Further Exploration of Outlier Detection by De-Biasing VAE Likelihoods

Fehmi Ayberk Uckun
uckun@kth.se

Yuxin Sun
yuxins@kth.se

Ezio Cristofoli
ezioc@kth.se

Abstract

This paper replicates and further explores relatively computationally inexpensive techniques to reduce the bias effects and improve outlier detection originally presented in (1). Typical metrics adopted for outlier detection are likelihoods computed by deep generative models, which are often biased. In this paper, we replicate the utilization of variational autoencoders in combination with a pre-processing technique "contrast stretching" and simple post-processing techniques for intensity bias correction to perform outlier detection. AUROC performances are presented and discussed for a selection of grayscale and natural datasets. Findings from an ablation study are also presented an discussed. Moreover, we go beyond the re-implementation and test our own method to challenge suggested contrast stretching. Both the cases of continuous Bernoulli and categorical visible decoder distributions have been considered for experiments.

1 Introduction

Deep neural networks are indeed extensively used in computer vision applications across a broad set of use cases. One drawback of these architectures consists in the fact that, in some specific conditions, they may produce wrong predictions with high confidence levels. This typically happens if the input data is sampled from a different distribution (out of distribution, OOD) than the data utilised for training the model.

A popular approach for outlier detection for unlabeled data consists in computing sample likelihood with generative models such as variational autoencoders (VAE) (2). Multiple studies though showed that VAE likelihoods are sensitive to phenomena like the number of zeros in a sample (3) or the variance in the distribution of pixel intensities (4). The majority of the proposed solutions to this problem shows or sizeable computational bottlenecks or requires sizeable amount of data. The solutions proposed by the paper "Robust outlier detection by de-biasing VAE likelihoods" (1) are instead showing the way for computationally not expensive and not data hungry solutions.

2 Code Repository

`https://github.com/ayberkuckun/CVAE_OOD`

3 Related Work

Studies which propose different approaches to compute correction factors for model likelihoods to improve outlier detection of deep generative models are various and include among others:

- The work of Serra et al. (2019) (5) who proposed a correction for model likelihoods based on “input complexity” (IC);

- The work of Ren et al. (2019) (3) which addressed the problem of bias in model likelihoods, for samples with many zero-valued pixels with a correction computed by training a second generative model with noise-corrupted samples;
- The work by Yong et al. (2020) (6) proposed to use bias-free Gaussian likelihoods and their variances for outlier detection;
- the work by Xiao et al. (2020) (7) which proposes a "likelihood regret" metric that requires the quantification of the improvement in marginal likelihood by retraining the encoder network to get the optimal likelihood for each sample.

While the above strategies appear to be effective in the contexts of their studies, they all come with drawbacks such as: high computational demand (7), requirement for high amount of data (3) or theoretical assumptions which in practice are difficult to be satisfied (6).

The approach proposed by (1), whose replication is the main objective of our paper, addresses the issue of outlier detection proposing a methodology that is:

- Affordable from the computational perspective, since it does not require any model retraining and just demands relatively simple pre-training processing (for contrast normalization 4.1) in combination with post-training per sample processing (for intensity bias correction 4.2);
- Affordable from the perspective of required amount of data, since all the proposed techniques are pretty "standard" and not "data hungry";
- Working across a broad set of scenarios: the proposed approach supports a wide range of visible decoder distributions with analytically computed corrections for intensity bias 4.2.1 (e.g. Gaussian, Bernoulli or Continuous Bernoulli) and algorithmically computed corrections 4.2.2 (e.g. for Categorical or Truncated Gaussian distributions when the analytical form is intractable).

4 Methods

This paragraph describes the techniques proposed in (1) (and replicated by our project) to reduce the effects of contrast bias 4.1 and intensity bias 4.2 on the likelihoods computed by the deep generative models - more specifically a CNN based deep convolutional generative adversarial network architecture DCGAN (9) has been implemented.

4.1 Correction for contrast bias: Normalization

There is experimental evidence that, due to practical limitations with achieving perfect reconstructions with VAEs, the variation in image contrast levels produces systematic biases in the likelihood and negative reconstruction error, both when considering continuous Bernoulli or categorical visible decoder distributions. The proposed solution to this problem in (1) is "contrast stretching", a standard image pre-processing technique which improves the robustness of deep classifier models (8).

The technique is applied to training and testing data-sets according to the following transformation: x_i = min(max(0, [x_i - a]/r), 1), where x_i refers to the i^{th} pixel of image **x**, r = $P_{95}(\mathbf{x}) - P_5(\mathbf{x})$, a = $P_5(\mathbf{x})$, P_j refers to the j_{th} percentile and **x** refers to the vectorized input sample tensor.

Results about our experiments with contrast normalization are discussed in 6.

4.2 Correction for intensity bias

The problem of intensity bias is discussed in detail in (4) and (3). The experimental exemplification of the problem consists in the fact that (simulated) images with different constant pixel intensities when processed by a VAE produce a U-shape likelihood (if plotted as a function of pixel intensity).

To address the effect of intensity bias on outlier detection two methods are proposed in (1):

- Analytical correction, described in 4.2.1, applicable in the cases of visible decoder distributions like Bernoulli, continuous Bernoulli or Gaussian where the likelihood correction can be expressed in an analytical closed form;

- Algorithmic correction, described in 4.2.2, applicable in the cases of visible decoder distributions like categorical or truncated Gaussian where the correction can **not** be expressed in an analytical closed form.

In both cases the method utilises the sample likelihood learned during the training phase to compute a per-image correction for the validation and test sets (thus providing the advantage of not requiring any additional training of the deep model).

Results about our experiments for both methods are discussed in 6.

4.2.1 Analytical correction for intensity bias

The driving idea behind the analytical correction for intensity bias discussed in (1) is that, as a consequence of the intensity bias, even if two input samples are perfectly reconstructed by the VAE, these will be assigned different likelihoods, depending on the average pixel intensity in each sample.

The idea is then to eliminate the intensity bias in the reconstruction error by dividing it by the error for perfect reconstruction.

In the case of a continuous Bernoulli decoder, the negative reconstruction error (NRE) is expressed by the following formula 1 from (1), where: **x** is an input image, composed by D pixels x_i, **z** is the corresponding latent representation learnt by the VAE encoder, θ are the VAE decoder parameters and λ_i is the shape parameter of the Bernoulli distribution for the i^{th} pixel of **x**:

$$NRE = logp_\theta(\mathbf{x}/\mathbf{z}) = \sum_{n=1}^{D} logC(\lambda_i) + x_i log\lambda_i + (1-x_i)log(1-\lambda_i) \tag{1}$$

Additionally the decoded pixel value $\hat{x}_i$ for a continuous Bernoulli decoder can be calculated from the corresponding shape parameters λ_i according to the following formulas 2:

$$\hat{x}_i = \begin{matrix} \frac{\lambda_i}{2\lambda_i - 1} + \frac{1}{2tanh^{-1}(1-2\lambda_i)}, & \text{if } \lambda_i \neq \frac{1}{2} \\ \frac{1}{2}, & \text{if } \lambda_i = \frac{1}{2} \end{matrix} \tag{2}$$

Setting x_i = $\hat{x}_i$ using formulas 1 and 2 we obtain a linear equation in λ_i which can be iteratively maximised with the SciPy's implementation of Nelder-Mead simplex algorithm. The Nelder-Mead algorithm returns then the value of λ_i^* which optimizes the NRE. The corresponding NRE value (obtained from 1) is the perfect reconstruction NRE which is used to correct the marginal image likelihoods calculated by the VAE.

4.2.2 Algorithmic correction for intensity bias

The high level approach adopted in the algorithmic solution is the following:

1. Calculate the average decoder distribution value for each pixel value of **each** image in the training set;
2. Calculate the average decoder distribution value for each pixel value **across all** the images in the training set;
3. For each test image compute the correction term as the average value across pixels, under the above computed average decoder visible distribution.

The detailed procedure is described in Figure 1, (1).

5 Data

Our replication study considered the well known greyscale and natural datasets summarised in tables 1 and 2 which are a subset of the ones considered in (1).

```
Algorithm 1: Algorithmic Bias correction
Data: Training Set $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, ...\mathbf{x_n}\}$ with $\mathbf{x_p}$ of
      shape $32 \times 32 \times nc$ (no. of channels),
      Encoder Parameters $\phi$, and Decoder
      Parameters $\theta$
Result: Log Correction Factor $\mathbf{C} : (v, k) \rightarrow$ Float
        for $v = 0, 1, \ldots 255$ and $k = 1, 2, \ldots nc$
Init: Map $\mathbf{A} : (v, k) \rightarrow$ EmptyList for
      $v = 0, 1, \ldots 255$ and $k = 1, 2, \ldots nc$
for $\mathbf{x_p} \in \mathbf{X}$ do
    Init: Map $\mathbf{B} : (v, k) \rightarrow$ EmptyList for
          $v = 0, 1, \ldots 255$ and $k = 1, 2, \ldots nc$
    $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x_p})$
    for $i \leftarrow 1$ to 32, $j \leftarrow 1$ to 32, $k \leftarrow 1$ to $nc$ do
        Append $p_\theta^{ijk}(x_p^{ijk}|\bar{\mathbf{z}})$ to $\mathbf{B}(x_p^{ijk}, k)$
    end
    for $v \leftarrow 0$ to 255, $k \leftarrow 1$ to $nc$ do
        Append Mean($\mathbf{B}(v, k)$) to $\mathbf{A}(v, k)$
    end
end
Init: Map $\mathbf{C} : (v, k) \rightarrow 0$ for $v = 0, 1, \ldots 255$ and
      $k = 1, 2, \ldots nc$
for $v \leftarrow 0$ to 255, $k \leftarrow 1$ to $nc$ do
    $\mathbf{C}(v, k) \leftarrow$ Log(Mean($\mathbf{A}(v, k)$))
end
```

Figure 1: Algorithm for intensity bias correction

Table 1: Greyscale Datasets

Dataset	N-train (N-val)	N-test
MNIST	54000 (6000)	10000
Fashion MNIST	54000 (6000)	10000
EMNIST-Letters	79920 (8880)	14800
Grey-Noise	-	10000

Table 2: Natural Datasets

Dataset	N-train (N-val)	N-test
CIFAR-10	45000 (5000)	10000
SVHN cropped	65932 (7325)	26032
GTSRB	35289 (3920)	12630
Color-Noise	-	10000

6 Experiments and Findings

In order to reproduce the results from the paper in question, we did 3 main experiments. All the numbers are AUROC scores. For concise reporting, we use the following naming conventions: **BC**: Bias-corrected/Bias-correction, **CS**: Contrast stretched/Contrast stretching.

The first experiment is the main experiment of the paper which proves that BC, together with CS, performs better outlier detection. The experiment compares the outlier detection results of the standard likelihood VAE with the outlier detection results of the BC-likelihood VAE when CS is applied during both training and test time. We repeat this experiment for both continuous Bernoulli and categorical distribution and also for both grayscale datasets and natural datasets.

The second experiment is an ablation study which proves that CS is required for a superior outlier detection performance. Here, we only use grayscale datasets. We compare the outlier detection results when CS is not applied, when it is only applied during test time and when it is applied during both test and training time. We also put standard-likelihood VAE results for reference.

The third and the last experiment is an extension of the experiments presented in the paper. The authors claim that applying CS results in observing more homogeneous distributions of per-channel variance across datasets. In this experiment, we wanted to test replacing batch normalization layers with instance normalization layers which are also known as contrast normalization layers that remove instance-specific contrast information from the content image. These layers can potentially achieve the wanted effect without applying CS. To test this hypothesis, we compared CS applied, BC batch-norm-VAE with no CS, BC instance-norm-VAE and with CS only applied on test time, BC instance-norm-VAE.

For the below results, averages don't include the score that is coming from the same outlier dataset as the training set (like MNIST-VAE/MNIST test set) since it is expected to be ended up with a 0.5 AUCROC score.

Table 3: Summary evaluation for EMNNIST VAE (CBERN)

Test set	LL CS	LL No CS	LL CS(Test only)
NOISE	0,95	0,92	0,91
MNIST	0,37	0,40	0,39
FMNIST	0,85	0,69	0,84
EMNIST	0,50	0,50	0,51
AVERAGE	0,72	0,67	0,71

CS(Test only) = Contrast Stretching is only applied during test time not during training time

Table 4: Summary evaluation for MNNIST VAE (CBERN)

Test set	LL CS	LL No CS	LL No CS Inst	LL CS Inst
NOISE	0,87	0,97	1,00	1,00
MNIST	0,50	0,50	0,50	0,50
FMNIST	0,87	0,90	1,00	1,00
EMNIST	0,60	0,62	1,00	1,00
AVERAGE	0,78	0,83	1,00	1,00

Inst = model with an instance normalization layer

Table 5: Summary evaluation for MNNIST VAE (CAT)

Test set	LL CS	LL No CS	LL No CS Inst	LL CS Inst
NOISE	1,00	1,00	1,00	1,00
MNIST	0,50	0,50	0,50	0,50
FMNIST	0,94	0,93	1,00	1,00
EMNIST	0,71	0,78	0,99	0,99
AVERAGE	0,88	0,90	1,00	1,00

Inst = model with an instance normalization layer

One surprising result is that models that are trained with our novel addition "instance normalization layers" performed almost perfectly on all outlier datasets even without using the bias correction method. This result can potentially mean that instance-specific normalization can make autoencoders unbiased yet this powerful claim requires more experimentation.

The results of our ablation study can be seen in Table 3. It shows that models trained with CS perform best while models trained with no CS can still perform well yet model that neither uses CS during train or test time perform poorly.

Unfortunately, at the time of writing this, it is only 10 minutes left to the deadline and even though we have more grayscale results and results for natural datasets we couldn't put them here. Also, we did some experiments with bias-corrected likelihoods too that gave us the results that we are discussing in the Challenges Section 7 but again there is no time for us the put these new results. Please see **Categorical visible distribution** and **Algorithmic bias calculation** subsections of Section 7 to understand what led to this incomplete reporting.

Table 6: Summary CIFAR10 VAE

Test set	LL CBERN
NOISE	1,00
CIFAR10	0,50
GTSRB	0,37
SVHN	0,48
AVERAGE	0,62

Table 7: Summary SVHN VAE

Test set	LL CBERN	LL CAT Bat
NOISE	1,00	0,00
CIFAR10	0,80	0,20
GTSRB	0,57	0,43
SVHN	0,50	0,50
AVERAGE	0,79	0,28

7 Challenges

Reimplementing the paper was quite a challenge since some small but very crucial implementation details are not described or assumed known in the paper. Let's discuss all the challenges and inconveniences we faced during the implementation.

- **Tensorflow 2.0**. Firstly, specifying the library as Tensorflow 2.0 is not specific enough. Tensorflow can change its default values, its implementation techniques etc. with each version and this explanation can mean any version from 2.0.0 to the currently latest 2.11.0. In order to match the specific version, we checked the author's code.
- **Paddings**. The network architecture doesn't specify the convolutional layer paddings. They reference another paper (7) by saying that their architecture is nearly identical to theirs. However, when we checked the paper, we saw that it doesn't include any network architecture definition and one needs to check that paper's GitHub repo too. As a result, to implement the same padding, we checked the author's code.
- **Importance-weighted lower bound**. The authors say that they estimate the log-likelihoods using n=100 importance-weighted lower bound. However using 100 samples during training caused too big a memory footprint and caused OOM errors. Later when we checked the author's code we saw that they only use importance weighting during evaluation and not while training. This detail wasn't specified on paper.
- **Categorical visible distribution**. The authors assume we are already familiar with the usage of the categorical distribution with VAEs and they don't explain the distribution-specific preprocessing that is applied to the dataset. Our first assumption was that categorical visible distribution means input image with values in range (0-255), an output with shape (32, 32, nc, 256) and later reconstruction error can be calculated between the input and output using $CategoricalDistribution(output).log\_logprob(input)$. However, after training nearly all the categorical distribution models, we realized this is not the case. We were getting NaN values in our evaluation, so we checked the authors' code and saw this: Even though you calculate the reconstruction loss using an input with values in range (0-255), the network input to produce the output must be with values in range (0-1). This may be a widespread implementation detail but we assumed that the input to the network must be the same as the reconstruction. This led to the failure of lots of experiments.
- **Algorithmic bias calculation**. This calculation takes way too much time. Even longer than the training itself and even though we made it really efficient using as much vector operation as possible. When we checked the authors' code, we saw that they only calculate this using the first 500 images of the training set as opposed to their claim of using the whole dataset. This also explains why their algorithm performs slowly because they didn't make it to run on the whole dataset. Moreover, they describe the algorithm as the one that calculates the corrections separately for each channel but in their code, they don't use channel-specific corrections but use a total average. This answers the question "Should I sum or average the channel-specific correction values of my 3-channel image?" which they don't specify.
- **Analytic Bias Correction**. Bias correction failed because of the numerical differences between math theory and discrete computer mathematics. Or we can say that this has failed because the authors' didn't specify the stabilizing trick they used in their implementation. Normally, for continuous Bernoulli visible distribution, the network outputs the logits of the distribution which transforms to the λ parameter using the equation $\lambda = sigmoid(logits)$. However, outputted logprob values by the

$tensorflow\_probability.distributions.ContinuousBernoulli$ drastically changes with varying too-high or too-low logit values. This leads to logprob values that can reach above 10k. This doesn't stop the network from training/learning so we didn't think there is a problem. However, it turned out that analytically calculated perfect reconstruction error which we need to subtract from the our logprob becomes too small compared to the logprob and can't achieve its purpose. In order to prevent this and make the two probabilities on the same scale, authors clipped the logit values between (-15.94, 15.94). We didn't know this kind of thing is required until the very end when we checked the author's code to understand why ours is not working. This caused failures in our experiments.

- **Number of models to train for experiments**. They are using 4 grayscale, 4 natural datasets and 4 different decoder visual distributions for each dataset. They claim to perform 3 different initialization with 2 different train-val splits thus in total 6 different runs per model. Also, there are 2 different scenarios where we need to train all models with and without CS plus with batch-norm layers or with instance-norm layers. A total combination of these, also considering the epoch count is 1000, makes the required training time for the reimplementation huge. Not the mention the required time for other calculations like algorithmic bias calculation. Thus, in total, we performed our experiments with 16 different models.

8 Conclusion

This project was extremely valuable in order to understand the challenges of re-implementing a paper. From the perspective of re-producibility, we can learn some takeaways from this experience. Never assume that your readers are all knowledgeable about all the common techniques that are used in the field. Even if you don't want to or can't explain every technique, just give a reference where an explanation can be found. If your novel method depends on numerical stability in the implementation and you already went through solving that problem, highlight it. The bias correction in this paper depends on having the same scale with model loss and perfect reconstruction from the distribution pdf and this situation caused problems since the authors didn't describe the trick they used to stabilise it. Anything can go wrong with reimplementing based on paper because it is a really slippery path. Many details depend on the tools we use. We believe it is quite impossible to give every little detail of the implementation in the paper itself thus experience and luck are very needed.

A future work we think that can benefit greatly from the experimentation is the usage of different augmentation methods in order to increase performance on the CIFAR-10 dataset. This can provide a deeper look into the OOD detection using VAEs on CIFAR-10 and provide methods to overcome one of the shortcomings of this paper.

9 Ethical Considerations

There is no ethical consideration of the paper and its methods.

10 Self Assessment

We believe that our project deserves the "Excellent Project" title. Even though we failed the reproduce the same results as the original paper this was one of the expected outcomes since not all papers are perfectly reproducible, especially if one only utilizes the paper itself and not the online code. Thus, we believe that we successfully re-implemented the paper albeit not reproducing the results. We performed ablation studies (CS/no CS) in order to test the different components of the approach. Moreover, we went beyond the project and tested our own hypothesis (instance normalization instead of CS). As a result, we believe that our project satisfies all the criteria for an "Excellent Project".

11 Self-nomination for Bonus

Difficulty of the implementation: We believe that the points we brought forward in Section 7 justify how difficult it was to implement this paper. One can copy the methods and the architecture from the available online repository however, with academic honesty, we tried to implement it using only the knowledge which is presented in the paper and this caused lots of problems because of the details authors chose to disregard presenting in the paper. In the end, it caused many experiments to give poor results and many experiments required a new retrained model. Moreover, the discovery of some of these small disregarded but required details were not possible early into the project since we needed both a working model, working bias correction and a working evaluation. It was only possible after we get the final results.

References

[1] Chauhan, Kushal and U, Barath Mohan and Shenoy, Pradeep and Gupta, Manish and Sridharan, Devarajan, Robust outlier detection by de-biasing VAE likelihoods, arXiv 2021: `https://arxiv.org/abs/2108.08760`

[2] Diederik P. Kingma and Max Welling, Auto-encoding variational bayes. In 2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings, 2014: `https://arxiv.org/pdf/1312.6114.pdf`

[3] Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan, Likelihood ratios for out-of-distribution detection. In Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019: `https://papers.nips.cc/paper/2019/file/1e79596878b2320cac26dd792a6c51c9-Paper.pdf`

[4] EricT.Nalisnick, AkihiroMatsukawa,YeeWhyeTeh,Dilan Gorur, and Balaji Lakshminarayanan, Do deep generative models know what they don't know? In 7th International Conference on Learning Representations, ICLR 2019. Open- Review.net, 2019: `https://arxiv.org/pdf/1810.09136.pdf`

[5] Joan Serra, David Alvarez, Input complexity and out-of-distribution detection with likelihood-based generative models. In 8th International Conference on Learning Representations, ICLR 2020. OpenReview.net, 2020: `https://arxiv.org/pdf/1909.11480.pdf`

[6] JBang Xiang Yong, Tim Pearce, and Alexandra Melike Brintrup, Bayesian autoencoders : Analysing and fixing the bernoulli likelihood for out-of distribution detection. In International Conference on Machine Learning Workshop on Uncertainty & Robustness in Deep Learning, 2020: `https://arxiv.org/pdf/2107.13304.pdf`

[7] Zhisheng Xiao, Qing Yan, and Yali Amit, Likelihood regret: An out-of-distribution detection score for variational auto-encoder. In Advances in Neural Information Process- ing Systems, volume 33, pages 20685–20696. Curran Asso- ciates, Inc., 2020: `https://proceedings.nips.cc/paper/2020/file/eddea82ad2755b24c4e168c5fc2ebd40-Paper.pdf`

[8] Dan Hendrycks and Thomas G. Dietterich, Benchmarking neural network robustness to common corruptions and perturbations. In 7th International Conference on Learning Representations, ICLR 2019. OpenReview.net, 2019.: `https://arxiv.org/pdf/1903.12261.pdf`

[9] Alec Radford, Luke Metz, and Soumith Chintala, Unsupervised representation learning with deep convolutional gener- ative adversarial networks. In 4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings, 2016.: `https://arxiv.org/pdf/1511.06434.pdf`