Appendices: Robust outlier detection by debiasing VAE likelihoods

Appendix A: VAE architecture and training

All experiments were performed using Tensorflow 2 and Tensorflow Probability libraries. We employed a convolutional VAE architecture that follows the DCGAN [18] structure (Table 1), nearly identical with that of [25]. We used the Adam optimizer [6] with a learning rate of 5e-4 for training all of our models. Each model was trained for 1000 epochs with a batch size of 64, and the checkpoint with best validation performance based on negative log likelihood was used for reporting results. We used the Xavier uniform initializer (default in Tensorflow 2) for initializing network weights.

For reporting results based on the bias-corrected log likelihood (BC-LL score) we used a VAE with a latent dimension (nz) of size 20. To examine the robustness of our remedies to the VAE architecture, we also trained four additional VAEs with latent dimensions of size 40, 60, 80, and 100 (see Appendix E.1, Fig. 7). The same architecture was used for training both grayscale and natural image VAEs, with two differences (grayscale: nf = 32, nc = 1; natural: nf = 64, nc = 3). Log likelihoods were estimated using the importance weighted lower bound (n=100 samples) [2].

Table 1. **VAE architecture**. nc: number of channels; nf: number of filters; nz: number of latent dimensions; BN: batch normalization; Conv: convolution layer; DeConv: deconvolution layer; ReLU: rectified linear unit

Encoder	Decoder
Input image of shape $32 \times 32 \times$ nc	Input latent code, reshape to $1 \times 1 \times$ nz
4×4 Conv_{nf} Stride=2, BN, ReLU	4×4 $\text{DeConv}_{4 \times \text{nf}}$ Stride=1, BN, ReLU
4×4 $\text{Conv}_{2 \times \text{nf}}$ Stride=2, BN, ReLU	4×4 $\text{DeConv}_{2 \times \text{nf}}$ Stride=2, BN, ReLU
4×4 $\text{Conv}_{4 \times \text{nf}}$ Stride=2, BN, ReLU	4×4 $\text{DeConv}_{\text{nf}}$ Stride=2, BN, ReLU
4×4 $\text{Conv}_{2 \times \text{nz}}$ Stride=1	4×4 $\text{DeConv}_{\text{nc}}$ Stride=2

Appendix B: Bias correction for Bernoulli decoder

For a VAE decoder with a Bernoulli visible distribution, the negative reconstruction error is given by:

$$\begin{aligned}\log p_\theta(\mathbf{x}|\mathbf{z}) &= \log p_{\text{B}}(\mathbf{x}; \hat{\mathbf{x}}_\theta(\mathbf{z})) \\ &= \sum_{i=1}^{D} x_i \log \hat{x}_i + (1 - x_i) \log(1 - \hat{x}_i)\end{aligned}$$

where x_i is the pixel value of the i^{th} pixel in the input sample and $\hat{x}_i$ (or $\hat{x}_i(\mathbf{z})$) is the corresponding pixel value in the image reconstructed by the decoder, and $\mathbf{z}$ is the latent representation corresponding to the input image (see [8] their Appendix C.1).

The negative reconstruction error for perfect reconstruction is simply calculated by setting $\hat{x}_i = x_i$, as:

$$\log p_{\text{B}}(\mathbf{x}; \mathbf{x}) = \sum_{i=1}^{D} x_i \log x_i + (1 - x_i) \log(1 - x_i)$$

Appendix C: Data sources and pre-processing

Data sources. We used Tensorflow Datasets[2] for all datasets except Sign Language MNIST, CompCars and GTSRB. We fetched the Sign Language MNIST[3], CompCars[4] surveillance-nature images and GTSRB[5] datasets from their respective official sources. For the EMNIST dataset, we selected only the "Letters" split. To generate noise patches, we used uniform random noise in range [0, 1], sampled independently across pixels and channels.

Data pre-processing and splits For both grayscale and natural image datasets, we resized all images to 32×32 pixels before VAE training and evaluation. For natural image datasets, we employed the cropped versions of SVHN and CelebA (default option in Tensorflow Datasets), such that the central object (numbers or faces, respectively) were approximately centered in each image. In addition, we preprocessed the images using "contrast stretching", where specified, with the method described in Section 3.3. For all datasets, we reserved 10% for the training data for validation (generated twice for two train-val splits, independently). Evaluation was performed with the test splits of each dataset, as indicated in their respective sources. The specific datasets and the number of training and testing samples, and representative exemplars are shown in Table 2.

Appendix D: Performance metrics and competing methods

Computing performance metrics. We computed three performance metrics for outlier detection [19]: i) area under the receiver-operating-characteristic (AUROC), which represents the area under the curve obtained by plotting the true-positive rate versus the false-positive rate; ii) area under the precision-recall curve (AUPRC), which similarly represents the area under the curve obtained by plotting the precision versus the recall values, and iii) the false-positive rate at 80% true positive rate (FPR@80%TPR). Higher values of the first two metrics indicate better outlier detection,

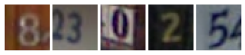
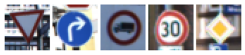
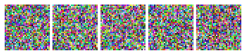
[2] https://www.tensorflow.org/datasets

[3] https://www.kaggle.com/datamunge/sign-language-mnist

[4] http://mmlab.ie.cuhk.edu.hk/datasets/comp_cars/

[5] https://benchmark.ini.rub.de/gtsrb_dataset.html

Table 2. **Dataset details.** Details of datasets used for outlier detection with VAE likelihoods.

Dataset	Type	Exemplars	N-train (N-val)	N-test	License
MNIST	Grayscale		54000 (6000)	10000	CC BY-SA 3.0
Fashion-MNIST	Grayscale		54000 (6000)	10000	MIT
EMNIST-Letters	Grayscale		79920 (8880)	14800	CC BY-SA 3.0
Sign Language MNIST	Grayscale		24720 (2735)	7172	CC0: Public Domain
Gray-Noise	Grayscale		-	10000	-
SVHN	Color		65932 (7325)	26032	Custom (non-commercial) [1]
CelebA	Color		146493 (16277)	19962	Custom (non-commercial) [2]
CompCars	Color		28034 (3114)	13333	Custom (non-commercial) [3]
GTSRB	Color		35289 (3920)	12630	CC0: Public Domain
CIFAR-10	Color		45000 (5000)	10000	MIT
Color-Noise	Color		-	10000	-

[1] http://ufldl.stanford.edu/housenumbers/
[2] http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
[3] http://mmlab.ie.cuhk.edu.hk/datasets/comp_cars/

and vice versa for the third metric. All metrics were computed with the *scikit-learn* library.

For Figures 4 and 5, as well as supplementary results reported in Appendix E, we computed these metrics with an all-versus-all comparison, by comparing each score distribution (e.g. BC-LL) for the test split of the inlier dataset against the respective score distribution for the test split of the outlier dataset (test versus test).

Implementing competing methods. For implementing competing methods, we used the same VAE architecture as in our paper, except that the number of hidden dimensions was set to 100 to match the likelihood regret paper [25]. Other details are specified below.

- *Input complexity (IC).* We computed input complexity by subtracting from the vanilla (uncorrected) log likelihood the "complexity" estimate $L(\mathbf{x})$ for each sample computed as $|C(\mathbf{x})|/d$, where the string of bits $C(\mathbf{x})$ was obtained with the PNG compressor (d is the dimensionality of $\mathbf{x}$). Log likelihoods were computed with the categorical visible distribution.

- *Likelihood ratio (LRat).* We trained a standard VAE and a background model using the noise-corruption procedure described in [19] with the mutation factor set to $\mu = 0.3$ for grayscale image VAEs and $\mu = 0.1$ for natural image VAEs. We also applied a large weight decay ($\lambda = 100$), as suggested by the authors. Log likelihoods were computed with the categorical visible distribution, as in the original study.

- *Likelihood regret (LReg).* We computed the likelihood regret score by quantifying the improvement in the likelihood for each sample by retraining the encoder for 100 epochs using the implementation provided by the authors [25]. Log likelihoods were computed with the categorical visible distribution, as in the original study.

- *Watanabe-Akaike Information Criterion (WAIC).* WAIC was computed with the formula $\mathbb{E}_\theta[\log p_\theta(\mathbf{x})] - \mathrm{Var}_\theta[\log p_\theta(\mathbf{x})]$, using the average log likelihood and variance across the ensemble of six VAEs. Log likelihoods were computed with a continuous Bernoulli visible distribution; the original study [3] used a Bernoulli visible distribution.

For Figures 4 and 5 that compare the performance of these other scores with our scores, AUROC values were computed by comparing each score distribution for the test split of the inlier dataset against the score distribution for test split of the outlier dataset (test versus test). The VAEs for competing approaches were trained 3 times with differ-

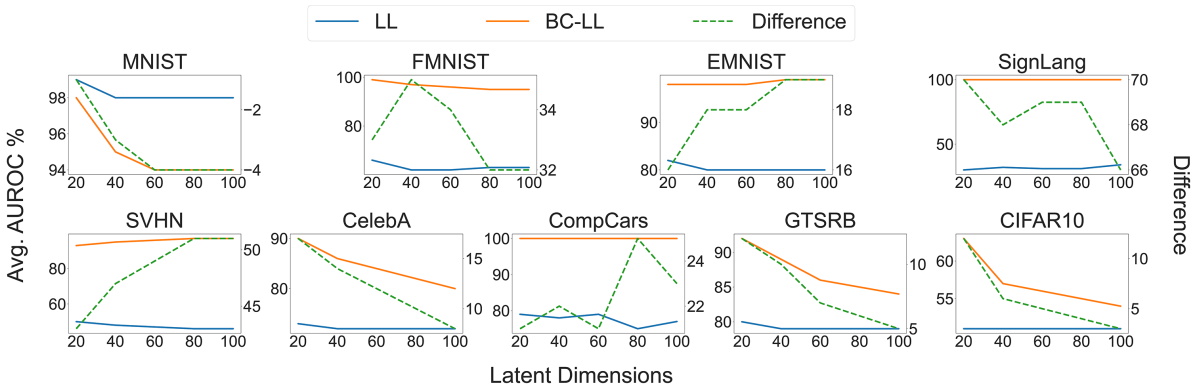


Figure 7. **Robustness of outlier detection performance to VAE architecture.** (*Top row*) AUROC variation with the number of latent dimensions for the four grayscale image VAEs, based on uncorrected log likelihoods (blue, left y-axis), bias-corrected log likelihoods (orange, left y-axis) and their difference (dashed green, right y-axis). AUROC values were averaged across all (n=4) outlier test datasets. (*Bottom row*) Same as in the top panel, but for five natural image VAEs, with AUROC values averaged across n=5 outlier test datasets.

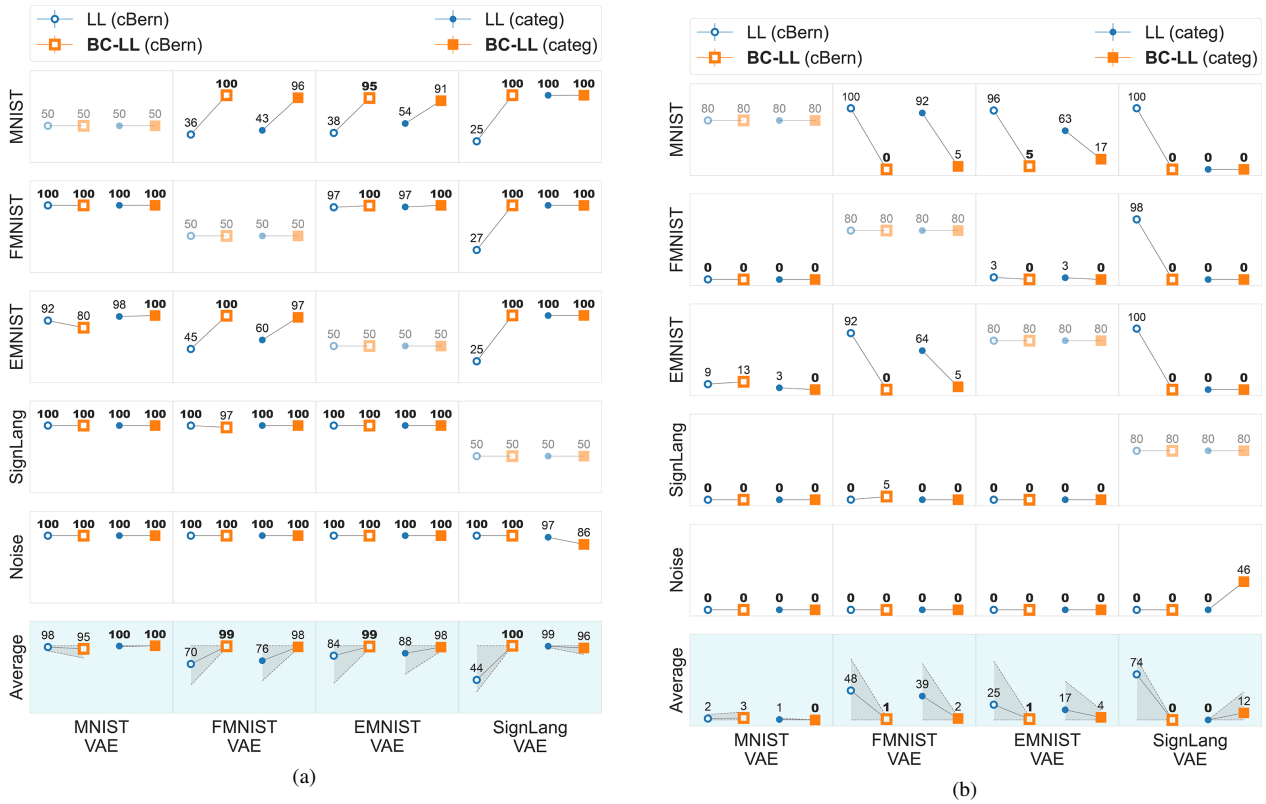


Figure 8. **Additional metrics for outlier detection: Grayscale datasets. (a)** Area under the precision recall curve (AUPRC) (higher is better). **(b)** False-positive rate (FPR) at a true-positive rate (TPR) of 80% (lower is better). Other conventions are the same as in Figure 4.

ent random initializations; we report average AUROC values across 3 runs.

Appendix E: Supplementary results on outlier detection

E.1 Outlier detection performance with varying numbers of VAE latent dimensions

We computed outlier detection AUROC for VAEs trained with different numbers of latent dimensions (nz).

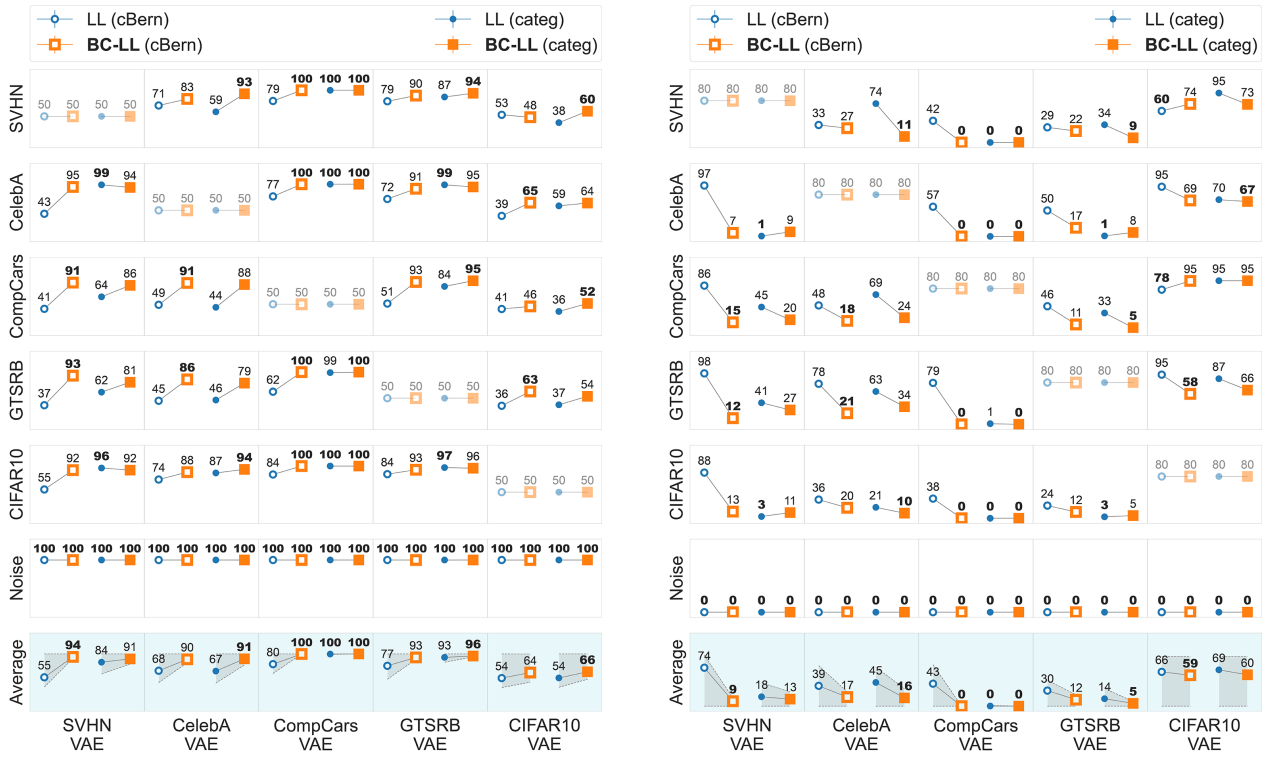


Figure 9. **Additional metrics for outlier detection: Natural image datasets.** Same as in Figures 8a and 8b, but showing additional metrics for outlier detection, **(a)** AUPRC and **(b)** FPR at 80%TPR based on natural image VAE likelihoods. Other conventions are the same as in Figure 8.

AUROC values remained more or less constant, or declined marginally with nz (Fig. 7, blue). In addition, in all cases bias correction improved AUROC values (Fig. 7, orange versus blue).

E.2 Grayscale image datasets: Additional performance metrics

In addition to AUROC (Fig. 4), we computed AUPRC and FPR at 80% TPR for outlier detection with the grayscale VAEs with the bias corrected likelihoods (BC-LL). AUPRC typically improved (Fig. 8a) and FPR decreased (Fig. 8b) for BC-LL scores (orange symbols) compared to uncorrected likelihoods (blue symbols).

E.3 Natural image datasets: Additional performance metrics

In addition to AUROC (Fig. 5), we computed AUPRC and FPR at 80% TPR for outlier detection with the natural image VAEs. AUPRC typically improved (Fig. 9a) and FPR decreased (Fig. 9b) for BC-LL scores (orange symbols) compared to uncorrected likelihoods (blue symbols). As with the AUROC metric (Fig. 5), outlier detection was generally poor with the CIFAR10 VAE.

E.4 Effect of contrast normalization on outlier detection

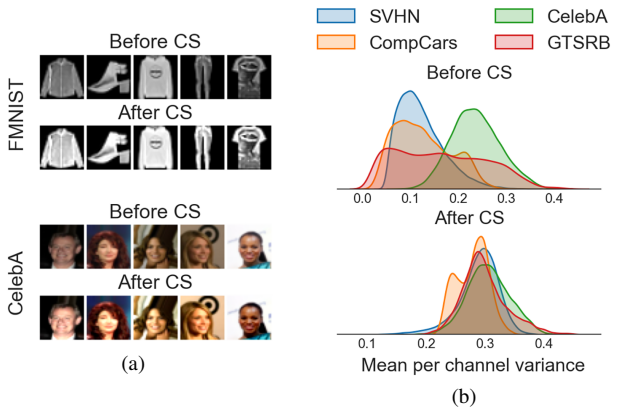


Figure 10. **Effect of contrast stretching on image statistics. (a)** Representative images from the Fashion-MNIST (top) and CelebA (bottom) datasets before (top row in each sub-panel) and after (bottom row in each sub-panel) contrast normalization (stretching). **(b)** Distributions of mean per-channel variance for the natural image datasets show greater overlap after (bottom), as compared to before (top), contrast stretching.

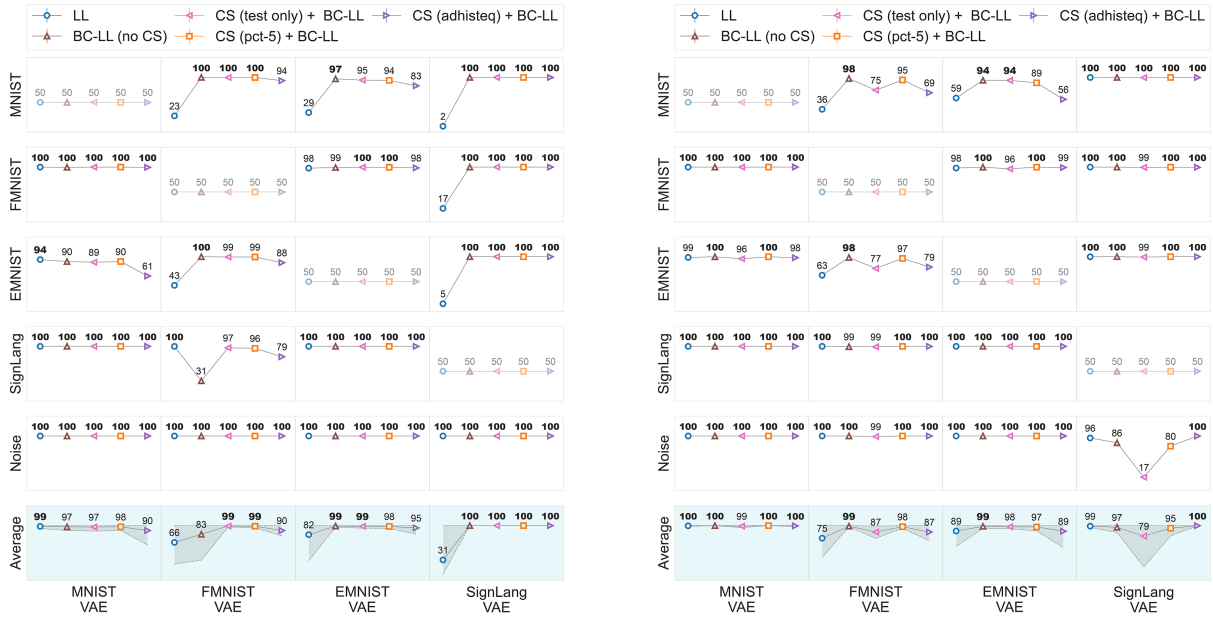


Figure 11. **Effect of contrast normalization on outlier detection: Grayscale datasets.** Outlier detection results using **(a)** continuous Bernoulli and **(b)** Categorical VAE likelihoods trained on grayscale image datasets. We report AUROC values computed with no contrast normalization (BC-LL (no CS), brown symbols), with contrast stretching applied only at test time (CS (test only) + BC-LL, pink symbols), with contrast stretching applied at both train and test time (CS (pct-5) + BC-LL) and with adaptive histogram equalization instead of percentile-based contrast stretching (CS (adhisteq) + BC-LL, magenta symbols), all with bias correction. Blue symbols: AUROC based on uncorrected likelihoods. We use adaptive (local) histogram equalization (adhisteq) for grayscale images because global histogram equalization produced unnatural variations in image contrast. Other conventions are the same as in Figure 4.

E.5 Outlier detection with milder perturbations

We tested the efficacy of bias correction to detect milder corruptions to the data (distribution shifts) at test time. We used continuous Bernoulli VAEs trained on multiple grayscale and natural image datasets. At test time, we corrupted ID images with four noise types – Gaussian noise, impulse noise, shot noise and speckle noise, applied with varying degrees of severity – following the procedure described in [5]. We computed AUROC values for near-OOD detection with the original images from the ID test sets as ID samples and their noise-corrupted variants as OOD samples, for both grayscale (Fig. 13a) and natural image (Fig. 13b) datasets. Bias correction (CS(pct-5) + BC-LL) improved outlier detection performance in most cases, typically in a manner that increased with noise severity level.

E.6 Augmented training baseline

As a baseline, we asked whether training each VAE using augmented samples, with varying contrasts and intensities, would improve outlier detection. We augmented the training samples by varying the pixel intensity uniformly across the image (up to ±75 pixel units), as well as by varying contrast using percentile-based contrast stretching (up to 10%). We trained continuous Bernoulli and categorical VAEs on the augmented training sets for all of the four grayscale and five natural image datasets. We then computed AUROC values for OOD detection on the original test sets (without augmentation); we report the results as a difference in AUROC values relative to those obtained with vanilla VAE likelihoods. In general, augmented training either did not improve, or marginally worsened, AUROC for outlier detection, relative to the vanilla likelihoods (Fig. 14, blue). By contrast, bias correction (CS(pct-5) + BC-LL) yielded significant improvements over vanilla likelihoods in nearly all cases (Fig. 14, orange).

E.7 Effect of preprocessing with ZCA whitening

We experimented with ZCA whitening, as an alternative to percentile based contrast stretching. The ZCA whitening transform was computed and applied for each image, to normalize per channel variance and to decorrelate values across the three color channels. We trained continuous Bernoulli VAEs on ZCA-whitened images and computed outlier detection AUROCs also on whitened images at test time, followed by bias correction. In general, ZCA whitening yielded outlier detection performance comparable with percentile-based contrast stretching for grayscale

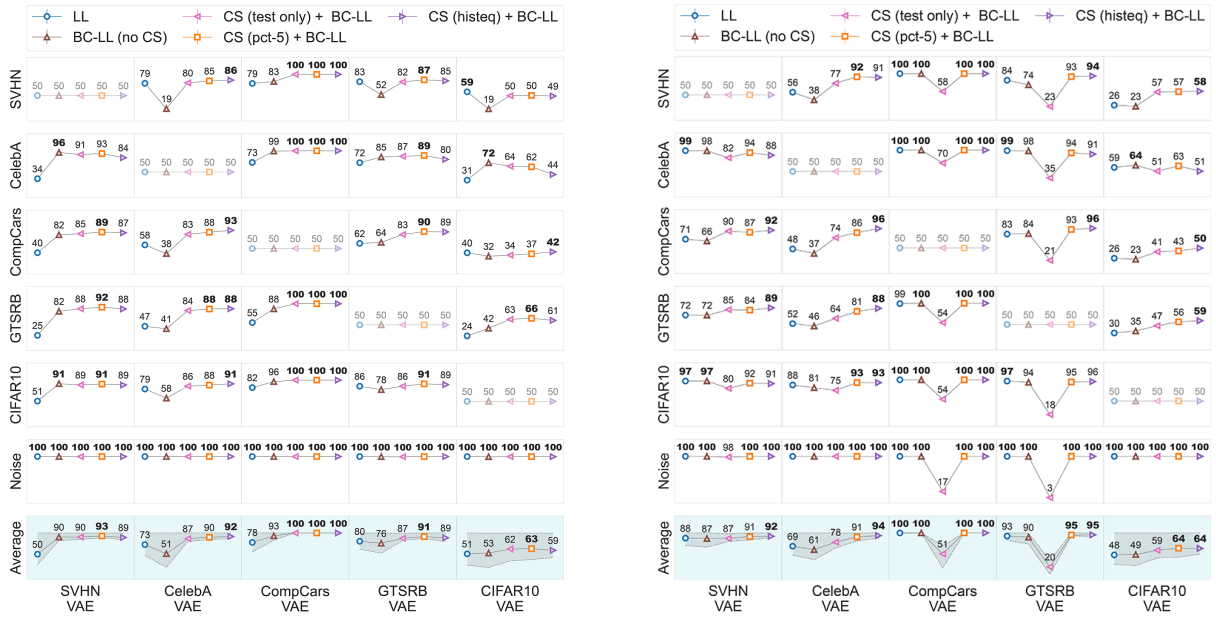


Figure 12. **Effect of contrast normalization on outlier detection: Natural image datasets.** Outlier detection results using **(a)** continuous Bernoulli and **(b)** Categorical VAE likelihoods trained on natural image datasets. Magenta symbols (CS (histeq) + BC-LL): AUROC values computed with (global) histogram equalization instead of percentile-based contrast stretching. Other conventions are the same as in Figure 11.

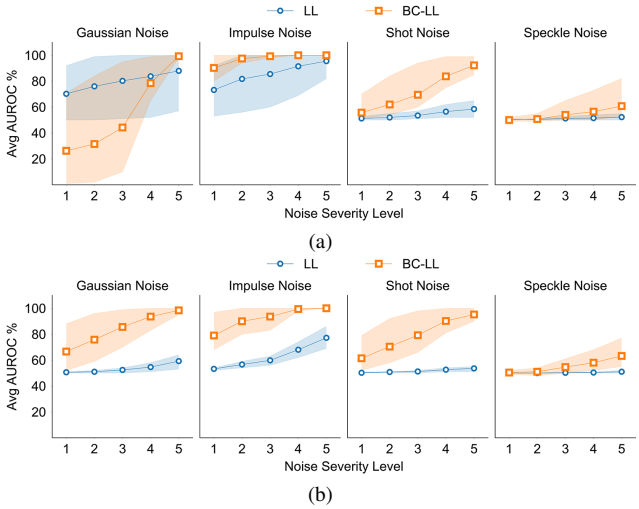


Figure 13. **Outlier detection with milder perturbations**: **(a)** AUROC values for near-OOD detection with four different types of noise – (from left to right) Gaussian, Impulse, Shot and Speckle – at 5 different levels of severity [5]. Values averaged across the four grayscale datasets for each noise type and severity level. **(b)** Same as in (a) but near-OOD AUROC values averaged across the five natural image datasets.

images (Table 3) but worse than contrast stretching for natural image datasets (Table 4).

Table 3. ZCA+BC-LL AUROC values for VAEs trained with grayscale image datasets. Columns indicate training datasets (ID) and rows indicate test datasets (OOD). (MN - MNIST, FM - Fashion MNIST, EM - EMNIST, SL - SignLang. MNIST)

MN	50	100	97	100
FM	100	50	99	100
EM	88	100	50	100
SL	100	42	100	50
Noise	100	100	100	100
OOD ↑ \ ID →	MN	FM	EM	SL

Table 4. ZCA+BC-LL AUROC values for VAEs trained with natural image datasets. Columns indicate training datasets (ID) and rows indicate test datasets (OOD). (SV - SVHN, CA - CelebA, CC - CompCars, GT - GTSRB, CF - CIFAR10.)

SV	50	95	100	98	85
CA	34	50	99	75	49
CC	5	39	50	38	17
GT	38	72	99	50	53
CF	39	73	99	79	50
Noise	100	99	100	100	100
OOD ↑ \ ID →	SV	CA	CC	GT	CF

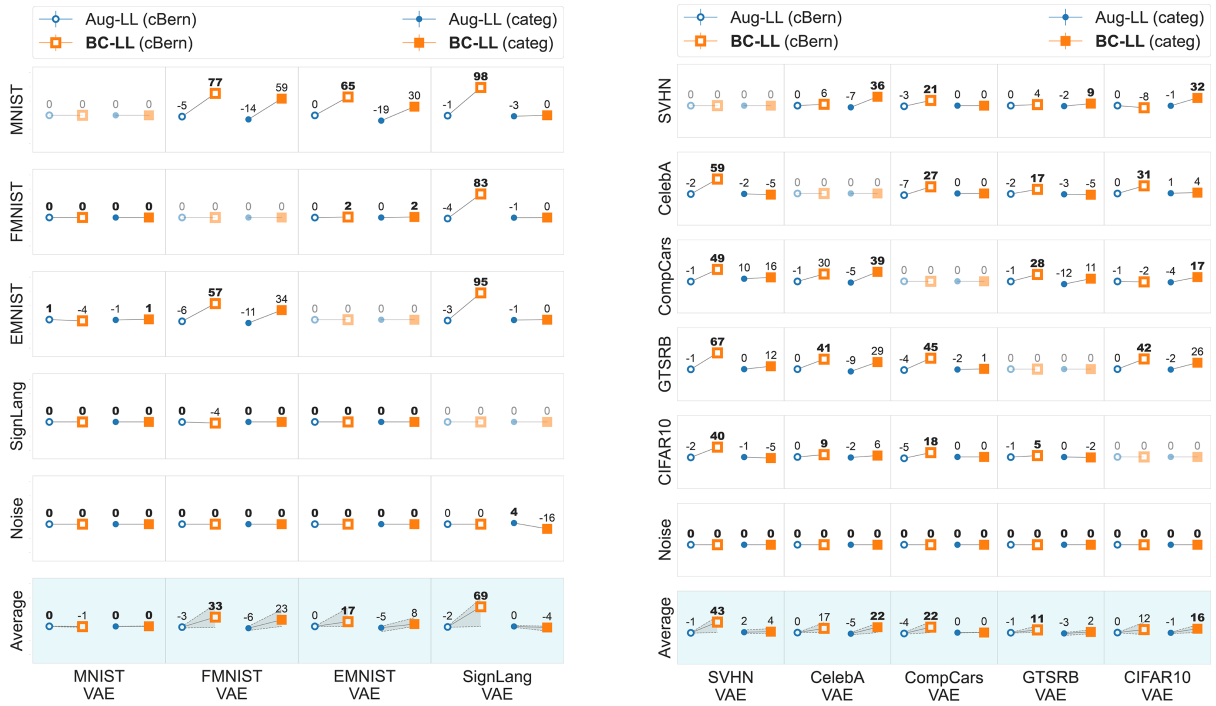


Figure 14. **Augmented training baseline**: Change in AUROC upon training the VAEs using augmented samples, with varying intensities and contrasts. Change in AUROC was measured relative to the vanilla log likelihoods AUROC (blue symbols) for (a) grayscale and (b) natural image datasets. Change in AUROC with bias correction (orange symbols) is also shown for each train-test pair, for reference. Other conventions are the same as in Figures 4 and 5.

Appendix F: Bias in alternative visible distributions

F.1 Bias in the Categorical visible distribution

In Figure 15, we revisit the empirical bias in the log likelihood of a CelebA VAE with Categorical visible distribution, for images with different uniform pixel intensities. We plot the VAE decoder categorical distribution outputs for 5 different target pixel intensity values across the range (0-255), by averaging across all occurrences of the respective pixel value in the CelebA test set (Fig. 15). The U-shaped log likelihood profile can be explained by the discrepancy in the entropy of these distributions across different pixel values. For example, for target pixel values close to full black or full white, the VAE decoder concentrates probability mass in the proximity of the target pixel value, with the output having narrower peaks and smaller overall entropy. For target values closer to the middle of the range (grays), the probability mass is more dispersed around the target pixel value, and has wider peaks and larger overall entropy in the output. These discrepancies in the output entropy across different pixel intensities arise from "edge effects": probability mass cannot be assigned to values <0 or >255, and the PMF has to sum to one for all target pixel outputs, resulting in accumulation of probability mass at the edges.

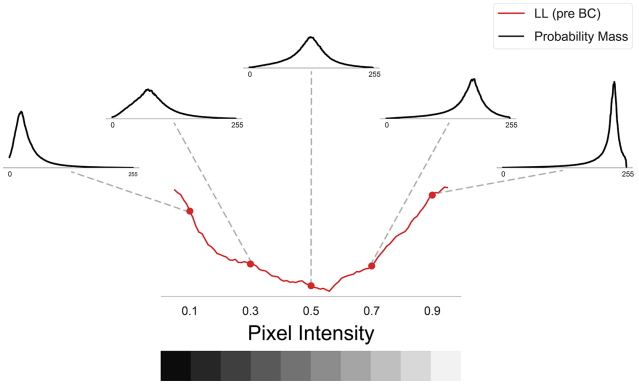


Figure 15. **Empirical bias arising from pixel intensities for a Categorical VAE.** (Red curve) Uncorrected log likelihoods for uniform pixel intensity images. (Insets, black) PMF plots: Categorical distribution outputs for specific target pixel values (from left to right) 25, 76, 127, 178 and 229

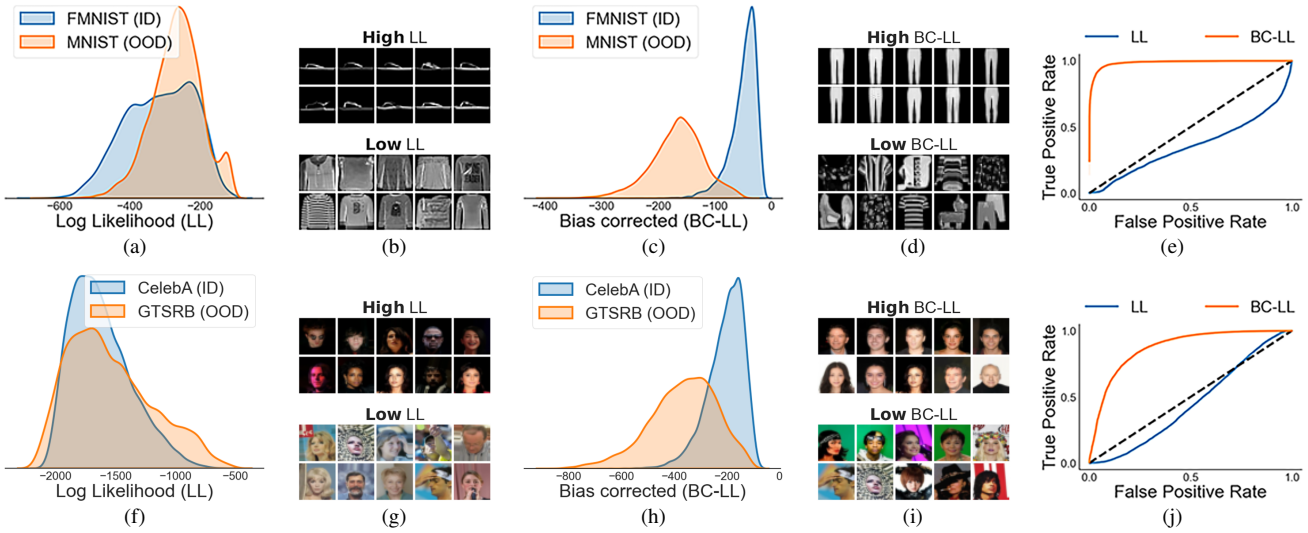


Figure 16. **Bias correction improves outlier detection for the Bernoulli VAE.** Conventions are same as in Figure 2, but for a VAE trained with the Bernoulli visible distribution.

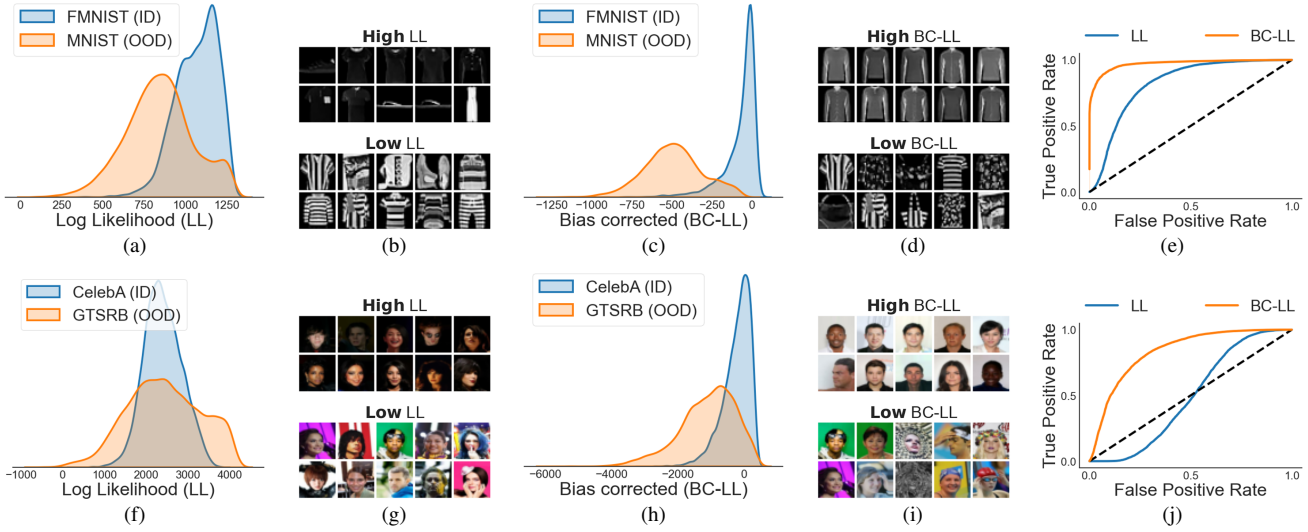


Figure 17. **Bias correction improves outlier detection for the truncated Gaussian VAE.** Conventions are same as in Figure 2, but for a VAE trained with the truncated Gaussian visible distribution.

F.2 Bias correction with alternative visible distributions

We report outlier detection results for Bernoulli (Fig. 16) and truncated Gaussian (Fig. 17) VAEs. To correct for the intensity bias in the Bernoulli VAE, we use the analytical approach discussed in Section 3.1 and Appendix B. For truncated Gaussian VAEs, we used the algorithmic correction discussed in Section 3.2 and Algorithm 1. For all VAEs, the images were contrast stretched during both training and testing phases.

Appendix G: Outlier detection with the CIFAR-10 VAE

Our BC-LL scores approached or exceeded state-of-the-art accuracies for outlier detection with multiple grayscale and natural image datasets (Figs. 4 and 5). Nonetheless, VAEs trained on the CIFAR-10 dataset yielded relatively low AUROC values with bias correction, ranging from 37-66 (Fig. 5, last column, all outlier datasets except Noise). Interestingly, other approaches based on VAE likelihoods also performed relatively poorly with this dataset (Fig. 5, last column)

We hypothesized that this failure could be due to the het-

erogeneity of the CIFAR-10 dataset: this dataset comprises 10 different categories of images, which are both visually and semantically unrelated to each other (e.g. airplanes, deer, frogs, ships). We tested this hypothesis by training four category-specific VAEs with images from four CIFAR-10 image categories (Airplane, Ship, Frog and Deer) and tested them against the other datasets. Outlier detection performance improved with bias correction for the VAEs trained on specific categories of images (Fig. 18, last 4 columns) relative to the one trained with all categories of images (Fig. 18, first column). On average, category-specific outlier detection with the BC-LL score improved by between 6-20 points (Fig. 18, last row) for the individual categories compared to the VAE trained on all categories.

These results suggest that the heterogeneity in CIFAR-10 categories was, in part, responsible for overall poor outlier detection with this dataset. Yet, even with these category-specific CIFAR-10 VAEs, outlier detection performance did not reach the superlative levels of accuracy that it did with the other natural image datasets (e.g. Fig. 5, CompCars, GTSRB).

Why do VAE likelihoods fail when other classifier-based outlier detection methods (e.g. [11]) succeed? To answer this question, we note that there is a key difference between VAEs and deep CNN classifiers. Deep CNN classifiers transform the input image into a spatially-invariant (typically translation-invariant) feature representation, on which a classification decision must be made. Even though the VAE encoder employs a CNN that achieves such a transformation, the VAE decoder needs to reconstruct the entire image pixel-for-pixel. Consequently, VAEs can ill-afford to ignore spatial relationships among the features and their positions relative to a coordinate frame that is locked to the "edge" of the image. In other words, to optimize its objective, a VAE must care not only about "what" features are present in an image, it must also encode "where" these features are positioned relative to each other and, importantly, relative to the image's bounding box. We propose that datasets in which such spatial relationships do not occur consistently (e.g. CIFAR-10) are particularly unsuited for outlier detection with VAEs.

We test this prediction with a simple set of experiments. We plotted the bias corrected likelihoods for representative VAEs (GTSRB, CelebA, CompCars), each following one simple affine transformation: a) x/y translation (uniform random shift of ±0-10 pixels along both x and y directions) (Fig. 19, top left, GTSRB), b) reflection about the horizontal axis (Fig. 19, top right, CelebA) or c) rotation by 90 degrees anti-clockwise (Fig. 19, bottom, CompCars). Each of these affine operations sufficed to hoodwink the VAEs into treating their, respective, inlier images as outliers: the VAEs consistently assigned lower likelihoods for these translated, reflected or rotated in-distribution images

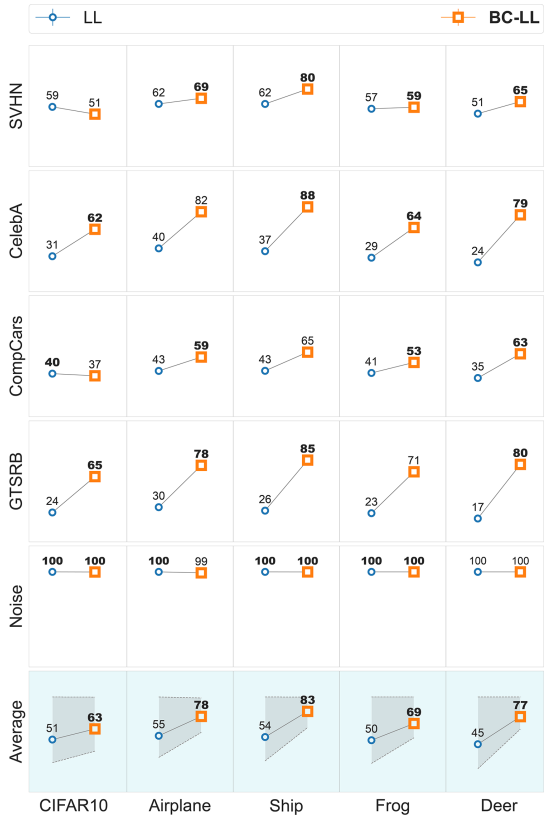


Figure 18. **Outlier detection with specific CIFAR-10 categories.** AUROC values for outlier detection based on VAEs trained with specific categories of images in the CIFAR-10 dataset. (Leftmost to rightmost columns) VAEs trained with all CIFAR-10 image categories, Airplane images, Ship images, Frog images and Deer images. Each row is an outlier dataset. Other conventions are the same as in Figure 5.

(Fig. 19, orange distributions), as compared to their original counterparts (Fig. 19, blue distributions). This was particularly surprising for the case of perturbation by translation, to which the encoder CNN typically learns invariance. In sum, even though all image features were exactly (or nearly) identical between the original and affine transformed images, VAE outlier detection failed because perturbed image features were in novel spatial positions relative to the edge of the image.

We discuss a few possibilities to overcome this limitation. One solution is to train the VAE, not directly on natural images, but after transforming them into a spatially-invariant "semantic" feature representation. Indeed, a recent study [9] attempted precisely this experiment with deep generative (Flow) models using features from a pretrained EfficientNet model, and showed that such features enabled robust outlier detection, even with the CIFAR-10 dataset. A second possibility is to perturb the output [5] and develop reconstruction error metrics that are invariant to these perturbations in the input data. These remedies could en-

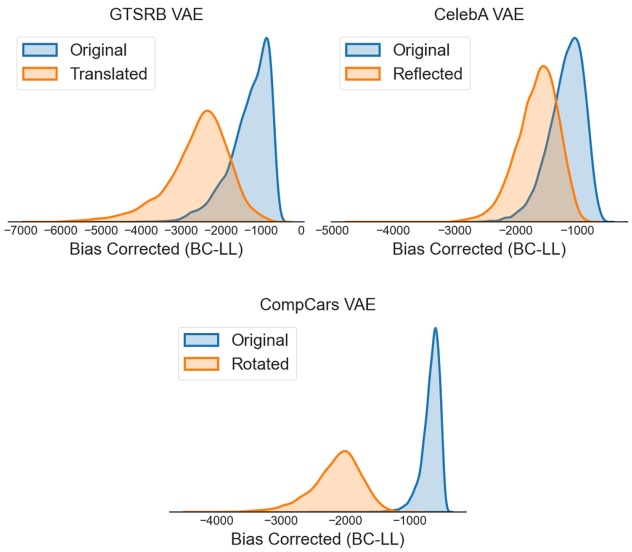


Figure 19. **VAE likelihoods following affine transformations to input data**. **(Top Left)** GTSRB VAE likelihoods (bias corrected) for the original images (blue distribution) or for the images following small, random wrap-around x/y translations (orange distribution). **(Top Right)** CelebA VAE likelihoods (bias corrected) for the original images (blue distribution) or for the images flipped vertically (orange distribution). **(Bottom)** CompCars VAE likelihoods (bias corrected) for the original images (blue distribution) or for the images rotated by 90 degrees anti-clockwise (orange distribution).

able VAEs to be employed in dynamic real-world settings where objects of interest may not be stationary relative to the edge of the bounding frame, for example, outlier detection in video data.