The SVHN Dataset Is Deceptive for Probabilistic Generative Models Due to a Distribution Mismatch

Tim Z. Xiao[1,2,*] **Johannes Zenn**[1,2,*] **Robert Bamler**[1]
[1]University of Tübingen [2]IMPRS-IS
[*]Equal contribution, order determined by coin flip.
{zhenzhong.xiao, johannes.zenn, robert.bamler}@uni-tuebingen.de

Abstract

The Street View House Numbers (SVHN) dataset [22] is a popular benchmark dataset in deep learning. Originally designed for digit classification tasks, the SVHN dataset has been widely used as a benchmark for various other tasks including generative modeling. However, with this work, we aim to warn the community about an issue of the SVHN dataset as a benchmark for generative modeling tasks: we discover that the official split into training set and test set of the SVHN dataset are not drawn from the same distribution. We empirically show that this distribution mismatch has little impact on the classification task (which may explain why this issue has not been detected before), but it severely affects the evaluation of probabilistic generative models, such as Variational Autoencoders and diffusion models. As a workaround, we propose to mix and re-split the official training and test set when SVHN is used for tasks other than classification. We publish a new split and the indices we used to create it at https://jzenn.github.io/svhn-remix/.

1 Introduction

The Street View House Numbers (SVHN) dataset [22] is a popular benchmark datasets originated that from computer vision. SVHN consists of real-world images from house numbers found on Google Street View and has 10 classes, one for each digit. It is often treated as a more difficult variant of the MNIST dataset [14]. The dataset is divided into a training set $\mathcal{D}_{\text{train}}$ with 73,257 samples, a test set $\mathcal{D}_{\text{test}}$ with 26,032 samples, and a less used extra training set of 531,131 simpler samples. In addition to classification tasks [29, 7, 16, 17], SVHN also serves as a benchmark for tasks such as generative modeling [1, 18], out-of-distribution detection [15, 30, 35, 21], and adversarial robustness [11, 32].

As a toy dataset consisting of color images, SVHN is often used during the development of new generative models such as Generative Adversarial Networks (GANs; [2]), Variational Autoencoders (VAEs; [10]), normalizing flows [23], and diffusion models [33, 5]. To evaluate generative models, one commonly measures the sample quality using Fréchet Inception Distance (FID; [4]) and Inception Score (IS; [26]). In the following, we refer to likelihood-based generative models (like VAEs, normalizing flows, and diffusion models but not, e.g., GANs) as *probabilistic* generative models as they explicitly model a probability distribution $p_\theta(\boldsymbol{x})$, where θ are the model parameters. One typically also evaluates these models using (an approximation of) their likelihoods on test data. This test set likelihood indicates how closely a model $p_\theta(\boldsymbol{x})$ approximates the true (usually inaccessible) data distribution $p_{\text{data}}(\boldsymbol{x})$ from which data points are drawn. It becomes particularly relevant if we want to use the model for tasks such as out-of-distribution detection and lossless data compression.

Surprisingly, we discover that SVHN, as a popular benchmark, has a *distribution mismatch* between its training set $\mathcal{D}_{\text{train}}$ and its test set $\mathcal{D}_{\text{test}}$. In other words, $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ do not seem to come from the same distribution. We find that this mismatch has little effect on classification tasks for supervised learning, or on sample quality for generative modeling. However, we show that for probabilistic

Workshop on Distribution Shifts, 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

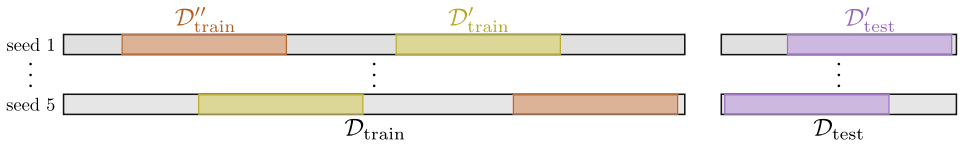


Figure 1: Five random splits (with reshuffle) of $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ into $\mathcal{D}'_{\text{train}}$, $\mathcal{D}''_{\text{train}}$, and $\mathcal{D}'_{\text{test}}$.

generative models such as VAEs and diffusion models, the mismatch leads to a false assessment of model performance when evaluating test set likelihoods: test set likelihoods on the SVHN dataset are deceptive since $\mathcal{D}_{\text{test}}$ appears to be drawn from a simpler distribution than $\mathcal{D}_{\text{train}}$. As a workaround, we merge the original $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$, then shuffle and re-split them. We empirically show that this remixing solves the problem of distribution mismatch, thus restoring the SVHN test set likelihood as an informative metric for probabilistic generative models. We also publish the new split we used in our experiments as a proposal of a canonical split for future research on generative models.

2 Distribution Mismatch in SVHN

In this section we show evidence for the distribution mismatch in SVHN between the training set $\mathcal{D}_{\text{train}}$ and the test set $\mathcal{D}_{\text{test}}$. We downloaded the SVHN dataset from its official website[1], which is also the default download address used by `Torchvision` [20] and `TensorFlow Datasets` [19].

2.1 Defining Distribution Mismatch

We often assume $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ in a benchmark dataset consists of i.i.d. samples from an underlying data distribution $p_{\text{data}}(\boldsymbol{x})$. Thus, given a distance metric $D(p_1(\boldsymbol{x}), p_2(\boldsymbol{x}))$ that measures the dissimilarity between two distributions $p_1(\boldsymbol{x})$ and $p_2(\boldsymbol{x})$, we expect that

$$D\big(p_{\text{data}}(\boldsymbol{x}), \mathcal{D}'_{\text{train}}\big) \approx D\big(p_{\text{data}}(\boldsymbol{x}), \mathcal{D}'_{\text{test}}\big) \tag{1}$$

and that both sides have a low value. Here, $\mathcal{D}'_{\text{train}}$ and $\mathcal{D}'_{\text{test}}$ are equally sized random subsets of $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$, respectively (this will simplify comparisons both within and across datasets below). In practice, we typically do not have access to $p_{\text{data}}(\boldsymbol{x})$. But we can bypass this issue by drawing an additional random subset $\mathcal{D}''_{\text{train}}$ of $\mathcal{D}_{\text{train}}$, which has the same size as $\mathcal{D}'_{\text{train}}$ and does not overlap with it (see Figure 1). Then, Eq. (1) also holds if we replace $\mathcal{D}'_{\text{train}}$ with $\mathcal{D}''_{\text{train}}$, and by combining the two variants of Eq. (1) for $\mathcal{D}'_{\text{train}}$ and $\mathcal{D}''_{\text{train}}$, we find by the triangle inequality,

$$D\big(\mathcal{D}''_{\text{train}}, \mathcal{D}'_{\text{train}}\big) \approx D\big(\mathcal{D}'_{\text{train}}, \mathcal{D}'_{\text{test}}\big) \tag{2}$$

and that, again, both sides should have a low value. Conversely, if $D(\mathcal{D}''_{\text{train}}, \mathcal{D}'_{\text{train}})$ differs substantially from $D(\mathcal{D}''_{\text{train}}, \mathcal{D}'_{\text{test}})$, it indicates a distribution mismatch. Note that Eq. (2) is a necessary but not a sufficient condition for matching distributions.

2.2 Evaluation Using Sample Quality Measures

To detect a distribution mismatch between any two sets of images, we need to find a well-motivated distance metric $D(\,\cdot\,,\cdot\,)$ to use in Eq. (2). Here, we draw inspiration from the Fréchet Inception Distance (FID) [4], which is usually used to measure sample quality, but which we repurpose for our setup. We contrast FID to Inception Score (IS) [28], which does *not* compare two sets of images.

Fréchet Inception Distance (FID) measures semantic dissimilarity between two finite sets $\mathcal{D}_1$ and $\mathcal{D}_2$ of images. One first maps both sets into a semantic feature space using a feature extractor $f(\cdot)$. Then, one computes the feature means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ and covariances $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$, which parameterize two Gaussian distributions. FID is defined as the Fréchet distance between these two Gaussians,

$$\text{FID}(\mathcal{D}_1, \mathcal{D}_2) = \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|_2^2 + \text{Tr}\left(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2 - 2\left(\boldsymbol{\Sigma}_1\boldsymbol{\Sigma}_2\right)^{1/2}\right). \tag{3}$$

For image data, $f(\cdot)$ is most commonly the activation at the penultimate layer of an Inception classifier [34] that was pre-trained on ImageNet. A lower FID indicates higher semantic similarity.

[1] http://ufldl.stanford.edu/housenumbers/

Table 1: FID (lower means larger similarity) and IS (higher means better sample quality) on three datasets, averaged over 5 random seeds. For SVHN, we find that the FID between random subsets of the training and test set (bold red) is significantly higher than the FID between non-overlapping subsets of the training set of the same size, while IS for $\mathcal{D}'_{\text{train}}$ and $\mathcal{D}'_{\text{test}}$ is similar within all datasets.

FID (↓), IS (↑)	SVHN	SVHN-Remix	CIFAR-10
$\text{FID}(\mathcal{D}''_{\text{train}}, \mathcal{D}'_{\text{train}})$	3.309 ± 0.029	3.334 ± 0.018	5.196 ± 0.040
$\text{FID}(\mathcal{D}''_{\text{train}}, \mathcal{D}'_{\text{test}})$	$\mathbf{16.687 \pm 0.325}$	3.326 ± 0.015	5.206 ± 0.031
$\text{IS}(\mathcal{D}'_{\text{train}} \mid \bar{\mathcal{D}}_{\text{train}})$	8.507 ± 0.114	8.348 ± 0.568	7.700 ± 0.043
$\text{IS}(\mathcal{D}'_{\text{test}} \mid \bar{\mathcal{D}}_{\text{train}})$	8.142 ± 0.501	8.269 ± 0.549	7.692 ± 0.023

FID is commonly used to evaluate sample quality by comparing samples $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$ from a trained generative model to $\mathcal{D}_{\text{train}}$. We instead apply FID directly as the distance metric D on both sides of Eq. (2), without training any generative model. Thus, we randomly sample (without replacement) three subsets $\mathcal{D}'_{\text{train}}$, $\mathcal{D}''_{\text{train}}$, and $\mathcal{D}'_{\text{test}}$ from $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$, respectively, each of size $10,000$. Then, we assess Eq. (2) by calculating $\text{FID}(\mathcal{D}''_{\text{train}}, \mathcal{D}'_{\text{train}})$ and $\text{FID}(\mathcal{D}''_{\text{train}}, \mathcal{D}'_{\text{test}})$. We repeat this procedure over 5 different random seeds as illustrated in Figure 1 and report FID means and standard deviations in Table 1. For comparison, we also evaluate CIFAR-10 [12] using the same procedure.

Table 1 shows that $\text{FID}(\mathcal{D}''_{\text{train}}, \mathcal{D}'_{\text{train}})$ differs significantly from $\text{FID}(\mathcal{D}''_{\text{train}}, \mathcal{D}'_{\text{test}})$, violating the necessary condition Eq. (2), therefore indicating that there is a distribution mismatch between $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ for SVHN. As a comparison, for CIFAR-10, the two FIDs are basically indistinguishable.

Inception Score (IS) is another common measure of sample quality. Unlike FID, IS does not build on a similarity metric between samples. Instead, it evaluates how well data points in a set $\mathcal{D}$ can be classified with a classifier $p_{\text{cls.}}(y \mid \boldsymbol{x})$ that was trained on $\mathcal{D}_{\text{train}}$, and how diverse their labels y are,

$$\text{IS}(\mathcal{D} \mid \mathcal{D}_{\text{train}}) = \exp\left(\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[D_{\text{KL}}\left[p_{\text{cls.}}(y \mid \boldsymbol{x}) \parallel p_{\text{cls.}}(y)\right]\right]\right), \tag{4}$$

where D_{KL} denotes the Kullback-Leibler divergence [13], and $p_{\text{cls.}}(y) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[p_{\text{cls.}}(y \mid \boldsymbol{x})]$.

Similar to FID discussed above, the generative modeling literature typically applies IS to samples $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$ from a trained generative model. We instead apply it directly to subsets of the SVHN dataset to measure their quality. We randomly sample (without replacement) subsets $\mathcal{D}'_{\text{train}}$ and $\mathcal{D}'_{\text{test}}$ of $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$, respectively, each with size $M = 10,000$. We then train a classifier $p_{\text{cls.}}(y \mid \boldsymbol{x})$ on $\bar{\mathcal{D}}_{\text{train}} := \mathcal{D}_{\text{train}} \backslash \mathcal{D}'_{\text{train}}$, and we evaluate the IS on both $\mathcal{D}'_{\text{train}}$ and $\mathcal{D}'_{\text{test}}$. See Appendix B for the model architecture of $p_{\text{cls.}}(y \mid \boldsymbol{x})$. We follow the same procedure for CIFAR-10.

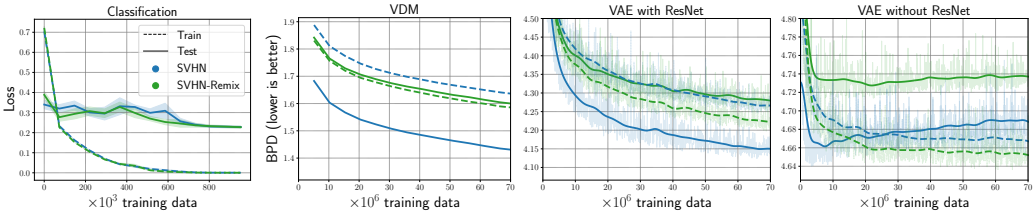
The results in Table 1 show that there is not much difference between $\text{IS}(\mathcal{D}'_{\text{train}} \mid \bar{\mathcal{D}}_{\text{train}})$ and $\text{IS}(\mathcal{D}'_{\text{test}} \mid \bar{\mathcal{D}}_{\text{train}})$ for both SVHN and CIFAR-10. Thus, in terms of the class distribution $p(y)$ and the data quality, $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ are similar for both SVHN and CIFAR-10, which is expected for classification benchmark datasets. It also tells us that if we want to measure the sample quality in terms of distribution similarity, we should not use IS as the metric.

3 SVHN-Remix

As a workaround to alleviate distribution mismatch in SVHN, we propose a new split called SVHN-Remix, which we created by joining the original $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$, random shuffling, and re-splitting them into $\mathcal{D}^{\text{remix}}_{\text{train}}$ and $\mathcal{D}^{\text{remix}}_{\text{test}}$. We make sure the size of the new training and test set is the same as before, and the number of samples for each class is also preserved for both the new training and test set. We evaluated SVHN-Remix using the same procedures as in Section 2.2 using FID and IS. The results in Table 1 show that $\text{FID}(\mathcal{D}^{\text{remix}\prime\prime}_{\text{train}}, \mathcal{D}^{\text{remix}\prime}_{\text{train}})$ is now very similar to $\text{FID}(\mathcal{D}^{\text{remix}\prime\prime}_{\text{train}}, \mathcal{D}^{\text{remix}\prime}_{\text{test}})$, i.e., just like CIFAR-10 it now satisfies Eq. (2). The result also shows that our remixing does not impact the IS, which further indicates that IS cannot be used for detecting distribution mismatch.

4 Implications on Supervised Learning

In the classification setting, we want to learn a conditional distribution $p_{\text{cls.}}(y \mid \boldsymbol{x})$ by maximizing the cross entropy $\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_{\text{train}}}[p_{\text{cls.}}(y \mid \boldsymbol{x})]$. We train classifiers on the training sets of both the



(a) Classification (b) VDM (c) VAEs

Figure 2: (a): classification loss evaluated on training set (dashed) and test set (solid) on SVHN (blue) and SVHN-Remix (green) for five random seeds (lines are means, shaded areas are $\pm\sigma$). The losses are similar (details in Section 4). (b) and (c): BPD evaluated as a function of training progress on the training set (dotted) and test set (solid) for SVHN (blue) and SVHN-Remix (green). For SVHN, the order of training and test set performance is flipped compared to SVHN-Remix (details in Section 5).

original SVHN and of SVHN-Remix (for classifier details see Appendix B). The results in Figure 2(a) show that the distribution mismatch in SVHN between $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ has minimal impact on the classification loss. This observation is consistent with the IS (which also relies on a classifier), where $\text{IS}(\mathcal{D}'_{\text{train}})$ and $\text{IS}(\mathcal{D}'_{\text{test}})$ are also not affected by the proposed remixing. Therefore, we suspect that the distribution mismatch in SVHN happens in-distribution, i.e., $\mathcal{D}_{\text{test}}$ covers only a subset of $\mathcal{D}_{\text{train}}$ but more densely (we further verify this in Section 5). Note other conceivable forms of distribution mismatch, such as $\mathcal{D}_{\text{test}}$ being out-of-distribution for $\mathcal{D}_{\text{train}}$, would seriously impact the classification performance of $p_{\text{cls.}}(y \mid \boldsymbol{x})$ on $\mathcal{D}_{\text{test}}$. The results in Figure 2(a) explain why the issue of distribution mismatch in SVHN has not been identified earlier, as SVHN is mostly used for classification.

5 Implications on Probabilistic Generative Models

Probabilistic generative models $p_\theta(\boldsymbol{x})$, such as Variational Autoencoders (VAEs; [10, 23]) and diffusion models [31, 33], aim to explicitly model the underlying data distribution $p_{\text{data}}(\boldsymbol{x})$. These probabilistic models are the backbone of foundation models. Developing and fast prototyping such models often involves using small benchmark datasets like SVHN. We normally evaluate these models by their likelihood on the test set, i.e., $p_\theta(\mathcal{D}_{\text{test}})$. Hence, a distribution mismatch between $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ can lead to false evaluation of these models.

In order to see the impact of distribution mismatch on probabilistic generative models, we train and evaluate a variational diffusion model (VDM; [9]) and two VAEs (one with ResNet [5] architecture, one without) on both SVHN and SVHN-Remix (details see Appendix A). We report model performance in bits per dimension (BPD), which is proportional to negative log-likelihood of the model on a dataset. Lower BPD corresponds to higher likelihood. Figure 2(b) and (c)(left) show that both VDM and VAE with ResNet have much lower BPD on $\mathcal{D}_{\text{test}}$ than on $\mathcal{D}_{\text{train}}$ (solid blue) when trained with SVHN. *This means that both models have higher likelihood on $\mathcal{D}_{\text{test}}$, even though they are trained to maximize the likelihood on $\mathcal{D}_{\text{train}}$.* This is exactly due to the distribution mismatch, where $\mathcal{D}_{\text{test}}$ is in-distribution for $\mathcal{D}_{\text{train}}$ but has much higher density on the 'easy' data. Otherwise, we would expect a slightly lower BPD on $\mathcal{D}_{\text{train}}$ than on $\mathcal{D}_{\text{test}}$, as we indeed observe when using SVHN-Remix (green lines). Additionally, Figure 2(c)(right) shows the results for the VAE without ResNet, where the solid blue line first goes below the dashed blue line, then goes above it. This occurs because the VAE begins to overfit $\mathcal{D}_{\text{train}}$, which does not invalidate our previous findings. In summary, when there is distribution mismatch between $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$, the likelihood evaluated on $\mathcal{D}_{\text{test}}$ can be misleading and does not provide meaningful information on the generalization performance of the model.

6 Conclusion

In this paper, we show that there is a distribution mismatch between the training set and test set in the SVHN dataset. This distribution mismatch affects the evaluation of probabilistic generative models such as VAEs and diffusion models, but does not harm classification. We provide a new split of the SVHN dataset resolving this issue. In a broader sense, this tells us that when creating benchmark datasets for (probabilistic) foundation models we have to be mindful of a distribution mismatch.

Acknowledgments

The authors would like to thank Takeru Miyato, Yingzhen Li, and Andi Zhang for helpful discussions. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC number 2064/1 – Project number 390727645. This work was supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A. Robert Bamler acknowledges funding by the German Research Foundation (DFG) for project 448588364 of the Emmy Noether Programme. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Tim Z. Xiao and Johannes Zenn.

Reproducibility Statement. We publish the new split and the indices we used to create it at https://jzenn.github.io/svhn-remix/.

References

[1] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2016. 1

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 2020. 1

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 8

[4] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017. 1, 2

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020. 1, 4

[6] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 8

[7] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems*, 2016. 1

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015. 8

[9] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, 2021. 4, 8

[10] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014. 1, 4

[11] Jernej Kos, Ian Fischer, and Dawn Song. Adversarial examples for generative models. In *IEEE Security and Privacy Workshop*, 2018. 1

[12] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 3

[13] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22, 1951. 3

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998. 1

[15] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, 2018. 1

[16] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 1

[17] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international Conference on Computer Vision*, 2017. 1

[18] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International Conference on Machine Learning*, 2016. 1

[19] TensorFlow Datasets maintainers and contributors. TensorFlow Datasets, a collection of ready-to-use datasets. https://github.com/tensorflow/datasets/blob/v4.9.3/tensorflow_datasets/datasets/svhn_cropped/svhn_cropped_dataset_builder.py. 2

[20] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision/blob/v0.15.2/torchvision/datasets/svhn.py, 2016. 2

[21] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2018. 1

[22] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 1

[23] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 2015. 1, 4

[24] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951. 8

[25] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323, 1986. 8

[26] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016. 1

[27] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: Improving the pixelCNN with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*, 2017. 8

[28] Tim Salimans, Han Zhang, Alec Radford, and Dimitris Metaxas. Improving gans using optimal transport. In *International Conference on Learning Representations*, 2018. 2

[29] Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *International Conference on Pattern Recognition*, 2012. 1

[30] Joan Serrà, David Álvarez, Vicenç Gómez, Olga Slizovskaia, José F Núñez, and Jordi Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. *arXiv preprint arXiv:1909.11480*, 2019. 1

[31] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015. 4

[32] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In *Advances in Neural Information Processing Systems*, 2018. 1

[33] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. 1, 4

[34] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2015. 2

[35] Zhisheng Xiao, Qing Yan, and Yali Amit. Likelihood regret: An out-of-distribution detection score for variational auto-encoder. In *Advances in Neural Information Processing Systems*, 2020. 1

[36] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern recognition*, 2010. 8

A Details on the Probabilistic Generative Models

A.1 VAEs

The VAEs in this work use standard Gaussian priors and diagonal Gaussian distributions for the inference network. Our VAEs are trained with two architectures: a residual architecture [3] and a non-residual architecture. The generative model uses a a discretized mixture of logistics (MoL) likelihood [27].

The VAE with residual architecture has two convolutional layers (kernel size: 4, stride: 2, padding: 1), a residual layer, and a final convolutional layer. The resulting latent has a dimension of 64. The residual layer sequentially processes the input by two convolutional layers (kernel size: 3, stride: 1, padding: 1 and kernel size: 1, stride: 1, padding: 0). For all convolutional layers we use BatchNorm [8]. The decoder mirrors the architecture of the encoder.

The non-residual architecture passes the input through three convolutional layers ((i) kernel size: 3, stride: 2, padding: 1; (ii) kernel size: 4, stride: 2, padding: 1; (iii) kernel size: 5, stride: 2, padding: 1) and maps the flattened output with a fully-connected layer to mean and variance, respectively. The latent dimension is 20. The decoder mirrors the architecture of the encoder (three convolutional layers with (i) & (ii) kernel size: 6, stride: 2, padding: 2; (iii) kernel size: 5, stride: 2, padding: 1) but uses transposed convolutions [36].

A.2 Diffusion Model

We use an open source implementation[2] of Variational Diffusion Models [9]. We train two diffusion models (on SVHN and SVHN-Remix), each on 4 NVIDIA A100 40GB GPUs with a batch size of 512 for approximately 2 days.

B Details on the Classifier

We train a classifier to compute the IS (see Section 2.2) and we train a classifier to compare training and test losses in Section 4.

We use a ResNet-18 [3] for the classifiers trained on SVHN, and we use a DenseNet-121 [6] for classifiers trained on CIFAR-10. Each classifier is trained for 14 epochs with a batch size of 256 on the corresponding dataset. We use stochastic gradient descent [24] with a learning rate of 0.001, a momentum [25] of 0.9, and weight decay of $5 \cdot 10^{-4}$.

[2] https://github.com/addtt/variational-diffusion-models