NOVAS: NON-CONVEX OPTIMIZATION VIA ADAPTIVE STOCHASTIC SEARCH FOR END-TO-END LEARNING AND CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work we propose the use of adaptive stochastic search as a building block for general, non-convex optimization operations within deep neural network architectures. Specifically, for an objective function located at some layer in the network and parameterized by some network parameters, we employ adaptive stochastic search to perform optimization over its output. This operation is differentiable and does not obstruct the passing of gradients during backpropagation, thus enabling us to incorporate it as a component in end-to-end learning. We study the proposed optimization module's properties and benchmark it against two existing alternatives on a synthetic energy-based structured prediction task, and further showcase its use in stochastic optimal control applications.

1 INTRODUCTION

Deep learning has experienced a drastic increase in the diversity of neural network architectures, both in terms of proposed structure, as well as in the repertoire of operations that define the interdependencies of its elements. With respect to the latter, a significant amount of attention has been devoted to incorporating *optimization blocks or modules* operating at some part of the network. This has been motivated by large number of applications, including meta-learning (Finn et al., 2017; Rusu et al., 2018; Bartunov et al., 2019), differentiable physics simulators (de Avila Belbute-Peres et al., 2018), classification (Amos et al., 2019), GANs (Metz et al., 2016), reinforcement learning with constraints, latent spaces, or safety (Amos & Kolter, 2017; Srinivas et al., 2018; Amos & Yarats, 2019; Cheng et al., 2019; Pereira et al., 2020), model predictive control (Amos et al., 2018; Pereira et al., 2018), as well as tasks relying on the use of energy networks (Belanger et al., 2017; Bartunov et al., 2019), among many others. Local[1] optimization modules lead to *nested optimization* operations, as they interact with the global, end-to-end training of the network that contains them. Consider some component within the neural network architecture, e.g. a single layer, whose input and output are $x_i \in \mathbb{R}^n$ and $x_{i+1} \in \mathbb{R}^m$, respectively. Within that layer, the input and output are linked via the solution of the following optimization problem:

$$x_{i+1} = \arg\min_{x} F(x; x_i, \theta), \tag{1}$$

that is, the output x_{i+1} is defined as the solution to an optimization problem for which the input x_i remains temporarily fixed, i.e., acts as a parameter. Here, $F(x; x_i, \theta) : \mathbb{R}^m \times \mathbb{R}^n \times \Theta \to \mathbb{R}$ is a function possibly further parameterized by some subset of the neural network parameters $\theta \in \Theta$. Note that x here is an independent variable which is free to vary. The result of this optimization could potentially also be subject to a set of (input-dependent) constraints, though in this paper we will consider only unconstrained optimization. It is also important to note that, depending on the problem, F can be a given function, or it can itself be represented by a multi-layer neural network (trained by the outer loop), in which case the aforementioned optimization layer consists of multiple sub-layers and is more accurately described as a *module* rather than a single layer. Examples of this type of optimization are structured prediction energy networks (e.g. Belanger et al. (2017)); another such example is Amos & Kolter (2017) which treats the case of convex $F(\cdot; x_i, \theta)$.

[1]To distinguish between the optimization of the entire network as opposed to that of the optimization module, we frequently refer to the former as global or *outer-loop* optimization and to the latter as local or *inner-loop* optimization.

In order to facilitate end-to-end learning over the entire network, the gradient of its loss function $\mathcal{L}$ with respect to θ will require during backpropagation passing the gradient of the module's output x^{i+1} with respect to parameters θ and x_i. Depending on the nature of the optimization problem under consideration, several procedures have been suggested; among them, particularly appealing is the case of convex optimization (Gould et al., 2016; Johnson et al., 2016; Amos et al., 2017; Amos & Kolter, 2017), in which the aforementioned gradients can be computed efficiently through an application of the implicit function theorem to a set of optimality conditions, such as the KKT conditions. In the case of non-convex functions however, obtaining such gradients is not as straight-forward; solutions involve either forming and solving a locally convex approximation of the problem, or unrolling gradient descent (Domke, 2012; Metz et al., 2016; Belanger et al., 2017; Finn et al., 2017; Srinivas et al., 2018; Rusu et al., 2018; Foerster et al., 2018; Amos et al., 2018). Unrolling gradient descent approximates the $\arg\min$ operator with a fixed number of gradient descent iterations during the forward pass and interprets these as an unrolled compute graph that can be differentiated through during the backward pass. One drawback in using this unrolled gradient descent operation however is the fact that doing so can lead to over-fitting to the selected gradient descent hyper-parameters, such as learning rate and number of iterations. Recently, Amos & Yarats (2019) demonstrated promising results in alleviating this phenomenon by replacing these iterations of gradient descent by iterations of sampling-based optimization, in particular a differentiable approximation of the cross-entropy method. While still unrolling the graph created by the fixed number of iterations, they showed empirically that no over-fitting to the hyper-parameters occurred by performing inference on the trained network with altered inner-loop optimization hyper-parameters. Another significant bottleneck in all methods involving graph unrolling is the number of iterations, which has to be kept low to prevent a prohibitively large graph during backprop, to avoid issues in training.

Note that in eq. (1) the variable of optimization is free to vary independently of the network. This is in contrast to many applications involving nested optimization, mainly in the field of meta-learning, in which the inner loop, rather than optimizing a free variable, performs *adaptation* to an initial value which is supplied to the inner loop by the outer part of the network. For example, MAML (Finn et al., 2017) performs the inner-loop adaptation $\theta \to \theta'$, in which the starting point θ is not arbitrary (as x is in eq. (1)) but is supplied by the network. Thus, in the context of *adaptation*, unrolling the inner-loop graph during back-prop is generally necessary to trace the adaptation back to the particular network-supplied initial value. Two notable exceptions are first-order MAML (Finn et al., 2017; Nichol et al., 2018), which ignores second derivative terms, and implicit MAML (Rajeswaran et al., 2019), which relies on local curvature estimation.

In this paper we propose Non-convex Optimization Via Adaptive Stochastic Search (NOVAS), a module for differentiable, non-convex optimization. The backbone of this module is adaptive stochastic search (Zhou & Hu, 2014), a sampling-based method within the field of stochastic optimization. The contributions of our work are as follows: (A). We demonstrate that the NOVAS module does not over-fit to optimization hyper-parameters and offers improved speed and convergence rate over its alternative (Amos & Yarats, 2019). (B). If the inner-loop variable of optimization is free to vary (i.e., the problem fits the definition given by eq. (1)), *we show that there is no need to unroll the graph during the back-propagation of gradients*. The latter advantage is critical, as it drastically reduces the size of the overall end-to-end computation graph, thus facilitating improved ability to learn with higher convergence rates, improved speed, and reduced memory requirements. Furthermore, it allows us to use a higher number of inner-loop iterations. (C). If the inner-loop represents an adaptation to a network-supplied value as it is the case in meta-learning applications, NOVAS may still be used in lieu of the gradient descent rule (though unrolling the graph may be necessary here). Testing NOVAS in such a setting is left for future work. (D). We combine the NOVAS module with the framework of *deep FBSDEs*, a neural network-based approach to solving nonlinear partial differential equations (PDEs). This combination allows us to solve Hamilton-Jacobi-Bellman (HJB) PDEs of the most general form, i.e., those in which the $\min$ operator does not have a closed-form solution, a class of problems that was previously impossible to address due to the non-convexity of the corresponding Hamiltonian. We validate the algorithm on a cart-pole task and demonstrate its scalability on a 101-dimensional continuous-time portfolio selection problem.

2 FURTHER BACKGROUND AND RELATED WORK

Relation to Differentiable Cross-Entropy: Particular importance should be given to Amos & Yarats (2019), since, to the best of our knowledge, it is the first to suggest sampling-based optimization instead of gradient descent, and features some similarities with our approach. The authors therein propose a differentiable approximation of the cross-entropy method (CEM) (Rubinstein, 2001; De Boer et al., 2005), called differentiable cross-entropy (DCEM). To obtain this approximation, they need to approximate CEM's eliteness threshold operation, which is non-differentiable. This is done by solving an additional, convex optimization problem separately for each inner loop step (and separately for each sample of x_i in the batch, resulting in a total of $N \times M \times K$ *additional convex optimization problems*, with N: batch size, M: number of inner loop iterations, K: number of outer loop iterations, i.e. training epochs). After CEM has been locally approximated by DCEM, they replace the usual inner-loop gradient descent steps with DCEM steps, and the entire inner-loop optimization graph is unrolled during the backward pass. Our method differs from this approach in the following ways: 1. we employ the already differentiable adaptive stochastic search algorithm, thus not having to solve any additional optimization problem to obtain a differentiable approximation (speed improvement), while also showing some convergence rate improvements, and most importantly 2. In the case of inner-loop optimization over an independent variable (e.g., such as the problem defined by eq. (1)), we *do not unroll the optimization graph, but instead pass the gradients only through the last inner-loop iteration*. This drastically reduces its size during backpropagation, increasing speed, reducing memory requirements, and facilitating easier learning.

Sampling-based Optimization: Adaptive stochastic search (Zhou & Hu, 2014) is a sampling-based method within stochastic optimization that transforms the original optimization problem via a probabilistic approximation. The core concept behind this algorithm is approximating the gradient of the objective function by evaluating random perturbations around some nominal value of the independent variable, a concept that also appears under the name Stochastic Variational Optimization and shares many similarities with natural evolution strategies (Bird et al., 2018). Another comparable approach is CEM (Rubinstein, 2001; De Boer et al., 2005). In contrast to adaptive stochastic search, CEM is non-differentiable (due to the eliteness threshold) and the parameters are typically updated *de novo* in each iteration, rather than as a gradient descent update to the parameter values of the previous iteration. In the case of Gaussian distributions, the difference between CEM and adaptive stochastic search boils down to the following: in adaptive stochastic search, the mean gets updated by calculating the average of all sampled variable values *weighted* by a typically exponential mapping of their corresponding objective function values, whereas in CEM only the top-k performing values are used, and are weighted equally. Furthermore, this difference can be made even smaller if one replaces the exponential mapping in the former method with a differentiable (sigmoid) function that approximates the eliteness operation. More details are available in the Appendix.

Deep Learning Approaches on PDEs and FBSDEs: There has been a recent surge in research and literature in applying deep learning to approximate solutions of high-dimensional PDEs. The transition from a PDE formulation to a trainable neural network is done via the concept of a system of Forward-Backward Stochastic Differential Equations (FBSDEs). Specifically, certain PDE solutions are linked to solutions of FBSDEs. Systems of FBSDEs can be interpreted as a stochastic equivalent to a two-point boundary value problem, and can be solved using a suitably defined deep neural network architecture. This is known in the literature as the *deep FBSDE* approach (Han et al., 2018; Raissi, 2018). While applied in high-dimensional PDEs, the aforementioned results have seen very limited applicability in the field of optimal control. Indeed, the HJB PDE in control theory has a much more complicated structure, and in its general form involves a $\min$ operator applied on its Hamiltonian term over the control input. Exploiting certain structures of system dynamics and cost functions that allowed for a closed-form expression for this operator, Exarchos & Theodorou (2018); Exarchos et al. (2018; 2019) developed a framework for control using FBSDEs, which was then translated to a deep neural network setting in Pereira et al. (2019); Wang et al. (2019). *In this work, we incorporate the NOVAS module inside deep FBSDE neural network architectures to account for PDEs lacking a closed-form expression for their* $\min$ *and/or* $\max$ *operators.* Thus, we are able to address the most general description of a HJB PDE in which the corresponding Hamiltonian is non-convex. More information concerning the deep FBSDE framework can be found in the Appendix.

3 NON-CONVEX OPTIMIZATION VIA ADAPTIVE STOCHASTIC SEARCH

The cornerstone of our approach is a method within stochastic optimization called *adaptive stochastic search* (Zhou & Hu, 2014). Adaptive stochastic search addresses the general maximization[2] problem

$$x^* \in \arg\max_{x \in \mathcal{X}} F(x), \qquad \mathcal{X} \subseteq \mathbb{R}^n, \tag{2}$$

with $\mathcal{X}$ being non-empty and compact, and $F : \mathcal{X} \to \mathbb{R}$ a real-valued, non-convex, potentially discontinuous and non-differentiable function. Instead of dealing with this function that lacks desirable properties such as smoothness, adaptive stochastic search proposes the solution of a stochastic approximation of this problem in which x is drawn from a selected probability distribution $f(x; \rho)$ of the exponential family with parameters ρ and solve

$$\rho^* = \arg\max_{\rho} \int F(x) f(x; \rho) \mathrm{d}x = \mathbb{E}_\rho \left[F(x) \right].$$

This new objective function, due to its probabilistic nature, exhibits desirable properties for optimization. Algorithmically, this can be facilitated by introducing a natural log and a shape function $S(\cdot) : \mathbb{R} \to \mathbb{R}^+$ which is continuous, non-decreasing, and with a non-negative lower bound (an example of such a function would be the exponential). Due to their properties, passing $F(x)$ through $S(\cdot)$ and the log does not affect the optimal solution. The final optimization problem is then

$$\rho^* = \arg\max_{\rho} \ln \int S(F(x)) f(x; \rho) \mathrm{d}x = \ln \mathbb{E}_\rho \left[S(F(x)) \right]. \tag{3}$$

To address this optimization problem, one can sample candidate solutions x from $f(x; \rho)$ in the solution space $\mathcal{X}$, and then use a gradient ascent method on eq. (3) to update the parameter ρ. Depending on the chosen probability distribution for sampling x, a closed-form solution for the gradient of the above objective function with respect to ρ is available. Thus, while still being a sampling-based method at its core, adaptive stochastic search employs gradient ascent on a *probabilistic mapping* of the initial objective function. While any probability density function of the exponential family will work, in our work we sample x from a Gaussian distribution. The resulting update scheme is shown in Alg. 1. More details concerning its derivation are included in the Appendix.

As described in the introduction, the standard approach employed in the literature for non-convex inner-loop optimization is to apply an optimization procedure (either gradient descent or DCEM) for a fixed number of iterations during the forward pass and interpret these as an unrolled compute graph that can be differentiated through during the backward pass. In this work we argue that this needs to be done only in cases of *adaptation* (e.g., in meta-learning): the variable to be adapted is supplied to the inner-loop by the outer part of the network, and is adapted using a rule such as gradient descent with respect to the inner-loop objective function for a fixed number of steps. Crucially, the process is not initialized with an arbitrary initial value for the adapted variable but with the one that is supplied by the network; the backward pass which needs to pass through the optimization module also needs to flow through the input of the variable of optimization in the module. Thus, the variable's values pre- and post-adaptation need to be linked via the unrolled computational graph of the adaptation. In contrast, in the case in which the variable of optimization is not an input to the layer, but rather is allowed to vary freely and independently of the outer part of the network (e.g. as described by problem (1)), such a process is not only unnecessary, but further leads to complications such as reduced network trainability and learning speed, as well as increased memory usage and computation time. Given that in this case the variable of optimization is initialized by what is typically no more than a random guess, there is no need to trace the gradients during backpropagation all the way back to that random guess.

After fixing a number of iterations for optimization, say, n, a simple way to implement NOVAS in a non-unrolled fashion is to take $n - 1$ of the iterations off the graph and perform only the n-th on graph. This amounts essentially to getting a good initial guess solution and performing a single step optimization. The latter is enough to supply gradient information during back-prop as it relates to the relevant, optimized value x^*, rather than the intermediate steps forming the trajectory from its

[2]While presented for maximization, we deploy it for minimization by switching the sign of the objective function.

Algorithm 1: Non-convex Optimization Via Adaptive Stochastic Search (NOVAS)

Given:
Neural network architecture containing NOVAS module, mini-batch dataset $\{X_j,\, Y_j\}_{j=1}^{J}$, NOVAS objective function $F(\cdot; x_i, \theta)$.
Parameters:
Neural network parameters: number of layers L, layer transformations f_i, network parameters θ.
NOVAS parameters: initial mean and standard deviation μ_0, σ_0, learning rate α, shape function S, number of samples M, number of iterations N, small positive number $\varepsilon = 10^{-3}$ (sampling variance lower bound).
Set $x_1 \leftarrow \{X_j\}$
for $i = 1$ **to** L **do**
  if f_i is NOVAS layer **then**
    Set $\mu \leftarrow \mu_0$, $\sigma \leftarrow \sigma_0$
    for $n = 1$ **to** $N - 1$ (***off-graph operations***) **do**
      $(\mu,\, \sigma) \leftarrow$ **NOVAS_Layer** $(\mu,\, \sigma,\, \alpha,\, S,\, M,\, N,\, \epsilon,\, F)$
    end for
    $(\mu,\, \sigma) \leftarrow$ **NOVAS_Layer** $(\mu,\, \sigma,\, \alpha,\, S,\, M,\, N,\, \epsilon,\, F)$
    $x_{i+1} = \mu$
  else
    $x_{i+1} = f_i(x_i; \theta)$
  end if
  $\{\hat{Y}_j\} = x_{L+1}$
  Compute Loss: $\mathcal{L} = \text{MSE}(Y_j,\, \hat{Y}_j)$
  Update Parameters: $\theta \leftarrow \text{Adam}(\mathcal{L},\, \theta)$
end for

$(\mu, \sigma) \leftarrow$ **NOVAS_Layer** $(\mu,\, \sigma,\, \alpha,\, S,\, M,\, N,\, \epsilon,\, F)$

Generate M samples of $x^m \sim \mathcal{N}(\mu, \sigma^2)$, $\quad m = 1, \ldots, M$;
for $m = 1$ **to** M ***(vectorized operation)*** **do**
  Evaluate $F^m = F(x^m)$ for maximization or $F^m = -F(x^m)$ for minimization;
  Subtract baseline $F^m = F^m - \min_m(F^m)$;
  Apply shape function $S^m = S(F^m)$;
  Normalize $S^m = S^m / \sum_{m=1}^{M} S^m$;
end for
Update $\mu = \mu + \alpha \sum_{m=1}^{M} S^m (x^m - \mu)$, $\quad \sigma = \text{sqrt}(\sum_{m=1}^{M} S^m (x^m - \mu)^2 + \varepsilon)$;

initial guess to its final optimal value. From a coding perspective, most automatic differentiation packages allow for localized deactivation of gradient information flow; in PyTorch (Paszke et al., 2019), this is as simple as adding a "`with torch.no_grad():`" for the $n - 1$ first iterations. With regards to the proposed Alg. 1, we would like to mention following: A. The forward pass of the given neural network containing the NOVAS module is a nested operation of layer transformations f_i given by $Y_j = f_L(f_{L-1}(f_{L-2}(\ldots\, f_1(X_j)\, \ldots)))$, wherein f_i can be either a **NOVAS_layer**, or standard linear or non-linear neural network layers. Also, note that the above transformation applies to recurrent layers with the forward pass unrolled wherein each f_i corresponds to a time step. B. For clarity of presentation, we outline the forward and backward (i.e. backprop) passes for a single sampled mini-batch from the training dataset. The user is free to choose a training strategy that best suits the problem and apply the proposed algorithm to every sampled mini-batch with any variant of stochastic gradient descent for the outer loop. C. The "subtract baseline" operation in **NOVAS_layer** is optional, but may lead to some numerical improvements. Essentially, rather than comparing objective function absolute values, their improvement over the worst one is compared, which may lead to a better numerical value range when passing through the shape function (which is typically an exponential). Further algorithm implementation details are given in the Appendix. Finally, an important remark is necessary concerning the differentiability of the module. Nonconvex optimization problems do not necessarily have a unique optimum, and as a result, the $\arg\min$ operator is a set-valued map and not differentiable in the classical sense; even when the optimum is unique almost everywhere in the parameter space it is possible for the optimal value to be discon-

tinuous in the parameter. For non-global optimizers, an initialization that avoids wrong local optima is crucial. With respect to the latter, there is some evidence (see, e.g., Bharadhwaj et al. (2020)) that sampling-based optimization methods are more robust compared to gradient descent as they evaluate the objective function over an extended area of the input space and, given enough sampling variability for exploration, have thus more chances of escaping a narrow local optimum.

4 APPLICATIONS

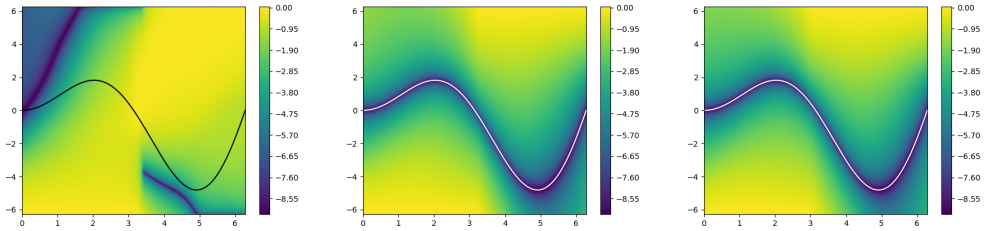
In this section we explore the properties of NOVAS and test its applicability in a few problems involving end-to-end learning. The first task is a Structured Prediction Energy Network (SPEN) learning task, which we adopted directly from Amos & Yarats (2019). We found it to be an ideal environment to test NOVAS against unrolled DCEM and unrolled gradient descent because it is simple, allows for fast training, and, being two-dimensional, one can visualize the results. We would like to stress that this is merely an example for illustrating various algorithm differences and behavior rather than a claim on state-of-the-art results in the domain of SPENs. We then address two optimal control problems by incorporating the NOVAS module in a deep FBSDE neural network architecture as shown in Fig. 3: the first problem is the cart-pole swing-up task, a low-dimensional problem that has been successfully addressed with already existing deep FBSDE approaches (Pereira et al., 2019) that exploit the structure of dynamics and cost (dynamics are affine in control and the cost is quadratic in the control) in order to perform minimization of the Hamiltonian explicitly. This problem merely serves as a means to validate the NOVAS-FBSDE algorithm. The second problem demonstrates the establishment of a new state-of-the-art in solving high-dimensional HJB PDEs using the deep FBSDE method; specifically, we address a continuous-time portfolio optimization problem that leads to a general (i.e., without an explicit solution for the $\min$ operator) HJB PDE in 101 dimensions. This HJB PDE form could not be addressed by deep FBSDE methods previously.

4.1 STRUCTURED PREDICTION ENERGY NETWORKS

The goal in energy-based learning is to estimate a conditional probability $\mathbb{P}(y|x)$ of an output $y \in \mathcal{Y}$ given an input $x \in \mathcal{X}$ using a parameterized energy function $E(x, y; \theta) : \mathcal{X} \times \mathcal{Y} \times \Theta \to \mathbb{R}$, wherein $\theta \in \Theta$ are the energy function's trainable parameters. The conditional probability is approximated as $\mathbb{P}(y|x) \propto \exp(-E(y; x, \theta))$. Predictions can be made by minimizing the trained energy function with respect to y:

$$\hat{y} = \arg\min_{y} E(x, y; \theta). \tag{4}$$

Initially studied in the context of linear energy functions (Taskar et al., 2005; LeCun et al., 2006), the field recently adopted deep neural networks called structured prediction energy networks (SPENs) (Belanger & McCallum, 2016) to increase the complexity of learned energy functions. In particular, Belanger et al. (2017) suggested training SPENs in a supervised manner; unrolled gradient descent is used to obtain $\hat{y}$ which then compared to the ground-truth y^* by a suitable loss function. Mimicking the unrolled gradient descent suggested by Belanger et al. (2017), Amos & Yarats (2019) replaced the gradient descent operations with differentiable cross entropy iterations, also using an unrolled computation graph during backpropagation. Here we adopt the same example as in Amos & Yarats (2019) to benchmark NOVAS against unrolled gradient descent and unrolled dcem, compare their properties, and further show that unrolling the inner-loop graph is not necessary. We consider the simple regression task where ground-truth data are generated from $f(x) = x\sin(x)$, $x \in [0, 2\pi]$, and we use a neural network of 4 hidden layers to approximate E. This problem belongs to the class of problems described by eq. (1) in the introduction: the variable of optimization, y, is not an input to the optimization module from the exterior part of the network. The energy function represented by the multi-layer neural network defines the objective function *within* the module (that is, it corresponds to $F(\cdot; x_i, \theta)$ of eq. (1)). While in this case the entire network is within the module, the input could instead be features extracted from an anterior part of the network, e.g. through convolution. The results are shown in Figs. 1 and 2. As can be seen from Fig. 2(a), unrolled gradient descent converges to a very low loss value (which implies good regression performance), but the trained energy function does not reflect the ground-truth relationship, Fig. 1(a). This implies a "faulty" inner-loop optimization, which the energy network itself learns to compensate for. The result resembles more ordinary regression than energy-based structured prediction, since no useful structure is learned; furthermore, changing the inner-loop optimization parameters during inference



(a) (b) (c)

Figure 1: **Learned energy functions:** plots depict the log of the normalized energy function trained using 10 inner loop iterations of (a) unrolled gradient descent (reproduced from Amos & Yarats (2019)), (b) NOVAS with the entire inner-loop graph unrolled, and (c) NOVAS without unrolling (only the last iteration on-graph). Black/white curve denotes the ground truth solution $x\sin(x)$.

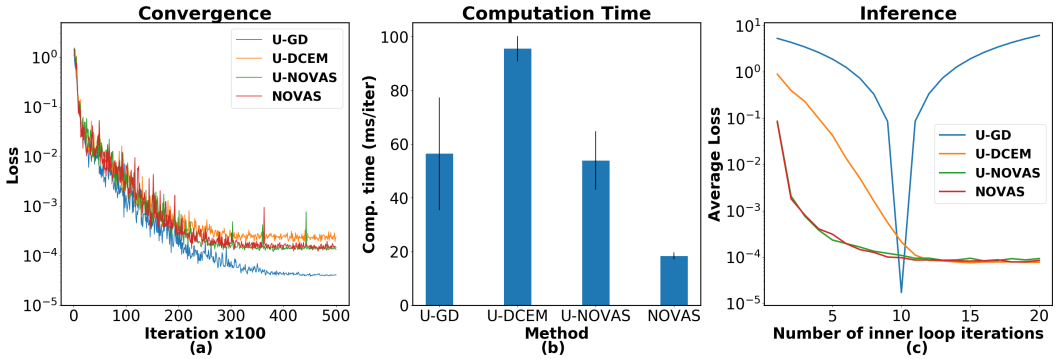


Figure 2: **Convergence analysis and computation time:** (a). convergence (test set loss) and (b). computation time for unrolled gradient descent (U-GD), unrolled DCEM (U-DCEM), NOVAS in unrolled mode (U-NOVAS) and without unrolling (NOVAS). In terms of speed, NOVAS offers an approximate speedup of 5x compared to U-DCEM, and if unrolled is on-par with U-GD. **(c). Inference on trained models with altered inner-loop parameters:** All models were trained using 10 inner loop iterations. Unrolled GD causes the energy network to over-fit to that number (as noted in Amos & Yarats (2019)). Unrolled DCEM does not suffer from this phenomenon, but optimization seems to be less efficient (more inner-loop iterations are required for the same loss as NOVAS). NOVAS, in both its regular and unrolled form does not over-fit, and is the most efficient.

(after training) leads to an operation that the energy network has not learned to compensate for, as seen in Fig. 2(c). The sampling-based methods of unrolled DCEM and NOVAS both alleviate this phenomenon by learning the correct energy landscape. Furthermore, as seen in Fig. 1(b) and (c), unrolling the graph is unnecessary, and avoiding it leads to a significant speed-up, 5x with respect to unrolled DCEM, Fig. 2(b). Interestingly, an additional benefit is that NOVAS seems to offer a greater inner-loop convergence rate than DCEM (Fig. 2(c)). Due to the simplicity of this example, there is no learning inhibition when using an unrolled graph, as seen from the comparison between NOVAS and unrolled NOVAS. However, this can be the case in more complex tasks and network architectures, as we shall see in Section 4.2. Further details are given in the Appendix.

4.2 Control using FBSDEs

4.2.1 Cart-pole swing-up task

We first validate the NOVAS-FBSDE algorithm (neural network architecture seen in Fig. 3) by solving a task whose special structure (dynamics affine in control and cost function quadratic in control) allows for a closed-form solution for the $\min$ operator of the Hamiltonian. Because of this special structure, this task can be solved by already existing deep FBSDE approaches (e.g., Pereira et al. (2019)). Here, we replace the minimization explicit solution with NOVAS. The results are shown in Fig. 4, and are in accordance with results obtained using explicit minimization (see Fig. 6 in Pereira et al. (2019)). Equations and implementation details are given in the Appendix.

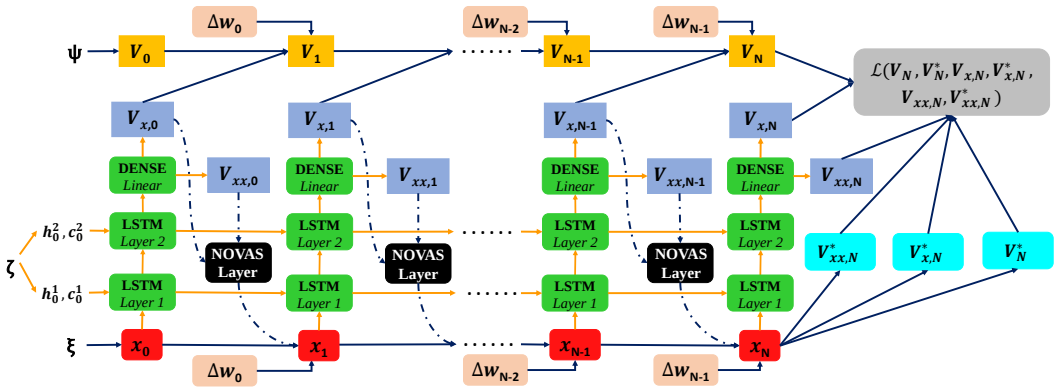


Figure 3: **NOVAS-FBSDE neural network architecture:** the NOVAS module is incorporated in the neural network to minimize the Hamiltonian over the control at each time step.

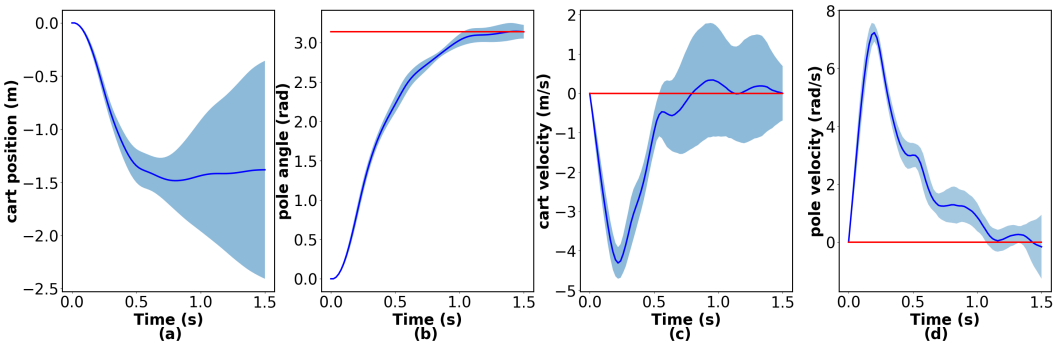


Figure 4: **Cartpole:** Validation of the NOVAS-FBSDE algorithm on a task that allows explicit minimization of the Hamiltonian with respect to the control. Replacing the explicit minimization with NOVAS leads to the same solution (compare with Fig. 6 in Pereira et al. (2019)). (a) cart position, (b) pole angle, (c) cart velocity (d) pole angular velocity. Blue line denotes mean trajectory, shaded regions show controller's response to injected noise. Target states are indicated with red. Plots above show statistics over 128 test trials.

4.2.2 HIGH-DIMENSIONAL, CONTINUOUS-TIME PORTFOLIO SELECTION

We now demonstrate that augmenting the deep FBSDE method with NOVAS allows us to solve general, high-dimensional HJB PDEs by employing NOVAS-FBSDE on a stock portfolio optimization problem, defined as follows: we consider a market index I that consists of $N = 100$ stocks, and select a subset of $M = 20$ of those for trading. There is also a risk-less asset with (relatively low) return rate. We may invest an initial wealth capital W among these $20 + 1$ assets, and the goal is to control the percentage allocation among these assets *over time* such that the wealth outperforms the market index *in probability*. This optimal control formulation leads to a HJB PDE on a state space of $100 + 1$ dimensions (100 stocks of the market plus the wealth process, which incorporates the risk-less asset dynamics. Volatility clearly dominates in such short-term horizons, so a successful trading strategy would be one that increases the odds of beating the market average compared to a random selection. *Index-tracking* and *wealth-maximization* have long been the subject of study from a controls perspective (Primbs, 2007; Bae et al., 2020), though with limited results due to the difficulty in dealing with such high uncertainties. Primbs (2007) investigates a low-dimensional variant of this problem (5 stocks, 3 traded) but does not enforce the constraint that allocation can only be positive (thus leading to negative investments, i.e. borrowing money from stocks), and *tracks* the index (thus paying a penalty also when the portfolio outperforms the index) due to the approach being restricted to consider only quadratic cost functions. We avoid these and enforce positive investments only by applying a `softmax` on the control input, and use $\left(\texttt{softplus}(I - W)\right)^2$ as cost function to incentivize outperforming rather than tracking the index. We consider two alternative investment strategies as baselines: a constant and equal allocation among all traded assets, as well

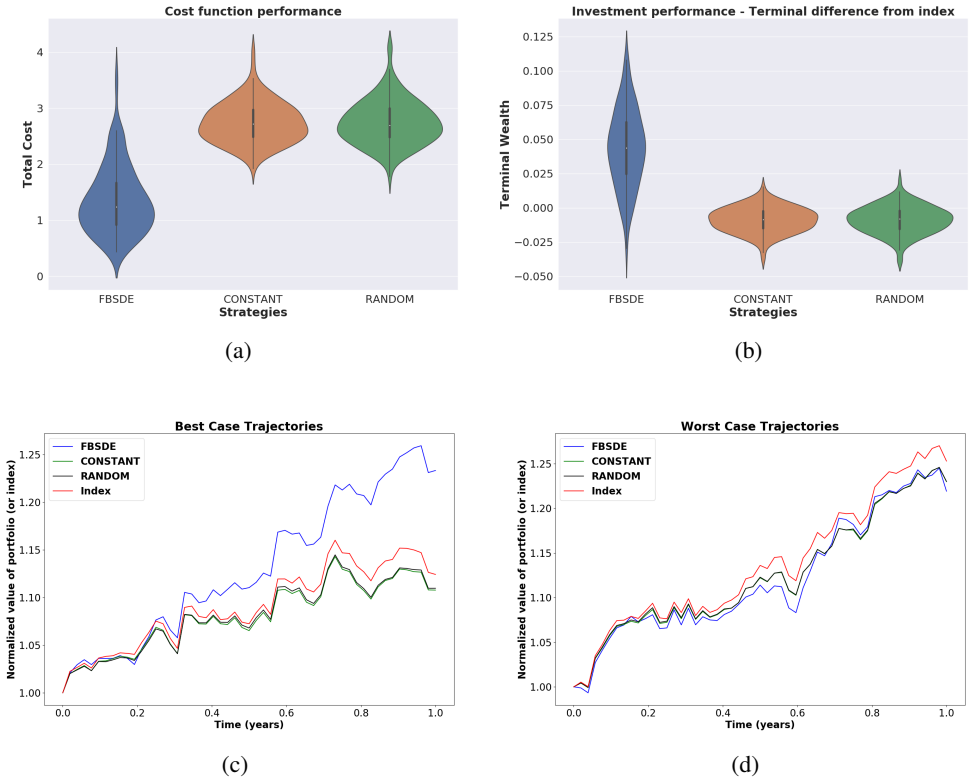


Figure 5: **Portfolio Optimization:** (a) Violin plots for the comparison of trading strategies in terms of the defined loss. The FBSDE controller outperforms in probability both strategies of equal and random allocation among traded stocks. (b) Wealth - market index difference ($W - I$) at the end of one year for the three investment strategies. (c) Trajectory corresponding to the lowest cost (best case). (d) Trajectory corresponding to the highest cost (worst case).

as random allocations. The results are shown in Fig. 5. As indicated by the violin plots, the FBSDE investment strategy outperforms in probability the two alternative strategies, and is the only one who outperforms the market index (by almost 5% on average) at the end of the planning horizon of one year. These statistics are obtained by running 128 market realizations with *test set* volatility profiles (noise profiles not seen during training or validation). All equations and implementation details are provided in the Appendix. *We note that solving this problem in an unrolled graph setting was not possible*: either because it was impossible to facilitate learning when a high number of inner-loop iterations was applied (presumably due to the excessive total graph depth), or because of memory issues. Thus, eliminating the unrolled graph is absolutely critical in this case.

5 Conclusions

In this paper we presented NOVAS, an optimization module for differentiable, non-convex inner-loop optimization that can be incorporated in end-to-end learning architectures involving nested optimizations. We demonstrated its advantages over alternative algorithms such as unrolled gradient decent and DCEM on a SPEN benchmark. We also showed that NOVAS allows us to expand the class of PDEs that can be addressed by the deep FBSDE method while resisting the curse of dimensionality, as demonstrated by the solution of a 101-dimensional HJB PDE associated with a portfolio optimization problem. NOVAS is a *general purpose* differentiable non-convex optimization approach and thus, owing to its broad description and its generality, could be useful in a plethora of other applications involving nested optimization operations. We hope that the results of this work will inspire continued investigation.

REFERENCES

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 136–145. JMLR. org, 2017.

Brandon Amos and Denis Yarats. The differentiable cross-entropy method. *arXiv preprint arXiv:1909.12830*, 2019.

Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 146–155. JMLR. org, 2017.

Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable MPC for end-to-end planning and control. In *Advances in Neural Information Processing Systems*, pp. 8289–8300, 2018.

Brandon Amos, Vladlen Koltun, and J Zico Kolter. The limited multi-label projection layer. *arXiv preprint arXiv:1906.08707*, 2019.

Se Yung Bae, Junkee Jeon, and Hyeng Keun Koo. Continuous-time portfolio selection: A cursory survey. *Frontiers in Applied Mathematics and Statistics*, 6:4, 2020. ISSN 2297-4687. doi: 10.3389/fams.2020.00004. URL https://www.frontiersin.org/article/10.3389/fams.2020.00004.

Sergey Bartunov, Jack W Rae, Simon Osindero, and Timothy P Lillicrap. Meta-learning deep energy-based memory models. *arXiv preprint arXiv:1910.02720*, 2019.

David Belanger and Andrew McCallum. Structured prediction energy networks. In *International Conference on Machine Learning*, pp. 983–992, 2016.

David Belanger, Bishan Yang, and Andrew McCallum. End-to-end learning for structured prediction energy networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 429–439. JMLR. org, 2017.

Homanga Bharadhwaj, Kevin Xie, and Florian Shkurti. Model-predictive control via cross-entropy and gradient-based optimization. *arXiv preprint arXiv:2004.08763*, 2020.

Thomas Bird, Julius Kunze, and David Barber. Stochastic variational optimization. *arXiv preprint arXiv:1809.04855*, 2018.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3387–3395, 2019.

Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter. End-to-end differentiable physics for learning and control. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 7178–7189. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7948-end-to-end-differentiable-physics-for-learning-and-control.pdf.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

Justin Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pp. 318–326, 2012.

Ioannis Exarchos and Evangelos A. Theodorou. Stochastic optimal control via forward and backward stochastic differential equations and importance sampling. *Automatica*, 87:159–165, 2018.

Ioannis Exarchos, Evangelos A Theodorou, and Panagiotis Tsiotras. Stochastic L^1-optimal control via forward and backward sampling. *Systems & Control Letters*, 118:101–108, 2018.

Ioannis Exarchos, Evangelos Theodorou, and Panagiotis Tsiotras. Stochastic differential games: A sampling approach via FBSDEs. *Dynamic Games and Applications*, 9(2):486–505, Jun 2019. ISSN 2153-0793. doi: 10.1007/s13235-018-0268-4. URL https://doi.org/10.1007/s13235-018-0268-4.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 122–130. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *CoRR*, abs/1607.05447, 2016. URL http://arxiv.org/abs/1607.05447.

Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

Matthew J Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pp. 2946–2954, 2016.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Marcus Pereira, David D Fan, Gabriel Nakajima An, and Evangelos Theodorou. MPC-inspired neural network policies for sequential decision making. *arXiv preprint arXiv:1802.05803*, 2018.

Marcus A Pereira, Ziyi Wang, Ioannis Exarchos, and Evangelos A Theodorou. Learning deep stochastic optimal control policies using forward-backward SDEs. In *Robotics: Science and Systems*, 2019.

Marcus A Pereira, Ziyi Wang, Ioannis Exarchos, and Evangelos A Theodorou. Safe optimal control using stochastic barrier functions and deep forward-backward SDEs. *arXiv preprint arXiv:2009.01196*, 2020.

J. A. Primbs. Portfolio optimization applications of stochastic receding horizon control. In *2007 American Control Conference*, pp. 1811–1816, July 2007. doi: 10.1109/ACC.2007.4282251.

Maziar Raissi. Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. *arXiv preprint arXiv:1804.07010*, 2018.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pp. 113–124, 2019.

Reuven Y Rubinstein. Combinatorial optimization, cross-entropy, ants and rare events. In *Stochastic optimization: algorithms and applications*, pp. 303–363. Springer, 2001.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks. *arXiv preprint arXiv:1804.00645*, 2018.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pp. 896–903, 2005.

Ziyi Wang, Keuntaek Lee, Marcus A Pereira, Ioannis Exarchos, and Evangelos A Theodorou. Deep forward-backward SDEs for min-max control. In *58th IEEE Conference on Decision and Control (CDC), Nice, France, December 11-13*, pp. 6807–6814, 2019.

Enlu Zhou and Jiaqiao Hu. Gradient-based adaptive stochastic search for non-differentiable optimization. *IEEE Transactions on Automatic Control*, 59(7):1818–1832, 2014.