Credit Assignment Techniques in Stochastic Computation Graphs

Théophane Weber[*,†] **Nicolas Heess**[*] **Lars Buesing** **David Silver**
DeepMind

Abstract

Stochastic computation graphs (SCGs) provide a formalism to represent structured optimization problems arising in artificial intelligence, including supervised, unsupervised, and reinforcement learning. Previous work has shown that an unbiased estimator of the gradient of the expected loss of SCGs can be derived from a single principle. However, this estimator often has high variance and requires a full model evaluation per data point, making this algorithm costly in large graphs. In this work, we address these problems by generalizing concepts from the reinforcement learning literature. We introduce the concepts of value functions, baselines and critics for arbitrary SCGs, and show how to use them to derive lower-variance gradient estimates from partial model evaluations, paving the way towards general and efficient credit assignment for gradient-based optimization. In doing so, we demonstrate how our results unify recent advances in the probabilistic inference and reinforcement learning literature.

1 Introduction

The machine learning community has recently seen breakthroughs in challenging problems in classification, density modeling, and reinforcement learning (RL). To a large extent, successful methods have relied on gradient-based optimization (in particular on the backpropagation algorithm (Rumelhart et al., 1985)) for credit assignment, i.e. for answering the question how individual parameters (or units) affect the value of the objective. Recently, Schulman et al. (2015a) have shown that such problems can be formalized as optimization in stochastic computation graphs (SCGs). Furthermore, they derive a general gradient estimator that remains valid even in the presence of stochastic or non-differentiable computational nodes. This unified view reveals that numerous previously proposed, domain-specific gradient estimators, such as the likelihood ratio estimator (Glasserman, 1992), also known as 'REINFORCE' (Williams, 1992), as well as the pathwise derivative estimator, also known as the "reparameterization trick" (Glasserman, 1991; Kingma & Welling, 2014; Rezende et al., 2014), can be regarded as instantiations of the general SCG estimator. While theoretically correct and conceptually satisfying, the resulting estimator often exhibits high variance in practice, and significant effort has gone into developing techniques to mitigate this problem (Ng et al., 1999; Sutton et al., 2000; Schulman et al., 2015b; Arjona-Medina et al., 2018). Moreover, like backpropagation, the general SCG estimator requires a full forward and backward pass through the entire graph for each gradient evaluation, making the learning dynamics global instead of local. This can become prohibitive for models consisting of hundreds of layers, or recurrent model trained over long temporal horizons.

In this paper, starting from the SCG framework, we target those limitations, by introducing a collection of results which unify and generalize a growing body of results dealing with credit assignment. In combination, they lead to a spectrum of approaches that provide estimates of model parameter gradients for very general deterministic or stochastic models. Taking advantage of the model structure, they allow to trade off bias and variance of these estimates in a flexible manner. Furthermore, they provide mechanisms to derive local and asynchronous gradient estimates that relieve the need for a full model evaluation. Our results are borrowing from and generalizing a class of methods popular primarily in the reinforcement learning literature, namely that of learned approximations to the surrogate loss or its gradients, also known as *value functions*, *baselines* and *critics*. As new models with increasing structure and complexity are actively being developed by the machine learning community, we expect these methods to contribute powerful new training algorithms in a variety of fields such as hierarchical RL, multi-agent RL, and probabilistic programming.

This paper is structured as follows. We review the stochastic computation graph framework and recall the core result

*: Equal contribution; †: theophane@google.com

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

of Schulman et al. (2015a) in section 2. In section 3 we discuss the notions of value functions, baselines, and critics in arbitrary computation graphs, and discuss when and how they can be used to obtain both lower variance and local estimates of the model gradients, as well as local learning rules. In section 4 we provide similar results for gradient critics, i.e. estimates or approximations of the downstream loss gradient. In section 5, we discuss how the techniques and concepts introduced in the previous sections can be combined in different ways to obtain a wide spectrum of different gradient estimators with different strengths and weaknesses. Many examples, as well as a discussion of how our results can be special-cased to recent results from the literature, are discussed in the appendix.

Notation for derivatives

We use a 'physics style' notation by representing a function and its output by the same letter. For any two variables x and y in a computation graph, we use the partial derivative $\frac{\partial y}{\partial x}$ to denote the direct derivative of y with respect to x, and the $\frac{dy}{dx}$ to denote the total derivative of y with respect to x, taking into account all paths (or effects) from x on y; we use this notation even if y is still effectively a function of multiple variables. For any variable x, we let $\widetilde{x}$ denote the value of x which we treat as a constant in the gradient formula; i.e. $\widetilde{x}$ can be thought of a the output of a 'function' which behaves as the identity, but has gradient zero everywhere[1]. Finally, the gradient of any sampling operation is assumed to be 0.

All proofs are omitted from the main text and can be found in the appendix.

2 Preliminaries

An important class of problems in machine learning can be formalized as optimizing an expected loss $\mathbb{E}_{x_1,\ldots,x_n\sim p_\theta(x_1,\ldots,x_n)}[\ell(x_1,x_2,\ldots,\theta)]$ over parameters θ, where *both* the sampling distribution p_θ as well as the loss function ℓ can depend on θ. As we explain in greater detail in the appendix, concrete examples of this setup are reinforcement learning (where p is a composition of known policy and potentially unknown system dynamics) and variational autoencoders (where p is a composition of data distribution and inference network); cf. Fig. 1. Because of the dependency of the distribution on θ, backpropagation does not directly apply to this problem.

Two well-known estimators, the score function estimator and the pathwise derivative, have been proposed in the literature (for a primer on both, see appendix B). Both turn the gradient of an expectation into an expectation of gradients and hence allow for unbiased Monte Carlo estimates by simulation from known distributions, thus opening the door to stochastic approximations of gradient-based algorithms such as gradient descent (Robbins & Monro, 1985). They can also be unified when seen from the lens of the stochastic computation graph framework, which we detail next.

[1] such an operation is often called 'stop gradient' in the deep learning community; (Schulman et al., 2015a) denote it $\hat{x}$.

2.1 Stochastic computation graphs

SCG framework. We quickly recall the main results from Schulman et al. (2015a).

Definition 1 (Stochastic Computation Graph)**.** *A stochastic computation graph $\mathcal{G}$ is a directed, acyclic graph , with two classes of nodes (also called variables): deterministic nodes, and stochastic nodes.*

1. ***Stochastic nodes**, (represented with circles, denoted $\mathcal{S}$), which are distributed conditionally given their parents.*
2. ***Deterministic nodes** (represented with squares, denoted $\mathcal{D}$), which are deterministic functions of their parents.*

We further specialize deterministic nodes as follows:

- *Certain deterministic nodes with no parents in the graphs are referred to as **input nodes** $\theta \in \Theta$, which are set externally, including the parameters we differentiate with respect to.*
- *Certain deterministic nodes are designated as **losses** or **costs** (represented with diamonds) and denoted $\ell \in \mathcal{L}$. We aim to compute the gradient of the expected sum of costs with respect to some input node θ.*

A parent v of a node w is connected to it by a directed edge (v, w). Let $L = \sum_{\ell\in\mathcal{L}} \ell$ be the total cost.

For a node v, we let h_v denote the set of parents of v in the graph. A path between v and w is a sequence of nodes $a_0 = v, a_1, \ldots, a_{K-1}, a_K = w$ such that all (a_{k-1}, a_k) are directed edges in the graph; if any such path exists, we say that w descends from v and denote it $v \prec w$. We say the path is blocked by a set of variables $\mathcal{V}$ if any of the $a_i \in \mathcal{V}$. By convention, v descends from itself. For a variable x and set $\mathcal{V}$, we say x can be deterministically computed from $\mathcal{V}$ if there is no path from a stochastic variable to x which is not blocked by $\mathcal{V}$ (cf. Fig. 2). Algorithmically, this means that knowing the values of all variables in $\mathcal{V}$ allows to compute x without sampling any other variables; mathematically, it means that conditional on $\mathcal{V}$, x is a constant. Finally, whenever we use the notion of conditional independence, we refer to the notion of conditional independence informed by the graph (i.e. d-separation) (Geiger et al., 1990; Koller & Friedman, 2009).

Gradient estimator for SCG. Consider the expected loss $\mathcal{J}(\theta) = \mathbb{E}_{s\in\mathcal{S}}[L]$. We present the general gradient estimator for the gradient of the expected loss $\frac{d}{d\theta}\mathcal{J}(\theta)$ derived in (Schulman et al., 2015a).

For any stochastic variable v, we let $\log p(v)$ denote the conditional log-probability of v given its parents, i. e. the value $\log p(v|h_v)$, and let $s(v,\theta)$ denote the score function $\frac{d\log p(v)}{d\theta}$.

Theorem 1. *[Theorem 1 from (Schulman et al., 2015a)] Under simple regularity conditions,*

$$\frac{d}{d\theta}\mathcal{J}(\theta) = \mathbb{E}\left[\sum_{\substack{v\in\mathcal{S}\\ \theta\prec v}} s(v,\theta)L + \sum_{\substack{\ell\in\mathcal{L}\\ \theta\prec\ell}} \frac{d\ell}{d\theta}\right]$$

Here, the first term corresponds to the influence θ has on the loss through the non-differentiable path mediated by stochastic nodes. Intuitively, when using this estimator for gradient descent, θ is changed so as to increase or 'reinforce' the probability of samples of v that empirically led to lower total cost L. The second term corresponds to the direct influence θ has on the total cost through differentiable paths. Note that differentiable paths include paths going through reparameterized random variables.

3 Value based methods

The gradient estimator from theorem 1 is very general and conceptually simple but it tends to have high variance (see for instance the analysis found in Mnih & Rezende, 2016), which affects convergence speed (see e.g. Schmidt et al., 2011). Furthermore, it requires a full evaluation of the graph. To address these issues, we first discuss several variations of the basic estimator in which the total cost L is replaced by its *deviation* from the expected total cost, or conditional expectations thereof, with the aim of reducing the variance of the estimator. We then discuss how approximations of these conditional expectations can be learned locally, leading to a scheme in which gradient computations from partial model evaluations become possible.

3.1 Values

In this section, we use the simple concept of conditional expectations to introduce a general definition of value function in a stochastic computation graph.

Definition 2 (Value function)**.** *Let $\mathcal{X}$ be an arbitrary subset of $\mathcal{G}$, $\mathbf{x}$ an assignment of possible values to variables in $\mathcal{X}$ and S an arbitrary scalar value in the graph. The value function for set $\mathcal{X}$ is the expectation of the quantity S conditioned on $\mathcal{X}$:*

$$V : \mathbf{x} \mapsto V(\mathbf{x}; S) = \mathbb{E}_{\mathcal{G}\setminus\mathcal{X}|\mathcal{X}=\mathbf{x}}[S].$$

Intuitively, a value function is an estimate of the cost which averages out the effect of stochastic variables not in $\mathcal{X}$, therefore the larger the set, the fewer variables are averaged out.

The definition of the value function as conditional expectation results in the following characterization:

Lemma 1. *For a given assignment $\mathbf{x}$ of $\mathcal{X}$, $V(\mathbf{x}; S)$ is the optimal mean-squared error estimator of S given input $\mathcal{X}$:*

$$V(\mathbf{x}; S) = \operatorname{argmin}_{v_{\mathbf{x}}} \mathbb{E}_{\mathcal{G}\setminus\mathcal{X}|\mathcal{X}=\mathbf{x}}\left[(S - v_{\mathbf{x}})^2\right].$$

Consider an arbitrary node $v \in \mathcal{G}$, and let $L(v) = \sum_{\substack{\ell\in\mathcal{L}\\ v\prec\ell}} \ell$ denote the v-rooted cost-to-go, i.e. the sum of costs 'downstreams' from v (similar notation is used for $L(V)$ if V is a set). The scalar S will often be the cost-to-go $L(v)$ for some fixed node v; furthermore, when clear from context, we use $\mathcal{X}$ to both refer to the variables and the values they take. For notational simplicity, we will denote the corresponding value function $V(\mathcal{X})$.

Fig. 3 shows multiple examples of value functions for different graphs. The above definition is broader than the typical one used in reinforcement learning. There, due to the simple chain structure of the Markov Decision Processes, the resulting Markov properties of the graph, and the particular choice of $\mathcal{X}$, the expectation is only with respect to downstream nodes. Importantly, according to Def. 2 the value can depend on $\mathcal{X}$ via ancestors of $\mathcal{X}$ (e.g. example in Fig. 3c). Lemma 1 remains valid nevertheless.

3.2 Baselines and critics

In this section, we will define the notions of baselines and critics and use them to introduce a generalization of theorem 1 which can be used to compute lower variance estimator of the gradient of the expected cost. We will then show how to use value functions to design baselines and critics.

Consider an arbitrary node v and input θ.

Definition 3 (Baseline)**.** *A* baseline *B for v is any function of the graph such that $\mathbb{E}[s(v,\theta)B] = 0$. A* baseline set *$\mathcal{B}$ is an arbitrary subset of the non-descendants of v.*

Baseline sets are of interest because of the following property:

Property 1. *Let B be an arbitrary scalar function of $\mathcal{B}$. Then B is a baseline for v.*

Common choices are constant baselines, i.e. $\mathcal{B} = \emptyset$, or baselines $B(h_v)$ only depending on the parents $\mathcal{B} = h_v$ of v.

Definition 4 (Critic)**.** *A* critic *Q of cost $L(v)$ for v is any function of the graph such that $\mathbb{E}[s(v,\theta)(L(v) - Q)] = 0$.*

By linearity of expectations, linear combinations of baselines are baselines, and convex combinations of critics are critics.

The use of the terms *critic* and *baseline* is motivated by their respective roles in the following theorem, which generalizes the policy gradient theorem (Sutton et al., 2000):

Theorem 2. *Consider an arbitrary baseline B_v and critic Q_v for each stochastic node v. Then,*

$$\frac{d}{d\theta}\mathcal{J}(\theta) = \mathbb{E}\left[\sum_{\substack{v\in\mathcal{S}\\ \theta\prec v}} G_v + \sum_{\substack{\ell\in\mathcal{L}\\ \theta\prec\ell}} \frac{d}{d\theta}\ell\right],$$

where $G_v = s(v,\theta)\Big(Q_v - B_v\Big)$.

The difference $Q_v - B_v$ between a critic and a baseline is called an *advantage* function.

Theorem 2 enables the derivation of a surrogate loss. Let L^s be defined as $L^s = L + \sum_{\substack{\ell\in\mathcal{L}\\ \theta\prec\ell}} \log p(v)\left(\widetilde{Q} - \widetilde{B}\right)$, where we recall that the tilde notation indicates a constant from the point of view of computing gradients. Then, the gradient of the expected cost $\mathcal{J}(\theta)$ equals the gradient of L^s in expectation: $\frac{d}{d\theta}\mathbb{E}[L] = \mathbb{E}\left[\frac{d}{d\theta}L^s\right]$.

Before providing intuition on this theorem, we see how value functions can be used to design baselines and critics:

Definition 5 (Baseline and critic value functions)**.** *For any node v and baseline set $\mathcal{B}$, a special case of a baseline is to choose the value function with set $\mathcal{B}$. Such a baseline is called a* ***baseline value function****.*
Let a critic set $\mathcal{C}$ be a set such that $v \in \mathcal{C}$, and $\log p(v)$ and $L(v)$ are conditionally independent given $\mathcal{C}$; a special case is when $\mathcal{C}$ is such that $\log p(v)$ is deterministically computable given $\mathcal{C}$. Then the value function for set $\mathcal{C}$ is a critic for v which we call a ***critic value function*** *for v.*

Figure 3 contains several examples of value functions which take the role of baselines and critics for different nodes. In the standard MDP setup of the RL literature, $\mathcal{C}$ consists of the state s and the action a which is taken by a stochastic policy π in state s with probability $\log\pi(a|s)$, which is a deterministic function of (s,a). Definition 5 is more general than this conventional usage of critics since it does not require $\mathcal{C}$ to contain all stochastic ancestor nodes that are required to evaluate $\log p(v)$. For instance, assume that the action is conditionally sampled from the state s and some source of noise ξ, for instance due to dropout, with distribution $\pi(a|\xi,s)$[2]. The critic set may but does not need to include ξ; if it does not, $\log\pi(a|\xi,s)$ is not a deterministic function of a and s. The corresponding critic remains useful and valid.

Three related ideas guide the derivation of theorem 2. To give intuition, let us analyze the term G_v, which replaces the score function weighted by the total cost $s(v,\theta)L$. First, the conditional distribution of v only influences the costs downstream from v, hence we only have to reinforce the probability of v with the cost-to-go $L(v)$ instead the total cost L. Second, the extent to which a particular sample v contributed to cost-to-go $L(v)$ should be compared to the cost-to-go the graph typically produces in the first place. This is the intuition behind subtracting the baseline B, also known as a *control variate*. Third, we ideally would like to understand the precise contribution of v to the cost-to-go, not for a particular value of downstream random variables, but on average. This is the idea behind the critic Q. The advantage (difference between critic and baseline) therefore provides an estimate of 'how much better than anticipated' the cost was, as a function of the random choice v.

Baseline value functions are often used as baselines as they approximate the optimal baseline (see Appendix G.1). Critic value functions are often used as they provide an expected downstream cost given the conditioning set. Furthermore, as we will see in the next section, value functions can be estimated in a recursive fashion, enabling local learning of the values, and sharing of value functions between baselines and critics. For these reasons, in the rest of this paper, we will only consider baseline value functions and critic value functions.

In the remainder of this section, we consider an arbitrary value function with conditioning set $\mathcal{X}$.

3.3 Recursive estimation and Markov properties

A fundamental principle in RL is given by the Bellman equation – which details how a value function can be defined recursively in terms of the value function at the next time step. In this section, we generalize the notion of recursive computation to arbitrary graphs.

The main result, which follows immediately from the law of iterated expectations, characterizes the value function for one set, as an expectation of a value function (or critic / baseline value function) of a larger set:

Lemma 2. *Consider two sets $\mathcal{X}^1 \subset \mathcal{X}^2$, and an arbitrary quantity S. Then we have: $\mathbb{E}[V(\mathcal{X}^2;S)|\mathcal{X}^1] = V(\mathcal{X}^1;S)$.*

This lemma is powerful, as it allows to relate value functions as average of over value function. A simple example in RL is the relation (here, in the infinite discounted case) between the Q function $Q^\pi(s,a) = \mathbb{E}[R|s,a]$ of a policy and the corresponding value function $V^\pi(s) = \mathbb{E}[R|s]$, which is given by $V^\pi(s) = \sum_a \pi(a|s)Q^\pi(s,a)$. Note this equation relates a critic value function to a value function typically used as baseline.

To fully leverage the lemma above, we proceed with a Markov property for graphs[3], which captures the following situation: given two conditioning sets $\mathcal{X}^1 \subset \mathcal{X}^2$, it may be the case that the additional information contained

[2]in this example, it is important ξ is used only once; it cannot be used to compute other actions.

[3]borrowed from well known conditional independence conditions in graphical models, and adapted to our purposes.

in $\mathcal{X}^2$ does not improve the accuracy of the cost prediction compared to the information contained in the smaller set $\mathcal{X}^1$.

Definition 6. *For conditioning set $\mathcal{X}$, we say that $\mathcal{X}$ is* ***Markov*** *(for $L(v)$) if for any w such that there exists a directed path from w to $L(v)$ not blocked by $\mathcal{X}$, none of the descendants of w are in $\mathcal{X}$.*

Let $\mathcal{X}^{\uparrow}$ be the set of all ancestors of nodes $\mathcal{X}$[4].

Property 2. *Let $\mathcal{X}$ be Markov, consider any $\mathcal{X}'$ such that $\mathcal{X} \subset \mathcal{X}' \subset \mathcal{X}^{\uparrow}$. For any x' assignment of values to the variables in $\mathcal{X}'$, let $x'_{|\mathcal{X}}$ be the restriction of x' to the variables in $\mathcal{X}$. Then, for all x', $V(\mathcal{X} = x'_{|\mathcal{X}}) = V(\mathcal{X}' = x')$,, which we will simply denote, with a slight abuse of notation, $V(\mathcal{X}') = V(\mathcal{X})$.*

In other words, the information contained in $\mathcal{X}^{\uparrow} \setminus \mathcal{X}$ is irrelevant in terms of cost prediction, given access to the information in $\mathcal{X}$. Several examples are shown in Fig. 4. It is worth noting that Def. 6 does not rule out changes in the expected value of $L(v)$ after adding additional nodes to $\mathcal{X}$ (cf. Fig. 4(d,e)). Instead it rules out correlations between $\mathcal{X}$ and $L(v)$ that are mediated via *ancestors* of nodes in $\mathcal{X}$ as in the example in Fig. 4(a,b,c)).

The notion of Markov set can be used to refine Lemma 2:

Lemma 3 (Generalized Bellman equation)**.** *Consider two sets $\mathcal{X}^1 \subset \mathcal{X}^{2\uparrow}$, and suppose $\mathcal{X}^2$ is Markov. Then we have: $\mathbb{E}[V(\mathcal{X}^2)|\mathcal{X}^1] = V(\mathcal{X}^1)$.*

The Markov assumption is critical in allowing to 'push' the boundary at which the expectation is defined; without it, lemma 2 only allows to relate value functions of sets which are subset of one another. But notice here that no such inclusion is required between $\mathcal{X}^1$ and $\mathcal{X}^2$ themselves. In the context of RL, this corresponds to equations of the type $V(s) = \sum_a \pi(s,a)\left(r(s,a) + \sum_{s'} P(s'|s,a)V(s')\right)$ (see Fig. 5), though to get the separation between the reward and the value at the next time step, we will need a slight refinement, which we detail in the next section.

3.4 Decomposed costs and bootstrap

In the previous sections we have considered a value function with respect to a node v which predicts an estimate of the cost-to-go $L(v)$ from node v (note $L(v)$ was implicit in most of our notation). In this section, we write the cost-to-go at a node as a funtion of cost-to-go from other nodes or collection of nodes, and leverage the linearity of expectation to turn these relations between costs into relation between value functions.

Definition 7 (Decomposed costs)**.** *For a node v and a collection $\mathbf{V} = \{V_0, V_1, \ldots, V_D\}$ in the graph, we say that the cost $L(v)$ can be decomposed with set $\mathbf{V}$ if $L(v) = \sum_i L(V_i)$.*

This implies that cost nodes can be grouped in disjoint sets corresponding to the descendents of different sets V_i, without double-counting. A common special case is a tree, where each V_i is a singleton containing a single child $\{v_i\}$ of v.

Theorem 3 (Bootstrap principle for SCGs)**.** *Suppose the cost-to-go $L(v)$ from node v can be decomposed with sets $\mathbf{V} = \{V_0, \ldots, V_D\}$, and consider an arbitrary set $\mathcal{X}_v$ with associated value function $V(\mathcal{X}_v, L(v))$. Furthermore, for each set V_i, consider a set $\mathcal{X}_{V_i}$ and associated value function: $V(\mathcal{X}_{V_i}, L(V_i))$. If for each i, $\mathcal{X}_v \subset \mathcal{X}_{V_i}$, or if for each i, $\mathcal{X}_{V_i}$ is Markov and $\mathcal{X}_v \subset \mathcal{X}^{\uparrow}_{V_i}$, then:*

$$V(\mathcal{X}_v) = \sum_i \mathbb{E}_{\mathcal{G}\setminus\mathcal{X}_v|\mathcal{X}_v}\left[V(\mathcal{X}_{V_i})\right].$$

Fig. 6 highlights potential difficulties of defining correct bootstrap equations for various graphs.

From the bootstrap equation follows a special case, which we call *partial averaging*, often used for critics:

Corollary 1 (Partial averages)**.** *Suppose that for each i, $\mathcal{X}_{V_i}$ is Markov and $\mathcal{X}_{V_i} \subset \mathcal{X}_v \subset \mathcal{X}^{\uparrow}_{V_i}$. Without loss of generality, define V_0 as the collection of all cost nodes which can be deterministically computed from $\mathcal{X}_v$. Then,*

$$V(\mathcal{X}_v) = \sum_i V(\mathcal{X}_{V_i}) = \sum_{\ell \in V_0} \ell + \sum_{i \geq 1} V(\mathcal{X}_{V_i}).$$

The term 'partial average' indicates that the value function is a conditional expectation (i.e. 'averaging' variables) but that it combines averaged cost estimates (the value terms $V(\mathcal{X}_{V_i})$) and empirical costs ($\sum_{\ell \in V_0} \ell_v$). Fig. 7 shows some examples for generic graphs.

In the case of RL for instance, a K-step return is a form of partial average, since the return R_t – sum of all rewards downstream from state s_t – can be written as the sum of all rewards in $V_0 = \{r_t, \ldots, r_{t+k-1}\}$ and downstream from $V_1 = \{s_{t+k}\}$; the critic value function $V(s_t, \ldots, s_{t+k})$ is therefore equal[5] to $\sum_{t'=t}^{t+k-1} r_{t'} + V(s_{t+k})$. This implies in turn that $V(s_t) = \mathbb{E}[V(s_t, \ldots, s_{t+k})]$ is also equal to $\mathbb{E}[\sum_{t'=t}^{t+k} r_{t'} + V(s_{t+k})]$.

3.5 Approximate Value functions

In practice, value functions often cannot be computed exactly. In such cases, one can resort to learning parametric approximations. For node v, conditioning set $\mathcal{X}$, we will consider an approximate value function $\hat{V}^{\phi}(\mathcal{X})$ as an approximation (with parameters ϕ) to the value function $V(\mathcal{X}) = \mathbb{E}_{\mathcal{G}\setminus\mathcal{X}_v|\mathcal{X}_v}\left[L(v)\right]$.

Following corollary 1, we know that for a possible assignment $\mathbf{x}$ of variables $\mathcal{X}$, $V(\mathbf{x})$ minimizes $\mathbb{E}_{\mathcal{G}\setminus\mathcal{X}|\mathcal{X}}\left[(L(v) - v_{\mathbf{x}})^2\right]$ over $v_{\mathbf{x}}$. We therefore elect to optimize ϕ by considering the following weighted average,

[4]Recall that by convention nodes are descendants of themselves, so $\mathcal{X} \subset \mathcal{X}^{\uparrow}$

[5]We assume for simplicity that the rewards are deterministic functions of the state; the result can be trivially generalized.

called a *regression on return* in reinforcement learning:

$$\begin{aligned}\mathcal{L}^\phi &= \mathbb{E}_\mathcal{X}\left[\mathbb{E}_{\mathcal{G}\setminus\mathcal{X}|\mathcal{X}}\left[(L(v)-\hat{V}^\phi(\mathcal{X}))^2\right]\right] \\ &= \mathbb{E}_\mathcal{G}\left[(L(v)-\hat{V}^\phi(\mathcal{X}))^2\right],\end{aligned}$$

from which we obtain:

$$\frac{d\mathcal{L}^\phi}{d\phi} = \mathbb{E}_\mathcal{G}\left[\frac{d}{d\phi}\hat{V}^\phi(\mathcal{X})\left(\hat{V}^\phi(\mathcal{X})-L(v)\right)\right] \quad (1)$$

which can easily be computed by forward sampling from $\mathcal{G}$, even if conditional sampling given $\mathcal{X}$ is difficult.

We now leverage the recursion methods from the previous sections in two different ways. The first is to use the combination of approximate value functions and partial averages to define other value functions. For a partial average as defined in theorem 1 and family of approximate value functions $\hat{V}^\phi_{v_i}(\mathcal{X}_{v_i})$, we can define an approximate value function through the bootstrap equation: $\sum_{\ell\in V_0}\ell_v+\sum_i\hat{V}^\phi(\mathcal{X}_{v_i})$. In other words, using the bootstrap equations, approximating value functions for certain sets automatically defines other approximate value functions for other sets.

In general, we can trade bias and variance by making V_0 larger (which will typically result in lower bias, higher variance) or not, i.e. by shifting the boundary at which variables are integrated out, for instance by using K-step returns or λ-weighted returns. An extreme case of a partial average is not an average at all, where $\mathcal{X}=\mathcal{G}$, in which case the value function is the empirical return $L(v)$.

The second way to use the bootstrap equation is to provide a different, lower variance target to the value function regression. By combining theorem 3 and equation 1, we obtain:

$$\frac{d\mathcal{L}_\phi}{d\phi} = \mathbb{E}_\mathcal{G}\left[\frac{d}{d\phi}\hat{V}^\phi(\mathcal{X})\left(\hat{V}^\phi(\mathcal{X})-\sum_i\hat{V}^\phi(\mathcal{X}_{v_i})\right)\right]$$

By following this gradient, the value function $\hat{V}^\phi(\mathcal{X}_v)$ will tend towards the bootstrap value $\sum_{\ell\in V_0}\ell_v+\sum_{i\geq 1}\hat{V}^\phi(\mathcal{X}_{v_i})$ instead of the return $L(v)$. Because the former has averaged out stochastic nodes, it is a lower variance target, and should in practice provide a stronger learning signal. Furthermore, as it can be evaluated as soon as $\mathcal{X}_{v_i}$ is evaluated, it provides a local or online learning rule for the value at v; by this we mean the corresponding gradient update can be computed as soon as all sets $\mathcal{X}_{v_i}$ are evaluated. In RL, this local learning property can be found in actor-critic schemes: when taking action a_t in state s_t, as soon as the immediate reward r_t is computed and next state s_{t+1} is evaluated, the value function $V(s_t)$ (which is a baseline for a_t) can be regressed against low-variance target $r_t+V(s_{t+1})$, which can also be used as critic for a_t and therefore used to update the corresponding policy.

4 Gradient-based methods

In the previous section, we developed techniques to lower the variance of the score-function terms $\mathbb{E}\left[\left(\frac{d}{d\theta}\log p(v)\right)\left(Q(\mathcal{C})-B(\mathcal{B}_v)\right)\right]$ in the gradient estimate. This led to the construction of a surrogate loss L^s which satisfies $\frac{d}{d\theta}\mathbb{E}\left[\mathcal{J}(\theta)\right]=\mathbb{E}\left[\frac{dL^s}{d\theta}\right]$.

In this section, we develop corresponding techniques to lower the variance estimates of the gradients of surrogate cost $\frac{d}{d\theta}L^s$. To this end, we will again make use of conditional expectations to partially average out variability from stochastic nodes. This leads to the idea of a *gradient-critic*, the equivalent of the value critic for gradient-based approaches.

4.1 Gradient-Critic

Definition 8 (Value-gradient)**.** *The **value-gradient** for v with set $\mathcal{C}$ is the following function of $\mathcal{C}$:*

$$g(\mathcal{C}) = \mathbb{E}_{\mathcal{G}\setminus\mathcal{C}|\mathcal{C}}\left[\frac{dL^s}{dv}\right].$$

Value-gradients are not directly useful in our derivations but we will see later that certain value-gradients can reduce the variance of our estimators. We call these value-gradient *gradient-critics*.

Definition 9 (Gradient-critic)**.** *Consider two nodes u and v, and a value-gradient g_v for node v with set $\mathcal{C}$. If $\frac{dv}{du}$ and $\frac{dL^s}{dv}$ are conditionally independent given $\mathcal{C}$[6], then we say the value-gradient is a **gradient-critic** for v with respect to u.*

Corollary 2. *If $\frac{dv}{du}$ is deterministically computable from $\mathcal{C}$, then $g_v(\mathcal{C})$ is a gradient-critic for v with respect to u.*

We can use gradient-critics in the backpropagation equation. First, we recall the equation for backpropagation and stochastic backpropagation. Let u be an arbitrary node of $\mathcal{H}$, and $\{v_1,\ldots,v_d\}$ be the children of u in $\mathcal{G}$. The backpropagation equations state that: $\frac{dL^s}{du}=\sum_i\frac{dL^s}{dv_i}\frac{\partial v_i}{\partial u}$. From this we obtain the stochastic backpropagation equations:

$$\mathbb{E}_\mathcal{G}\left[\frac{dL^s}{du}\right] = \mathbb{E}_\mathcal{G}\left[\sum_i\frac{dL^s}{dv_i}\frac{\partial v_i}{\partial u}\right]$$

Gradient-critics allow for replacing these stochastic estimates by conditional expectations, potentially achieving lower variance:

Theorem 4. *For each child v_i of v, let g_{v_i} be a gradient-critic for v_i with respect to u. We then have:*

$$\mathbb{E}_\mathcal{G}\left[\frac{dL^s}{du}\right] = \mathbb{E}_\mathcal{G}\left[\sum_i g_{v_i}\frac{\partial v_i}{\partial u}\right].$$

[6] See lemma 7 in Appendix for a characterization of conditional independence between total derivatives.

Note a similar intuition as the idea of critic defined in the previous section. In both cases, we want to evaluate the expectation of a product of two correlated random variables, and replace one by its expectation given a set which makes the variables conditionally independent.

4.2 Horizon gradient-critic and gradient-critic bootstrap

More generally, we do not have to limit ourselves to $\{v_1, v_2, \dots, v_d\}$ being children of u. We define a *separator set* for u in $\mathcal{H}$ to be a set $\{v_1, v_2, \dots, v_d\}$ such that every deterministic path from u to the loss L^s is blocked by a $v_i \in \mathcal{H}$. For simplicity, we further require the separator set to be *unordered*, which means that for any $i \neq j$, v_j cannot be an ancestor to v_i; we drop this assumption for a generalized result in the appendix C. Under these assumptions, the backpropagation rule can be rewritten (see (Naumann, 2008; Parmas, 2018)):

$$\mathbb{E}_{\mathcal{G}}\left[\frac{dL^s}{du}\right] = \mathbb{E}_{\mathcal{G}}\left[\sum_i \frac{dL^s}{dv_i}\frac{dv_i}{du}\right]. \tag{2}$$

Theorem 5. *Assume that for every i, g_{v_i} is a gradient critic for v_i with respect to u. We then have:*

$$\mathbb{E}_{\mathcal{G}}\left[\frac{dL^s}{du}\right] = \mathbb{E}_{\mathcal{G}}\left[\sum_i g_{v_i}\frac{dv_i}{du}\right].$$

This theorem allows us to 'push' the horizon after which we start using gradient-critics. It constitutes the gradient equivalent of partial averaging, since it combines stochastic backpropagation (the terms $\frac{dv_i}{du}$) and gradient critics g_{v_i}.

We now show how this theorem to derive a generic notion of bootstrapping for gradient-critics:

Theorem 6 (Gradient-critic bootstrap)**.** *Consider a node u, unordered separator set $\{v_1, \dots, v_d\}$. Consider value-gradient g_u with set $\mathcal{C}_u$ for node u, and $(g_{v_1}, g_{v_2}, \dots, g_{v_d})$ with Markov sets $(\mathcal{C}_{v_1}, \dots, \mathcal{C}_{v_d})$ critics for v_i with respect to u. Suppose that for all i, $\mathcal{C}_u \subset \mathcal{C}_{v_i}$. Then,*

$$g_u = \sum_i \mathbb{E}_{\mathcal{C}_{v_i}|\mathcal{C}_u}\left[g_{v_i}\frac{dv_i}{du}\right]. \tag{3}$$

4.3 Gradient-critic and gradient of critic

The section above proposes an operational definition of a gradient critic, in that one can replace the sampled gradient $\frac{dL^s}{du}$ by the expectation of the gradient g_u. A natural question follows – is a value-gradient the gradient of a value function? Similarly, is a gradient-critic the gradient of a critic function?

It is in general not true that the value-gradient must be the gradient of a value function. However, if the critic set is Markov, the gradient-critic is the gradient of the critic.

Theorem 7. *Consider a node v and critic set $\mathcal{C}$, and corresponding critic value function $Q(\mathcal{C})$ and gradient-critic $g_v(\mathcal{C})$. If $\mathcal{C}$ is Markov for v, then we have: $\frac{dQ(\mathcal{C})}{dv} = g_v(\mathcal{C})$.*

This characterization of the gradient-critic as gradient of a critic plays a key role in using reparametrization techniques when gradients are not computable. For instance, in a continuous control application of reinforcement learning, the state of the environment can be assumed to be an unknown but differentiable function of the previous state and of the action. In this context, a critic can readily be learned by predicting total costs. By the argument above, the gradient of this critic actually corresponds to the gradient-critic of the unknown environment dynamics. This technique is at the heart of differentiable policy gradients (Lillicrap et al., 2015) and stochastic value gradients (Heess et al., 2015).

4.4 Gradient-critic approximation and computation

Following the arguments regarding conditional expectation and square minimization from section 3.1, we know that g_v satisfies the following minimization problem:

$$g_v(\mathcal{C}) = \text{argmin}_{g_{c_v}}\ \mathbb{E}_{\mathcal{G}\setminus\mathcal{C}|\mathcal{C}}\left[\left(g_{c_v} - \frac{dL^s}{dv}\right)^2\right]$$

For a parametric approximation g_v^ϕ, and using the same weighting scheme as section 3.5, it follows that:

$$\begin{aligned}\mathcal{L}^\phi &= \mathbb{E}_{\mathcal{G}}\left[\left(g_v^\phi(\mathcal{C}) - \frac{dL^s}{dv}\right)^2\right] \\ \frac{d\mathcal{L}^\phi}{d\phi} &= \mathbb{E}_{\mathcal{G}}\left[\frac{d}{d\phi}g_v^\phi(\mathcal{C})\left(g_v^\phi(\mathcal{C}) - \frac{dL^s}{dv}\right)\right]\end{aligned} \tag{4}$$

Finally, if $\mathcal{C}$ is Markovian for v, from Theorem 7, the gradient-critic g_v^ϕ can be defined in two ways: first, as the critic of a gradient ($\mathbb{E}[\frac{dL^s}{dv}|\mathcal{C}]$), and second, as the gradient of a critic ($\frac{d}{dv}\mathbb{E}[L|\mathcal{C}] = \frac{d}{dv}Q(\mathcal{C})$).

In this case, it makes sense to parameterize g_v^ϕ as the derivative of a function $Q^\phi(\mathcal{C})$, where $v \in \mathcal{C}$, i.e. define $g_v^\phi(\mathcal{C}) = \frac{dQ^\phi(\mathcal{C})}{dv}$. The gradient-critic can therefore defined directly by the gradient-critic loss, and indirectly by the critic loss. It therefore makes sense to combine them:

$$\mathcal{L}^\phi = \mathbb{E}_{\mathcal{G}}\left[\alpha(Q^\phi(\mathcal{C}) - L)^2 + \beta\left(\frac{dQ^\phi(\mathcal{C})}{dv} - \frac{dL^s}{dv}\right)^2\right] \tag{5}$$

where α, β are relative weights for each norm. This is called a *Sobolev* norm, see also (Czarnecki et al., 2017).

5 Combination of estimators and critics

The techniques outlined above suggest a "menu" of choices for constructing gradient estimators for stochastic computation graphs. We lay out a few of these choices, highlighting how are results strictly generalize known methods from the literature.

Reparameterization; use of score function and pathwise derivative estimators. Many distributions can be reparameterized, including discrete random variables (Maddison et al., 2016; Jang et al., 2016). This opens a choice between SF and PD estimator. The latter allows gradients to flow through the graph. Where exact gradients are not available (e.g. in MDPs or in probabilistic programs and approximate Bayes computation (Meeds & Welling, 2014; Ong et al., 2018)) gradients of critics can under certain conditions be used in combination with reparameterized distributions (see e.g. (Heess et al., 2015).

Grouping of random variables. For many graphs there is a natural grouping of random variables that suggests obvious baseline and critic choices. Taking into account the detailed Markov structure of the computation graph may however reveal interesting alternatives. For instance, independent action dimensions allow updates in which baselines are conditioned on the values chosen for other action dimensions. Such ideas have been exploited e.g. in work on action-dependent baselines (Wu et al., 2018), and in multiagent domains (Foerster et al., 2017). Other applications have been found in hierarchical RL, for instance in (Bacon et al., 2017), where the relation between options and actions directly informs the computation graph structure, in turn defining the correct value bootstrap equations and corresponding policy gradient theorem.

Use of value critics and baselines. The discussion in 3.2 highlights there are typically many different choices for constructing baselines and critics even beyond the choice of a particular variable grouping (e.g. the use of K-step returns or generalized advantage estimates in RL, Schulman et al. (2015a)) even for simple graphs (such as chains).

Use of gradient-critics. For reparameterized variables or general deterministic pathways through a computation graph gradient critics can be used. Gradient critics for a given node in the graph can be obtained by either directly approximating the (expected) gradient of the downstream loss, or by approximating the value of the future loss terms (as for value critics) and then using the gradient of this approximation. Gradient-critics allow to conceptualize the links between related notions of value-gradient found in (Fairbank & Alonso, 2012; Fairbank, 2014), stochastic value-gradients (Heess et al., 2015), and synthetic gradients (Jaderberg et al., 2016; Czarnecki et al., 2017).

Debiasing of estimators through policy-gradient correction. The use of critics or gradient-critics results in biased estimators. When using gradient-critic, it is often possible to debias the use of the gradient-critic by adding a correction term corresponding to the critic error. The resulting scheme is unbiased, but may or may not have lower variance than 'naive' estimators which do not use critics at all. This is sometimes known as 'action-conditional' baselines in the literature (Tucker et al., 2018), and is also strongly related to Stein variational gradient (Liu & Wang, 2016). See App. G.3 for more details.

Bootstrapping. Targets for baselines and critics can be obtained in a variety of ways: For instance, they can be regressed directly onto empirical sums of downstream losses ("Monte Carlo" returns in reinforcement learning). But targets can also constructed from other, downstream value or gradient approximations (e.g. "K-step returns" or "λ-weighted returns" in reinforcement learning), an idea discussed above under the name *bootstrapping* (sections 3.3 and 4.2). The appropriate choice here will again be highly application specific.

Decoupled updates. In its original form Theorem 1 requires a full and backward pass through the entire computation graph to compute a single sample approximation to its gradient. Through appropriate combination of surrogate signals and bootstrapping, however, updates for different parts of the graph can be decoupled to different extents. For instance, in reinforcement learning actor-critic algorithms compute updates to the policy parameters from single transitions s_t, a_t, r_t, s_{t+1}. The same ideas can be applied to general computation graphs where additional freedom can allow even more flexible schemes (e.g. individual parts of the graph can be updated more frequently than others).

6 Conclusion

In this paper, we have provided a detailed discussion and mathematical analysis of credit assignment techniques for stochastic computation graphs. Our discussion explains and unifies existing algorithms, practices, and results obtained in a number of particular models and different fields of the ML literature. They also provide insights about the particular form of algorithms, highlighting how they naturally result from the constraints imposed by the computation graph structure, instead of ad-hoc solutions to particular problems.

The conceptual understanding and tools developed in this work do not just allow the derivation of existing solutions as special cases. Instead, they also highlight the fact that for any given model there typically is a menu of choices, each of which gives rise to a different gradient estimator with different advantages and disadvantages. For new models, these tools provide methodological guidance for the development of appropriate algorithms. In that sense our work emphasizes a similar separation of model and algorithm that has been proven fruitful in other domains, for instance in the probabilistic modeling and inference literature.

We believe that this separation as well as a good understanding of the underlying principles will become increasingly important as both models and training schemes become more complex and the distinction between different model classes blurs.

Acknowledgments

Wed like to thank the many people for useful discussions and feedback on the research and the manuscript, including Ziyu Wang, Sébastien Racanière, Yori Zwols, Chris Maddison, Arthur Guez and Andriy Mnih.

References

Archer, Evan, Park, Il Memming, Buesing, Lars, Cunningham, John, and Paninski, Liam. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

Arjona-Medina, Jose A, Gillhofer, Michael, Widrich, Michael, Unterthiner, Thomas, and Hochreiter, Sepp. Rudder: Return decomposition for delayed rewards. *arXiv preprint arXiv:1806.07857*, 2018.

Bacon, Pierre-Luc, Harb, Jean, and Precup, Doina. The option-critic architecture. In *AAAI*, pp. 1726–1734, 2017.

Bayer, Justin and Osendorfer, Christian. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

Bengio, Yoshua, Léonard, Nicholas, and Courville, Aaron. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Buesing, Lars, Weber, Theophane, Racaniere, Sebastien, Eslami, SM, Rezende, Danilo, Reichert, David P, Viola, Fabio, Besse, Frederic, Gregor, Karol, Hassabis, Demis, et al. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

Chung, Junyoung, Kastner, Kyle, Dinh, Laurent, Goel, Kratarth, Courville, Aaron C, and Bengio, Yoshua. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

Czarnecki, Wojciech M, Osindero, Simon, Jaderberg, Max, Swirszcz, Grzegorz, and Pascanu, Razvan. Sobolev training for neural networks. In *Advances in Neural Information Processing Systems*, pp. 4278–4287, 2017.

Eslami, SM Ali, Heess, Nicolas, Weber, Theophane, Tassa, Yuval, Szepesvari, David, Hinton, Geoffrey E, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pp. 3225–3233, 2016.

Fairbank, Michael. *Value-gradient learning*. PhD thesis, City University London, 2014.

Fairbank, Michael and Alonso, Eduardo. Value-gradient learning. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pp. 1–8. IEEE, 2012.

Figurnov, Michael, Mohamed, Shakir, and Mnih, Andriy. Implicit reparameterization gradients. *arXiv preprint arXiv:1805.08498*, 2018.

Foerster, Jakob N., Farquhar, Gregory, Afouras, Triantafyllos, Nardelli, Nantas, and Whiteson, Shimon. Counterfactual multi-agent policy gradients. *CoRR*, abs/1705.08926, 2017.

Fortunato, Meire, Azar, Mohammad Gheshlaghi, Piot, Bilal, Menick, Jacob, Osband, Ian, Graves, Alex, Mnih, Vlad, Munos, Remi, Hassabis, Demis, Pietquin, Olivier, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Fraccaro, Marco, Sønderby, Søren Kaae, Paquet, Ulrich, and Winther, Ole. Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pp. 2199–2207, 2016.

Gan, Zhe, Li, Chunyuan, Henao, Ricardo, Carlson, David E, and Carin, Lawrence. Deep temporal sigmoid belief networks for sequence modeling. In *Advances in Neural Information Processing Systems*, pp. 2467–2475, 2015.

Geiger, Dan, Verma, Thomas, and Pearl, Judea. Identifying independence in bayesian networks. *Networks*, 20(5): 507–534, 1990.

Gershman, Samuel and Goodman, Noah. Amortized inference in probabilistic reasoning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 36, 2014.

Glasserman, Paul. *Gradient estimation via perturbation analysis*. Springer Science & Business Media, 1991.

Glasserman, Paul. Smoothing complements and randomized score functions. *Annals of Operations Research*, 39 (1):41–67, 1992.

Grathwohl, Will, Choi, Dami, Wu, Yuhuai, Roeder, Geoff, and Duvenaud, David. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

Greensmith, Evan, Bartlett, Peter L, and Baxter, Jonathan. Variance reduction techniques for gradient estimates in reinforcement learning. *The Journal of Machine Learning Research*, 5:1471–1530, 2004.

Gregor, Karol, Papamakarios, George, Besse, Frederic, Buesing, Lars, and Weber, Theophane. Temporal difference variational auto-encoder. *arXiv preprint arXiv:1806.03107*, 2018.

Gu, Shixiang, Levine, Sergey, Sutskever, Ilya, and Mnih, Andriy. Muprop: Unbiased backpropagation for stochastic neural networks. *arXiv preprint arXiv:1511.05176*, 2015.

Haarnoja, Tuomas, Zhou, Aurick, Abbeel, Pieter, and Levine, Sergey. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Heess, Nicolas, Wayne, Gregory, Silver, David, Lillicrap, Tim, Erez, Tom, and Tassa, Yuval. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

Heess, Nicolas, Wayne, Greg, Tassa, Yuval, Lillicrap, Timothy, Riedmiller, Martin, and Silver, David. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Heng, Jeremy, Bishop, Adrian N, Deligiannidis, George, and Doucet, Arnaud. Controlled sequential monte carlo. *arXiv preprint arXiv:1708.08396*, 2017.

Igl, Maximilian, Zintgraf, Luisa, Le, Tuan Anh, Wood, Frank, and Whiteson, Shimon. Deep variational reinforcement learning for POMDPs. *arXiv preprint arXiv:1806.02426*, 2018.

Jaderberg, Max, Czarnecki, Wojciech Marian, Osindero, Simon, Vinyals, Oriol, Graves, Alex, Silver, David, and Kavukcuoglu, Koray. Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*, 2016.

Jang, Eric, Gu, Shixiang, and Poole, Ben. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kingma, Diederik P and Welling, Max. Auto-encoding variational Bayes. *arXiv:1312.6114*, 2013.

Kingma, Diederik P and Welling, Max. Efficient gradient-based inference through transformations between bayes nets and neural nets. *arXiv preprint arXiv:1402.0480*, 2014.

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

Kosiorek, Adam R, Kim, Hyunjik, Posner, Ingmar, and Teh, Yee Whye. Sequential attend, infer, repeat: Generative modelling of moving objects. *arXiv preprint arXiv:1806.01794*, 2018.

Krishnan, Rahul G, Shalit, Uri, and Sontag, David. Deep Kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Levine, Sergey. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Lillicrap, Timothy P, Hunt, Jonathan J, Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liu, Hao, He, Lirong, Bai, Haoli, and Xu, Zenglin. Efficient structured inference for stochastic recurrent neural networks. 2017.

Liu, Qiang and Wang, Dilin. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pp. 2378–2386, 2016.

Lowe, Ryan, Wu, Yi, Tamar, Aviv, Harb, Jean, Abbeel, Pieter, and Mordatch, Igor. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 6382–6393, 2017.

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

Meeds, Edward and Welling, Max. Gps-abc: Gaussian process surrogate approximate bayesian computation. *arXiv preprint arXiv:1401.2838*, 2014.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. *arXiv:1402.0030*, 2014.

Mnih, Andriy and Rezende, Danilo J. Variational inference for monte carlo objectives. *arXiv preprint arXiv:1602.06725*, 2016.

Moreno, Pol, Humplik, Jan, Papamakarios, George, Avila Pires, Bernardo, Buesing, Lars, Heess, Nicolas, and Weber, Theophane. Neural belief states for partially observed domains. *NeurIPS 2018 workshop on Reinforcement Learning under Partial Observability*, 2018.

Naumann, Uwe. Optimal jacobian accumulation is np-complete. *Mathematical Programming*, 112(2):427–441, 2008.

Ng, Andrew Y, Harada, Daishi, and Russell, Stuart. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

Ong, Victor MH, Nott, David J, Tran, Minh-Ngoc, Sisson, Scott A, and Drovandi, Christopher C. Variational bayes with synthetic likelihood. *Statistics and Computing*, 28 (4):971–988, 2018.

Paisley, John, Blei, David, and Jordan, Michael. Variational bayesian inference with stochastic search. *arXiv preprint arXiv:1206.6430*, 2012.

Parmas, Paavo. Total stochastic gradient algorithms and applications in reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10224–10234. 2018.

Peng, Xue Bin, Andrychowicz, Marcin, Zaremba, Wojciech, and Abbeel, Pieter. Sim-to-real transfer of robotic control with dynamics randomization. *arXiv preprint arXiv:1710.06537*, 2017.

Piché, Alexandre, Thomas, Valentin, Ibrahim, Cyril, Bengio, Yoshua, and Pal, Chris. Probabilistic planning with sequential monte carlo methods. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByetGn0cYX.

Plappert, Matthias, Houthooft, Rein, Dhariwal, Prafulla, Sidor, Szymon, Chen, Richard Y, Chen, Xi, Asfour, Tamim, Abbeel, Pieter, and Andrychowicz, Marcin. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Ranganath, Rajesh, Gerrish, Sean, and Blei, David M. Black box variational inference. *arXiv preprint arXiv:1401.0118*, 2013.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. *arXiv:1401.4082*, 2014.

Robbins, Herbert and Monro, Sutton. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pp. 102–109. Springer, 1985.

Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Schmidt, Mark, Roux, Nicolas L, and Bach, Francis R. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in neural information processing systems*, pp. 1458–1466, 2011.

Schulman, John, Heess, Nicolas, Weber, Theophane, and Abbeel, Pieter. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pp. 3528–3536, 2015a.

Schulman, John, Moritz, Philipp, Levine, Sergey, Jordan, Michael, and Abbeel, Pieter. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

Silver, David, Lever, Guy, Heess, Nicolas, Degris, Thomas, Wierstra, Daan, and Riedmiller, Martin. Deterministic policy gradient algorithms. In *ICML*, 2014.

Sutton, Richard S, McAllester, David A, Singh, Satinder P, and Mansour, Yishay. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Tucker, George, Mnih, Andriy, Maddison, Chris J, Lawson, John, and Sohl-Dickstein, Jascha. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pp. 2627–2636, 2017.

Tucker, George, Bhupatiraju, Surya, Gu, Shixiang, Turner, Richard E, Ghahramani, Zoubin, and Levine, Sergey. The mirage of action-dependent baselines in reinforcement learning. *arXiv preprint arXiv:1802.10031*, 2018.

Weber, Theophane, Heess, Nicolas, Eslami, Ali, Schulman, John, Wingate, David, and Silver, David. Reinforced variational inference. In *Advances in Neural Information Processing Systems (NIPS) Workshops*, 2015.

Werbos, Paul J. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pp. 762–770. Springer, 1982.

Wierstra, Daan and Schmidhuber, Jürgen. Policy gradient critics. In *European Conference on Machine Learning*, pp. 466–477. Springer, 2007.

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Wingate, David and Weber, Theophane. Automated variational inference in probabilistic programming. *arXiv preprint arXiv:1301.1299*, 2013.

Wu, Cathy, Rajeswaran, Aravind, Duan, Yan, Kumar, Vikash, Bayen, Alexandre M., Kakade, Sham, Mordatch, Igor, and Abbeel, Pieter. Variance reduction for policy gradient with action-dependent factorized baselines. *CoRR*, abs/1803.07246, 2018. URL http://arxiv.org/abs/1803.07246.