Constrained Image Generation Using Binarized Neural Networks with Decision Procedures

Svyatoslav Korneev[1], Nina Narodytska[2], Luca Pulina[3], Armando Tacchella[4], Nikolaj Bjorner[5], and Mooly Sagiv[26]

[1] Dept. of Energy Resources Engineering, Stanford, USA, skorneev@stanford.edu
[2] VMware Research, Palo Alto, USA, nnarodytska@vmware.com
[3] Chemistry and Pharmacy Dept., University of Sassari, Italy, lpulina@uniss.it
[4] DIBRIS, University of Genoa, Italy, armando.tacchella@unige.it
[5] Microsoft Research, USA, nbjorner@microsoft.com
[6] Tel Aviv University, msagiv@acm.org

Abstract. We consider the problem of binary image generation with given properties. This problem arises in a number of practical applications, including generation of artificial porous medium for an electrode of lithium-ion batteries, for composed materials, etc. A generated image represents a porous medium and, as such, it is subject to two sets of constraints: topological constraints on the structure and process constraints on the physical process over this structure. To perform image generation we need to define a mapping from a porous medium to its physical process parameters. For a given geometry of a porous medium, this mapping can be done by solving a partial differential equation (PDE). However, embedding a PDE solver into the search procedure is computationally expensive. We use a binarized neural network to approximate a PDE solver. This allows us to encode the entire problem as a logical formula. Our main contribution is that, for the first time, we show that this problem can be tackled using decision procedures. Our experiments show that our model is able to produce random constrained images that satisfy both topological and process constraints.

1 Introduction

We consider the problem of constrained image generation of a porous medium with given properties. Porus media occur, e.g., in lithium-ion batteries and composed materials [1,2]; the problem of generating porus media with a given set of properties is relevant in practical applications of material design [3,4,5]. Artificial porous media are useful during the manufacturing process as they allow the designer to synthesize new materials with predefined properties. For example, generated images can be used in designing a new porous medium for an electrode of lithium-ion batteries. It is well-known that ions macro-scale transport and reactions rates are sensitive to the topological properties of the porous medium of the electrode. Therefore, manufacturing the porous electrode with given properties allows improving the battery performance [1].

Images of porous media[7] are black and white images that represent an abstraction of the physical structure. Solid parts (or so called grains) are encoded as a set of connected

---

[7] Specifically, we are looking at a transitionally periodic "unit cell" of porous medium assuming that porous medium has a periodic structure [5].

black pixels; a void area is encoded a set of connected white pixels. There are two important groups of restrictions that images of a porous medium have to satisfy. The first group constitutes a set of "geometric" constraints that come from the problem domain and control the total surface area of grains. For example, an image contains two isolated solid parts. Figure 1(a) shows examples of 16x16 images from our datasets with two (the top row) and three (the bottom row) grains. The second set of restrictions comes from

Fig. 1: (a) Examples of images from train sets with two and three grains; (b) Examples of images generated by a GAN on the dataset with two grains. Examples of generated images with (c) $d \in [40, 50)$, (d) $d \in [60, 70)$, and (e) $d \in [90, 100]$.

the physical process that is defined for the corresponding porous medium. In this paper, we consider the macro-scale transportation process that can be described by a set of dispersion coefficients depending on the transportation direction. For example, we might want to generate images that have two grains such that the dispersion coefficient along the x-axis is between 0.5 and 0.6. The dispersion coefficient is defined for the given geometry of a porous medium. It can be obtained as a numerical solution of the diffusion Partial Differential Equation (PDE). We refer to these restrictions on the parameters of the physical process as process constraints.

The state of the art approach to generating synthetic images is to use generative adversarial networks (GANs) [6]. However, GANs are not able learn geometric, three-dimensional perspective, and counting constraints which is a known issue with this approach [7,8]. Our experiments with GAN-generated images also reveal this problem. There are no methods that allow embedding of declarative constraints in the image generation procedure at the moment.

In this work we show that the image generation problem can be solved using decision procedures for porous media. We show that both geometric and process constraints can be encoded as a logical formula. Geometric constraints are encoded as a set of linear constraints. To encode process constraints, we first approximate the diffusion PDE solver with a Neural Network(NN) [9,10]. We use a special class of NN, called BNN, as these networks can be encoded as logical formulas. Process constraints are encoded as restrictions on outputs of the network. This provides us with an encoding of the image generation problem as a single logical formula. The contributions of this paper can be summarized as follows: (i) We show that constrained image generation can be encoded as a logical formula and tackled using decision procedures. (ii) We experimentally investigate a GAN-based approach to constrained image generation and analyse their advantages and disadvantages compared to the constraint-based approach. (iii) We demonstrate that our constraint-based approach is capable of generating random images that have given properties, i.e., satisfy process constraints.

2   Problem description

We describe a constrained image generation problem. We denote $I \in \{0,1\}^{t \times t}$ an image that encodes a porous medium and $d \in \mathbb{Z}^m$ a vector of parameters of the physical process defined for this porous material. We use an image and a porous medium interchangeably to refer to I. We assume that there is a mapping function M that maps an image I to the corresponding parameters vector d, $\mathrm{M} : I \to \mathbb{Z}^m$. We denote as $C_g(I)$ the geometric constraints on the structure of the image I and as $C_p(d)$ the process constraints on the vector of parameters d. Given a set of geometric and process constraints and a mapping function M, we need to generate a random image I that satisfies C_g and C_p. Next we overview geometric and process constraints and discuss the mapping function.

The geometric constraints C_g define a topological structure of the image. For example, they can ensure that a given number of grains is present on an image and these grains do not overlap. Another type of constraints focuses on a single grain. They can restrict the shape of a grain, e.g., a convex grain, its size or position on the image. The third type of constraints are boundary constraints that ensure that the boundary of the image must be in a void area. Process constraints define restrictions on the vector of parameters. For example, we might want to generate images with $d_i^j \in [a_j, b_j]$, $j = 1, \ldots, m$.

Next we consider a mapping function M. A standard way to define M is by solving a system of partial differential equations. However, solving these PDEs is a computationally demanding task and, more importantly, it is not clear how to 'reverse' them to generate images with given properties. Hence, we take an alternative approach of approximating a PDE solver using a neural network [9,10]. To train such an approximation, we build a training set of pairs (I_i, d_i), $i = 1, \ldots, n$, where I_i is an input image of a porous medium and d_i, obtained by solving the PDE given I, is its label. In this work, we use a special class of deep neural networks — Binarized Neural Networks (BNN) that admit an exact encoding into a logical formula. We assume that M is represented as a BNN and is given as part of input. We will elaborate on the training procedure in Section 5.

3   The generative neural network approach

One approach to tackle the constrained image generation problem is to use generative adversarial networks (GANs) [6,11]. GANs are successfully used to produce samples of realistic images for commonly used datasets, e.g. interior design, clothes, animals, etc. A GAN can be described as a game between the image generator that produces synthetic (fake) images and a discriminator that distinguishes between fake and real images. The cost function is defined in such a way that the generator and the discriminator aim to maximize and minimize this cost function, respectively, turning the learning process into a minimax game between these two players. Each payer is usually represented as a neural network. To apply GANs to our problem, we take a set of images $\{I_1, \ldots, I_n\}$ and pass them to the GAN. These images are samples of real images for the GAN. After the training procedure is completed, the generator network produces artificial images that look like real images. The main advantage of GANs is that it is a generic approach that can be applied to any type of images and can handle complex concepts, like animals,

scenes, etc.[8] However, the main issue with this approach is that there is no way to explicitly pass declarative constraints into the training procedure. One might expect that GANs are able to learn these constraints from the set of examples. However, this is not the case at the moment, e.g., GANs cannot capture counting constraints, like four legs, two eyes, etc. [7]. Figure 1 shows examples of images that GAN produces on a dataset with two grains per image. As can be seen from these examples, GAN produces images with an arbitrary number of grains between 1 and 5 per image. In some simple cases, it is easy to filter wrong images. If we have more sophisticated constraints like convexity or size of grains, then most images will be invalid. On top of this, to take into account process constraints, we need additional restrictions on the training procedure. Overall, it is an interesting research question how to extend the GAN training procedure with physical constraints, which is beyond the scope of this paper [13]. Next we consider our approach to the image generation problem.

4   The constraint-based approach

The main idea behind our approach is to encode the image generation problem as a logical formula. To do so, we need to encode all problem constraints and the mapping between an image and its label as a set of constraints. We start with constraints that encode an approximate PDE solver. We denote $[N]$ a range of numbers from 1 to N.

4.1   Approximation of a PDE solver.

One way to approximate a diffusion PDE solver is to use a neural network [9,10]. A neural network is trained on a set of binary images I_i and their labels d_i, $i = 1, \ldots, n$. During the training procedure, the networks takes an image I_i as an input and outputs its estimate of the parameter vector $\hat{d}_i$. As we have ground truth parameters d_i for each image, we can use the mean square error or absolute value error as a cost function to perform optimization [14]. In this work, we take the same approach. However, we use a special type of networks: Binarized Neural Networks (BNN). BNN is a feedforward network where weights and activations are binary [15]. It was shown in [14,16] that BNNs allow exact encoding as logical formulas, namely, they can be encoded a set of reified linear constraints over binary variables. We use BNNs as they have a relatively simple structure and decision procedures scale to reason about small and medium size networks of this type. In theory, we can use any exact encoding to represent a more general network, e.g., MILP encodings that are used to check robustness properties of neural networks [17,18]. However, the scalability of decision procedures is the main limitation in the use of more general networks. We use the ILP encoding as in [14] with a minor modification of the last layer as we have numeric outputs instead of categorical outputs. We denote $\text{ENCBNN}(I, d)$ a logical formula that encodes BNN using reified linear constraints over Boolean variables (Section 4, ILP encoding [14]).

$$\text{ENCBNN}(I, d) = \left(\bigwedge_{k=1}^{q-1} \text{ENCBLK}_k(\mathbf{x}_k, \mathbf{x}_{k+1}) \right) \wedge \text{ENCO}(\mathbf{x}_q, d), \qquad (1)$$

---

[8] GANs exhibit well-known issues with poor convergence that we did not observe as our dataset is quite simple [12].

where $x_1 = I$ is an input of the network, q is the number of layers in the network and d is the output of the network. ENCBLK_k denotes encoding of the intermediate layer and ENCO denotes encoding of the last layer that maps the output of the qth layer (x_q) to the dispersion value d.

4.2 Geometric and process constraints.

Geometric constraints can be roughly divided into three types. The first type of constraints defines the high-level structure of the image. The high-level structure of our images is defined by the number of grains present in the image. Let w be the number of grains per image. We define a grid of size $t \times t$. Figure 2(a) shows an example of a grid of size 4×4. We refer to a cell (i, j) on the grid as a pixel as this grid encodes an image of size $t \times t$. Next we define the neighbor relation on the grid. We say that a cell (h, g) is a neighbour of (i, j) if these cells share a side. For example, $(2, 3)$ is a neighbour of $(2, 4)$ as the right side of $(2, 3)$ is shared with $(2, 4)$. Let $\text{NB}(i, j)$ be the set of neighbors of (i, j) on the gird. For example, $\text{NB}(2, 3) = \{(1, 3), (2, 2), (2, 4), (3, 3)\}$.
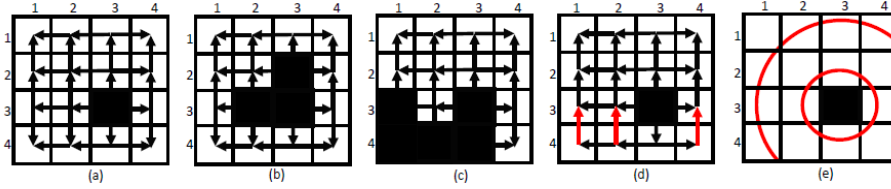

Fig. 2: Illustrative examples of additional structures used by constraint-based model.

Variables. For each cell we introduce a Boolean variable $c_{i,j,r}$, $i, j \in [t]$, $r \in [w + 1]$. $c_{i,j,r} = 1$ iff the cell (i, j) belongs to the rth grain, $r = 1, \ldots, w$. Similarly, $c_{i,j,w+1} = 1$ iff the cell (i, j) represents a void area.

Each cell is either a black or white pixel. We enforce that each cell contains either a grain or a void area.

$$\sum_{r=1}^{w+1} c_{i,j,r} = 1 \qquad j, i \in [t] \tag{2}$$

Grains do not overlap. Two cells that belong to different grains cannot be neighbours.

$$c_{i,j,r} \rightarrow \neg c_{h,g,r'} \qquad (h, g) \in \text{NB}(i, j), r' \in [w] \setminus \{r\} \tag{3}$$

Grains are connected areas. We enforce connectivity constraints for each grain. By connectivity we mean that there is a path between two cells of the same grain using only cells that belong to this grain. Unfortunately, enforcing connectivity constraints is very expensive. Encoding the path constraint results in a prohibitively large encoding. To deal with this explosion, we restrict the space of possible grain shapes. First, we assume that we know the position of one pixel of this grain that we pick randomly. Let $s_r = (i, j)$ be a random cell, $r \in [w]$. Then we implicitly build a directed acyclic graph (DAG) G starting from this cell s_r that covers the entire grid. Each cell of a grid is a node in this graph. The node that corresponds to the cell s_r does not have incoming arcs. There are multiple ways to build a G from s_r. Figure 2(a) and (d) show two possible ways to build

a G that covers a grid starting from cell $(3, 3)$. We enforce that cell i, j belongs to the rth grain if its center s_r is equal to (i, j).

$$(c_{i,j,r}), \qquad s_r = (i, j), r \in [w + 1]. \tag{4}$$

Next we define a parent relation in G. Let $\text{PR}_G(i, j)$ be the set of parents of cell (i, j) in G. For example, $\text{PR}_G(2, 2) = \{(2, 3), (3, 2)\}$ in our example on Figure 2(a). Given a DAG G, we can easily enforce connectivity relation w.r.t. G. The following constraint ensures that a cell (i, j) cannot belong to the rth grain if none of its parents in G belongs to the same grain.

$$\left(\wedge_{(h,g) \in \text{PR}_G(i,j)} \neg c_{h,g,r} \right) \rightarrow \neg c_{i,j,r}, \qquad j, i \in [t], r \in [w + 1]. \tag{5}$$

Note that by enforcing connectivity constraints on the void area, we make sure that grains do not contain isolated void areas inside them.

Given a DAG G, we can generate grains of multiple shapes. For example, Figure 2(b) shows one possible grain. However, we also lose some valid shapes that are ruled out by the choice of graph G. For example, Figure 2(c) gives an example of a shape that is not possible to build using G in Figure 2(a). Note that there is no path from the center $(3, 3)$ of G to the cell $(3, 1)$ that does not visit $(3, 2)$. However, if we select a different random DAG G', e.g., Figure 2(d), then this shape is one of the possible shapes for G'. In general, we can pick s_r and DAG randomly, so we generate a variety of shapes.

Compactness of a grain. The second set of constraints is about restrictions on a single grain. The compactness constraint is a form of convexity constraint. We want to ensure that any two boundary points of a grain are close to each other. The reason for this constraint is that grains are unlikely to have a long snake-like appearance as solid particles tend to group together. Sometimes, we need to enforce the convexity constraint, which is an extreme case of compactness. To enforce this constraint, we again trade-off the variety of shapes and the size of the encoding. Now we assume that s_r is the center of the grain. Then we build virtual circles around this center that cover the grid. Figure 2(e) shows examples of such circles. Let $C_r(i, j) = \{C_r^1, \ldots, C_r^q\}$ be a set of circles that are built with the cell s_r as a center. The following constraint enforces that a cell that belongs to the circle C_r^v can be in the rth grain only if all cells from the inner circle C_r^{v-s} belong to the rth grain, where s is a parameter.

$$\vee_{c_{h,g,r} \in C_r^{v-s}} \neg c_{h,g,r} \rightarrow \neg c_{i,j,r} \qquad c_{i,j,r} \in C_r^v, v \in [q], r \in [w] \tag{6}$$

Note that if $s = 1$ then we generate convex grains. In this case, every pixel from C_r^v has to belong to the rth grain before we can add a pixel from the circle C_r^{v+1} to this grain.

Boundary constraints. We also have a technical constraint that all cells on the boundary of the grid must be void pixels. They are required to define boundary conditions for PDEs on generated images.

$$(c_{i,j,w+1}) \qquad j = t \vee i = t \tag{7}$$

Connecting with BNN. We need to connect variables $c_{i,j,r}$ with the inputs of the network. We recall that an input image is a black and white image, where black pixels correspond to solid parts. Hence, if a cell belongs to a grain, i.e. $c_{i,j,r}$ is true and $r \neq w + 1$, then it maps to a black pixel. Otherwise, it maps to a white pixel.

$$
\begin{aligned}
c_{i,j,r} \rightarrow I_{i,j} = 1 \qquad & j, i \in [t], r \in [w], \\
c_{i,j,w+1} \rightarrow I_{i,j} = 0 \qquad & j, i \in [t].
\end{aligned}
\tag{8}
$$

Process constraints. Process constraints are enforced on the output of the network. Given ranges $[a_i, b_i]$, $i \in [m]$ we have:

$$
a_i \leq d_i \leq b_i \qquad i \in [m]
\tag{9}
$$

Summary. To solve the constrained random image generation problem, we solve the conjunctions of constraint (1)–(9). Randomness comes from the random seed that is passed to the solver, a random choice of s_r and G.

5 Experiments

We conduct a set of experiments with our constraint based approach. We ran our experiments on Intel(R) Xeon(R) 3.30GHz. We use the timeout of 600 sec in all runs.

Training procedure. For the training sets, we consider the synthetic random images of the unit cell of a periodic porous medium. For each generated image, we solve the partial differential equation to define the rate of the transport process, d. We use synthetic images because high-quality images of the natural porous medium are not available. The usage of synthetic images is a standard practice when data supply is strictly limited. We generated two datasets, D_2 with 10K images and D_3 with 5K images. Each image in D_2 contains two grains and each image in D_3 contains three grains. All images are black and white images of size 16 by 16. These images were labeled with dispersion coefficients along the x-axis which is a number between 0.4 and 1. We performed quantization on the dispersion coefficient value to map d into an interval of integers between 40 and 100. Intuitively, the larger the volume of the solid grains in the domain the lower the dispersion value, since the grain creates an obstacle for the transport. In the datasets, we don't use images with large enough grains to have the dispersion rate lower than 0.4 (or re-scaled to 40 for BNN), since the shape diversity of large grains is low. We use mean absolute error (MAE) to train BNN. BNN consists of three blocks with 100 neurons per inner layers and one output. The MAE is 4.2 for D_2 and 5.1 for D_3. We lose accuracy compared to non-binarized networks, e.g, MAE for the same non-binarized network is 2.5 for D_2. However, BNNs are much easier to reason about, so we work with this subclass of networks.

Image generation. We use CPLEX and the SMT solver Z3 to solve instances produced by constraints (1)–(9). In principle, other solvers could be evaluated on these instances. The best mode for Z3 was to use an SMT core based on CDCL and a theory solver for *nested* Pseudo-Boolean and cardinality constraints. We noted that bit-blasting into sorting circuits did not scale, and Z3's theory of linear integer arithmetic was also inadequate. We considered six process constraints for d, namely, $d \in [a, b]$,

$[a, b] \in \{[40, 50), \dots, [90, 100]\}$. For each interval $[a, b]$, we generate 100 random constrained problems. The randomization comes from a random seed that is passed to the solver, the position of centers of each grain and the parameter s in the constraint (6). We used the same DAG G construction as in Figure 2(a) in all problems.

Solver	D_2						D_3					
	[40,50)	[50,60)	[60,70)	[70,80)	[80,90)	[90,100]	[40,50)	[50,60)	[60,70)	[70,80)	[80,90)	[90,100]
CPLEX	100	99	99	98	100	41	100	100	96	99	100	84
Z3	98	89	81	74	56	12	100	97	97	97	96	54

Table 1: The number of solved instances in each interval $[a, b]$.

Table 1 shows summary of our results for CPLEX and Z3 solvers. As can be seen from this table, these instances are relatively easy for the CPLEX solver. It can solve most of them within the given timeout. The average time for D_2 is 25s and for D_3 is 12s with CPLEX. Z3 handles most benchmarks, but we observed it gets stuck on examples that are very easy for CPLEX, e.g. the interval $[80, 90)$ for D_2. We hypothesize that this is due to how watch literals are tracked in a very general way on nested cardinality constraints (Z3 maintains a predicate for each nested PB constraint and refreshes the watch list whenever the predicate changes assignment), when one could instead exploit the limited way that CPLEX allows conditional constraints. The average time for the dataset D_2 is 94s and for the dataset D_3 is 64s with Z3.

Figures 1(c)–(e) show examples of generated images for ranges $[40, 50)$, $[60, 70)$ and $[90, 100]$ for D_2 (the top row) and D_3 (the bottom row). For the process we consider, as the value of the dispersion coefficient grows, the black area should decrease as there should be fewer grain obstacles for a flow to go through the porous medium. Indeed, images in Figures 1(c)–(e) follow this pattern, i.e. the black area on images with $d \in [40, 50)$ is significantly larger than on images with $d \in [90, 100]$. Moreover, by construction, they satisfy geometric constraints that GANs cannot handle. For each im-

Fig. 3: The absolute error between d and its true value.

age we generated, we run a PDE solver to compute the true value of the dispersion coefficient on this image. Then we compute the absolute error between the value of d that our model computes and the true value of the coefficient. Figure 3 shows absolute errors for all benchmarks that were solved by CPLEX. We ordered solved benchmarks by dispersion coefficient values, breaking ties arbitrarily. First, this figure shows that our model generates images with given properties. The mean absolute error is about 10 on these instances. Taking into account that BNN has MAE of 4.2 on D_2, MAE of 10 on new generated instances is a reasonable result. Ideally, we would like MAE to be zero. However, this error depends purely on the BNN we used. To reduce this error, we need to improve the accuracy of BNN as it serves as an approximator of a PDE solver. For example, we can use more binarized layers or use additional non-binarized layers. Of
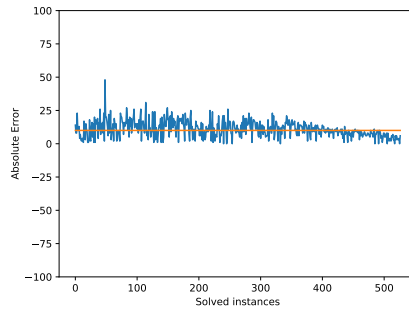
course, increasing the power of the network leads to computational challenges solving the corresponding logical formulas.

Limitation of the approach. The main limitation of the presented approach is its scalability. In our experiments, we used small images that represent a unit cell of a periodic porous medium. Conceptually, our approach can handle larger images, but scalability of the underlying decision procedures becomes the main bottleneck. Unfortunately, scalability is the main limiting factor in all decision based approaches to analysis of NNs at the moment [17,19,14]. In the future, we are hoping to develop efficient domain specific decision procedures for NN analysis. The second limitation is the set of topological constraints that we can handle. Many real applications require sophisticated restrictions on the topology of solid and void areas and do not exhibit periodic structure. It is an interesting research direction to formalize these constraints and use decision procedures to generate rich topologies.

6 Related work

There are two lines of work related. The first line of work uses constraint to enhance machine learning techniques with declarative constraints, e.g. in solving constrained clustering problems and in data mining techniques that handle domain specific constraints [20,21,22]. One recent example is the work of Ganji *et al.* [21] who proposed a logical model for constrained community detection. The second line of research explores embedding of domain-specific constraints in the GAN training procedure [13,23,24,8,25]. Work in this area is targeting various applications in physics and medicine that impose constraints, like sparsity constraints, high dynamic range requirements (e.g. when pixel intensity in an image varies by orders of magnitude), location specificity constraints (e.g. shifting pixel locations can change important image properties), etc. However, this research area is emerging and the results are still preliminary.

7 Conclusion

In this paper we considered the constrained image generation problem for a physical process. We showed that this problem can be encoded as a logical formula over Boolean variables. For small porous media, we show that the generation process is computationally feasible for modern decision procedures.There are a lot of interesting future research directions. First, the main limitation of our approach is scalability, as we cannot use large networks with a number of weights in the order of hundreds of thousands, as it is required by industrial applications. However, constraints that are used to encode, for example, binarized neural networks are mostly pseudo-Boolean constraints with unary coefficients. Hence, it would be interesting to design specialized procedures to deal with this fragment of constraints. Second, we need to investigate different types of neural networks that admit encoding into SMT or ILP. For instance, there is a lot of work on quantized networks that use a small number of bits to encode each weight, e.g. [26]. Finally, can we use similar techniques to reveal vulnerabilities in neural networks? For example, we might be able to generate constrained adversarial examples or other special types of images that expose undesired network behaviour.

References

1. Arunachalam, H., Korneev, S., Battiato, I., Onori, S.: Multiscale modeling approach to determine effective lithium-ion transport properties. In: 2017 American Control Conference (ACC). (May 2017) 92–97
2. Battiato, I., Tartakovsky, D.: Applicability regimes for macroscopic models of reactive transport in porous media. **120-121** (03 2011) 18–26
3. Hermann, H., Elsner, A.: Geometric models for isotropic random porous media: A review. **2014** (04 2014)
4. Pyrcz, M., Deutsch, C.: Geostatistical Reservoir Modeling. (2014)
5. Hornung, U., ed.: Homogenization and Porous Media. Springer-Verlag New York, Inc., New York, NY, USA (1997)
6. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., eds.: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada. (2014) 2672–2680
7. Goodfellow, I.J.: NIPS 2016 tutorial: Generative adversarial networks. CoRR **abs/1701.00160** (2017)
8. Osokin, A., Chessel, A., Salas, R.E.C., Vaggi, F.: Gans for biological image synthesis. In: 2017 IEEE International Conference on Computer Vision (ICCV). (Oct 2017) 2252–2261
9. Korneev, S.: Using convolutional neural network to calculate effective properties of porous electrode. https://sccs.stanford.edu/events/sccs-winter-seminar-dr-slava-korneev
10. Harikesh Arunachalam, Svyatoslav Korneev, S.O., Battiato, I.: Using convolutional neural network to calculate effective properties of porous electrode. In: Journal of The Electrochemical Society(in preparation to submit). (February 2018)
11. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. CoRR **abs/1511.06434** (2015)
12. Chintala, S.: How to train a GAN? tips and tricks to make GANs work. https://github.com/soumith/ganhacks
13. Luke de Oliveira, M.P., Nachman, B.: Tips and tricks for training gans with physics constraints. In: Workshop at the 31st Conference on Neural Information Processing Systems (NIPS), Deep Learning for Physical Sciences. (December 2017)
14. Narodytska, N., Kasiviswanathan, S.P., Ryzhyk, L., Sagiv, M., Walsh, T.: Verifying properties of binarized deep neural networks. CoRR **abs/1709.06662** (2017)
15. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized Neural Networks. In Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R., eds.: Advances in Neural Information Processing Systems 29. Curran Associates, Inc. (2016) 4107–4115
16. Cheng, C., Nührenberg, G., Ruess, H.: Verification of binarized neural networks. CoRR **abs/1710.03107** (2017)
17. Katz, G., Barrett, C., Dill, D., Julian, K., Kochenderfer, M.: Reluplex: An efficient smt solver for verifying deep neural networks. arXiv preprint arXiv:1702.01135 (2017)
18. Cheng, C., Nührenberg, G., Ruess, H.: Maximum resilience of artificial neural networks. In: Automated Technology for Verification and Analysis. Volume 10482. (2017) 251–268
19. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety Verification of Deep Neural Networks. In: CAV. Volume 10426 of Lecture Notes in Computer Science., Springer (2017) 3–29
20. Dao, T., Duong, K., Vrain, C.: Constrained clustering by constraint programming. Artif. Intell. **244** (2017) 70–94

21. Ganji, M., Bailey, J., Stuckey, P.J.: A declarative approach to constrained community detection. In Beck, J.C., ed.: Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings. Volume 10416 of Lecture Notes in Computer Science., Springer (2017) 477–494

22. Guns, T., Dries, A., Nijssen, S., Tack, G., Raedt, L.D.: Miningzinc: A declarative framework for constraint-based mining. Artif. Intell. **244** (2017) 6–29

23. Luke de Oliveira, M.P., Nachman, B.: Generative adversarial networks for simulation. In: 18th International Workshop on Advanced Computing and Analysis Techniques in Physics Research. (August 2017)

24. Hu, Y., Gibson, E., Lee, L., Xie, W., Barratt, D.C., Vercauteren, T., Noble, J.A.: Freehand ultrasound image simulation with spatially-conditioned generative adversarial networks. In: Molecular Imaging, Reconstruction and Analysis of Moving Body Organs, and Stroke Imaging and Treatment,CMMI 2017, RAMBO 2017, SWITCH 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, 2017, Proceedings. Volume 10555 of Lecture Notes in Computer Science., Springer (2017) 105–115

25. Ravanbakhsh, S., Lanusse, F., Mandelbaum, R., Schneider, J.G., Póczos, B.: Enabling dark energy science with deep generative models of galaxy images. In Singh, S.P., Markovitch, S., eds.: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA., AAAI Press (2017) 1488–1494

26. Deng, L., Jiao, P., Pei, J., Wu, Z., Li, G.: Gated XNOR networks: Deep neural networks with ternary weights and activations under a unified discretization framework. CoRR **abs/1705.09283** (2017)