A GAN based solver of black-box inverse problems

Michael Gillhofer **Hubert Ramsauer** **Johannes Brandstetter** **Bernhard Schäfl**

Sepp Hochreiter

LIT AI Lab
Institute for Machine Learning
Johannes Kepler University Linz, Austria
`{gillhofer, ramsauer, brandstetter, schaefl, hochreit}@ml.jku.at`

Abstract

We propose a GAN based approach to solve inverse problems which have non-differentiable or even black-box forward relations. The idea is to find solutions via an adversarial game where the generator has to propose new samples and the discriminator has to assess the quality of the samples with respect to the forward relation f. However, instead of attempting to approximate f directly, the discriminator only has to solve a binary classification task in local regions populated by the generated samples. We demonstrate the efficacy of our approach by applying it to an artificially generated topology optimization problem. We show that our method leads to similar results like more traditional topology optimization methods.

1 Introduction

Inverse problems is a long established field of research with applications in science and engineering [11]. Their essence is calculating the causal factors $\boldsymbol{x}$ that produce a set of observations via a forward relation f. Inverse problems frequently appear in the fields of physics [2] or medicine [3]. However, they are typically ill-posed in the sense that the solution does not depend continuously on the data. Problems get even more severe when a non-differentiable or even non-continuous forward relation is involved.

We propose a novel method to solve inverse problems by applying core principles of Generative Adversarial Networks (GANs) [5]. Related work comprises the MetricGAN [4], which aims at generating data with improved metric scores, and Generative Adversarial Self-Imitation Learning [6], which encourages the agent in reinforcement learning tasks to imitate past good trajectories via generative adversarial imitation learning framework. In our work, the discriminator D is used as a continuous proxy of f. The two main aspects are that (i) instead of trying to approximate f directly, we reformulate the task such that D is a binary classifier, and (ii) the generator G is used to produce samples in a local region of the input domain $\mathcal{X}$ of the forward relation. The generator can be seen as a spotlight which illuminates parts of $\mathcal{X}$ and thereby enables the discriminator to locally learn a continuous approximate of f. The usage of a generator allows to encode prior knowledge of the problem and thus helps to restrict the search space in $\mathcal{X}$ and avoids non-plausible or infeasible solutions.

One major advantage of our proposed method is, that we find meaningful update directions in the sample space $\mathcal{X}$ without having access to a derivative of f. Our method is therefore capable of performing derivative-free optimization [1] as well as black-box optimization [9]. Furthermore, it does not directly depend on the response of f and thus is less prone to rapid changes or discontinuities of f which makes the method especially suited for the ill-posedness of inverse problems.

2 Problem characterization

In the following, we define the class of inverse problems our method intends to solve. We consider an arbitrarily complicated (black-box) forward relation $f : \mathcal{X} \to \mathbb{R}$. The only two restrictions we need to impose on f are the following: i) given some $\boldsymbol{x} \in \mathcal{X}$ it is computationally feasible to retrieve the value $f(\boldsymbol{x})$, and ii) a ranking on the input domain $\mathcal{X}$ is imposed by f. More precisely, we demand a *weak ordering* of the input domain w.r.t f. The inverse problem at hand is the task to find a non-empty subset of samples $\mathcal{S} \subseteq \mathcal{X}$ that lead to a specific response $c \in \mathbb{R}$: $\boldsymbol{s} \in S \implies f(\boldsymbol{s}) = c$, where for the rest of the paper we make the arbitrary assumption that a lower c is better. In practice, it is often sufficient to find a solution which meets a certain quality threshold. We can exploit this line of thought and relax the problem by not specifying an exact response value c, but rather an upper bound u. This way, we end up with the following problem formulation: given an upper bound $u \in \mathbb{R}$ and a forward relation $f : \mathcal{X} \to \mathbb{R}$ that suffices restrictions i) and ii), we want to find a non-empty set

$$\mathcal{S} \subseteq \{\boldsymbol{x} \in \mathcal{X} \mid f(\boldsymbol{x}) \leq u\} \tag{1}$$

of samples that lead to a response equal to or lower than u. We call any set of the above form a solution to our inverse problem. Note that we do not explicitly state constraints that our solutions must admit to, but rather assume that all samples from $\mathcal{X}$ already fulfill this constraints.

3 How GANs solve complex non-linear inverse problems

To solve the inverse problem, the GAN approach needs to be modified. In the original GAN setting the target distribution $\mathbb{P}_r$ is fixed and can be accessed through the training data set. However, this is not the case for the inverse problem. We do not know any sample from the set $\{\boldsymbol{x} \in \mathcal{X} \mid f(\boldsymbol{x}) \leq u\}$ and we do not require to learn the whole distribution of samples with a score lower than u. In principle, one solution is sufficient, and therefore our generator is not required to produce a diverse set of samples.

The initially generated samples are unlikely to overlap with the set of possible solutions. Therefore, it has to be ensured that during training the generator is pushed into regions of $\mathcal{X}$ with lower response. We note that for an update only local information can be used, i.e. we can only use the current subspace that is populated by the generated samples. Further we make the assumption that even if only local information is used, generalization beyond the current co-domain of the generator might be achieved through an inductive bias on $G_{\boldsymbol{\theta}}$ and $D_{\boldsymbol{w}}$, where θ and $\boldsymbol{w}$ are the weights of the generator and the discriminator, respectively.

Our approach is to train $D_{\boldsymbol{w}}$ to classify the generator's co-domain $\mathcal{G}$ into a region $\mathcal{G}^0$ with higher responses and a region $\mathcal{G}^1$ with lower responses. By imposing a weak ordering on $\mathcal{G}$ via f and choosing $p \in (0, 1)$, we split the co-domain into two sets, where $\mathcal{G}^1$ contains the best $((1-p) \cdot 100)\%$ of samples with lowest response and $\mathcal{G}^0$ contains the rest. The task of the generator is to produce more samples with lower response. We can track the progress of the generator by following the responses of the best samples $\mathcal{G}^1$. More precisely, we track the value of the highest (i.e. worst) response τ of our best samples:

$$\tau = \sup_{\boldsymbol{x} \in \mathcal{G}^1} f(\boldsymbol{x}), \tag{2}$$

$$f(\tilde{\boldsymbol{x}}) \geq \; \tau \geq f(\boldsymbol{x}) \,, \quad \forall \tilde{\boldsymbol{x}} \in \mathcal{G}^0 \,, \, \forall \boldsymbol{x} \in \mathcal{G}^1 \,, \tag{3}$$

where $((1-p) \cdot 100)\%$ of the highest scoring samples lie in $\mathcal{G}^1$. A set of solutions is found if this value has arrived at the upper bound: $\tau \leq u$.

The task for $D_{\boldsymbol{w}}$ is to discriminate between samples of the two sets. This is a classification task where the samples $\boldsymbol{x} \in \mathcal{G}$ are tagged with labels:

$$y(\boldsymbol{x}) = \mathbb{I}[\boldsymbol{x} \in \mathcal{G}^1] \,, \tag{4}$$

where $\mathbb{I}[\cdot]$ is the indicator function. We minimize the discriminator loss

$$L_{\boldsymbol{w}} = -\mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\boldsymbol{\theta}}}[\, y(\boldsymbol{x}) \ \log(D_{\boldsymbol{w}}(\boldsymbol{x})) + (1 - y(\boldsymbol{x})) \ \log(1 - D_{\boldsymbol{w}}(\boldsymbol{x})) \,] \tag{5}$$

where $\mathbb{P}_{\boldsymbol{\theta}}$ is the output distribution of the generator. Now let's assume that for a fixed update step the discriminator is able to perfectly classify the co-domain of the generator. As discussed before, we assume that even though the discriminator was trained only locally, it has learned features that reflect the structure of the problem. Its decision boundary therefore should, to some extent, be meaningful even in regions outside of $\mathcal{G}$, i.e. we assume that samples from one side of the decision boundary are prone to have higher responses than samples from the other side. The task of the generator is to move its mass into the direction of samples with lower response. For that we can leverage the discriminator, which has learned the decision boundary between samples with higher and lower response than τ. Only for generated samples with a response higher than the threshold, the weights of the generator $\boldsymbol{\theta}$ need to be adjusted. We minimize the generator loss:

$$L_{\hat{\boldsymbol{\theta}}} = -\mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\, (1 - y(G_{\boldsymbol{\theta}}(\boldsymbol{z}))) \log(1 - D_{\boldsymbol{w}}(G_{\hat{\boldsymbol{\theta}}}(\boldsymbol{z}))) \,] \ , \tag{6}$$

where we ignore the labeling function when taking the derivative

$$\frac{\partial L_{\hat{\boldsymbol{\theta}}}}{\partial \hat{\boldsymbol{\theta}}}\Big|_{\hat{\boldsymbol{\theta}}=\boldsymbol{\theta}} = -\frac{\partial}{\partial\, \hat{\boldsymbol{\theta}}}\Big|_{\hat{\boldsymbol{\theta}}=\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\, (1 - y(G_{\boldsymbol{\theta}}(\boldsymbol{z}))) \ \log(1 - D_{\boldsymbol{w}}(G_{\hat{\boldsymbol{\theta}}}(\boldsymbol{z}))) \,] \tag{7}$$

$$= -\mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\, (1 - y(G_{\boldsymbol{\theta}}(\boldsymbol{z}))) \ \frac{\partial}{\partial\, \hat{\boldsymbol{\theta}}}\Big|_{\hat{\boldsymbol{\theta}}=\boldsymbol{\theta}} \log(1 - D_{\boldsymbol{w}}(G_{\hat{\boldsymbol{\theta}}}(\boldsymbol{z}))) \,] \ . \tag{8}$$

If we update the generator accordingly, its output distribution $\mathbb{P}_{\boldsymbol{\theta}}$ will move into a region of samples with lower score and thus will also lead to a decrease of τ.

In practice, initially we draw samples $\hat{\mathcal{G}} = \{G_{\boldsymbol{\theta}}(\boldsymbol{z}_i)\}_{i=1}^{n}$ from the generator's co-domain. We split $\hat{\mathcal{G}}$ according to the percentile p with respect to the responses $\{f(G_{\boldsymbol{\theta}}(\boldsymbol{z}_i))\}_{i=1}^{n}$ into the sets $\hat{\mathcal{G}}^0$ and $\hat{\mathcal{G}}^1$, we set our threshold to $\hat{\tau} = \max_{\boldsymbol{x} \in \hat{\mathcal{G}}^1} f(\boldsymbol{x})$, and we label the samples according to $y_i = \mathbb{I}[\hat{\tau} \geq f(G_{\boldsymbol{\theta}}(\boldsymbol{z}_i))]$. However, instead of drawing a completely new set from the generator after every update, we use $\hat{\mathcal{G}}$ as a buffer, where we only replace samples if they have a score lower than one of the samples in the buffer. While this decouples the set $\hat{\mathcal{G}}$ to some extent from $\mathbb{P}_{\boldsymbol{\theta}}$, the hope is to gain a stronger pull towards smaller responses.

4 Application: Topology optimization

In the field of Computer Aided Design (CAD), the goal often is to find the stiffness or its inverse, the compliance $c \in \mathbb{R}$, of a given design $\boldsymbol{x}$. This design is exposed to external forces $\mathbf{F}$ and is fixed at one or more anchor points where the load-dependent displacement is fixed to zero. Here, our forward relation is the mapping

$$f_{\text{FEM}} : \ [0,1]^{\dim(\boldsymbol{x})} \to \mathbb{R} \tag{9}$$

$$\boldsymbol{x} \mapsto f_{\text{FEM}}(\boldsymbol{x}) \ , \tag{10}$$

where $c = f_{\text{FEM}}(\boldsymbol{x})$ is the compliance of sample $\boldsymbol{x}$. Such class of problems is often solved by the Finite Elements Method (FEM) [12]. In topology optimization, one usually wants to find a design $\boldsymbol{x}$ whose quality in terms of compliance is minimized [10].

We use a CNN-based [8] generator consisting of 5 convolutional layers to propose designs $\boldsymbol{x}$. The forward relation $f_{\text{FEM}}(\boldsymbol{x}) = \mathbf{U}^T\mathbf{K}\mathbf{U}$ assigns a value c to generated designs $\boldsymbol{x}$ and thus introduces a weak ordering to the set of proposed designs. The global displacement vector $\mathbf{U}$ is found by solving the finite element equation system $\mathbf{K}\mathbf{U} = \mathbf{F}$, where the sparse global stiffness matrix $\mathbf{K}$ is a function of $\boldsymbol{x}$, and $\mathbf{F}$ describes the external forces acting on the system. Anchor points are treated as additional constraints on $\mathbf{U}$. To enable local exploration we added random noise on the boundaries of the generated designs. This is achieved by masking the noise with edge-detector convolutions applied to the design $\boldsymbol{x}$.

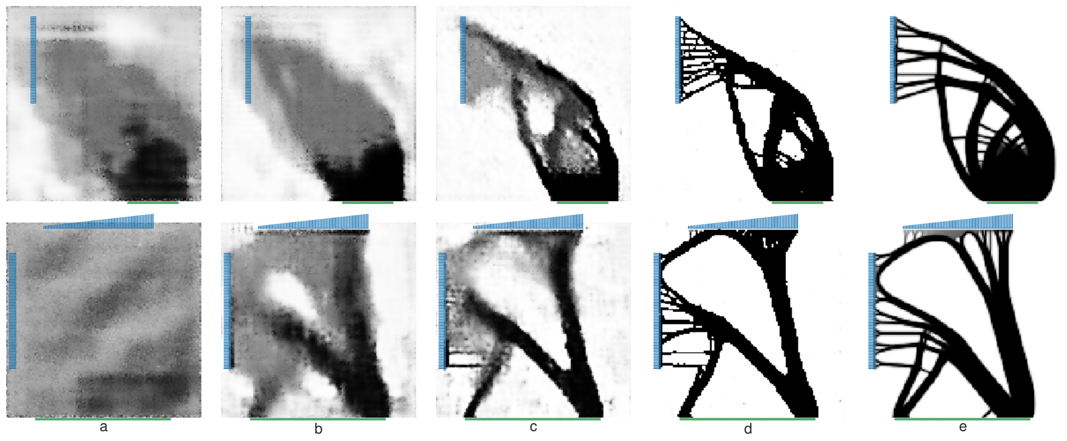

Figure 1: Topology optimization: The two rows represent two different problem settings. Each pixel corresponds to a node in the FEM. The darkness of a node corresponds to the used material. Possible anchor points are drawn in green and distributed loads are drawn in blue. The thickness of the bar corresponds to the acting force on the corresponding node. In columns a to d different stages of the GAN based optimization are shown progressing from earlier to later stages in the optimization. Column e shows the result of an established topology optimization method [7].

5 Results

We evaluate our topology optimization method on artificially designed problems on a 128×128 FEM grid and compare it to standard topology optimization software [7] on the same problem with identical model hyper-parameters. The problem setup consisted of (i) ranges where the designed part can be anchored and (ii) areas with distributed loads. The task is to design a structure which minimizes the compliance while using only 25% of the available design-space. This constraint is enforced by rescaling the output of the generator. While standard software performs smoothing of solutions we omitted this step. Training our GAN based solver on the topology optimization task took about 5 hours on a single GPU. Two results of the design process are shown in Fig. 1.

Although our GAN based optimization approach does not have access to a derivative or any other information of the model f_{FEM}, it delivers similar structured parts than a standard optimization software [7]. The structured parts have on par to slightly worse quality scores depending on the problem. Our system is capable of coming up with a variety of solutions which are local minima of the artificially designed problems by changing the initial parameters of generator and discriminator. One could change the initial solution for traditional methods too. However, in practice they often converge to a single minimum solution regardless of initialization.

6 Discussion

We propose a GAN based method to solve non-differentiable or even black-box inverse problems. The optimization process is reformulated as a game between a generator which proposes samples, and a discriminator which groups the proposed samples into two classes, namely samples with low response and samples with high response. Our GAN based approach solves inverse problems without relying on the derivative of the forward relation and is insensitive to rapid changes or discontinuities of the forward relation. We show the efficacy of our approach by applying it to a topology optimization problem. We observe a generator which's designs gradually become better.

Our goal is to further leverage the GAN setting and produce a distribution of solutions which yield similar quality scores. Besides the variety in solutions future work comprises a profound mathematical treatment of convergence criteria as well as stability analyses. Further we want to apply the method on several problems where we expect it to perform well in high dimensions since GANs are particularly good at approximating high-dimensional distributions.

References

[1] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*. Vol. 8. Siam, 2009.

[2] Ian JD Craig and John C Brown. “Inverse problems in astronomy: a guide to inversion strategies for remotely sensed data”. In: *Research supported by SERC. Bristol, England and Boston, MA, Adam Hilger, Ltd., 1986, 159 p.* (1986).

[3] Jeffrey A Fessler. “Model-based image reconstruction for MRI”. In: *IEEE Signal Processing Magazine* 27.4 (2010), pp. 81–89.

[4] Szu-Wei Fu et al. “MetricGAN: Generative Adversarial Networks based Black-box Metric Scores Optimization for Speech Enhancement”. In: *CoRR* abs/1905.04874 (2019). arXiv: `1905.04874`. URL: `http://arxiv.org/abs/1905.04874`.

[5] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[6] Yijie Guo et al. “Generative Adversarial Self-Imitation Learning”. In: *CoRR* abs/1812.00950 (2018). arXiv: `1812.00950`. URL: `http://arxiv.org/abs/1812.00950`.

[7] A.J.J. Lagerweij. *topopt*. `https://github.com/AJJLagerweij/topopt`. 2019.

[8] Yann LeCun, Yoshua Bengio, et al. “Convolutional networks for images, speech, and time series”. In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[9] Songqing Shan and G Gary Wang. “Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions”. In: *Structural and Multidisciplinary Optimization* 41.2 (2010), pp. 219–241.

[10] O. Sigmund. “A 99 line topology optimization code written in Matlab”. en. In: *Structural and Multidisciplinary Optimization* 21.2 (Apr. 2001), pp. 120–127. ISSN: 1615-1488. DOI: `10.1007/s001580050176`. URL: `https://doi.org/10.1007/s001580050176` (visited on 08/21/2019).

[11] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation*. Vol. 89. siam, 2005.

[12] Olgierd Cecil Zienkiewicz et al. *The finite element method*. Vol. 3. McGraw-hill London, 1977.