Elevating Perceptual Sample Quality in Probabilistic Circuits through Differentiable Sampling

Steven Lang
TU Darmstadt
steven.lang@cs.tu-darmstadt.de

Martin Mundt
TU Darmstadt
martin.mundt@cs.tu-darmstadt.de

Fabrizio Ventola
TU Darmstadt
ventola@cs.tu-darmstadt.de

Robert Peharz
TU Graz
r.peharz@tue.nl

Kristian Kersting
TU Darmstadt
kersting@cs.tu-darmstadt.de

Abstract

Deep generative models have seen a dramatic improvement in recent years, due to the use of alternative losses based on perceptual assessment of generated samples. This improvement has not yet been applied to the model class of probabilistic circuits (PCs), presumably due to significant technical challenges concerning differentiable sampling, which is a key requirement for optimizing perceptual losses. This is unfortunate, since PCs allow a much wider range of probabilistic inference routines than main-stream generative models, such as exact and efficient marginalization and conditioning. Motivated by the success of loss reframing in deep generative models, we incorporate perceptual metrics into the PC learning objective. To this aim, we introduce a differentiable sampling procedure for PCs, where the central challenge is the non-differentiability of sampling from the categorical distribution over latent PC variables. We take advantage of the Gumbel-Softmax trick and develop a novel inference pass to smoothly interpolate child samples as a strategy to circumvent non-differentiability of sum node sampling. Our hypothesis is that perceptual losses, unlocked by our novel differentiable sampling procedure, will elevate the generative power of PCs and improve their sample quality to be on par with neural counterparts like probabilistic auto-encoders and generative adversarial networks.

1 Introduction

The central task of approximating data-generating distributions by means of probabilistic models has experienced impressive improvements with the advent of deep learning. However, when considering the in-depth novelties that underlie the corresponding advancements, it seems that several improvements are independent of the neural network and its parameters. The neural architecture of encoders and decoders involved in state-of-the-art deep generative models is predominantly shared among methods. Instead, it could be argued that major achievements were an immediate result of iterative stages of reframing the optimization and the particular employed objectives.

For instance, for natural images, the initial sampled generation on the basis of denoising auto-encoders [2] (DAE), or variational inference in neural networks [11, 20] (VAE), was rapidly improved upon with the inclusion of a min-max objective that learns to separate real from fake data in a discriminator of a generative adversarial network (GAN) [10]. The latter approach, promoted to emphasize a "perceptual" measure over pixel values, has then been brought back into a hybrid VAEGAN model [24]. Subsequently, several concurrent realizations of adversarial training have argued that an encoder-decoder is sufficient to formulate a min-max adversarial objective [17, 41], without the presence of an extra discriminator. The loop has seemingly been closed with the proposal of the perceptual auto-encoder [45], linking advancements back to the original neural starting point. Orthogonal developments have simultaneously improved generation quality through the addition of autoregressive sampling steps [4, 13] and losses motivated from a perspective of optimal transport with the incorporation of Wasserstein distances [1, 39].

Motivated by this evolution of deep generative models, we posit that the prevalence of (deep) neural networks is not primarily due to the nature of their neural architectural design and computational operations, but rather from objectives that directly relate samples from the approximated distribution to ground-truth data instances, in an attempt to introduce perceptual metrics into training. To date, the latter practice is absent from tractable probabilistic model alternatives, such as the family of probabilistic circuits (PCs) [5, 6, 16, 22, 33, 42], which focus their training efforts on maximum likelihood estimation (MLE). Despite these models having a crucial advantage of providing general and computationally efficient inference, a large scale demonstration of their generative modeling capabilities comparable to their neural network counterparts still remains open. *Our paper's central hypothesis is that the perceived gap to neural networks can be bridged and probabilistic circuits provide an equally adequate alternative, if their objective is posed from an analogous perspective. That is, we introduce to PCs a formulation of objectives over samples from the approximated distribution and ground-truth instances.*

To gather experimental evidence in support of our hypothesis, we propose to introduce differentiable sampling into PCs. This provides the currently missing component required for flexible loss formulation to include similarity measures between samples and data. Therefore, we present a novel differentiable sampling procedure for PCs, which provides continuous approximations to the non-differentiable operations in the standard sampling procedure. Finally, an evaluation of PCs on methods such as probabilistic auto-encoding [20], adversarial training [10], and maximum mean discrepancy optimization [12] will be enabled.

2 Related Work

Probabilistic circuits (PCs) are a set of expressive probabilistic models that provide exact and tractable inference for a wide range of probabilistic queries. Well-known representatives of PCs are arithmetic circuits [5, 6], cutset networks [33], probabilistic sentential decision diagrams [22], and sum-product networks (SPNs) [16]. PCs are graphical models which connect a set of random variables (RVs), modeled in leaf nodes, by repeatedly and hierarchically assuming independence between RVs (product node) and building convex mixtures over these local independencies (sum node). For our purpose, we concentrate on SPNs, although the findings of this work will be equally applicable to every member of PCs. SPNs constrain their graphs to be *smooth*, i.e. all children of a sum node must have equal scope, in other words, the distribution they encode should be defined over the same set of RVs. Moreover, SPNs are *decomposable*, i.e. all children of a product node must have pairwise disjoint scope. These constraints allow for efficient marginalization and conditioning routines, providing great representational power and efficiency for probabilistic queries. Recent work on the computational efficiency [34, 36], and specialized SPN structures such as EinsumNetworks [30] and LibSPN [32], leverage modern GPU architectures to further scale SPN model complexity and reduce the gap to deep neural networks in terms of model capacity.

SPN parameters, i.e. sum node weights and leaf node distribution parameters, are optimized by means of maximum likelihood estimation. To date, SPNs cannot make use of flexible formulations of optimization objectives that include perceptual loss terms involving the generated samples. In contrast, due to the continuous nature of the way deep neural components are arranged, the generation of samples in deep probabilistic models admits direct formulation of objectives $\mathcal{L}\left(\boldsymbol{x}, \boldsymbol{x}^*\right)$ in the input domain, where $\boldsymbol{x}$ is a data instance and $\boldsymbol{x}^*$ is a sample generated by the model.

2.1 Flexible Loss Formulations in Neural Networks

As an early example, the generalization of denoising auto-encoders in Bengio et al. [2] has formulated the denoising procedure from a probabilistic perspective. That is, given a data instance $\boldsymbol{x}$, a noisy sample $\hat{\boldsymbol{x}} \sim C(\hat{\boldsymbol{x}} \mid \boldsymbol{x})$ is generated from a corruption process C and the expected value of $-\log P_\theta(\boldsymbol{x} \mid \hat{\boldsymbol{x}})$, a reconstruction loss, is then minimized. A variational inference approach to auto-encoding has been suggested in Kingma and Welling [20]. Here, a variational approximation $q_\theta(\boldsymbol{z} \mid \boldsymbol{x})$, encoded through the parameters of a neural network, serves as the approximation to the true posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$. A lower bound on the data likelihood $p(\boldsymbol{x})$, consisting of a reconstruction error between the data instance and a reconstruction $\boldsymbol{x}^* \sim p(\boldsymbol{x} \mid \boldsymbol{z})$ from the decoder, and a negative Kullback-Leibler divergence term that represents the distance between the variational posterior and a typically isotropic Gaussian prior $p(\boldsymbol{z}) = \mathcal{N}(0, 1)$, are optimized jointly with the help of a reparametrization trick.

A different strategy has been proposed by Goodfellow et al. [10] who construct a min-max game objective. The introduced model consists of two components i.e. a generator G, generating samples, and a discriminator D, that discerns whether an input is generated by G (fake) or whether it is a true data instance (real) by incorporating a perceptual loss on the quality of samples generated by G. The objectives are formulated such that D maximizes its accuracy, while G is pushed to generate samples that fool D, thus, reducing its accuracy. Since the sample generation by G and the fake-real discrimination by D are differentiable, the two components can be trained in an end-to-end fashion, allowing for direct optimization of perceived sample quality.

Therefore, a crucial component seems to be differentiability in sample generation, which is still missing from PC formulations. In this work, we address this challenge and thus bring a broader set of optimization objectives to PCs, similar to the one found in deep neural models. In Section 3, we first summarize the parts of the standard sampling in PCs that break differentiability and then continue to propose a novel procedure that permits gradient flow from model samples to parameters, therefore, making PCs amenable to the aforementioned flexible loss formulations.

2.2 Complementary Advances

Whereas the previous section has highlighted the role of differentiability in sampling and the consequent possibilities of flexible loss formulations as a key element of sample quality improvements, there naturally exist other avenues that have lead to further advances in recent years. For example, sophisticated data-augmentation techniques have been proposed and are commonly employed in practice. To name a few, geometric transformations, color space augmentations, image mixing, random erasing, feature space augmentation, all lead to respective model performance boosts, as summarized in several timely surveys [8, 35, 43]. While interesting, these are fully complementary to perceptual loss formulations and the concept under investigation in this work. Therefore, the additional inclusion of these auxiliary efforts should be inspected in separate future work.

Another line of recent concurrent work are diffusion models [15, 21, 37]. The latter sample by reversing a gradual noising process and, regarding sample quality, have been shown to be on par with GANs [7]. Whereas diffusion models are conceptually different, we argue that they can also be perceived as a fundamental way of re-framing the optimization objective from a perceptual point of view, i.e. learning to produce a slightly less noisy $\boldsymbol{x}_{t-1}$ from $\boldsymbol{x}_t$ until reaching a final sample $\boldsymbol{x}_0$.

3 Method

To gain comparable benefits from employing perceptual loss quantities as deep neural models do, we hypothesize that permitting gradient flow during sample generation w.r.t. the model's parameters can elevate PCs to similar data generation quality as deep neural models.

3.1 Sampling in Sum-Product Networks

Sampling in SPNs is performed in a top-down fashion by starting at the root node, descending into either all of its children, in the case of a product node, or one of its children, in the case of a sum node. When a leaf node is reached, the distribution modeled by that leaf node is sampled. The particular sampling procedures for sum and product nodes are defined in the following.

Product nodes in SPNs are decomposable and, thus, have children with pairwise non-overlapping scopes. Therefore, the sampling procedure for a product node P simply requires to sample from all of its children $\mathbf{ch}(\mathsf{P})$, i.e. $\boldsymbol{x}^* = \{\boldsymbol{x} \sim \mathsf{N} \mid \forall \mathsf{N} \in \mathbf{ch}(\mathsf{P})\}$. Figure 1a depicts the sampling procedure for a sum node S sampling procedure as a computation graph. Interpreting the sum node weights $\boldsymbol{w}_{\mathsf{S}} = \{\boldsymbol{w}_{\mathsf{S},\mathsf{N}} \mid \mathsf{N} \in \mathbf{ch}(\mathsf{S})\}$ as categorical probabilities, we first sample from a categorical distribution $i \sim \text{Cat}(\boldsymbol{w}_{\mathsf{S}})$, where i is used as an index to obtain the sampled sum node child $\mathsf{N}_i^* = \mathbf{ch}(\mathsf{S})\,[i]$ and we finally obtain the actual sample $\boldsymbol{x}^* \sim \mathsf{N}_i^*$.

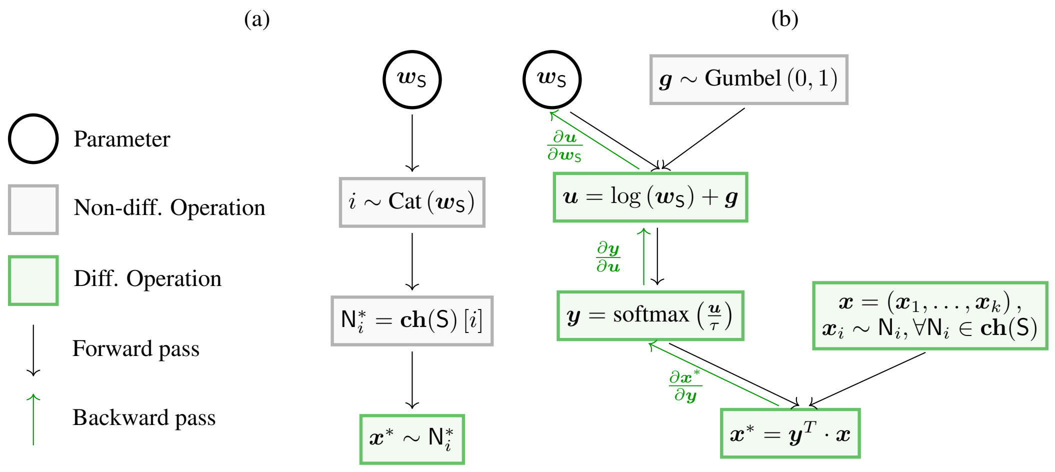
(a)
(b)
Parameter
Non-diff. Operation
Diff. Operation
Forward pass
Backward pass
$\boldsymbol{w}_{\mathsf{S}}$
$i \sim \mathrm{Cat}(\boldsymbol{w}_{\mathsf{S}})$
$\mathsf{N}_i^* = \mathbf{ch}(\mathsf{S})[i]$
$\boldsymbol{x}^* \sim \mathsf{N}_i^*$
$\boldsymbol{g} \sim \mathrm{Gumbel}(0,1)$
$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{w}_{\mathsf{S}}}$
$\boldsymbol{u} = \log(\boldsymbol{w}_{\mathsf{S}}) + \boldsymbol{g}$
$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{u}}$
$\boldsymbol{y} = \mathrm{softmax}\left(\frac{\boldsymbol{u}}{\tau}\right)$
$\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$,
$\boldsymbol{x}_i \sim \mathsf{N}_i, \forall \mathsf{N}_i \in \mathbf{ch}(\mathsf{S})$
$\frac{\partial \boldsymbol{x}^*}{\partial \boldsymbol{y}}$
$\boldsymbol{x}^* = \boldsymbol{y}^T \cdot \boldsymbol{x}$

Figure 1: Computation graphs for the (a) standard and the (b) differentiable sampling procedure in an SPN sum node S. Gray nodes indicate non-differentiable operations which break the gradient flow on the path $\boldsymbol{w}_{\mathsf{S}} \rightarrow \boldsymbol{x}^*$. Using the Gumbel-Softmax Trick combined with smooth child interpolation, we are able to construct a computation path $\boldsymbol{w}_{\mathsf{S}} \rightarrow \boldsymbol{x}^*$, which consists of only smooth functions, allowing gradient computation $\partial \boldsymbol{x}^* / \partial \boldsymbol{w}_{\mathsf{S}}$ between the generated sample $\boldsymbol{x}^*$ and the sum node parameters $\boldsymbol{w}_{\mathsf{S}}$ using the chain-rule.

The sum node sampling steps break differentiability in two points as highlighted in Figure 1a. The first operation is the drawing from a categorical distribution, which is a not differentiable due to its discrete nature. The second issue arises when obtaining the sampled child node N^* by indexing the set of all children $\mathbf{ch}(\mathsf{S})$ which is a discrete operation. Indexing prohibits gradients w.r.t. the index itself, thus, breaking differentiability. In the following, we explain how to tackle these challenges and formulate a differentiable sampling procedure for SPNs.

3.2 Differentiable Sampling in Sum-Product Networks

To leverage differentiability during sampling in SPNs, we propose a novel sampling procedure that differs from the standard sampling procedure in two ways. We first replace the discrete categorical sampling in sum nodes by a continuous approximation, making use of the well-known Gumbel-Softmax Trick [18, 28]. Then, we perform smooth interpolation of sum node children to overcome the discretization introduced by indexing a single child.

Gumbel-Softmax Trick Let z be a categorical variable with probabilities $\pi_1, \ldots, \pi_k$, the Gumbel-*Max* trick allows us to draw samples z from a categorical distribution $z = \arg\max_i \{\log(\pi_i) + g_i\}$, with gumbel distributed noise $g_i \sim \mathrm{Gumbel}(0,1)$. To relax the discretization introduced by the $\arg\max$ operation, Jang et al. [18] have proposed to use the softmax function as a continuous, differentiable approximation to $\arg\max$, leading to a k-dimensional sample vector $\boldsymbol{y}$ in the simplex Δ^{k-1} where $\boldsymbol{y} = \mathrm{softmax}((\boldsymbol{\pi} + \boldsymbol{g})/\tau)$. The temperature τ controls the approximation precision. As τ goes towards zero, the Gumbel-Softmax Distribution [18] converges to the categorical distribution with probabilities π_i. For $\tau > 0$, the Gumbel-Softmax distribution is smooth and has a well-defined gradient $\partial y_i / \partial \pi_i$ with respect to its parameters π_i.

Smooth Child Interpolation For the sake of simplicity, let us first explore the issue of discretization induced by child node indexing for the case when $\tau \rightarrow 0$. After applying the Gumbel-Softmax Trick, sampling from a sum node S results in a one-hot encoded vector $\boldsymbol{y}$ with $y_i = 1$ at exactly one position (the sampled child node index) and $y_j = 0$ for all $j \neq i$. Let's assume a scenario in which we have already sampled from all children of S, i.e. we have a vector $\boldsymbol{x}^* = (\boldsymbol{x}_1^* \sim \mathsf{N}_1, \ldots, \boldsymbol{x}_k^* \sim \mathsf{N}_k)^T$. In the conventional non-differentiable perspective, we are now required to index this vector with the sampled index i. Due to the one-hot encoding of the vector $\boldsymbol{y}$, we can perform the dot-product between $\boldsymbol{x}^*$ and $\boldsymbol{y}$ to obtain an interpolation between all samples $(\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_k^*)^T \cdot \boldsymbol{y} = \boldsymbol{x}_i^*$. In the general case of $\tau > 0$, then, the dot product will result in a mixture over all possible child samples.

Each child sample contributes to the final sample according to the sampled mixture weight y_i. The computation graph of the differentiable sampling procedure is shown in Figure 1b. Alternatively, we can also use the Straight-Through Gumbel Estimator [18] by discretizing $\boldsymbol{y}$ with $\arg\max$ in the forward pass, but use the continuous approximation $\nabla_{\boldsymbol{\pi}} \boldsymbol{z} \approx \nabla_{\boldsymbol{\pi}} \boldsymbol{y}$ in the backward pass.

While this sampling procedure is differentiable, it is not identical to the sampling from the corresponding categorical distribution for $\tau > 0$. This leads to a trade-off during parameter learning, where small τ results in almost one-hot encoded samples and gradients with high variance, while large τ leads to smooth samples with low gradient variance. To allow for stable training, the temperature τ can either be annealed or learned as a parameter, which can be interpreted as entropy regularization [31, 38]. In this case, the Gumbel-Softmax distribution adaptively adjusts the "confidence" of the samples during the training process [18].

Moreover, by creating these smooth top-down evaluations through the network graph using the Gumbel-Softmax Trick, while out of scope for the investigation of this work, it is now possible to further consider variance reduction techniques to deal with discrete distributions in differentiable programs, such as REBAR [40], wake-sleep algorithms [14], or REINFORCE [44].

4 Experimental Protocol

Our experimental protocol is devised to demonstrate the effectiveness of our contributions, namely, the introduced differentiable sampling for PCs to enable flexible objectives. Our main hypothesis is that PCs equipped with the latter can be on par with deep neural models like GANs and VAEs in sample generation. For this purpose, we leverage the implementation of PCs by Peharz et al. [30], where we implement our proposed method, Section 3.1, to take advantage of the flexible loss formulations as seen in neural networks and investigate their impact in PCs. Given that this is uncharted territory for PCs, as previous works did not prioritize perceived image sample quality, our focus is to investigate SPNs with differentiable sampling in the following three concrete settings.

4.1 Flexible Loss Experiments

1. Adversarial Training: We adopt the min-max optimization objective of GANs [10]:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})} \left[\log\left(1 - D(G(\boldsymbol{z}))\right)\right] .$$

In our case, the generator G is an SPN $\mathcal{S}_G$ and the second term of the objective becomes the expectation over samples drawn from $\mathcal{S}_G$: $\mathbb{E}_{\boldsymbol{x} \sim \mathcal{S}_G(\boldsymbol{x})} \left[\log\left(1 - D(\boldsymbol{x})\right)\right]$. The discriminator D can be either: 1) The generator SPN $\mathcal{S}_G$ itself, similar in spirit to adversarial training with introspection [17], 2) a separate, discriminative, SPN $\mathcal{S}_D$ [9], 3) another arbitrary model, e.g. a neural network. We then compare the sample quality with the original GAN, where G and D are modeled by a neural network. For a fair comparison, we will also use the discriminative SPN $\mathcal{S}_D$ with the neural generator G.

2. Maximum Mean Discrepancy: We train SPNs by using and minimizing the Maximum Mean Discrepancy [12] distance:

$$\text{MMD}^2(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\binom{n}{2}} \sum_{i \neq j} k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \frac{2}{\binom{n}{2}} \sum_{i \neq j} k(\boldsymbol{x}_i, \boldsymbol{y}_j) + \frac{1}{\binom{n}{2}} \sum_{i \neq j} k(\boldsymbol{y}_i, \boldsymbol{y}_j) ,$$

as proposed in MMD-GANs [26]. The core idea is to employ the kernel two-sample test, denoted by k in above equation, instead of a discriminator with the goal of minimizing the distance between the target and the estimated distributions. In our evaluation, we contrast against MMD-GAN [26] and employ adversarially learned kernels for both, the MMD-GAN and the SPN.

3. Probabilistic Auto-Encoding: We extend SPNs by explicitly modeling the distribution of the latent space, similar to VAEs. That is, we learn an SPN $\mathcal{S}(\boldsymbol{x}, \boldsymbol{z})$ over the joint of the input $\boldsymbol{x}$ and some latent space variables $\boldsymbol{z}$ via auto-encoding. Given a data point $\boldsymbol{x}_i \sim p_{\text{data}}(\boldsymbol{x})$, we compute its latent representation with the most probable explanation $\boldsymbol{z}_i = \arg\max_i \mathcal{S}(\boldsymbol{z} \mid \boldsymbol{x}_i)$. Then, we generate its reconstruction by sampling $\hat{\boldsymbol{x}}_i \sim \mathcal{S}(\boldsymbol{x} \mid \boldsymbol{z}_i)$. We compare the results with a VAE [20] that has a Gaussian prior. For a fair comparison, we also model the SPN latent variables $\boldsymbol{z}$ as Gaussians.

4.2 Fair Experiment and Evaluation Protocol

We analyze the above three loss formulations and their respective comparison with neural networks on four commonly investigated image datasets: MNIST [25], CIFAR-10 [23], SVHN [29], and CelebA [27]. For each dataset we will use their default data splits for train, test, and validation sets and employ preprocessing, i.e. ensuring zero mean and unit variance for each feature across the dataset. Furthermore, for each proposed experiment we will also compare with a baseline SPN, optimized purely using maximum likelihood. As SPNs and deep neural networks deviate in their model structure, we practice particular caution to warrant a fair comparison that provides conclusive evidence with respect to our postulated hypothesis. For each experiment, we will perform five randomly seeded runs to measure statistical deviations. Specifically:

Optimization: We use Adam [19] as the optimization algorithm in all models. Since the aforementioned loss objectives are novel territory for SPNs, we conduct a grid-search over a discrete set of learning rates and the stochasticity induced by mini-batch size using a separate validation set. The same process is conducted for the neural counterparts. A set of hyper-parameters is then selected individually for each model to assure an adequate operating point.

Metrics: For evaluation purposes we primarily employ the predominant Fréchet inception distance (FID). However, this metric is known to suffer from minor perturbation and resolution changes [3]. Therefore, we also analyze the generative power of each model by training a separate classifier on the corresponding generations, and we assess its classification performance on the original test sets. For the auto-encoding setting, we will further report the reconstruction losses.

Model Capacity: To keep the comparison between original models and the SPN variants fair, we employ the same amount of parameters for each model in direct comparisons.

Data Efficiency and Convergence: Since deep neural networks and PCs have fundamentally different structures, they may have different data efficiency and convergence rate. For this purpose, we evaluate the aforementioned losses and metrics under the lens of different amounts of training instances available for optimization. In addition, to further underline the fairness w.r.t. the arguments in the previous points on optimization and metrics, we also test all models at different points over the course of training, in order to assess generation quality at potentially different convergence speeds.

Ablation: Since the Gumbel-Softmax Trick exposes an additional hyper-parameter, i.e. the temperature τ, we will perform an ablation study between constant, annealed, and interactively learned temperature values. We perform this study on CIFAR-10 for each of the investigations in Section 4 with a fixed learning rate and mini-batch size, to keep the number of experiments practical.

Acknowledgments

This work has been supported by the project "safeFBDC - Financial Big Data Cluster" (FKZ: 01MK21002K), funded by the German Federal Ministry for Economics Affairs and Energy as part of the GAIA-x initiative, the Hessian Ministry of Higher Education, Research, Science and the Arts (HMWK) project "The Third Wave of AI", and the Federal Ministry of Education and Research (BMBF; Competence Center for AI and Labour; "kompAKI", FKZ02L19C150).

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint, arXiv:1701.07875*, 2017.

[2] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. *Conference on Neural Information Processing Systems (NeurIPS)*, 2013.

[3] Ali Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[4] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational Lossy Autoencoder. *International Conference on Learning Representations (ICLR)*, 2017.

[5] Adnan Darwiche. A logical approach to factoring belief networks. *Knowledge Representation and Reasoning (KR)*, 2:409–420, 2002.

[6] Adnan Darwiche. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3):280–305, 2003.

[7] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *arXiv preprint, arXiv:2105.05233*, 2021.

[8] Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for NLP. In *Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 968–988, 2021.

[9] Robert Gens and Pedro Domingos. Discriminative learning of sum-product networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 25, 2012.

[10] I. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

[11] Alex Graves. Practical variational inference for neural networks. *Conference on Neural Information Processing Systems (NeurIPS)*, 2011.

[12] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research (JMLR)*, 13:723–773, 2012.

[13] Ishaan Gulrajani, Kundan Kumar, Ahmed Faruk, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. PixelVAE: a Latent Variable Model for Natural Images. *International Conference on Learning Representations (ICLR)*, 2017.

[14] Geoffrey E. Hinton, Peter Dayan, Brendan J. Frey, and Radford M. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268 5214:1158–61, 1995.

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint, arXiv:2006.11239*, 2020.

[16] Poon Hoifung and Domingos Pedro. Sum-product networks: A new deep architecture. In *Uncertainty in Artificial Intelligence (UAI)*, 2011.

[17] Huaibo Huang, Zhihang Li, Ran He, Zhenan Sun, and Tieniu Tan. Introvae: Introspective variational autoencoders for photographic image synthesis. *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[18] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint, arXiv:1611.01144*, 2017.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[20] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *International Conference on Learning Representations (ICLR)*, 2013.

[21] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *arXiv preprint, arXiv:2107.00630*, 2021.

[22] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic Sentential Decision Diagrams. In *Knowledge Representation and Reasoning (KR)*, 2014.

[23] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Ontario, 2009.

[24] Anders Boesen Lindbo Larsen, Soren Kaae Sonderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *International Conference on Machine Learning (ICML)*, 2016.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *IEEE*, 86(11):2278–2324, 1998.

[26] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. *arXiv preprint, arXiv:1705.08584*, 2017.

[27] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision (ICCV)*, 2015.

[28] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint, arXiv:1611.00712*, 2017.

[29] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Conference on Neural Information Processing Systems (NeurIPS) Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[30] Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van Den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *International Conference on Machine Learning (ICML)*, 2020.

[31] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint, arXiv:1701.06548*, 2017.

[32] Andrzej Pronobis, Avinash Ranganath, and Rajesh P. N. Rao. LibSPN: A library for learning and inference with Sum-Product Networks and TensorFlow. In *International Conference on Machine Learning (ICML) Workshop on Principled Approaches to Deep Learning*, 2017.

[33] Tahrima Rahman, Prasanna V. Kothalkar, and Vibhav Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKKD)*, 2014.

[34] Nimish Shah, Laura I. Galindez Olascoaga, Wannes Meert, and Marian Verhelst. Acceleration of probabilistic reasoning through custom processor architecture. *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020.

[35] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:1–48, 2019.

[36] Lukas Sommer, Michael Halkenhäuser, Cristian Axenie, and Andreas Koch. Spnc: Accelerating sum-product network inference on cpus and gpus. In *IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 53–56, 2021.

[37] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint, arXiv:2011.13456*, 2021.

[38] Christian Szegedy, V. Vanhoucke, S. Ioffe, Jonathon Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[39] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. Wasserstein auto-encoders. *International Conference on Learning Representations (ICLR)*, 2018.

[40] George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 30. Curran Associates, Inc., 2017.

[41] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. It Takes (Only) Two : Adversarial Generator-Encoder Networks. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[42] Antonio Vergari, YooJung Choi, Robert Peharz, and Guy Van den Broeck. Probabilistic circuits: Representations, inference, learning and applications, 2020. Tutorial at AAAI 2020.

[43] Qingsong Wen, Liang Sun, Xiaomin Song, Jing Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[44] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[45] Zijun Zhang, Ruixiang Zhang, Zongpeng Li, Yoshua Bengio, and Liam Paull. Perceptual generative autoencoders. *International Conference on Machine Learning (ICML)*, 2020.