EFFICIENT GENERATION OF STRUCTURED OBJECTS WITH CONSTRAINED ADVERSARIAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite their success, generative adversarial networks (GANs) cannot easily generate *structured objects* like molecules or game maps. The issue is that such objects must satisfy structural requirements (e.g., molecules must be chemically valid, game maps must guarantee reachability of the end goal) that are difficult to capture with examples alone. As a remedy, we propose constrained adversarial networks (CANs), which embed the constraints into the model during training by penalizing the generator whenever it outputs invalid structures. As in unconstrained GANs, new objects can be sampled straightforwardly from the generator, but in addition they satisfy the constraints with high probability. Our approach handles arbitrary logical constraints and leverages knowledge compilation techniques to efficiently evaluate the expected disagreement between the model and the constraints. This setup is further extended to hybrid logical-neural constraints for capturing complex requirements like graph reachability. An extensive empirical analysis on constrained images, molecules, and video game levels shows that CANs efficiently generate valid structures that are both high-quality and novel.

1 INTRODUCTION

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have shown impressive performance on challenging tasks – like image generation (Karras et al., 2018), text-to-image (Zhang et al., 2017), and style transfer (Zhu et al., 2017) – where the goal is to produce believable configurations.

A number of important applications, however, require to generate objects that are *both* credible *and* feasible with respect to hard structural constraints. Examples include generating drug molecules, which must satisfy chemical validity requirements, and game levels, where the goal must be reachable from the starting position. Recent studies (Guimaraes et al., 2017; De Cao & Kipf, 2018; Xue & van Hoeve, 2019) observed that GANs struggle in these tasks. The main reason is that the training examples alone are insufficient to capture the feasibility constraints and thus to guide the model toward producing valid objects.

As a remedy, we propose Constrained Adversarial Networks (CANs), a class of generative models that extend GANs to structured domains. Given a set of examples drawn from a latent distribution and a set of structural constraints, CANs learn to output valid structured with high probability. This is achieved by augmenting the standard GAN loss with a penalty term that discourages the model from producing infeasible structures. Since CANs inject the constraints directly into the learned model, valid structures can be obtained with high probability by performing forward inference on the generator, avoiding the need for costly sampling or optimization steps (Volz et al., 2018).

The penalty term is implemented using the semantic loss (SL) (Xu et al., 2018). The SL leverages knowledge compilation (Darwiche, 2011) to represent arbitrary Boolean constraint as a circuit and uses the latter to measure the mass allocated by the generator to infeasible objects. The procedure is probabilistically sound, exact, and does not require sampling. Notably, the circuit, which can be quite large depending on the complexity of the constraints, can be thrown away after learning.

In addition, we show how to extend the SL complex constraints that would normally lead to intractably large circuits. This is accomplished by first using a neural network to map configurations to a different space in which the structural constraints can be compactly encoded, and then applying the SL to the latter. This enables us to deal with reachability on a graph, which is beyond the reach of standard SL. Finally, we show how the validity constraints baked into the generator can be enabled or disabled during inference using ideas from InfoGANs Chen et al. (2016).

In summary, we contribute:

- Constrained Adversarial Networks (CANs), a new class of GANs in which the generator is encouraged (at training time) to output valid structures with high probability via forward inference.
- An extensive empirical analysis on structured objects like constrained images, molecules, and video game levels showing that CANs generate structures that are stylistically coherent with the training data.
- A decomposition of intractably complex constraints into a neural and logical components, showcased on graph reachability for video game level synthesis.

2 Unconstrained GANs

Let $\mathcal{X}$ be the object space and $\mathcal{Z}$ some latent space. GANs (Goodfellow et al., 2014) are composed of a generator $g : \mathcal{Z} \to \mathcal{X}$ and a discriminator $d : \mathcal{X} \to [0, 1]$, both implemented as neural networks. The discriminator is trained to recognize "real" objects $\boldsymbol{x}$ taken from the data distribution P_r, while the generator is trained to map random latent vectors $\boldsymbol{z} \in \mathbb{R}^m$ to "fake" objects that fool the discriminator. Learning amounts to solving the minimax game:

$$\min_g \max_d f_{\text{GAN}}(g, d) \qquad f_{\text{GAN}}(g, d) := \mathbb{E}_{\boldsymbol{x}\sim P_r}[\log P_d(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim P_g}[\log(1 - P_d(\boldsymbol{x}))] \quad (1)$$

Here $P_g(\boldsymbol{x})$ is the distribution induced by the generator and $P_d(\boldsymbol{x}) := P_d(\text{real} \mid \boldsymbol{x})$ by the discriminator. It is worth noting that the value of the game defines (a constrained form of) the Jensen-Shannon entropy and that Eq. 1 can be viewed as divergence minimization (Nowozin et al., 2016; Mescheder et al., 2017). After training, new objects can be generated by sampling random vectors $\boldsymbol{z}$ and mapping them to object space with the generator $\boldsymbol{x} = g(\boldsymbol{z})$.

Goodfellow et al. (2014) have shown that, under idealized assumptions, the learned generator g matches the data distribution, that is $P_g = P_r$. We report their theorem here for completeness:

Theorem 1. *So long as (i) g and d are non-parametric, and (ii) the leftmost expectation in Eq. 1 is approximated arbitrarily well by the examples alone, any global equilibrium (g^*, d^*) of Eq. 1 satisfies $P_{d^*} \equiv \frac{1}{2}$ and $P_{g^*} \equiv P_r$.*

Finding Nash equilibria of non-convex games like Eq. 1 is notoriously hard, rendering GAN training very challenging. The most common optimization algorithm is alternating gradient descent, whereby d and g are optimized sequentially, or the more well-understood simultaneous gradient descent (Mescheder et al., 2017). A common failure state is mode collapse, whereby the generator outputs objects concentrated in a tiny portion of the object space. A plethora of theoretical and empirical remedies have been proposed, cf. (Salimans et al., 2016; Mescheder et al., 2018), including using alternative divergences (Nowozin et al., 2016; Arjovsky et al., 2017) and encouraging smoothness of the discriminator by, e.g., controlling the spectral norm of its parameters (Miyato et al., 2018). In our experiments, we apply some of these strategies to stabilize training.

In structured tasks, the objects of interest are usually discrete. GANs can be adapted to such settings by having the generator output a *categorical distribution* $\boldsymbol{\theta}(\boldsymbol{z})$ over $\mathcal{X}$ and sampling objects from the latter. Below, we will focus on stochastic generators of this kind, although alternatives do exist (Gulrajani et al., 2017). In this setting $P_g(\boldsymbol{x}) = \int P_g(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z} = \int \boldsymbol{\theta}(\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z} = \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{\theta}(\boldsymbol{z})]$.

3 Generating Structures with CANs

Our goal is to learn to generate structures $\boldsymbol{x}$ consistent with respect to some validity constraint ψ according to some unobserved distribution P_r. Throughout, we will make the following assumptions:

1. A single validity constraint ψ is provided as input. This is without loss of generality: if multiple constraints are necessary, then ψ can be taken to be their conjunction.

2. The constraint ψ is compatible with the data distribution, that is, the support of P_r falls entirely within the feasible region determined by ψ.

These assumptions hold in many tasks of interest, including all generative problems with known well-formedness requirements. For the sake of simplicity, we also restrict our study to binary (i.e. 0–1) variables and logical constraints (aka formulas) only. This is a very general setup: any discrete structured space can be encoded using binary variables and formulas, at the cost of a larger model.

Limitations of GANs Generators learned with the standard GAN training rule can output invalid structures, for two main reasons. First, for non-trivial constraints ψ, any finite set of examples that satisfy the constraint is insufficient to capture the full semantics of ψ. Second, in many cases of interest the examples are not even consistent with ψ. This more challenging case shows that regular GANs are easily lured into learning *not* to satisfy the constraint. More formally:

Corollary 1. *Under the assumptions of Theorem 1, given a target distribution P_r, a constraint ψ consistent with it, and a dataset of examples $\boldsymbol{x}$ sampled i.i.d. from a corrupted distribution $\tilde{P}_r \neq P_r$ inconsistent with ψ, GANs associate non-zero mass to infeasible objects.*

Indeed, by Theorem 1, the optimal generator satisfies $P_g \equiv \tilde{P}_r$, which is inconsistent with ψ. Ergo, $\sum_{\boldsymbol{x}} \mathbb{1}\{\boldsymbol{x} \not\models \psi\} P_g(\boldsymbol{x}) > 0$. We stress that Theorem 1 captures the *intent* of GAN training, and thus this simple corollary shows that GANs are *by design* incapable of handling invalid examples.

Constrained adversarial networks In order to avoid these issues, constrained adversarial networks (CANs) take both the examples and the validity constraint into account during learning. More specifically, the CAN value function is designed so that the generator maximizes the probability of generating valid structures $P_g(\psi) := \mathbb{E}_{\boldsymbol{x} \sim P_g}[\mathbb{1}\{\boldsymbol{x} \models \psi\}]$, that is:

$$f_{\text{CAN}}(g, d) := f_{\text{GAN}}(g, d) - \lambda \log P_g(\psi) \tag{2}$$

Here $\lambda > 0$ is a hyper-parameter controlling the importance of the constraint. Since the second term is always non-negative, the CAN value function upper bounds the GAN one (Eq. 1).

The second term is the so-called *semantic loss* (SL), proposed in (Xu et al., 2018) to inject knowledge into neural networks, and it is formally defined as $SL_\psi(g) \propto -\log P_g(\psi)$, where:

$$P_g(\psi) = \textstyle\sum_{\boldsymbol{x}'} \mathbb{1}\{\boldsymbol{x}' \models \psi\} P_g(\boldsymbol{x}') = \mathbb{E}_{\boldsymbol{z}}[\sum_{\boldsymbol{x}' \in \mathcal{X} : \boldsymbol{x}' \models \psi} \prod_{i \,:\, x'_i = 1} \theta_i \prod_{i \,:\, x'_i = 0} (1 - \theta_i)] \tag{3}$$

Here $\boldsymbol{x}' \models \psi$ means that $\boldsymbol{x}'$ satisfies constraint ψ and the sum runs over all configurations $\boldsymbol{x}'$ consistent with ψ. The SL can also be viewed as the negative log-likelihood of ψ w.r.t. the generator. This shows that, in Eq. 2, the SL rewards the generator g proportionally to the mass it allocates to valid structures. Since the SL is the negative logarithm of a polynomial in $\boldsymbol{\theta}$, it is fully differentiable (so long as $P_\psi(g) \neq 0$, which is always the case in practice).

If the SL is given large enough weight, CANs are strongly encouraged to generate valid structures in expectation. Under the preconditions of Theorem 1, it can be shown that or $\lambda \to \infty$ CANs generate valid structures *only*:

Proposition 1. *Under the assumptions of Corollary 1, CANs associate zero mass to infeasible objects, irrespective of the discrepancy between P_r and $\tilde{P}_r$.*

Intuitively, this holds because with $\lambda = \infty$ any global equilibrium (g^*, d^*) of $\min_g \max_d f_{\text{CAN}}(g, d)$ *must* minimize the second term. If g is non-parametric, then the minimum is attained with $\log P_{g^*}(\psi) = 0$, which is equivalent to $P_{g^*}(\psi) = 1$. This in turn implies that $P_{g^*}(\neg\psi) = 0$, proving the claim. Of course, as with standard GANs, the prerequisites are often violated in practice. Regardless, Proposition 1 works as a sanity check, and shows that – in contrast to GANs – CANs are appropriate for constrained generative tasks.

Computing the semantic loss Naïve evaluation of the SL involves summing over all (exponentially many) possible configurations $\boldsymbol{x}'$, which is intractable. More generally, computing the SL amounts to weighted model counting, which is #P-complete (Chavira & Darwiche, 2008).

For many constraints of interest, however, the polynomial in Eq. 3 can be factorized into a much more compact form and evaluated very efficiently. Following Xu et al. (2018), we make use of knowledge compilation to automatically factorize the polynomial by compiling it into an arithmetic circuit (more specifically, into a sentential decision diagram (SDD) (Darwiche, 2011)). When the resulting circuit is small enough, evaluation of the SL and of its gradient are extremely efficient, as shown in (Xu et al., 2018).

The main downside of knowledge compilation is that, depending on the complexity of the constraint ψ at hand, the compiled circuit may be very large. This is less of a problem during training, which is often performed on powerful machines, but it can be an issue for inference – especially on embedded devices. However, in CANs the circuit is not required for inference (as it consists of a simple forward pass over the generator), and thus it can be thrown away after training. This means that CANs incur no space penalty during inference when compared to GANs.

Dealing with intractable constraints When fed a particularly complex constraint, knowledge compilation may produce a circuit too large even for the training stage. In this case, we approximate the semantic loss by first mapping the objects from $\mathcal{X}$ to an application-specific space where ψ can be expressed in compact form, and then use the semantic loss on top of the transformed objects. We successfully employed this technique to synthesize mario levels where the goal tile is reachable from the starting tile; all details are provided below. The same technique can be exploited for dealing with very complex logical formulas beyond the reach of state-of-the-art knowledge compilation.

4 Experiments

We implemented CANs using Tensorflow and PySDD[1] and tested them using different generator architectures on one synthetic and two real-world structured generative tasks. In all cases, we compared the objects generated by CANs against those output by strong baselines using three metrics (adopted from Samanta et al. (2018)):

- **validity** is the proportion of sampled objects that are valid;
- **novelty** is the proportion of valid sampled objects that are not present in the training data;
- **uniqueness** is the proportion of valid unique (non-repeated) sampled objects;

Our experimental evaluation aims at answering the following questions:

- **Q1** Can CANs with tractable constraints achieve better results than GANs?
- **Q2** Can CANs with intractable constraints achieve better results than GANs?
- **Q3** Can constraints be combined with rewards to achieve better results than using rewards only?

4.1 Constrained images

In this synthetic experiment, we used CANs to generate small, strongly constrained bitmap images. The training set is composed of $23,040$ black-and-white 20×20 images, each with a black background and two randomly placed random polygons – either a triangle (30% of the cases), a square (30%), or a diamond (40%). The two polygons are always different from each other, fully contained in the canvas, and do not overlap. The leftmost and rightmost columns are special, in that they contain parity information about the image. More specifically, row r of the leftmost column encodes the parity bit for all pixels in the left half of row r, see Fig. 1 (left) for an illustration. The rightmost column does the same for the right half of the image. Fig. 1 (middle) shows a sample image. As a baseline GAN, we implemented the generator and discriminator as deconvolutional and convolutional networks, respectively, sampling from the concrete distribution (Maddison et al., 2016) and using the vanilla loss. The CAN uses the same architecture, except for the extra semantic loss term which encodes the parity constraints using XOR formulas.

Fig. 1 (right) shows the results in terms of validity, novelty and uniqueness computed over 300 objects generated using CANs (with $\lambda = 1$) and Vanilla GANs, where novelty and uniqueness are computed over the subset of valid objects. CANs generate more valid objects, improving at the same

[1]URL:`https://pypi.org/project/PySDD/`

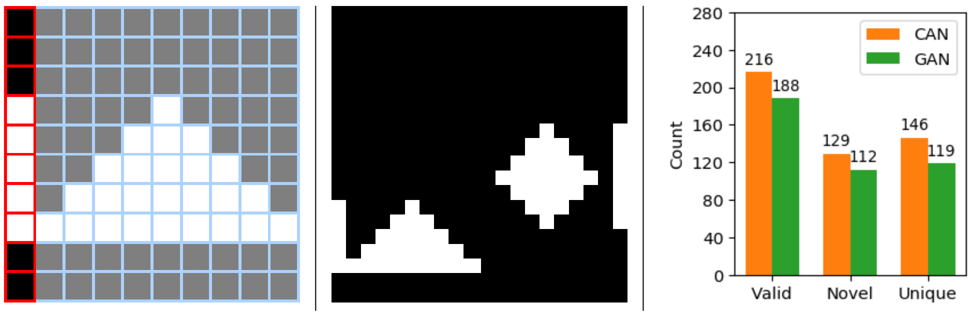

Figure 1: Left: illustration of the parity constraint at the left border. Middle: example from the synthetic dataset. Right: comparison of validity, novelty and uniqueness between CAN and GAN.

time both novelty and uniqueness. These results allow to answer **Q1** affirmatively. Note that in this experiment, using CANs over Vanilla GANs results in a negligible training overhead. On average, training CANs takes 30 minutes against the 27 minutes taken by Vanilla GANs.

4.2 Super Mario Bros level generation

In the next experiment we show how CANs can help in the challenging task of learning to generate videogame levels from user-authored content. While procedural approaches to videogame level generation have successfully been used for decades, the application of machine learning techniques in the creation of (functional) content is a relatively new area of research (Summerville et al., 2018). On the one hand, modern video game levels are characterized by aesthetical features that cannot be formally encoded and thus are difficult to implement in a procedure, which motivates the use of ML techniques for task. On the other hand, the levels have often to satisfy a set of functional (hard) constraints that are easy to guarantee when the generator is hand-coded but pose challenges for current machine learning models.

In the following, we show how the semantic loss can be used to encode useful hard constraints in the context of videogame level generation. These constraints might be functional requirements that apply to every generated object or might be contextually used to steer the generation towards objects with certain properties. In our empirical analysis, we focus on *Super Mario Bros* (SMB), possibly one of the most studied video games in tile-based level generation. See Section 5 for related work in SMB level generation.

Recently, Volz et al. (2018) applied Wasserstein GANs (WGANs) (Arjovsky et al., 2017) to SMB level generation. The approach works by first training a generator in the usual way, then using an evolutionary algorithm called Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to search for the best latent vectors according to a user-defined fitness function on the corresponding levels. We stress that this technique is orthogonal to CANs and the two can be combined together. Nonetheless, in the following we compare CANs with CMA-ES, as both techniques can be used to steer the network towards the generation of *playable* levels, i.e. having a feasible path[2] from the left-most to the right-most column of the level. Pictures in 2 shows two SMB levels generated by our CAN using the semantic loss.

In CMA-ES, the fitness function doesn't have to be differentiable and the playability can be computed on the output of an A* agent playing the level. Having the SL to steer the generation towards playable levels is not trivial, since it requires a differentiable definition of playability. Encoding the reachability of the right-most column with a propositional formula on the level variables $\boldsymbol{x}$ is unfeasible (consider the size of the formula resulting from unrolling and grounding a compact first-order encoding). For this reason, our CAN applies the semantic loss to a different probability distribution $\boldsymbol{\theta}$ over binary variables $\boldsymbol{r}$, with the intended meaning that r_i is true if and only if the tile x_i is reachable (in expectation) from the first column in the level $\boldsymbol{x}$, with $\boldsymbol{x}$ sampled from the generator's multinomial distribution γ over tiles. Encoding the playability of a level is straightforward in $\boldsymbol{r}$, although the problem becomes finding a fully differentiable transformation $P_R : \boldsymbol{\Gamma} \rightarrow \boldsymbol{\Theta}$.

[2]According to the game's physics.

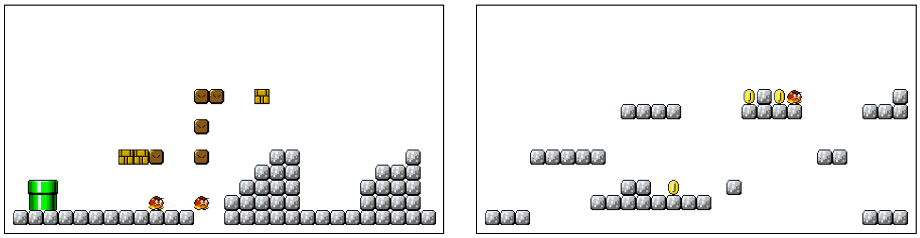

Figure 2: Illustration of two 14×28 tiles Super Mario Bros levels generated by CANs.

Network type	Mario Level	Validity	Valid samples	Uniqueness	Unique samples	Novely	Novel samples
Baseline	mario-1-3	61%	122	100%	122	100%	122
MarioGAN + CMA-ES	mario-1-3	69,5%	139	100%	139	100%	139
CAN	**mario-1-3**	**70,5%**	**141**	**100%**	**141**	**100%**	**141**
Baseline	mario-3-3	48,5%	97	100%	97	100%	97
MarioGAN + CMA-ES	mario-3-3	56,5%	113	100%	113	100%	113
CAN	**mario-3-3**	**93,50%**	**187**	**100%**	**187**	**100%**	**187**

Table 1: Results of using the semantic loss instead of post-processing with CMA-ES. Levels mario-1-3 and mario-3-3 has been chosen due to their higher complexity in being solved. The table shows the validity, uniqueness and novelty of samples after a testing session with A*. A valid sample is one that can be completely solved by the A* agent. The semantic loss has been activated from epoch 3000. Each run has lasted 5000 epochs with all the default hyper parameters defined in Volz et al. (2018)

We approximate P_R with a feedforward neural network $\hat{P_R}$. Our approach consists in pretraining $\hat{P_R}$ with synthetic data $\{(\gamma_i, \hat{\theta}_i)\}$, consisting of syntetic distributions labelled using the tile-based pathfinder provided in Summerville et al. (2016). Inputs γ_i are collected from previous runs of the generator, while $\hat{\theta}_i$ are computed by sampling 100 levels from γ_i and averaging the reachability of each tile using the pathfinder's output. The data generation process is inexpensive, allowing to train $\hat{P_R}$ to reasonable performance for the task.

We adopt the same experimental setting, WGAN architecture and training procedure of Volz et al. (2018). The structured objects are 14×28 tile-based representations of SMB levels (e.g. Fig. 2) and the training data is obtained by sliding a 28 tiles window over levels from the *Video game level corpus* (Summerville et al., 2016).

Our semantic loss, applied on the output of the approximated reachability map $\hat{P_R}$, encodes that at least 4 tiles of the right-most column are reachable. This choice is supported by two observations: if a tile is reachable, the same will probably be neighbours and the approximation given by $\hat{P_R}$ may satisfy too easily a constraint asking for only 1 reachable tile in the rightmost column.

Table 1 shows the playability, novelty and uniqueness of a batch of 200 levels generated respectively by the CAN with a forward pass (with $\lambda = 0.01$, which validation experiments showed to be a reasonable trade-off between SL and discriminator loss) and by CMA-ES using the default parameters for the search. Novelty and Uniqueness are really good because it is quite impossible to generate a level equal to an other in a space with $13^{(14\times28)}$ solutions (13 is the number of possible types for each tile). Results shows that CANs achieve comparable results on the first level, and substantial improvements on the second. Moreover, no significant negative effects can be seen on the generated material with respect to the quality of the ones generated by the original GAN.

Moreover, at the cost of pretraining $\hat{P_R}$ (1h35m on a 8-core machine with a GTX1080Ti), CANs avoid the execution of the A* agent during the generation, sampling high quality objects in milliseconds. On the other side, the training cost is slightly lower ($\sim 20\%$ less), but each run of CMA-ES, to find a single best individual, takes between 20 and 25 minutes on the previously described machine. Combining the two approaches and refining the constraints for the generation of state-of-the-art Super Mario levels is a promising research direction and is left for future work.

With these results, we can answer **Q2** affirmatively.

Reward for	Semantic loss	validity	uniqueness	diversity	QED	SA	logP
QED + SA + logP	False	97.4	2.4	91.0	47.0	84.0	65.0
	True	96.59 (2.52)	2.52 (0.25)	98.81 (2.04)	51.75 (1.55)	90.74 (5.48)	73.62 (1.14)
uniqueness	False	99.19 (0.19)	4.75 (2.35)	66.87 (6.39)	50.10 (1.68)	37.26 (9.55)	32.89 (4.55)
	True	99.27 (0.16)	17.41 (3.07)	91.28 (2.87)	41.54 (1.89)	43.99 (8.38)	33.94 (2.16)

Table 2: Results of using the semantic loss on the MolGAN architecture. The diversity score is obtained by comparing sub-structures of generated samples against a random subset of the dataset. A lower score indicates a higher amount of repetitions between the generated samples and the dataset. The first row refers to the results reported in the MolGAN paper. Experiments were run 8 times each (rows 2, 3, 4) to obtain mean and std values, the latter in parentheses.

4.3 Molecule generation

In the next experiment, we test how effective is SL in conjunction with different forms of supervision on the task of generating molecules with certain desirable chemical properties. Specifically, we consider MolGANs (De Cao & Kipf, 2018), a model that combines the adversarial loss with a *reinforcement learning objective*, used to maximize the **druglikeness**, **sythesizability** and **solubility** of the generated molecules. The reward is computed by a network that is trained to match the score provided by an external cheminformatics software. In contrast with our previous experimental settings, here the structured objects are undirected graphs of bounded maximum size, represented by discrete tensors that encode the atom/node type (**padding atom** (no atom), **Carbon**, **Nitogren**, **Oxygen**, **Fluorine**) and the bound/edge type (**padding bond** (no bond), **single**, **double**, **triple** and **aromatic** bond).

As confirmed by both the results reported by the authors of MolGANs and our previous experiments, providing an additional loss term might perturbate the already unstable adversarial game, possibly incurring in mode collapse. In the following, we explore the use of SL beyond the satisfaction of given constraints. Specifically, we observe that the formula can involve the latent variables too, and we show how this can be leveraged to increase the diversity of the generator's output and to mitigate the mode collapse. This can be achieved by using a portion of the latent vector (whose values are in the $[0, 1]$ domain) to trigger on and off SL terms that promote the presence of specific atoms in the molecule. Specifically, we apply the SL to MolGANs, making use of 5 latent dimensions to control the presence of one of the 5 types of atoms considered in the experiment. Each dimension represents the probability of having *at least one* atom of that type in the molecule, no matter the position. Assigning SL terms to specific subregion of the latent space is a general approach that can potentially be used to achieve some level of controllable constrained generation via the latent codes. We defer this research direction to future work.

In this experiment, we augment MolGAN with our constraints conditioned on the latent variables, starting from the first epoch. We consider two different variants of the reward network. In the first setting, the network implicitly rewards validity and the maximization of the three chemical properties at once: **QED** (druglikeness), **SA** (synthesizability) and **logP** (solubility). The experimental setting and evaluation metrics are identical to De Cao & Kipf (2018) except for the introduction of the SL, we thus report the same results for the baseline. In the second setting, the reward is proportional to the diversity of the generated batch, thus boosting the generator **uniqueness**. In this case, the training stops when the validity reaches 0.99.

The results, as shown in Table 2, indicate that the SL term is boosting the diversity of the generated molecules without negatively affecting the other metrics with both reward functions. This preliminary results seem to suggest that CANs can be successfully coupled with a reinforcement learning objective, answering **Q3** affermatively. In this setting, using CANs produced a negligible overhead during the training with respect to the original model, providing further evidence that the technique doesn't heavily impact on the training.

5 Related Work

Deep generative modeling has recently enjoyed substantial progress with the introduction of autoregressive models (Van Oord et al., 2016), variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014), flow-based approaches (Dinh et al., 2014), and GANs (Goodfellow et al., 2014). However, none of these approaches is designed to generate structures.

Traditional approaches to such tasks, like graphical models (Koller & Friedman, 2009; Richardson & Domingos, 2006) and probabilistic grammars (Talton et al., 2012), are ill-suited for complex tasks like image generation and do not support efficient inference under constraints. Tractable probabilistic circuits (e.g., probabilistic sentential decision diagrams (Kisa et al., 2014)) leverage knowledge compilation techniques – like CANs – and sport high capacity, efficient inference, and support for constraints, but they require the circuit even at inference time, meaning that inference can have large space requirements. In contrast, inference in CANs boils down to a forward pass over the generator and does not rely on the circuit output by knowledge compilation.

Injecting knowledge into learned models, and neural networks in particular, is a long-standing aim of machine learning. Early approaches include grounding-specifica Markov Logic Networks (Lippi & Frasconi, 2009), which combine Markov logic with neural networks. More recent models include target-specific architectures (Rocktäschel & Riedel, 2017; Donadello et al., 2017), as well as frameworks based on probabilistic logics (Manhaeve et al., 2018) and T-norms (Marra et al., 2019). Other recent approaches include also constraint learning component (Wang et al., 2019; Sourek et al., 2018). However, none of these models is at the same time generative and probabilistic.

A number of machine learning techniques were applied to Super Mario Bros level generation, e.g. LSTMs (Summerville & Mateas, 2016), probabilistic graphical models (Guzdial & Riedl, 2016) and multi-dimensional Markov Chain Monte Carlo (Snodgrass & Ontanón, 2016). Many approaches to molecule generation use VAEs (Gómez-Bombarelli et al., 2018; Kusner et al., 2017; Dai et al., 2018). Closest to MolGANs are ORGANs (Guimaraes et al., 2017), that used a different reinforcement learning objective with SeqGANs to generate molecules as sequences (in SMILE encoding).

6 Conclusion

We presented Constrained Adversarial Networks (CANs), a new class of GANs in which the generator is encouraged *during training* to output valid structures. CANs make use of the semantic loss (Xu et al., 2018) to measure the mass allocated by the generator to invalid structures and penalize the latter accordingly. As in GANs, generating (likely) valid structures then amounts to a simple forward pass of the generator. Importantly, the data structures used by the SL – which can be large if the structural constraints are very complex – can be discarded after training. Our framework was proven to be effective in different structured generative tasks, improving the validity of the generated structures (on average) while keeping the computational cost of training under control and substantially reducing inference runtimes. We also showed how the constraints to be turned on and off at inference time, suggesting different uses for the SL, namely promoting diversity of the generator's outputs. Finally, our level generation experiment shows how to transform structures to a space in which the structural constraints are easier to encode and more compact.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223, 2017.

Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799, 2008.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for molecule generation. In *Proceedings of the International Conference on Learning Representations*, 2018.

Adnan Darwiche. Sdd: A new canonical representation of propositional knowledge bases. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

I Donadello, L Serafini, and AS d'Avila Garcez. Logic tensor networks for semantic image interpretation. IJCAI, 2017.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in neural information processing systems*, pp. 5767–5777, 2017.

Matthew Guzdial and Mark Riedl. Game level generation from gameplay videos. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. 2018.

Diederik Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In *Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2014.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1945–1954. JMLR. org, 2017.

Marco Lippi and Paolo Frasconi. Prediction of protein β-residue contacts by markov logic networks with grounding-specific weights. *Bioinformatics*, 25(18):2326–2333, 2009.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, pp. 3749–3759, 2018.

Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. LYRICS: a General Interface Layer to Integrate AI and Deep Learning. *arXiv preprint arXiv:1903.07534*, 2019.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Advances in Neural Information Processing Systems*, pp. 1825–1835, 2017.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pp. 271–279, 2016.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286, 2014.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2): 107–136, 2006.

Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems*, pp. 3788–3800, 2017.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pp. 2234–2242, 2016.

Bidisha Samanta, Abir De, Niloy Ganguly, and Manuel Gomez-Rodriguez. Designing random graph models using variational autoencoders with applications to chemical design. *arXiv preprint arXiv:1802.05283*, 2018.

Sam Snodgrass and Santiago Ontanón. Controllable procedural content generation via constrained multi-dimensional markov chain sampling. In *IJCAI*, pp. 780–786, 2016.

Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezny, Steven Schockaert, and Ondrej Kuzelka. Lifted relational neural networks: Efficient learning of latent relational structures. *Journal of Artificial Intelligence Research*, 62:69–100, 2018.

Adam Summerville and Michael Mateas. Super mario as a string: Platformer level generation via lstms. *arXiv preprint arXiv:1603.00930*, 2016.

Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, 10(3):257–270, 2018.

Adam James Summerville, Sam Snodgrass, Michael Mateas, and Santiago Ontanón. The vglc: The video game level corpus. *arXiv preprint arXiv:1606.07487*, 2016.

Jerry Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah Goodman, and Radomír Měch. Learning design patterns with bayesian grammar induction. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pp. 63–74. ACM, 2012.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pp. 1747–1756, 2016.

Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M Lucas, Adam Smith, and Sebastian Risi. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 221–228. ACM, 2018.

Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pp. 6545–6554, 2019.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. A semantic loss function for deep learning with symbolic knowledge. In *International Conference on Machine Learning*, pp. 5498–5507, 2018.

Yexiang Xue and Willem-Jan van Hoeve. Embedding decision diagrams into generative adversarial networks. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 616–632. Springer, 2019.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5907–5915, 2017.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.