On the Out-of-distribution Generalization of Probabilistic Image Modelling

Anonymous Author(s)
Affiliation
Address
email

Abstract

Out-of-distribution (OOD) detection and lossless compression constitute two problems that can be solved by the training of probabilistic models on a first dataset with subsequent likelihood evaluation on a second dataset, where data distributions differ. By defining the generalization of probabilistic models in terms of likelihood we show that, in the case of image models, the OOD generalization ability is dominated by local features. This motivates our proposal of a *Local Autoregressive* model that exclusively models local image features towards improving OOD performance. We apply the proposed model to OOD detection tasks and achieve state-of-the-art unsupervised OOD detection performance *without* the introduction of additional data. Additionally we employ our model to build a new lossless image compressor: NeLLoC (Neural Local Lossless Compressor) and report state-of-the-art compression rates and model size.

1 Introduction

Probabilistic modeling has achieved great success in the modeling of images. Likelihood based models, *e.g*. Variational Auto-Encoders (VAE) [19], Flow [18], Pixel CNN [44, 36] are shown to successfully generate high quality images, in addition to estimating underlying densities.

Further to image generation, there also exists a close connection between probabilistic models and lossless compression [29]. Let $\{x_1, \dots, x_M\}$ be an image dataset to compress, where x_m is sampled i.i.d from some underlying distribution with probability mass function (pmf) p_d. If p_d is known, we can design a compression scheme to compress each data x_m to a bit string with length approximately $-\log_2 p_d(x_m)$[1]. The total length of the compressed dataset can therefore be approximated as $\sum_{m=1}^{M} -\log_2 p_d(x_m)$. As $M \to \infty$, the total compression length will approach the entropy of the distribution p_d, that is; $\sum_{m=1}^{M} -\log_2 p_d(x_m) \to \mathrm{H}(p_d)$ where $\mathrm{H}(\cdot)$ denotes entropy. In this case, the compression length is *optimal* under Shannon's source coding theorem [40] *i.e*. we cannot find another compression scheme with compression length less than $\mathrm{H}(p_d)$. In practice, however, the true data distribution p_d is unknown and we must find a suitable model p_θ to approximate p_d, typically by minimizing some divergence between p_d and p_θ. A popular choice for this is the KL divergence

$$\mathrm{KL}(p_d || p_\theta) = -\underbrace{\mathrm{H}(p_d)}_{\text{const.}} - \int \log p_\theta(x) p_d(x) dx, \tag{1}$$

[1]For a compression method like Arithmetic Coding [45], the message length is always within two bits of $-\log_2 p_d(x_m)$ [29], also see Section 5.1.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

where the entropy of the data distribution is a constant. Since we only have access to finite p_d data samples, $x_1, \ldots, x_N$ the second term is typically approximated using a Monte Carlo estimation

$$\mathrm{KL}(p_d || p_\theta) \approx -\frac{1}{N} \sum_{n=1}^{N} \log p_\theta(x_n) - const.. \tag{2}$$

Estimation of p_θ, by minimizing the KL divergence, is equivalent to Maximum Likelihood Estimation; as $\mathrm{KL}(p_d || p_\theta) \to 0$, $p_\theta \to p_d$. We can therefore use p_θ in place of p_d to build the lossless compressor.

Recent work successfully apply deep generative models such as VAEs [42, 43, 20], Flow [15, 3] towards conducting lossless image compression. We note that the underlying models of these methods are designed to focus on the test data and follows the same distribution as the training data: $x' \sim p_d$. However, in practical compression applications, training and test data distributions may differ: the test dataset $\mathcal{X}^o_{test} = \{x'_1, \ldots, x'_M\}$ where $x' \sim p_o \neq p_d$. Surprisingly, empirical results e.g. [43, 15] show that we can still use model p_θ (which is trained to approximate p_d) to compress test data $\mathcal{X}^o_{test}$ with reasonable compression rates. In this work we explore the lack of intuition as to when a model p_θ, trained to approximate p_d, will generalize well to (*i.e.* will successfully compress) data from $p_o \neq p_d$, and what affects the generalization ability. We use the the negative log-likelihood $-\frac{1}{M}\sum_{i=1}^{M} \log p_\theta(x'_i)$ to indicate the Out-of-distribution (OOD) generalization ability of model p_θ on dataset $\mathcal{X}^o_{test}$ (lower is better). To investigate these questions, we firstly consider recent advances in likelihood-based OOD detection, leading to our proposal of a new model that we show can capture the features that dominates the generalization ability, with downstream benefits to both OOD detection and lossless compression applications.

2 Unsupervised OOD Detection

Given a set of unlabeled data points, sampled from p_d, and test data point x' then the goal of OOD detection is to distinguish whether or not x' originates from p_d. A natural approach involves fitting a probabilistic model p_θ to approximate p_d and treat sample x' as in-distribution if its (log) likelihood is larger than a threshold $\log p_\theta(x') > \epsilon$. However, recent work [31] shows the surprising result that popular deep generative models including *e.g.* VAE [19], Flow [18] and PixelCNN [36] can assign OOD data higher density values than in-distribution samples, where such OOD data may contain differing semantics, *c.f.* the samples used for maximum likelihood training. We demonstrate this phenomenon using PixelCNN models, trained on Fashion MNIST (CIFAR10) and tested using MNIST (SVHN). Figure 1 provides histograms of model evaluation using negative bits-per-dimension (BPD), that is; the $\log_2$ likelihood normalized by data sample dimension (larger negative BPD corresponds to larger likelihood). We corroborate previous work and observe that tested models assign higher likelihood to the OOD data, in both cases.

This counter intuitive phenomenon suggests that likelihood-based approaches may not make for good OOD image detection criterion, yet also illustrates that a probabilistic model, trained using one dataset, may be employed to compress data originating from a different distribution with a potentially higher compression rate. This intuition builds a connection between OOD detection and lossless compression. Inspired by this link, we investigate the underlying latent causes of image model generalizability, towards improving lossless compressor generalization.

Previous work studies the potential causes of the considered phenomenon; [32] used a typical set to reason about the source of the effect, while the work of [35] argues that likelihoods are significantly affected by image background statistics or by the size and smoothness of the background [22]. In this work, we alternatively consider a recent hypothesis proposed by [38]: *low-level local features, learned by (CNN-based) probabilistic models, are common to all images and dominate the likelihood.* In their work, the authors investigated this hypothesis by studying the differences between individual pixel values and neighbourhood mean values and additionally considered the correlation between models trained on small image patches and trained, alternatively, on full images. To further investigate this hypothesis, we rather propose to *directly model the in-distribution dataset, using only local feature information*. By contrasting such an approach with a standard *full model*, that considers both local and non-local features, we are thus able to study the contribution that local features make to the full model likelihood.

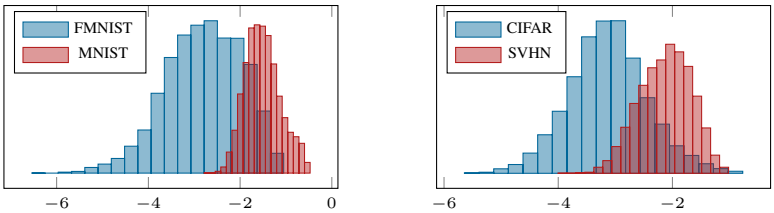


Figure 1: Left: log likelihood of FashionMNIST, MNIST test data using a full PixelCNN model, trained with FashionMNIST training set. Right: log likelihood of CIFAR10, SVHN test data using a PixelCNN model, trained with CIFAR10 training set. The x-axis indicates the value of the log-likelihood (negative BPD), the y-axis provides data sample counts.

3 Local Image Model

3.1 Local Autoregressive Model with Masked CNN

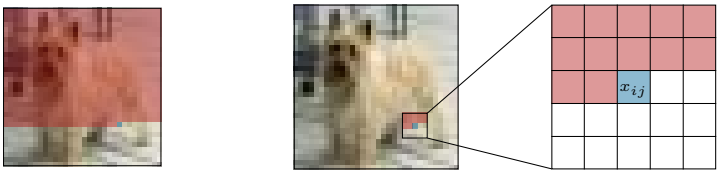
Autoregressive models have proven popular for modeling image datasets and common models are PixelCNN [44, 36], PixelRNN [44] or Transformer based models [6]. Assuming data dimension D, the Autoregressive model $p_f(x)$ can be written as

$$p_f(x) = p(x_1) \prod_{d=2}^{D} p(x_k | x_1, \ldots, x_{d-1}), \tag{3}$$

informally we refer to this type of model as a 'full model' since it can represent all dependencies between each dimension (pixel). Similarly, we can define a local autoregressive model $p_l(x)$ where pixel x_{ij}, at image row i column j, depends on *previous* pixels with fixed horizon h:

$$p_l(x) = \prod_{ij} p(x_{ij} | x_{[i-h:i-1, j-h:j+h]}, x_{[i, j-h:j-1]}), \tag{4}$$

with zero-padding used in cases where i or j are smaller than h. Figure 2b illustrates the resulting local autoregressive model dependency relationships. We implement this model using a masked CNN [44], with kernel size $k{=}2{\times}h{+}1$ in our first network layer to mask out future pixel dependency. A full PixelCNN model would then proceed to stack multiple masked CNN layers, where increasing kernel depth affords receptive field increases. In contrast, we employ masked CNN with $1{\times}1$ convolutions in subsequent layers. Such $1{\times}1$ convolutions can model the correlation between channels, as in [18], and additionally prevent our local model obtaining information from pixels outwith the local feature region, defined by h. Pixel dependencies are therefore defined solely using the kernel size of the first masked CNN layer, allowing for easy control over model local feature size. We note that the proposed local autoregressive model can also be implemented using alternative backbones *e.g.* Pixel RNN [44] or Transformers [6]. We also plot the samples of the local model, see Figure 4. Unlike the full autoregressive models [44, 36] which are used to generate semantically coherent samples, we find the samples from the local model are locally consistent yet have no clear semantic meaning.



(a) Full Autoregressive Model (b) Local Autoregressive Model

Figure 2: (a) full autoregressive model pixel dependencies; the distribution of the current pixel (blue) depends on all *previous* pixels (red); (b) local autoregressive model dependencies, with $h = 2$. The distribution of x_{ij} (blue) depends on only the pixels in a local region (red).

3.2 Local Model Generalization

To investigate the generalization ability of our local autoregressive model, we fit the model to Fashion MNIST (grayscale) and CIFAR10 (color) training datasets and test using in-distribution (ID)

images (respective dataset test images) and additional out-of-distribution (OOD) datasets: MNIST, KMNIST (grayscale) and SVHN, CelebA[2] (color). Both models use the discretized mixture of logistic distributions [36] with 10 mixtures for the predictive distribution and a ResNet architecture [44, 10]. We use a local horizon length $h = 3$ (kernel size $k = 7$) for both grayscale and color image data. We compare our local autoregressive model to a standard full autoregressive model (*i.e.* a standard PixelCNN), additional network architecture and training details can be found in Appendix A. Tables 1, 2 report comparisons in terms of BPD (where lower values entail higher likelihood) for Fasion MNIST and CIFAR10, respectively. We observe that for in-distribution (ID) data, the full model has better generalization ability *c.f.* the local model (0.21 and 0.13 BPD, respectively). This is unsurprising as training and test data originate from the same distribution; both local and non-local features, as learned by the full model, help ID generalization. For OOD data, we observe that the local model has generalization ability similar to the full model, exhibiting very small empirical gaps (only ≈ 0.02 BPD on average). This phenomenon is consistent with our intuition: local features are shared between different image datasets and dominate the generalization ability of the model, whereas non-local features (*e.g.* semantics) does not allow for easy generalization to OOD datasets.

Table 1: Test BPD (Trained on Fashion MNIST)

Test Dataset	Full	Local
Fashion MNIST (ID)	2.78	2.99
MNIST (OOD)	1.50	1.49
KMNIST (OOD)	2.48	2.44

Table 2: Test BPD (Trained on CIFAR10)

Test Dataset	Full	Local
CIFAR10 (ID)	3.12	3.25
SVHN (OOD)	2.13	2.14
CelebA (OOD)	3.33	3.36

Further, we also empirically study the effect of varying local feature scale on generalization ability, see Table 5. We find the model has poor generalization performance when local features are too small and increasing the horizon size does not compensate for this towards improving generalization. We hypothesize that optimal local feature scale likely varies with both image semantics and resolution. We leave the study of selecting optimal local scales to future work.

In the following sections, we demonstrate how to apply our local autoregressive model to OOD detection (Section 4) and lossless compression tasks (Section 5).

4 OOD Detection

As discussed in Section 2, local image features appear largely common across the real-world image manifold and can be treated as a domain-prior. Therefore, to detect if an image is out-of-distribution or not we propose to leverage non-local features, which typically contain semantic image information. Ideally, we would thus want to build a model that captures non-local components of the image distribution. We assume the full model, $p_f(x)$ can be written as a product of experts [13]:

$$p_f(x) = \frac{p_l(x)p_{nl}(x)}{Z} \tag{5}$$

where $Z = \int p_l(x)p_{nl}(x)dx$ is the normalization constant and two experts p_l (local expert) and p_{nl} (non-local expert) are independent. The non-local density is thus proportional to the likelihood ratio:

$$p_{nl}(x) \propto \frac{p_f(x)}{p_l(x)} \equiv \hat{p}_{nl}(x), \tag{6}$$

where we use $\hat{p}_{nl}(x)$ to denote the unnormalized density. For the OOD detection task, we require only $\hat{p}_{nl}(x)$ towards providing a score to classify if test data x is OOD or not and therefore do not require to estimate Z. Figure 3 shows the histogram of $\hat{p}_{nl}(\cdot)$ for both ID and OOD test datasets. It may be observed that the majority of ID test data obtains higher likelihood than OOD data, illustrating the effectiveness of the non-local model.

Likelihood ratio variants have been previously explored for OOD detection. We briefly discuss related works and highlight where our method differs from relevant literature.

In [35], it is assumed that each data sample x can be factorized as $x = \{x_b, x_s\}$, where x_b is a background component, characterized by population level background statistics: p_b. Further, x_s

[2] We down-sampled the original CelebA to $32 \times 32 \times 3$, see Appendix A.2 for details.

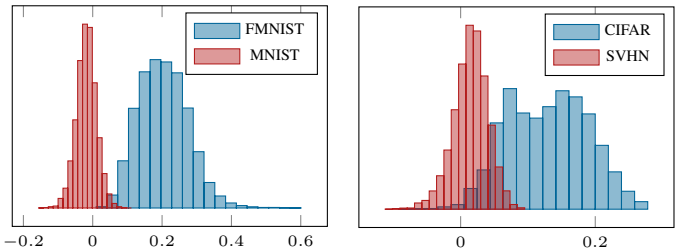


Figure 3: Unnormalized log likelihood of the non-local model $\hat{p}_{nl}(x)=\frac{p_f(x)}{p_l(x)}$ for both ID test datasets (FashionMNIST, CIFAR10) and OOD datasets (MNIST, SVHN). ID test datasets obtain significantly higher likelihoods, on average, in each case.

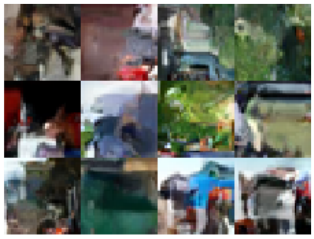


Figure 4: Samples from the local autoregressive model, see Appendix B for details.

then constitutes a semantic component, characterized by patterns belonging specifically to the in-distribution data: p_s. A full model, fitted on the original data, (*e.g.* Flow) can then be factorized as $p_f(x) = p_f(x_b, x_s) = p_b(x_b)p_s(x_s)$ and the semantic model can be correspondingly defined as a ratio: $p_s(x) = \frac{p_f(x)}{p_b(x)}$, where $p_f(x)$ is a full model. In order to estimate $p_b(x_b)$, the authors of [35] design a perturbation scheme and construct samples from $p_b(x_b)$ by adding random perturbation to the input data. A further full generative model is then fitted to the samples towards estimating p_b. In our method, both $p_l(x)$ and $p_{nl}(x)$ constitute distributions of the same sample space (that of the original image x) whereas p_s and p_b in [35] form distributions in different sample spaces (that of x_s and x_b, respectively). Additionally, in comparison with our local and non-local experts factorization of the model distribution, their decomposition of an image into 'background' and 'semantic' parts may not be suitable for all image content and the construction of samples from $p_b(x_b)$ (adding random perturbation) lacks principled explanation.

In [39, 38], the score for OOD detection is defined as $s(x) = \frac{p_f(x)}{p_c(x)}$ where p_f is a full model and p_c is a complexity measure containing an image domain prior. In practice, p_c is estimated using a traditional lossless compressor (*e.g.* PNG or FLIF), or a model trained on an additional, larger dataset in an attempt to capture general domain information [38]. In comparison, our method does not require the introduction of new datasets and our explicit local feature model, p_l, can be considered more transparent than PNG of FLIF. Additionally, our likelihood ratio can be explained as the (unnormalized) likelihood of the non-local model for the in-distribution dataset whereas the score described by [38, 39] does not offer a likelihood interpretation. In the following Section 5, we discuss how the proposed local model may be utilized to build a lossless compressor, enhancing the connection between our OOD detection framework and those using traditional lossless compressors (*e.g.* PNG or FLIF). In Table 4, we report experimental results where we can show that a lossless compressor based on our model significantly improves compression rates *c.f.* PNG and FLIF, which further suggests the benefits of our OOD detection method.

We use our method to conduct OOD detection in five different dataset-pairs that can be considered challenging [31]: Fashion MNIST (ID) *vs.* MNSIT (OOD); Fashion MNIST (ID) *vs.* KMNSIT (OOD); Fashion MNIST (ID) *vs.* OMNIGLOT (OOD); CIFAR10 (ID) *vs.* SVHN (OOD); CIFAR10 (ID) *vs.* CelebA (OOD). Additional experimental details and information are provided in Appendix A. In Table 3, we report the comparisons of 'area under the receiver operating characteristic curve' (AUROC), a common measure for the OOD detection task [11]. We compare against methods, some of which require additional label information[3] or datasets. Our method achieves state-of-the-art performance in most cases without requiring additional information. We observed that the model-ensemble method, WAIC [7] achieves higher AUROC in the experiments involving color images but is outperformed by our approach in the case of grayscale data, such ensemble methods may also be combined with our method, which we leave to future work.

[3]In principle, methods that require labels correspond to *classification*, task-dependent OOD detection, which may be considered fundamentally different from task-independent OOD detection (able to access only image space information), see [1] for details. We select to compare against both method classes, for completeness.

Table 3: OOD detection comparisons (AUROC) where higher values indicate improved performance and results are rounded to three decimal places. Results are reported in each case directly using the original references except in the cases of ODIN [35, 25] and VIB [7]. Results for Typicality test are from [39], corresponding to batches of two samples of the same type. (a) The MAHALANOBIS requires knowledge of the validation data (OOD distribution). (b) A full PixelCNN (see Appendix A) is trained on the ID dataset and its likelihood evaluations are then used to calculate AUROC.

ID dataset:	FashionMNIST			CIFAR10	
OOD dataset:	MNIST	Omniglot	KMNIST	SVHN	CelebA
Using Labels					
ODIN [27]	0.697	-	-	0.966	-
VIB [2]	0.941	0.943	-	0.528	0.735
Mahalanobis[a] [12]	0.986	-	-	0.991	-
Gram-Matrix [37]	-	-	-	0.995	-
Using Additional Datasets					
Outlier Exposure [12]	-	-	-	0.758	0.615
Glow diff to Tiny-Glow [38]	-	-	-	0.939	-
PCNN diff to Tiny-PCNN [38]	-	-	-	0.944	-
Not Using Additional Information					
WAIC (model ensemble) [7]	0.766	0.796	-	**1.000**	-
Glow diff to PNG [38]	-	-	-	0.754	-
PixelCNN diff to PNG [38]	-	-	-	0.823	-
Likelihood Ratio in [35]	0.997	-	-	0.912	-
S using Glow and FLIF [39]	0.998	**1.000**	-	0.950	0.736
S using PCNN and FLIF [39]	0.967	**1.000**	-	0.929	0.535
Full PixelCNN likelihood[b]	0.074	0.361	0.391	0.113	0.602
Our method	**1.000**	**1.000**	**1.000**	0.969	**0.949**

5 Lossless Compression

Recent deep generative model based compressors [42, 3, 20, 14] are designed under the assumption that data to be compressed originates from the same distribution (source) as model training data. However, in practical scenarios, test images may come from a diverse set of categories or domains and training images may be comparatively limited [29]. Obtaining a single method capable of offering strong compresson performance on data from different sources remains an open problem and related study involves consideration of 'universal' compression methods [29]. Based on our previous intuitions relating to generalization ability; for building such a 'universal' compressor in the image domain, we believe a promising route involves leveraging models that only depend on common local features, shared between differing image distributions. We thus propose a new 'universal' image lossless compressor: NeLLoC (Neural Local Lossless Compressor), built upon the proposed local autoregressive model and the concept of Arithmetic Coding [45]. In comparison with alternative recent deep generative model based compressors, we find that NeLLoC has competitive compression rates on a diverse set of data, yet requires significantly smaller model sizes which in turn reduces storage space and computation costs. We note that due to our design choices, and in contrast to alternatives, NeLLoC can compress images of *arbitrary* size.

In the remaining parts of this section, we firstly discuss NeLLoC implementation and then provide further details on the most important resulting properties of the method.

5.1 Implementation of NeLLoC

Our NeLLoC implementation uses the same network backbone as that of our OOD detection experiment (Section 4): a Masked CNN with kernel size $k = 2 \times h + 1$ (h is the horizon size) in the first layer and followed by several residual blocks with 1×1 convolution, see Appendix A for the network architecture and training details. To realize the predictive distribution for each pixel, we propose to use a discretized Logistic-Uniform mixture distribution, which we now introduce.

Discretized Logistic-Uniform Mixture Distribution The discretized Logistic mixture distribution, proposed by [36], has shown promising results for the task of modeling color images. In order to ensure numerical stability, the original implementation can provide only an (accurate) approximation and therefore cannot be used for our task of lossless compression, which requires exact numerical evaluation. We therefore propose to use the discretized Logistic-Uniform Mixture distribution, which mixes the original discretized Logistic mixture distribution with a discrete uniform distribution:

$$x \sim (1-\alpha)\left(\sum_{i=1}^{K} \pi_i \mathrm{Logistic}(\mu_i, s_i)\right) + \alpha \mathrm{U}(0, \ldots, 255), \tag{7}$$

where $\mathrm{U}(0, \ldots, 255)$ is the discrete uniform distribution over the support $\{0, \ldots, 255\}$. The proposed mixture distribution can explicitly avoid numerical issues and its PMF and CDF can be easily evaluated without requiring approximation. We can then use this CDF evaluation in relation to Arithmetic Coding. In practice; we set $\alpha = 10^{-4}$ to balance between numerical stability and model flexibility. We use $K = 10$ (mixture components) for all models in the compression task.

Arithmetic Coding Arithmetic coding (AC) [45] is a form of entropy encoding which utilizes a (discrete) probabilistic model $p(\cdot)$ to map a datapoint x to an interval $[0, 1]$. One fraction, lying in the interval, can then be used to represent the data uniquely. If we convert the fraction into a large message stream, the length of the message is always within two bits of $-\log_2 p(x)$ (the Shannon Information content). For an image with size $32\times32\times3 = 3072$, the BPD (bits per dimension) overhead of the arithmetic coding is less than $2/3072 \approx 0.00065$ bits, which is negligible if we calculate the BPD to two decimal places. Our NeLLoC implementation, makes use of the vanilla (decimal) version of AC.

5.2 Properties of NeLLoC

Universal Image Compressor As discussed previously, the motivation for designing NeLLoC is to realize an image compressor that is applicable (generalizable) to images originating from differing distributions. Towards this, NeLLoC conducts compression depending on local features which are shown to constitute a domain prior for all images. In addition to universality properties, we next discuss other important axes towards making NeLLoC practical when considering real applications.

Arbitrary Image Size Common generative models, *e.g*. VAE or Flow, can only model image distributions with fixed dimension. Therefore, lossless compressors based on such models [42, 15, 3, 14, 20] can only compress images of fixed size. Recently, HiLLoC [43] explore fully convolutional networks, capable of accommodating variable size input images, yet still requiring even height and width[4]. L3C [30], based on a multi-scale autoencoder, can compress large images yet also requires height and width to be even. NeLLoC (based on our local autoregressive model) is able to compress images with arbitrary size based on an alternative and simple intuition: *we only model the conditional distribution, based on local neighbouring pixels*. We thus do not model the distribution of the entire image and can therefore, in contrast to HiLLoC and L3C, compress arbitrary image sizes without padding requirements.

To validate method properties, we compare the compression performance of NeLLoC to other traditional image compressors and recently proposed generative model based compressors. We train NeLLoC with horizon length[5] $h = 3$ on two (training) datasets: CIFAR10 (32×32) and ImageNet32 (32×32) and test on the previously introduced test sets, including both ID and OOD data. We also test on ImageNet64 with size 64×64 and a full ImageNet[6] test set (with average size 500×374). Table 4 provides details of the comparison. We find NeLLoC achieves better BPD in a majority of cases. Exceptions include LBB [14] having better ID generalization for CIFAR and HiLLoC [43] having better OOD generalization in the full ImageNet, when the model is trained on ImageNet32.

Small Model Size In comparison with traditional codecs such as PNG or FLIF, one major limitation of current deep generative model based compressors is that they require to generate and store models

[4]For images with odd height or width, padding is required.

[5]See Table 5 for the use of other horizon size. We find when the too small horizon size: $h = 1$, will hurt both ID and OOD generalization performance, but when we increase the horizon length from $h = 3$ to $h = 5$, the ID generalization improves but OOD generalization remains roughly the same, therefore we use $h = 3$ to keep the best of both efficiency (see the discussion about Computational complexity below) and generalization.

[6]Images with height or width greater than 1000 are removed, resulting in a total of 49032 test images.

of very large size. For example, HiLLoC [43] contains 24 stochastic hidden layers, resulting in a capacity and parameter size of 156 MegaBytes (MB) using 32-bit floating point model weights. This poses practical challenges relating to both storage and transmission of such models to real, often resource-limited, edge-devices. Since NeLLoC only models the local region, a small network: three Residual blocks with 1×1 convolutions (except the first layer) is enough to achieve state-of-the-art compression performance. The parameter size requirement is only 2.75 MB accordingly. We also investigate the compression task under a NeLLoC instantiated with one residual block ($r = 1$); a resulting parameter size of 1.34 MB and yet still observe respectable performance. We report model size comparisons in Table 4. In principle, NeLLoC can be also combined with other resource saving techniques such as binary network weights to realize the generative model components. This would further reduce model size [4], which we consider a promising line of future investigation.

Computation Cost and Speed The computational complexity and speed for a neural based lossless compressor depends on two stages: 1. Inference stage: to compress or decompress the pixel x_d, a predictive distribution of x_d, needs to be generated by the probabilistic models; 2. Coding stage: uses the predictive distribution and the pixel value x_d to generate the binary code representing x_d (encoding) or using the predictive distribution and the binary code to recover x_d (decoding). The computation cost and speed of the second stage heavily depends on the implementation (*e.g.* programming language) and the choice of coding algorithms. For example, there exists Integer-based AC [45, 16], which can be used to achieve a trade-off between coding speed and overhead. Although the second stage is a crucial research problem for real applications, we consider it out of scope of the current work. Rather, we discuss the computational cost pertaining to the first inference stage. Our further discussion consists of two parts:

1. Computational complexity: given an image with size $N\times N$, for a full autoregressive model the distribution of pixel x_{ij} depends on all previews pixels $p(x_{ij}|x_{[1:i,1:j]})$, such that the computational cost of conditional distribution calculation scales as $\mathcal{O}(N^2)$. For latent variable models (*e.g.* [43, 42]) or flow models (*e.g.* [3]) the predictive distribution of x_i depends on all other pixels and therefore also scales with $\mathcal{O}(N^2)$. In direct contrast, our local autoregressive model only depends on local regions with horizon h, so computation of the distribution $p(x_{ij}|x_{[i-h:i-1,j-h:j+h]}, x_{[i,j-h:j-1]})$ scales only with $\mathcal{O}(h^2)$ and typically $h\ll N$ in practice. This results in significant reduction of the computational cost of our approach, pragmatically. A second key property affecting compression speed is parallelizability, discussed below.
2. Parallelizability: recall that in the compression phase of NeLLoC, all image pixels are observed. This enables the predictive distribution for all image pixels to be generated in parallel. Conversely, in the decompression phase, the predictive distribution of each pixel must be generated in a sequential order, due to the auto-regressive nature of NeLLoC. Thus the decompression stage cannot be directly parallelized and decompression therefore scales with image size. Alternative neural compressors make use of latent variable models [42, 43] and therefore allow for the distribution of each pixel to be independent (given decompressed latents). Flow models with an isotropic base distribution will also result in attractive decompression times on average. Towards speeding up NeLLoc on this axis, we note it also possible to split an image into patches and compress each patch individually, thus parallelizing the decompression phase. Table 6 shows the compression BPD when we split the full ImageNet images into patches. We find that BPD slightly increases when we increase the number of patches. This is because splitting an image into patches assumes each patch is independent, which will weaken the prediction performance. However, this scheme results in a trade-off between compression speed and rate: $\approx$0.1 BPD overhead can offer a $\sim\times400$ speed up in both compression, decompression (assuming infinite computation power), which may prove extremely valuable for practical applications.

6 Conclusion and Future Work

In this work, we propose the Local autoregressive model to study generalization in probabilistic image modelling. We applied the proposed model to OOD detection and built a practical lossless compressor: NeLLoC. Future work will look to study the generalization ability of probabilistic models in other domains *e.g.* text or audio, in order to widen the benefits of the proposed OOD detectors and lossless compressors.

Table 4: Lossless Compression Comparisons. We compare NeLLoC to other lossless compressors including traditional images compression methods and neural network based models. For neural models, we report the results where models are trained on CIFAR10 or ImageNet32 and tested on other ID or OOD test datasets. Results are either obtained from the original work or we test using publicly released models. We use † to represent the best ID generalization and $\star$ to represent the best OOD generalization. (a) We down-sampled the CelebA to 32×32, see Appendix A.2. (b) The BPD 3.15 reported in [43] is tested on 2000 random samples from the full ImageNet testset, whereas we test the HiLLoC on the whole testset with 49032 images. The reported BPD of NeLLoC has standard deviation ~0.02 across multiple random seeds.

Method	Size(MB)	CIFAR	SVHN	CelebA[a]	ImgNet32	ImgNet64	ImgNet
Generic							
PNG [5]	N/A	5.87	5.68	6.62	6.58	5.71	5.12
WebP [26]	N/A	4.61	3.04	4.68	4.68	4.64	3.66
JPEG2000 [34]	N/A	5.56	4.10	5.70	5.60	5.10	3.74
FLIF [41]	N/A	4.19	2.93	4.44	4.52	4.19	3.51
Train/test on one distribution		Train			Train	Train	
LBB [14]	-	**3.12**†	-	-	3.88	3.70	-
IDF++[3]	-	3.26	-	-	4.12	4.81	-
Trained on CIFAR		Train					
IDF [15]	223.0	3.34	-	-	4.18	3.90	-
Bit-Swap [20]	44.7	3.78	2.55	3.82	5.37	-	-
HiLLoC [43]	156.4	3.32	2.29	3.54	4.89	4.46	3.42
L3C [30]	19.11	3.39	3.17	4.44	4.97	4.77	4.88
NeLLoC ($r=1$)	**1.34**	3.28	2.16	3.37	4.07	3.74	3.25
NeLLoC ($r=3$)	2.75	3.24	**2.13***	**3.35***	**4.02***	**3.69***	**3.24***
Trained on ImgNet32					Train		
IDF [15]	223.0	3.60	-	-	4.18	3.94	-
Bit-Swap [20]	44.9	3.97	3.00	3.87	4.23	-	-
HiLLoC [43]	156.4	3.56	2.35	3.52	4.20	3.89	**3.25**$^{\star\,b}$
L3C [30]	19.11	4.34	3.21	4.27	4.55	4.30	4.34
NeLLoC ($r=1$)	**1.34**	3.56	2.26	3.47	3.85	3.55	3.31
NeLLoC ($r=3$)	2.75	**3.51***	**2.21**	**3.43***	**3.82**†	**3.53***	3.29

Table 5: Effect of horizon size for NeLLoC. All the models have residual block number $r=1$ and trained on CIFAR10. The reported BPD has standard deviation 0.02 across multiple random seeds.

Horizon	Size (MB)	CIFAR	SVHN	CelebA	ImgNet32	ImgNet64	ImgNet
$h=2$	1.18	3.38	2.21	3.42	4.08	3.75	3.30
$h=3$	1.25	3.28	2.16	3.37	4.07	3.74	3.25
$h=4$	1.34	3.26	2.15	3.37	4.07	3.73	3.24
$h=5$	1.46	3.25	2.15	3.37	4.06	3.73	3.24

Table 6: Parallelization using patches on full ImageNet. The model uses $h=3$, $r=1$ and is trained on CIFAR10. We use *e.g.* '400x' to denote an image is split into 400 patches. The reported BPD has standard deviation ~0.02 across multiple random seeds.

Method	1x	16x	25x	100x	400x
NeLLoC	3.25	3.26	3.27	3.29	3.34

References

[1] F. Ahmed and A. Courville. Detecting semantic anomalies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3154–3162, 2020.

[2] A. A. Alemi, I. Fischer, and J. V. Dillon. Uncertainty in the variational information bottleneck. *arXiv preprint arXiv:1807.00906*, 2018.

[3] R. v. d. Berg, A. A. Gritsenko, M. Dehghani, C. K. Sønderby, and T. Salimans. Idf++: Analyzing and improving integer discrete flows for lossless compression. *arXiv preprint arXiv:2006.12459*, 2020.

[4] T. Bird, F. H. Kingma, and D. Barber. Reducing the computational cost of deep generative models with binary neural networks. *arXiv preprint arXiv:2010.13476*, 2020.

[5] T. Boutell and T. Lane. Png (portable network graphics) specification version 1.0. *Network Working Group*, pages 1–102, 1997.

[6] R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[7] H. Choi, E. Jang, and A. A. Alemi. Waic, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

[8] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha. Deep learning for classical japanese literature, 2018.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[12] D. Hendrycks, M. Mazeika, and T. Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[13] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[14] J. Ho, E. Lohn, and P. Abbeel. Compression with flows via local bits-back coding. *arXiv preprint arXiv:1905.08500*, 2019.

[15] E. Hoogeboom, J. W. Peters, R. v. d. Berg, and M. Welling. Integer discrete flows and lossless compression. *arXiv preprint arXiv:1905.07376*, 2019.

[16] P. G. Howard and J. S. Vitter. Practical implementations of arithmetic coding. In *Image and text compression*, pages 85–112. Springer, 1992.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] F. Kingma, P. Abbeel, and J. Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In *International Conference on Machine Learning*, pages 3408–3417. PMLR, 2019.

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

[22] R. Krusinga, S. Shah, M. Zwicker, T. Goldstein, and D. Jacobs. Understanding the (un) interpretability of natural image distributions using generative models. *arXiv preprint arXiv:1901.01499*, 2019.

[23] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[24] Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[25] K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177, 2018.

[26] L. Lian and W. Shilei. Webp: A new image compression format based on vp8 encoding. *Microcontrollers & Embedded Systems*, 3, 2012.

[27] S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[28] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[29] D. J. MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[30] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10629–10638, 2019.

[31] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Do deep generative models know what they don't know? *International Conference on Learning Representations*, 2019.

[32] E. Nalisnick, A. Matsukawa, Y. W. Teh, and B. Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using a test for typicality. *arXiv preprint arXiv:1906.02994*, 5, 2019.

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

[34] M. Rabbani. Jpeg2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, 11(2):286, 2002.

[35] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. Depristo, J. Dillon, and B. Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, pages 14707–14718, 2019.

[36] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[37] C. S. Sastry and S. Oore. Detecting out-of-distribution examples with in-distribution examples and gram matrices. *arXiv preprint arXiv:1912.12510*, 2019.

[38] R. T. Schirrmeister, Y. Zhou, T. Ball, and D. Zhang. Understanding anomaly detection with deep invertible networks through hierarchies of distributions and features. *arXiv preprint arXiv:2006.10848*, 2020.

[39] J. Serrà, D. Álvarez, V. Gómez, O. Slizovskaia, J. F. Núñez, and J. Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. *arXiv preprint arXiv:1909.11480*, 2019.

[40] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

[41] J. Sneyers and P. Wuille. Flif: Free lossless image format based on maniac compression. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 66–70. IEEE, 2016.

[42] J. Townsend, T. Bird, and D. Barber. Practical lossless compression with latent variables using bits back coding. *arXiv preprint arXiv:1901.04866*, 2019.

[43] J. Townsend, T. Bird, J. Kunze, and D. Barber. Hilloc: Lossless image compression with hierarchical latent variable models. *arXiv preprint arXiv:1912.09953*, 2019.

[44] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. PMLR, 2016.

[45] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[46] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See section 4 and section 5.
   (b) Did you describe the limitations of your work? [Yes] See Parallelizability part in section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix D.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We have read them.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We don't have theoretical results.
   (b) Did you include complete proofs of all theoretical results? [N/A] We don't have theoretical results.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix A.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix C.
   (b) Did you mention the license of the assets? [Yes] See Appendix C.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A] We didn't use any new assests.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] The datasets are open for research.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Appendix C.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We do not use crowdsourcing or conducted research with human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We do not use crowdsourcing or conducted research with human subjects.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We do not use crowdsourcing or conducted research with human subjects.