf{z}_n$ in the ViT architecture, we simply split the output patch embedding into two separate embeddings (of reduced embedding dimensions): one of these embeddings is average-pooled to define $\mathbf{z}$, and the remaining one is vector-quantized [124] to define a nuisance representation $\mathbf{z}_n$, as described in the next paragraph in more details. Figure 6 illustrates an overview of our proposed AENIB for ViT-based architectures.

VQ-based nuisance modeling for ViT-AENIB. Remark that our current ViT-based design allocates *different* numbers of feature dimension for $\mathbf{z}$ and $\mathbf{z}_n$: specifically, we only apply global average pooling for $\mathbf{z}$ (not for $\mathbf{z}_n$), so that its dimensionality becomes independent to the input resolution, while the nuisance $\mathbf{z}_n$ would still get an increasing dimensionality for higher-resolution inputs. In practice, this difference in feature dimension may cause some training difficulties in AENIB training: (a) it makes harder to balance between the objectives given that AENIB is essentially a "competition" between two information channels, *i.e.*, $\mathbf{z}$ and $\mathbf{z}_n$; also, (b) it becomes increasingly difficult to force $\mathbf{z}_n$ to follow the independent Gaussian marginal for a tractable sampling as $\mathbf{z}_n$ gets higher dimensions, unlike our ConvNet-based design. To alleviate these issues, we propose to apply the *vector quantization* (VQ) [114, 124] to the nuisance embedding: namely, we train the output of the nuisnace head, say $\hat{\mathbf{z}}_n$, to have one of (discrete) vectors in a learned dictionary $\mathbf{e}$. With this, now $\mathbf{z}_n$ becomes independent to the dimensionality of per-patch embeddings, which allows a more scalable balancing with $\mathbf{z}$, as well as offering a tractable marginal distribution to sample: given that $\mathbf{z}_n$ now gets a sequence of discrete distribution, one can apply a post-hoc generative modeling (*e.g.*, with an autoregressive prior [114], or with a diffusion model [32]) to allow an efficient sampling. Specifically, we add the following objective upon our proposed AENIB objective (10) to enable VQ-based nuisance modeling:

$$L_{\texttt{VQ}}(\mathbf{x}; \mathbf{e}) := \|\texttt{sg}[\hat{\mathbf{z}}_n(\mathbf{x})] - e\|_2^2 + \beta\|\hat{\mathbf{z}}_n(\mathbf{x}) - \texttt{sg}[e]\|_2^2, \tag{14}$$

where $e := \min_i \|\hat{\mathbf{z}}_n(\mathbf{x}) - e_i\|_2^2$, and $\texttt{sg}(\cdot)$ denotes the stop-gradient operator defined by $\texttt{sg}(x) \equiv x$ and $\frac{d}{dx}\texttt{sg}(x) \equiv 0$. Here, the commitment hyperparameter β is set to 0.25 following [114, 124]. We allocate 32 per-patch dimensions for $\mathbf{z}_n$, with an embedding dictionary $\mathbf{e}$ of size 256. We adopt the embedding normalization [124] as we found it consistently improves the stability of VQ-based training.

SSIM-based D_2^2 reconstruction loss. Recall that our proposed AENIB is based on minimizing reconstruction loss (5) to implement $I(\mathbf{x}; \mathbf{z}, \mathbf{z}_n)$ in NIB (4). Although we introduce the *normalized mean-squared error* (NMSE) as a default design choice, the choice may not be limited to that: here, we demonstrate a SSIM-based [118] reconstruction loss as an alternative, and show its effectiveness on improving corruption robustness. Specifically, for a given pair of images[7] (x, y), the *structural similarity index measure* (SSIM) defines a similarity metric between x and y considering differences in luminance (represented by S_1) and structures (represented by S_2):

$$\text{SSIM}(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \cdot \frac{2\sigma_{xy} + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} =: S_1 \cdot S_2, \tag{15}$$

where $c_1 := 0.01^2$ and $c_2 := 0.03^2$ are small constants for numerical stability, as well as to simulate the saturation effects of visual system under low luminance (and contrast) [10]. Given that SSIM itself is not a distance metric (*e.g.*, it often allows the value to be negative), however, we instead consider the following modification of SSIM, the *squared-D_2* (D_2^2), as our reconstruction loss, where D_2 is originally defined by [10] that is shown to be a distance:

$$D_2^2(x, y) := \sqrt{(1 - S_1) + (1 - S_2)}^2 = 2 - S_1 - S_2. \tag{16}$$

In Table 7, we compare the effect of having different reconstruction losses in AENIB between the default choice of NMSE and D_2^2: the results on CIFAR-10/100-C with ViT-S/4 show that D_2^2-based reconstruction loss can reliable improve corruption robustness of the AENIB models over NMSE. This confirms that the choice of reconstruction loss impacts the final robustness of AENIB, and also suggests that a more perceptually-aligned similarity metric could possibly make the model less biased toward spurious features that are not necessary to build a robust representation.

[7] In practice, SSIM is often computed in per-patch basis for a sliding window of a certain kernel size, *e.g.*, 8. The values are then averaged to define the metric. In our experiments, we also follow this implementation.

In this respect, we adopt the D_2^2-based loss in AENIB for ViT-based models in our experiments: somewhat interestingly, we found the objective becomes much harder to be minimized for ConvNet-based models, where we keep the default choice of NMSE. This is possibly because that there can be a discrepancy between what ConvNets typically extract and those from a D_2^2-based reconstruction.

AENIB (ViT-S/4)	Loss	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic*	*Pixelate*	*JPEG*	*Average*
CIFAR-10-C	NMSE	20.0	15.8	19.2	**10.2**	**18.3**	**12.7**	**11.9**	9.76	10.0	11.0	6.33	11.3	**10.5**	**13.4**	14.7	13.0
	D_2^2	**17.8**	**14.0**	**17.6**	**10.2**	19.7	13.2	12.4	**9.12**	**8.87**	**9.56**	**5.86**	**8.29**	10.7	14.3	**13.5**	**12.3**
CIFAR-100-C	NMSE	48.2	42.8	43.2	32.4	48.6	36.1	35.6	31.4	32.3	35.2	25.3	33.7	33.2	36.2	40.9	36.9
	D_2^2	**45.7**	**40.1**	**42.1**	**32.2**	**47.9**	**35.8**	**35.1**	**31.1**	**31.7**	**34.1**	**24.4**	**31.5**	**32.4**	**34.5**	**38.4**	**35.8**

Table 7. Comparison of average per-corruption error rates (%; ↓) on CIFAR-10/100-C [35]. We use ViT-S/4 for this experiment. All the models reported here are trained via AENIB but with different reconstruction losses.

D. Ablation study

Figure 7. Reconstructions under random nuisance $\mathbf{z}_n$ (upper rows) and random semantic $\mathbf{z}$ (lower row). The leftmost per row shows the original reconstruction.

β	$L_{\texttt{recon}}$	$L_{\texttt{nuis}}$	$L_{\texttt{ind}}$	$L_{\texttt{sim}}$	Err.	C-Err.	FID
1e−4	✓	✓	✓	✓	7.07	23.3	33.3
1e−3	✓	✓	✓	✓	7.32	24.5	31.0
1e−2	✓	✓	✓	✓	7.38	26.3	30.8
1e−4	✗	✓	✓	✓	8.29	29.2	33.8
1e−4	✓	✗	✓	✓	8.01	24.1	29.2
1e−4	✓	✓	✗	✓	7.31	22.4	78.3
1e−4	✓	✓	✓	✗	7.95	28.6	83.1

Table 8. Comparison of the test error rate (Err.; %, ↓), corruption error (C-Err.; %, ↓) and FID on CIFAR-10 across ablations. We use ResNet-18 architecture for this experiment.

We further perform an ablation study on CIFAR-10 for a detailed analysis of the proposed AENIB. We use ResNet-18 based AENIB throughout this section.

Effect of β. As also introduced in the original IB objective, $\beta \geq 0$ plays the key role in AENIB training as it controls the information balance between the semantic $\mathbf{z}$ and the nuisance $\mathbf{z}_n$. Here, Figure 7 examine how using different value of β affect the actual representations, by comparing the reconstructed samples for a fixed input while randomizing the nuisance $\mathbf{z}_n$. Indeed, we observe a clear trend from this comparison demonstrating the effect of β: having larger β makes the model to push more "semantic" information into $\mathbf{z}_n$ regarding it as the nuisance. Without information bottleneck, *i.e.*, in case when $\beta = 0.0$, we qualitatively observe that the network rather encodes most information in $\mathbf{z}$, due to the minimax loss applied to the nuisance $\mathbf{z}_n$. Quantitatively, this behavior is further evidenced in Table 8 as an increase in the corruption errors when using larger β.

Reconstruction loss. The reconstruction loss $L_{\texttt{recon}}$ is one of essential part to make AENIB work as a "nuisance modeling": in Table 8, we provide an ablation when this loss is omitted, showing a significant degradation in the final accuracy, and more crucially in the corruption error. This confirms the necessity of reconstruction loss to obtain a robust representation in AENIB. Nevertheless, due to the adversarial similarity loss $L_{\texttt{sim}}$ that can also work (while not perfectly) as a reconstruction loss, one can still observe that the FID of the model can be moderately preserved.

Nuisance loss. From the ablation of $L_{\texttt{nuis}}$ given in Table 8, we observe not only a considerable degradation in clean accuracy but also in its corruption robustness. This shows that strictly forcing the nuisance-ness to $\mathbf{z}_n$ (against $\mathbf{y}$) indeed helps $\mathbf{z}$ to learn a more robust representation, possibly from encouraging $\mathbf{z}$ to extract more diverse class-related information in a faithful manner by keeping the remainder information in $\mathbf{z}_n$ sufficient to infer $\mathbf{x}$.

Independence loss. The independence loss $L_{\texttt{ind}}$ in our current design, which essentially performs a GAN training toward $p(\mathbf{z}, \mathbf{z}_n) \sim \mathcal{N}(0, \mathbf{I})$, not only forces $\mathbf{z} \perp \mathbf{z}_n$ but also leads $\mathbf{z}$ and $\mathbf{z}_n$ to have a tractable marginal distribution: so that one could efficiently perform a sampling from the learned decoder. In a practical aspect, therefore, omitting $L_{\texttt{ind}}$ in AENIB can directly harm its generation quality as given in Table 8. Nevertheless, it is still remarkable that the ablation could rather improve the corruption error: this suggests that our current design of forcing the full Gaussian may be restrictive. An alternative design for the future work could assume a weaker condition for $\mathbf{z}$ and $\mathbf{z}_n$, instead with a more sophisticated sampling to obtain a valid generative model from AENIB.

Adversarial similarity. As detailed in Appendix C.1, we introduce an additional adversarial reconstruction loss $L_{\texttt{sim}}$ (13) for cases when the backbone is convolutional. When the $L_{\texttt{sim}}$ is omitted in the ConvNet-based AENIB training, we observe a significant degradation in both FID and accuracy, in a similar manner to $L_{\texttt{recon}}$ but with a more impact on FID. This is reasonable given that $L_{\texttt{sim}}$ and $L_{\texttt{recon}}$ have the same goal of reconstructing input but in different metrics. Overall, it confirms the effectiveness of the proposed adversarial similarity based guidance to improve decoder performance.

E. Proof of Lemma 1

Lemma 1. *Let $\mathbf{x} \in \mathcal{X}$, and $\mathbf{y} \in \mathcal{Y}$ be random variables, $\hat{\mathbf{x}}$ be a noisy observation of $\mathbf{x}$ with $I(\mathbf{x}; \mathbf{y}) = I(\hat{\mathbf{x}}; \mathbf{y})$. Given that a representation $[\hat{\mathbf{z}}, \hat{\mathbf{z}}_n] := f(\hat{\mathbf{x}})$ of $\hat{\mathbf{x}}$ satisfies (a) $H(\hat{\mathbf{x}}|\hat{\mathbf{z}}, \hat{\mathbf{z}}_n) = 0$, (b) $I(\hat{\mathbf{z}}_n; \mathbf{y}) = 0$, and (c) $\hat{\mathbf{z}} \perp \hat{\mathbf{z}}_n$, it holds $I(\hat{\mathbf{z}}; \mathbf{y}) = I(\mathbf{x}; \mathbf{y})$.*

Proof. The condition $H(\hat{\mathbf{x}}|\hat{\mathbf{z}}, \hat{\mathbf{z}}_n) = 0$ implies that there exists a deterministic g such that $g(\hat{\mathbf{z}}, \hat{\mathbf{z}}_n) = \hat{\mathbf{x}}$ almost surely. Denoting $f(\cdot)$ by $f_0(\cdot, \boldsymbol{\epsilon})$ with a certain deterministic mapping f_0 and an *i.i.d.* random variable $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ via reparametrization trick, remark that a family of determinsitic functions $f_0^{\epsilon}(\cdot) := f_0(\cdot, \epsilon)$ becomes invertible almost surely for $\epsilon \sim p(\boldsymbol{\epsilon})$ and $\hat{\mathbf{x}} \sim p(\hat{\mathbf{x}})$, *i.e.*, as $g(f_0^{\epsilon}(\hat{\mathbf{x}})) = \hat{\mathbf{x}}$. Then, the statement follows from the information preservation of invertible mapping and the chain rule of mutual information (and that of conditional mutual information), as well as by applying (b) and (c):

$$I(\mathbf{x}; \mathbf{y}) = I(\hat{\mathbf{x}}; \mathbf{y}) = I(\mathbf{y}; \hat{\mathbf{z}}, \hat{\mathbf{z}}_n) = I(\mathbf{y}; \hat{\mathbf{z}}_n) + I(\mathbf{y}; \hat{\mathbf{z}}|\hat{\mathbf{z}}_n) \tag{17}$$

$$= I(\mathbf{y}; \hat{\mathbf{z}}) + H(\hat{\mathbf{z}}_n|\mathbf{y}) + H(\hat{\mathbf{z}}_n|\hat{\mathbf{z}}) - H(\hat{\mathbf{z}}_n|\mathbf{y}, \hat{\mathbf{z}}) - H(\hat{\mathbf{z}}_n) \tag{18}$$

$$= I(\mathbf{y}; \hat{\mathbf{z}}) = I(\hat{\mathbf{z}}; \mathbf{y}). \tag{19}$$

□

F. Additional background

F.1. Detailed survey on related work

Out-of-distribution robustness. Since the seminal works [2, 90, 102] revealing the fragility of neural networks for out-of-distribution inputs, there have been significant attempts on identifying and improving various notions of robustness: *e.g.*, detecting novel inputs [36–38, 71, 72, 103, 120], robustness against corruptions [22, 27, 35, 39, 116], and adversarial noise [5, 11, 17, 29, 82, 128], to name a few. Due to its fundamental challenges in making neural network to extrapolate, however, most of the advances in the robustness literature has been made under assuming priors closely related to the individual problems: *e.g.*, *Outlier Exposure* [37] and *AugMix* [39] assume an external dataset or a pipeline of data augmentations to improve the performances in novelty detection and corruption robustness, respectively; *Tent* [116] leverages extra information available from a batch of samples in test-time to adapt a given neural network; [52, 112] observe that neural networks robust to a certain type of adversarial attack (*e.g.*, an ℓ_∞-constrained adversary) do not necessarily robust to other types of adversary (*e.g.*, an ℓ_1 adversary), *i.e.*, adversarial robustness hardly generalizes from the adversary assumed *a priori* for training. In this work, we aim to improve multiple notions of robustness without assuming such priors, through a new training scheme that extends the standard information bottleneck principle under noisy observations.

Hybrid generative-discriminative modeling. Our proposed method can be also viewed as a new approach of improving the robustness of discriminative models by incorporating a generative model, in the context that has been explored in recent works [31, 72, 99, 123]: for example, [72, 73] have shown that assuming a simple Gaussian mixture model on the deep discriminative representations can improve novelty detection and robustness to noisy labels, respectively; [99] develop an empirical defense against adversarial examples via generative classifiers; A line of research on *Joint Energy-based Models* (JEM) [31, 123] assumes the entire discriminative model as a joint generative model by interpreting the logits of $p(\mathbf{y}|\mathbf{x})$ as unnormalized log-densities of $p(\mathbf{x}|\mathbf{y})$, and shows that modeling $p(\mathbf{x}|\mathbf{y})$ as well as $p(\mathbf{y}|\mathbf{x})$ can improve out-of-distribution generalization of the classifier. Nevertheless, it is still an unexplored and open question that how to

"better" incorporate generative representation into discriminative models: in case of novelty detection, for example, several recent works [89, 95, 100, 120] observe that existing likelihood-based generative models are not accurate enough to detect out-of-distribution datasets, suggesting that relying solely on (likelihood-based) deep generative representation may not enough for robust classification [25]. In case of JEM, on the other hand, it has been shown that directly assuming a joint generative-discriminative representation often makes a significant training instability. In this work, we propose to introduce an autoencoder-based model to avoid the training instability, and consider a design that the *nuisance* can succinctly supplement the given discriminative representation to be generative.

Invertible representations and nuisance modeling. The idea of incorporating nuisances can be also considered in the context of *invertible* modeling, or as known as *flow-based models* [6, 13, 23, 30, 47, 59],[8] which maps a given input $\mathbf{x}$ into a representation $\mathbf{z}$ of the same dimension so that one can construct an inverse of $\mathbf{z}$ to $\mathbf{x}$: here, the nuisance can be naturally defined as the remainder information of $\mathbf{z}$ for a given subspace of interest, *e.g.*, to model $\mathbf{y}$. For example, [46] adopt a fully-invertible variant of i-RevNet [47] to analyze *excessive invariance* in neural networks, *i.e.*, the existence of pairs of completely different samples with the same representation in a neural network, and proposes to maximize the cross-entropy for the nuisances in a similar manner to our proposed minimax-based nuisance loss ((6) in the main text); [4], on the other hand, leverages invertible neural network to model a Gaussian mixture based generative classifier in the representation space, so that nuisance information can be preserved until its representation. Compared to such approaches relying on invertible neural networks, our autoencoder-based nuisance modeling does not guarantee the "full" invertibility for arbitrary inputs: instead, it only focuses on inverting the data manifold given, and this enables (a) a much flexible encoder design in practice, *i.e.*, other than flow-based designs, and (b) a more scalable generative modeling of nuisance representation $\mathbf{z}_n$ while forcing its *independence* to the semantic space $\mathbf{z}$. This is due to that it works on a compact space rather than those proportional to the input dimension, which is an important benefit of our modeling in terms of the scalability of nuisance-aware training, *e.g.*, beyond an MNIST-scale as done by [46]. More closer related works [48, 49, 92] in this respect instead introduce a separate encoder for nuisance factors, where the nuisanceness is induced by the independence to $\mathbf{z}$: *e.g.*, DisenIB [92] applies FactorVAE [57] between semantic and nuisance embeddings to force their independence.[9] Yet, similarly to the invertible approach, the literature has been questioned on that the idea can be scaled-up beyond, *e.g.*, MNIST, and our work does explore and establish a practical design that is applicable for recent architectures and datasets addressing modern security metrics, *e.g.*, corruption robustness. On the technical side, we find that, *e.g.*, the "nuisanceness to $\mathbf{y}$" is more important for $\mathbf{z}_n$ than the "independence with $\mathbf{z}$" (as usually done in the previous works [48, 49, 92]) to induce a robust representation, as verified in our ablation study in Appendix D, which can be a useful practice for the future research concerning robust representation learning.

Autoencoder-based generative models. There have been steady advances in generative modeling based on autoencoder architectures, especially since the development in *variational autoencoders* (VAEs) [61]: due to its ability of estimating data likelihoods, and its flexibility to implement various statistical assumptions [57, 60, 80]. With the advances in its training objectives [41, 83, 115] as well as the architectural improvements [15, 113], VAE-based models are currently considered as one of state-of-the-art approaches in likelihood based generative modeling: *e.g.*, a state-of-the-art *diffusion models* [43, 101] is built upon the denoising autoencoders under Gaussian perturbations, and recently-proposed *hierarchical VAEs* [15, 113] have shown that VAEs can benefit from scaling up its architectures into deeper encoder networks. In perspectives of viewing our method as a *generative modeling*, AENIB is based on *adversarial autoencoders* [83] that replaces the KL-divergence based regularization in standard VAEs with a GAN-based adversarial loss, with a novel encoder architecture that is based on the *internal feature statistics* of discriminative models: so that the model can better encode lower-level features without changing the backbone architecture. We observe that this design enables autoencoder-based modeling even from a large, pre-trained discriminative models, and this "projection" of internal features can significantly benefit the generation quality, as well as for generative adversarial networks (GANs) as observed in Table 11.

F.2. Technical background

Variational information bottleneck. Although the information bottleneck (IB) principle given in (2) [107] suggests a useful definition on what we mean by a "good" representation, computing mutual information of two random variables is generally hard and this makes the IB objective infeasible in practice. To overcome this, *variational information bottleneck* (VIB) [1, 12] applies variational inference to obtain a lower bound on the IB objective (2). Specifically, it approximates: (a) $p(\mathbf{y}|\mathbf{z})$ by a (parametrized) "decoder" neural network $q(\mathbf{y}|\mathbf{z})$, and (b) $p(\mathbf{z})$ by an "easier" distribution $r(\mathbf{z})$, *e.g.*, isotropic Gaussian $\mathcal{N}(\mathbf{z}|0, \mathbf{I})$. Having such (variational) approximations in computing (2) as well as the Markov chain property $\mathbf{y} - \mathbf{x} - \mathbf{z}$ of

[8] A more complete survey on flow-based models can be found in [63].

[9] We provide a more direct empirical comparison with DisenIB [92] to AENIB in Section 4.4.

neural networks, one yields the following lower bound on the IB objective (2):

$$I(\mathbf{z};\mathbf{y}) - \beta I(\mathbf{z},\mathbf{x}) \geq \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\int dz \left(p(z|\mathbf{x}) \log q(\mathbf{y}|z) - \beta p(z|\mathbf{x}) \log \frac{p(z|\mathbf{x})}{r(z)}\right)\right]. \tag{20}$$

This bound can now be approximated with the empirical distribution $p(\mathbf{x},\mathbf{y}) \approx \frac{1}{n}\sum_i \delta_{x_i}(\mathbf{x})\delta_{y_i}(\mathbf{y})$ from data. By further assuming a Gaussian encoder $p(\mathbf{z}|\mathbf{x}) := \mathcal{N}(\mathbf{z}|f^\mu(\mathbf{x}), f^\sigma(\mathbf{x}))$ as defined in (1) and applying the reprarametrization trick [61], we get the following VIB objective:

$$L^\beta_{\texttt{VIB}} := \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{\epsilon}}[-\log q(y_i|f(x_i,\boldsymbol{\epsilon}))] + \beta \,\mathrm{KL}\,(p(\mathbf{z}|x_i)\|r(\mathbf{z})). \tag{21}$$

Generative adversarial networks. *Generative adversarial network* (GAN) [28] considers the problem of learning a generative model p_g from given data $\{x_i\}_{i=1}^n$, where $x_i \sim p_d(\mathbf{x})$ and $\mathbf{x} \in \mathcal{X}$. Specifically, GAN consists of two neural networks: (a) a *generator* network $G : \mathcal{Z} \to \mathcal{X}$ that maps a latent variable $z \sim p(\mathbf{z})$ into $\mathcal{X}$, where $p(\mathbf{z})$ is a specific prior distribution, and (b) a *discriminator* network $D : \mathcal{X} \to [0,1]$ that discriminates samples from p_d and those from the implicit distribution p_g derived from $G(\mathbf{z})$. The primitive form of training G and D is the following:

$$\min_G \max_D V(G,D) := \mathbb{E}_{\mathbf{x}}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]. \tag{22}$$

For a fixed G, the inner maximization objective (22) with respect to D leads to the following *optimal discriminator* D^*_G, and consequently the outer minimization objective with respect to G becomes to minimize the *Jensen-Shannon divergence* between p_d and p_g, namely $D^*_G := \frac{p_d}{p_d+p_g}$.

G. Results on MNIST-C

Method	*Clean*	*AUROC* (↑)	*Shot*	*Impulse*	*Glass*	*Motion*	*Shear*	*Scale*	*Rotate*	*Brightness*	*Translate*	*Stripe*	*Fog*	*Spatter*	*Dotted line*	*Zigzag*	*Canny edges*	*Average*
Cross-entropy	0.45	0.987	4.69	69.6	60.3	46.5	1.41	2.97	**4.80**	88.7	2.45	76.6	88.7	27.3	5.64	**27.3**	44.1	34.5
VIB [1]	**0.44**	0.988	4.52	73.5	73.8	71.8	1.73	2.84	5.85	90.1	2.15	78.1	89.8	28.4	5.85	28.5	44.0	37.6
sq-VIB [106]	0.48	0.955	4.32	71.5	63.5	62.3	1.62	2.70	5.74	90.5	2.43	80.3	90.3	24.8	5.91	32.0	43.4	36.4
NLIB [65]	1.15	0.974	7.13	67.9	62.5	57.9	2.15	4.00	7.06	**86.9**	3.28	81.8	88.7	30.1	8.97	31.0	41.8	36.4
sq-NLIB [106]	3.19	0.908	9.90	73.3	66.7	64.7	4.25	6.19	9.21	88.7	6.43	**72.4**	89.8	32.4	9.69	36.2	72.5	40.3
DisenIB [92]	0.54	0.997	4.60	68.8	56.4	50.4	1.11	**2.04**	4.84	88.7	2.01	74.3	**88.5**	20.1	4.75	27.4	69.0	35.2
AENIB (ours)	0.72	**1.000**	**3.71**	**48.8**	**44.0**	**27.1**	**0.99**	3.15	4.82	89.7	**0.88**	82.0	89.7	**16.4**	**4.14**	33.9	**25.9**	**29.8**

Table 9. Comparison of (a) clean error (%; ↓), (b) AUROC (↑) on detecting Gaussian noise (higher is better), and (c) corruption errors (%; ↓) per corruption type on MNIST-C [88]. Each classifier is trained on MNIST with random translation as augmentation. We highlight our results as blue whenever the value improves the baselines more than 3% in absolute values.

We also evaluate our proposed AENIB training on MNIST-C [88], a collection of corrupted versions of the MNIST [69] test dataset of 15 corruption types constructed in a similar manner to CIFAR-10/100-C [35], to get a clearer view on the effectiveness of our method on a simpler setup. For this experiments, we use a simple 4-layer convolutional network (with batch normalization [45]) as the encoder architecture, and trained every model on the (clean) MNIST training dataset for 100K updates following other training details of the CIFAR experiments (see Appendix B.1): again, we notice that the training does not assume specific prior on the corruptions. We compare AENIB with the direct ablations of cross-entropy and VIB based models, as well as some variants of VIB, namely Nonlinear-VIB [65], Squared-VIB/NLIB [106], and DisenIB [92].

Table 9 summarizes the results: overall, we observe that the effectiveness of AENIB training still applies to MNIST-C, *e.g.*, our AENIB training improves the average corruption error from the baseline cross-entropy based training from $34.5\% \to 29.8\%$, which could not be obtained by simply sweeping on the baseline VIB training. Given that MNIST-C allows

a visually clearer distinction between contents and corruptions compared to CIFAR-10/100-C, one can better interpret the behavior of given models on each corruption types: here, we observe that our training can dramatically improve robustness for certain types of corruptions where the baselines shows poor performances, *e.g.*, Impulse, Glass, and Motion, while still some types of corruptions are still remaining challenging even with AENIB, *e.g.*, especially for low-frequency biased corruptions such as Brightness and Stripe. Compared to DisenIB, on the other hand, we observe that the effectiveness from DisenIB, e.g., its gain in AUROC (as conducted by [92]), could not be further generalized on MNIST-C, where AENIB still improves upon it as well as achieving the perfect score at the same OOD task.

H. Additional results on corruption robustness

	CIFAR-10-C							CIFAR-100-C						
Severity	Clean	1	2	3	4	5	Avg.	Clean	1	2	3	4	5	Avg.
Cross-entropy	5.71	12.9	18.1	24.3	31.7	43.5	26.1	26.9	39.2	46.9	53.2	59.8	69.3	53.7
VIB [1]	**5.47**	**12.5**	17.5	23.6	30.7	42.5	25.4	**26.5**	39.7	47.5	53.8	60.5	70.1	54.3
AENIB (ours)	7.07	13.2	**17.2**	**21.7**	**27.5**	**37.0**	**23.3**	28.0	**39.0**	**45.5**	**51.4**	**57.6**	**67.0**	**52.1**

Table 10. Comparison of average corruption error rates (%; ↓) per severity level on CIFAR-10/100-C [35] with ResNet-18 architecture. Bold and underline denote the best and runner-up, respectively. All the models are trained only using random translation as data augmentation.

Table 10 summarizes our experiments on corruption robustness with ResNet-18 architectures. Again, our AENIB consistently improves the robustness at common corruptions benefiting from nuisance modeling compared to VIB objectives. For example, we obtain an absolute 6.5% of reduction in error rates on CIFAR-10-C of the highest severity, *i.e.*, by $43.5\% \rightarrow 37.0\%$. Here, we highlight that we do not utilize any data augmentations but random translation for training ResNet-18 models; the gain from our method indeed comes from the better robustness itself, unlike common practices in improving the corruption robustness by carefully designing augmentations. Interestingly, we observe that the impact of AENIB in the clean error can be different depending on the encoder architecture: compared to ViT results as given Table 2, AENIB with ResNet-18 makes a slight decrease in clean accuracy, while the robust accuracy is still improved. In this case, it is more clear to see that AENIB could improve the *effective robustness* of the learned representation. This is possibly due to that the representation induced via AENIB can be extracted better with non-local (attention-based) operations such as ViT.

I. Experiments on image generation

CIFAR-10, Unconditional	Augment.	FID (↓)	IS (↑)
StyleGAN2 [55]	HFlip	11.1	9.18
+ DiffAug [130]	Trans, CutOut	9.89	9.40
+ ContraD [51]	SimCLR	9.80	9.47
+ ADA* [53]	Dynamic	**7.01***	-
+ FSD (R-18; ours)	HFlip, Trans	8.43	9.68
+ FSD (R-50; ours)	HFlip, Trans	7.39	**10.0**
FastGAN [75]	HFlip, Trans	34.5	6.52
+ Proj-GAN (R-18)	HFlip, Trans	8.48	9.40
+ FSD (R-18; ours)	HFlip, Trans	**7.80**	**9.65**

Table 11. Test FID and IS of GANs on CIFAR-10. Bold and underline indicate the best and runner-up, respectively. We note that the value of ADA* [53] is taken after $2\times$ longer training steps.

Method	CIFAR-10		CelebA
	FID ↓	IS ↑	FID ↓
VAE [93]	115.8	3.8	-
VAE/GAN [93]	39.8	7.4	-
2s-VAE [19]	72.9	-	44.4
Perceptual AE [129]	51.5	-	13.8
NCP-VAE [3]	24.1	-	**5.25**
NVAE [113]	56.0	5.19	13.5
DC-VAE [93]	17.9	8.2	19.9
$L_{\texttt{recon}}$ (5) only	65.0	5.73	50.1
+ Adv. similarity (13)	46.8	6.29	25.1
+ Projection (R-18)	**12.6**	**8.86**	6.91

Table 12. Test FID and IS of VAE models on unconditional generation of CIFAR-10 and CelebA. Bold and underline denote the best and runner-up, respectively.

I.1. Feature statistics discriminator for GANs

Designing a stable discriminator has been crucial for GANs: a usual practice in the literature is to have a separate, carefully-designed network with a comparable generator, but with a significant overhead. We observe that the *internal feature statistics* of a convolutional encoder f can be a surprisingly effective representation to define a simple yet efficient discriminator. Concretely, for a given encoder f and an input $\mathbf{x}$, we consider L intermediate feature maps of $\mathbf{x}$, namely $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(L)}$ from $f(\mathbf{x})$, the *features statistics discriminator* (FSD) we consider here is then simply a 3-layer MLP applied on $\Pi_f(\mathbf{x})$ (9). In the following, we empirically confirm that this simplest design of discriminator can dramatically accelerate GAN training, particularly when applied upon pre-trained discriminative encoders: similarly to [98] but with a simpler architecture.

We evaluate the effect of the proposed feature statistics discriminator to the generation quality of GANs: here, we consider ImageNet-pretrained ResNet-18 (R-18) and ResNet-50 (R-50) [33], and define GAN discriminators via FSD upon the pre-trained models. We adopt StyleGAN2 [55] and FastGAN [75] for the generator architectures. For the StyleGAN2-based models, we follow the training details of DiffAug [130] and ADA [53] in their CIFAR experiments: specifically, we use Adam with $(\alpha, \beta_1, \beta_2) = (0.002, 0.0, 0.99)$ for optimization with batch size of 64. We use non-saturating loss for training, and use R_1 regularization [84] with $\gamma = 0.01$. We do not use, however, the path length regularization and the lazy regularization [55] in training. We take exponential moving average on the generator weights with half-life of 500K samples. We stop training after 800K generator updates, which is about the half of those conducted for the ADA baseline [53]. For the FastGAN baseline, on the other hand, we run the official implementation of FastGAN[10] [75] on CIFAR-10 for the length of 6.4M samples with batch size 16. For the "Projected GAN" baseline, we adapt the official implementation[11] [98] onto the ImageNet pre-trained ResNet-18, and trained for 6.4M samples with batch size 64. Our results ("FSD") follows the same training details, but with a difference in its discriminator.

Table 11 summarizes the results. Overall, we observe that FSD can aid GAN training of given generator network surprisingly effectively: by leveraging pre-trained representations, FSD could achieve FID competitive with a state-of-the-art level approach of ADA [53] even with using much weaker data augmentation. Compared to Projected GAN [98] that also leverages pre-trained models to stabilize GANs, our approach offers a more simpler approach to leverage the given representations, *i.e.*, by just aggregating the features statistics, yet achieving a better FID.

I.2. Feature statistics encoder for autoencoders

We also evaluate our proposed architecture and method as a *generative modeling*, especially focusing on the effectiveness of the *feature statistics encoder* (Section 3.2) and the *adversarial similarity* based training for ConvNet-based AENIB autoencoders on CIFAR-10 [66] and CelebA [77] datasets. To this end, we consider an "unsupervised" version of AENIB which omits the VIB loss ($L^{\beta}_{\texttt{VIB}}$; (20)) and the nuisance loss ($L_{\texttt{nuis}}$; (6)) in training, so that the model can assume the setup of unconditional generation.

Table 12 summarizes the quantitative generation results of our AENIB models optimized with different objectives.[12] Firstly, it confirms the effectiveness of adversarial similarity based training: when it is solely applied upon $L_{\texttt{recon}}$ ("$L_{\texttt{recon}}$ only"; equivalent to [83]) it makes a significant improvements in both FID and IS. To further investigate the effectiveness of our proposed feature statistics encoder, we also test a scenario that the encoder is *fixed* by ResNet-18 pre-trained on ImageNet, akin to the setup of Table 11: we observe that our encoder design can surprisingly benefit from using better representation, *e.g.*, "+ Projection (R-18)" in Table 12 further improves FID on CIFAR-10 from $46.8 \rightarrow 12.6$, better than the best results among considered VAE-based models, by only training an MLP upon the feature statistics of the (fixed) model. It is notable that the gain only appears when we apply the adversarial similarity based training: *i.e.*, even with the pre-trained model, it only achieves 67.5 in FID on CIFAR-10 without the training. This observation suggests an interesting direction to scale-up autoencoder-based models with large pre-trained representations, in a similar vein as [98] as presented in the context of GANs.

[10] https://github.com/odegeasslbc/FastGAN-pytorch

[11] https://github.com/autonomousvision/projected_gan

[12] Following other baselines, we compute FIDs from 50,000 generated samples against the training dataset.

I.3. Qualitative results

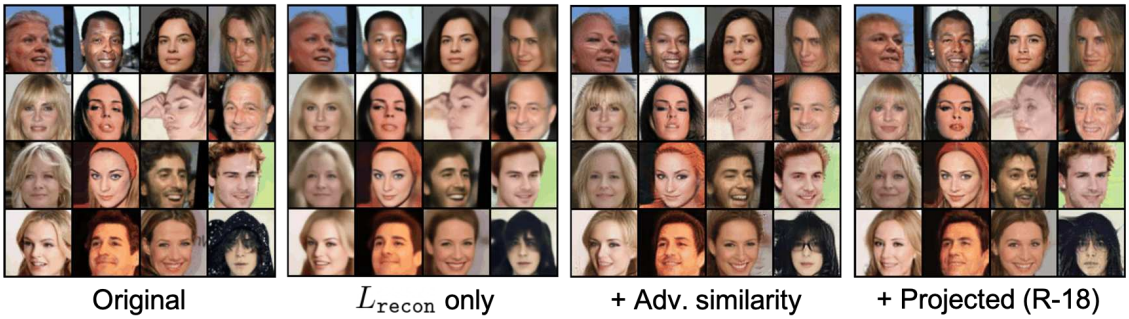
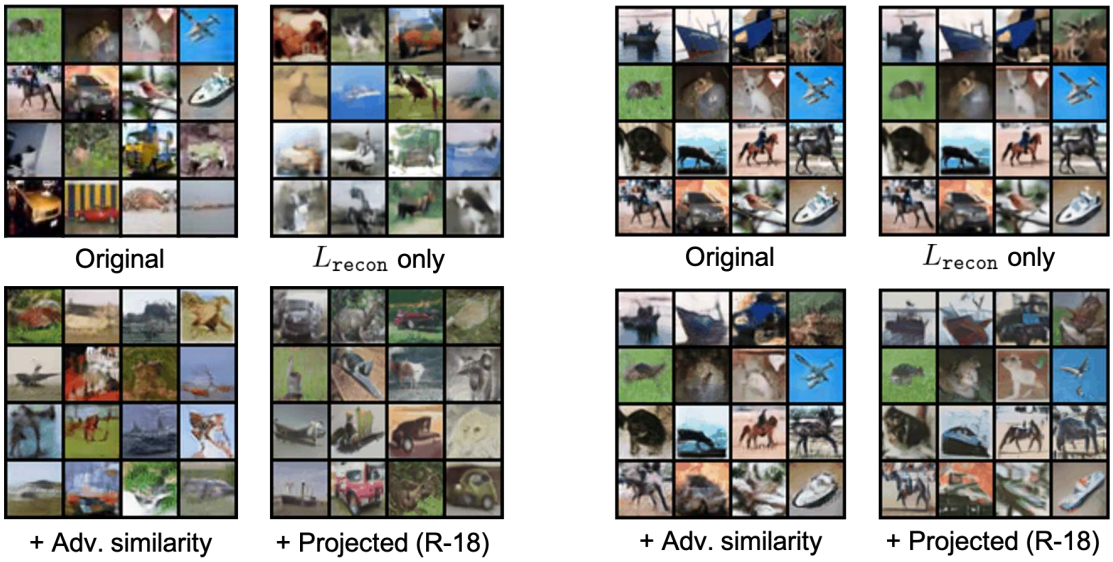


Figure 8. Qualitative comparison of reconstructions of fixed samples of unconditional AENIB model (and its ablations) on CelebA.

(a) $L_{\texttt{recon}}$ only (FID: 50.1) (b) + Adv. similarity (FID: 25.1) (c) + Projected (FID: 6.91)

Figure 9. Qualitative comparison on uncurated random samples from unconditional AENIB model (and its ablations) on CelebA.



(a) Reconstruction (b) Sampling

Figure 10. Qualitative comparison on uncurated random samples from unconditional AENIB model (and its ablations) on CelebA.

J. Application to model debugging

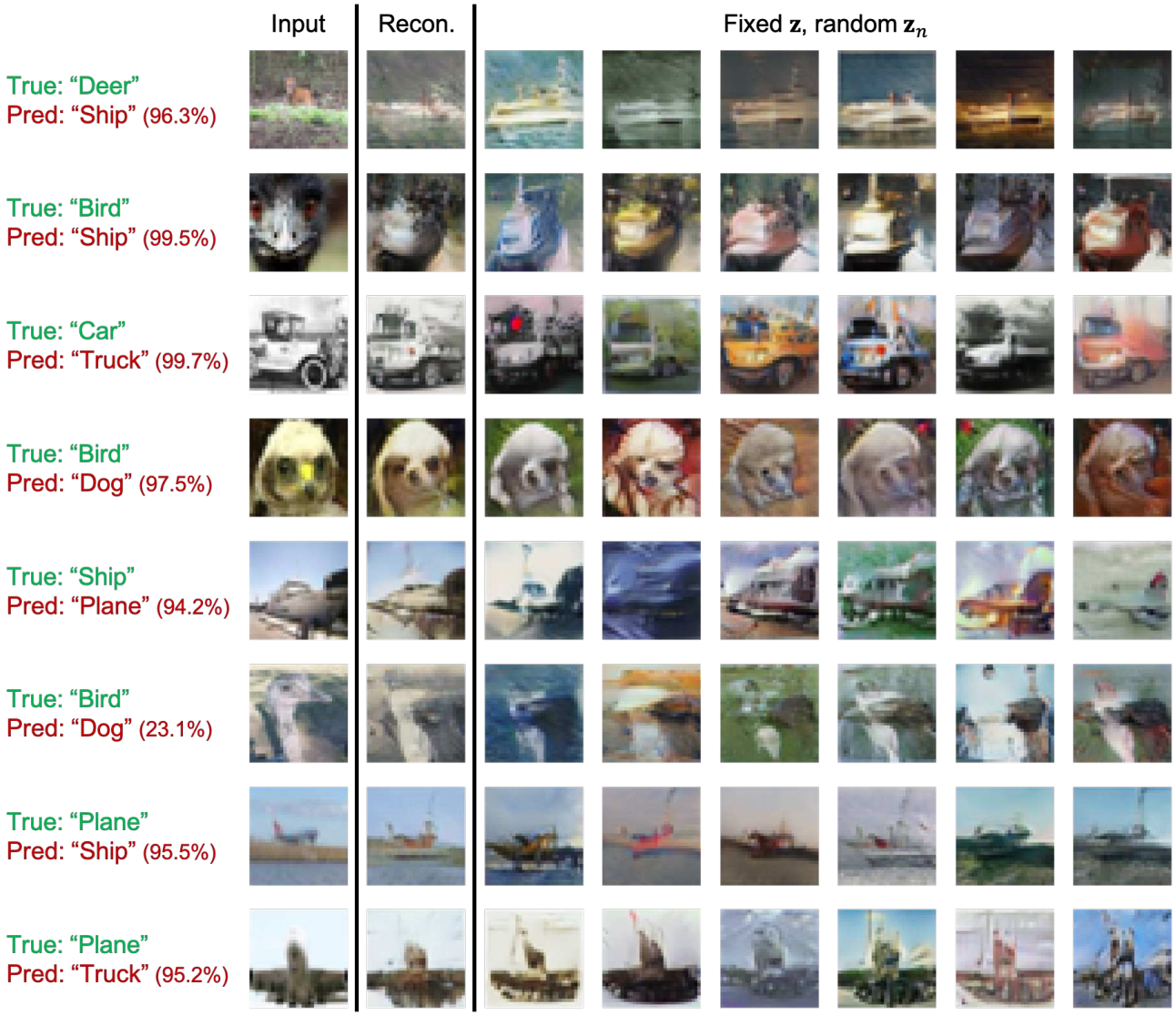


Figure 11. Qualitative comparison between (a) original input (the leftmost column), (b) its reconstruction (the second column), and (c) its further reconstructions with random nuisance $\mathbf{z}_n$ (the remaining columns), examined for test samples misclassified by a CIFAR-10 AENIB model.

To further understand how the proposed AENIB model internally works with its representation $\mathbf{z}$ and $\mathbf{z}_n$, we examine an AENIB model trained on CIFAR-10 to analyze how the model reconstruct given inputs when the model incorrectly classifies them. Specifically, Figure 11 illustrates a subset of CIFAR-10 test samples misclassified by an AENIB model by comparing the original input with its reconstructed samples from the model. Overall, we observe that such a qualitative comparison can provide a useful signal to interpret model errors: it effectively visualizes which visual cues of a given input negatively affected the decision making process of the given model, also visualizing the closest (misclassified) realizations that the model decodes for a given representation, *i.e.*, what the model actually perceived. For example, for the test input given at the first row of Figure 11, one can observe that the model essentially "ignored" the tiny part that represent the true semantic, *i.e.*, the "deer", and reconstructed the remaining part as a "ship".