Multiplicative Normalizing flows for Reinforcement Learning

Salma Tarmoun
Master MVA
Université Paris Saclay
salma.tarmoun@ens-paris-saclay.fr

Contents

1 Introduction

Normalizing flows is a simple yet powerful technique to transform distributions that has been successfully adopted in a range of areas in recent years. Its success comes from its vast number of applications from density estimation to variational inference and more recently reinforcement learning. Throughout this review, we will see that the historically preferred Gaussian distribution, while justifiably successful due to its simplicity, can be limiting in various cases. More complex distributions have been considered throughout the years but this same complexity makes them less practical. Normalizing flows suffered from this exact problem until recent works introduced new efficient architectures which unleashed its full potential.

In this review, we will focus on the application of normalizing flows in the reinforcement learning field based on our reading of articles [18], [17], [3] and [19]. In order to understand the concepts introduced in the RL papers, we will provide an overview of the previous works on normalizing flows in the areas of density estimation and variational inference. The report is thus organized as follows. In section 2, we present the mathematical setting of the normalizing flows approach with a special mention of its applications in density estimation and variational inference. While this may seem outside the scope of RL, it sheds a light on the motivations behind the normalizing flows technique and the introduction of variational inference will be useful in understanding [19]. This is followed by a non-exhaustive overview of the types of normalizing flows. In section 3, we focus on the applications in reinforcement learning and we provide a review of the 4 articles. The review of each paper is neither exhaustive nor comprehensive as each one contains rich concepts by itself and for which normalizing flows is mostly an addition and a powerful tool that improves those concepts. We will however highlight the strengths and motivations behind normalizing flows as well as the implementation choices in each article.

2 Normalizing flows

A *normalizing flow* as described by Rezende and Mohammed in [11] is "the transformation of a probability density through a sequence of invertible mapping". The result of this transformation is also a valid probability distribution.

The idea behind this transformation is the basic change of variable formula. Given an observed data variable $x \in X$ with probability density $p_X(x)$ and a smooth invertible mapping $f : X \to Y$ that transforms $x \sim p_X$ to a sample $y \sim p_Y$, we have the following.

$$p_X(x) = p_Y\big(y\big)\Big|\det\Big(\frac{\partial f(x)}{\partial x^T}\Big)\Big| \tag{1}$$

which can also be written as

$$p_Y\big(y\big) = p_X(x)\Big|\det\Big(\frac{\partial f^{-1}(y)}{\partial y^T}\Big)\Big| = p_X(x)\Big|\det\Big(\frac{\partial f(y)}{\partial y^T}\Big)\Big|^{-1} \tag{2}$$

By successively applying a chain of K transformations f_k to a random variable x_0 of density p_0, we obtain variable x_K of density p_K :

$$x_K = f_K \circ \cdots \circ f_2 \circ f_1(x_0) \tag{3}$$

$$\ln p_K(x_K) = \ln p_0(x_0) - \sum_{k=1}^{K} \ln\Big|\det\frac{\partial f_k}{\partial x_{k-1}}\Big| \tag{4}$$

The path traversed by the random variables $x_k = f_K(x_{k-1})$ starting from the initial distribution $p_0(x_0)$ is called the *flow* and the path formed by the successive distributions p_k is called a *normalizing flow*. We note that the previous definition was given for a finite sequence of transformations but the concept of flows can be extended to the infinite case and define what is referred to as *infinitesimal flows*. In that case, we consider that the density evolves through time and the dynamics are instead described by a partial differential equation.

2.1 Normalizing flows for density estimation

The normalizing flows technique was first proposed by Tabak et al. in [16] and [15] in the context of non-parametric density estimation. The idea behind it was to transform the original data $\mathbf{x} \sim p_X$ with a sequence of maps $(\phi_t)_t$ into a new set of variables y with known probability density μ. The target distribution μ is chosen to be an isotropic Gaussian because of its desired properties hence the name normalizing. The goal is then to maximize the log-likelihood of the data using the expression in equation 4. The initial distribution can be retrieved by inverting the transformations once the target distribution has been estimated.

2.2 Normalizing flows for variational inference

Variational inference [5] is a method used in Bayesian statistics to approximate posterior densities through optimization. Let's consider a probabilistic model with observations $\mathbf{x}$, continuous latent variables $\mathbf{z}$ and model parameters θ. We are interested in learning the parameters θ which maximize the likelihood of observations of the latent variable model $p_\theta(\mathbf{x}, \mathbf{z})$. Computing the likelihood $p_\theta(\mathbf{x})$ involves marginalizing over the latent variables. Unfortunately, this integration is generally intractable. Variational inference consists of introducing a variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$ with learnable parameters ϕ and maximizing the following Evidence Lower Bound (ELBO):

$$\text{ELBO} = \mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \leq \log p_\theta(\mathbf{x}) \tag{5}$$

A central problem in variational inference (VI) is the choice of the family of distributions q_ϕ. In fact, the complexity of the family determines the complexity of the optimization. Most research in VI has been focusing on the mean-field variational family which assumes that the latent variables are mutually independent. Moreover, the simplest and most widely used choice of distribution for each latent component is the gaussian distribution. The problem with such simple distributions is that the resulting estimated parameters can be highly biased as the gaussian distribution isn't necessarily a good representation for the underlying data. In this context, normalizing flows were adopted to perform the opposite of their original goal seen previously. In variational inference, they are used to transform a simple base distribution into a more complex one that is used to approximate the posterior, thus allowing a richer and more flexible family of posterior distributions. We will see in section 3 that reinforcement learning makes the same use of normalizing flows to define richer and more expressive distributions.

2.3 Types of normalizing flows

A central problem in using the normalizing flows technique is the choice of building blocks to use as maps. Tabak and Turner [15] defined criteria for choosing these elementary maps which included robustness, computational simplicity and good behaviour in high dimensional settings.

In fact, in order to use equation 1, we must be able to efficiently compute the Jacobian determinant. The complexity of such computation is typically $O(d^3)$ where d is the dimension of the domain $(f : \mathbb{R}^d \to \mathbb{R}^d)$. We must also be able to invert the maps in order to retrieve the initial distribution in the case of density estimation. All these computations are very expensive in general and without a good choice of invertible maps, the method of normalizing flows can be impractical.

In this section, we will present the different types of normalizing flows proposed in literature and discuss the advantages and drawbacks of each type.

2.3.1 Simple flows

Rezende and Mohamed [11] introduced two classes of transformations: planar flows and radial flows in the context of stochastic variational inference. Their Jacobians derived in [11] are tractable and were shown to be effective for problems in low dimensional spaces.

Planar flows: This family of transformations takes the form:

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + b) \tag{6}$$

where h is a smooth non-linearity and $\lambda = \{\mathbf{w} \in \mathbb{R}^D, \mathbf{u} \in \mathbb{R}^D, b \in \mathbb{R}\}$. They are called planar flows because they apply a series of contractions and expansions in the direction perpendicular to the hyperplane $\mathbf{w}^T\mathbf{z} + b = 0$ as can be seen in figure 1 presented in [11].

Radial flows: This family of transformations takes the form:

$$f(\mathbf{z}) = \mathbf{z} + \beta h(\alpha, r)(\mathbf{z} - \mathbf{z}_0) \tag{7}$$

where $\mathbf{z}_0$ is a reference point, $r = |\mathbf{z} - \mathbf{z}_0|$, $h(\alpha, r) = \frac{1}{\alpha + r}$ and the parameters of the map are $\lambda = \{\mathbf{w} \in \mathbb{R}^D, \mathbf{u} \in \mathbb{R}^D, b \in \mathbb{R}\}$. They are called radial flows because they apply a series of contractions and expansions around the reference point $\mathbf{z}_0$ (figure 1).

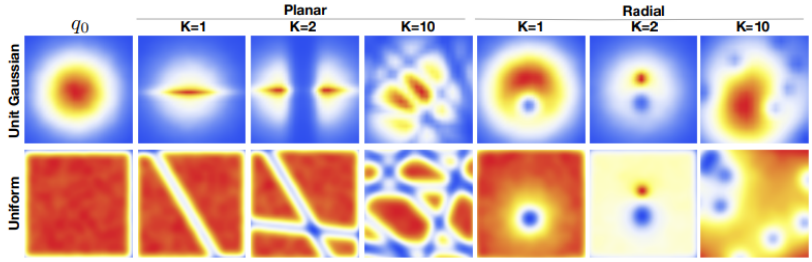


Figure 1: Planar and radial normalizing flows

Experiments have shown that planar flows are effective for small problems but require a long chain of transformations in high dimensional settings. Kingma et al. [6] interpret each transformation as a MLP with a single unit hidden layer and a skip connection. They argue that a single hidden layer is not expressive enough which would explain the need for a high number of transformations. To overcome the single-hidden-unit bottleneck of these simple flows, Berg et al. [20] recently proposed a new family of flows called Sylvester Normalizing Flows which generalizes the planar flows.

2.3.2 Modern normalizing flows

A number of invertible transformations with tractable Jacobians have been proposed in recent years. In the context of density estimation, Dinh et al. [2] proposed the Real-valued non-volume-perserving (Real NVP) estimator which was an improvement of their earlier work on Non-linear independent component estimation NICE [1]. Concurrently, more expressive flows based on autoregressive models have been proposed such as Inverse Autoregressive flows (IAF) [6] and Masked Autoregressive flows (MAF) [10]. IAF and MAF can both be seen as generalizations of Real NVP but they have their own computational drawbacks. For MAF, sampling can't be parallelized which makes it slow, while IAF is slower to train. In contrast, Real NVP can both sample and estimate the density with one forward pass. This last class of transformations was used in three of the four Reinforcement Learning papers we will be reviewing and will therefore be our main focus. We will also briefly present the autoregressive framework as [19] implements the IAF transformations for its variational inference problem.

2.3.2.1 Real NVP

Real NVP [2] is based on a careful design of bijective maps which are simple to invert and possess tractable Jacobians. It exploits the fact that the determinant of a triangular matrix is simply the product of its diagonal terms. The approach defines bijective functions which update only part of the input vector and refers to these functions as affine coupling layers. Given an input vector $x \in \mathbb{R}^D$ and $d < D$, the output y of such functions follows the equations:

$$y_{1:d} = x_{1:d} \tag{8}$$

$$y_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d}) \tag{9}$$

where s and t stand for scale and translation, and are functions from $\mathbb{R}^d \to \mathbb{R}^{D-d}$, and $\odot$ is the element-wise product.

The Jacobian of this function is:

$$\frac{\partial y}{x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d-1:D}}{\partial x_{1:d}^T} & \text{diag}(\exp[s(x_{1:d})]) \end{bmatrix} \tag{10}$$

Its determinant can thus be computed efficiently and is $|\frac{\partial y}{x^T}| = \exp[\sum_j s(x_{1:d})_j]$ where $s(x_{1:d})_j$ refers to the j-th component of $s(x_{1:d})$.

It can be shown that computing the inverse of the coupling layers does not require computing the inverse of s or t. These functions can therefore be arbitrarily complex. In general, s and t are chosen as neural networks. Furthermore, for each coupling layer, we permute the input of x so as to couple different components across layers.

Recently, Kingma and Dhariwal [7] introduced the Glow model which extends Real NVP by replacing the reverse permutation operation with invertible 1x1 convolutions and performs better on density estimation tasks.

2.3.2.2 Autoregressive Flows

Autoregressive flows can be seen as a generalization of Real NVP as they introduce more complex dependencies between the dimensions. They are also carefully designed to obtain a triangular jacobian by making each dimension depend only on the previous dimensions. Let $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}_+^d$. We introduce the random variable $z \in \mathbb{R}^d$ which is transformed by the normalizing flows into random variable y using the following equations.
For Masked Autoregressive Flows :

$$y_1 = \mu_1 + \sigma_1 z_1 \tag{11}$$
$$y_i = \mu(y_{1:i-1}) + \sigma(y_{1:i-1}) z_i \tag{12}$$

We can see that this transformation is sequential and can't be parallelised which explains why sampling with MAF can be slow. On the other hand, density estimation can be done with a single forward pass since the inverse of the previous transformation takes the form :

$$z_i = \frac{y_i - \mu(y_{1:i-1})}{\sigma(y_{1:i-1})} \tag{13}$$

For Inverse Autoregressive Flow, we face the opposite problem since it transforms the variable z using an inverse autoregressive mapping.

$$y_i = z_i \sigma(z_{1:i-1}) + \mu(y_{1:i-1}) \tag{14}$$

We notice that this time, sampling only requires one forward pass while density estimation using the inverse transformation will be sequential. This makes it more adapted to variational inference.

3 Normalizing flows in Reinforcement Learning

Normalizing flows have caught the attention of researchers in the reinforcement learning field as well and recent works have used it to improve upon several reinforcement learning methods. In fact, as we have seen in the previous section, the normalizing flows method is extremely powerful in its ability to represent complex distributions. This is particularly useful in reinforcement learning whenever we want to introduce probability distributions to represent policies ([18], [17], [3]) or value functions [19].

3.1 Normalizing flows for more expressive policies

Most reinforcement learning algorithms today implement some form of stochastic policies. One obvious example of this stochasticity is perturbing the greedy action with some random noise for exploration ie. ϵ-greedy or Boltzmann exploration. In this case, the final optimal policy can nonetheless be deterministic. But learning a stochastic policy can also be desirable. In fact, as explained by

Harnooja et al. [4], stochastic policies are preferred in multi-modal reward environments, they are also more robust in the face of adversarial perturbations. Moreover, several algorithms, especially policy gradient based approaches, are designed to learn a stochastic parametrized policy. In most cases of countinuous control, the policy is assumed to be following a unimodal gaussian distribution but this assumption can be very limiting due to its inability to model more complex policies and behaviours. In this context, normalizing flows can surpass this limitation by enriching the family of distributions.

3.1.1 Implicit policy for reinforcement learning

Tang and Agrawal were one of the first to investigate the effect of using normalizing flows policies in reinforcement learning. Their paper [18] shows that entropy regularization combined with a rich class of policies like normalizing flows can have desired properties such as robustness and multi-modality.

The paper addresses the problem of convergence to locally optimal solutions in policy gradient methods. In recent years, several solutions have been proposed to prevent this issue. One interesting idea to achieve this has been to take into account the entropy of the policy. This idea finds its origin in the maximum entropy principle which aims to find the probability distribution with the highest entropy while satisfying the observed statistics. Intuitively, this means that given several probability distributions which satisfy our current constraints, we should choose the one which makes the least assumptions about the unknowns. We can use this in reinforcement learning to say that the best policy is one which, with the current state of knowledge, optimizes the cumulative reward without making any additional assumptions, such as favoring one action over another when the previous observations did not warrant such behaviour. In this context, maximum entropy objective was introduced in a number of prior works by adding an entropy term to the original cumulative reward objective. The new goal is thus to to find the policy which achieves a high reward while being as random as possible. The problem with maximum entropy objective is that it considers the entropy of the distribution over entire trajectories thus making it hard to implement in a simple and scalable way using on-policy methods. Another variant is entropy regularization which adds the entropy term directly to the gradient update. This makes it easy to implement but it greedily optimizes the entropy at each time step. With a lack of global objective, Tang and Agrawal argue that entropy regularization is hard to analyze. Nonetheless, they go on to show that "entropy regularization maximizes a lower bound of maximum entropy objective when consecutive policy iterates are close". According to their paper, this result implies that entropy regularization captures some of the same effects that maximum objective is intended for even though the optimal solutions of both methods are different.

The other important aspect of maximum entropy objective is the choice of the family of policies. In policy gradient methods, a conventional choice has been the unimodal gaussian policy centered at the maximal Q-value and extending to the neighboring actions to provide noise for exploration. But this can be limiting in the case of multi-modal objectives. And more generally, we want to avoid making such assumptions in the maximum entropy framework. A richer and more expressive class of distributions is therefore highly desirable. This idea may seem obvious but the difficulty arises when we think about what constraints such distribution must satisfy. In order to use a policy gradient method, we need the policy to be differentiable to compute the gradient update and it must be easy to sample from. Interestingly, we saw in section 2 that recent research on the use of normalizing flows aimed to find a class of functions which satisfies similar constraints. In particular, Real NVP is a good candidate for such task, which explains why it was chosen in the implementations for this paper.

Now that we have an expressive family of policies, we can look at how our entropy regularized models behave and if they indeed capture the desired properties of maximum entropy objective. The experiments conducted compare the performance of the normalizing flows policy (NFP) implemented with Proximal Policy Optimization (PPO) [13] with several other gradient based methods (TRPO [12] , DDPG [14], SQL [4], PPO) implemented with a unimodal policy. On locomotion tasks, the NFP outperforms DDPG, TRPO and SQL on almost all tasks but it fails to perform better than PPO with a unimodal policy. Since NFP was implemented with PPO, it's hard to say if its success agaisnt other algorithms is due to the normalizing flows policies or the strength of the PPO algorithm especially when the performance under a unimodal policy is better when used with PPO. It might have been more interesting to study the effects of normalizing flows on each algo-

rithm instead of combining it with PPO in all experiments. There might be other works in the future that investigate this aspect and the second paper that we will review already addresses this for the TRPO algorithm. We can still conclude from these experiments that NFP with PPO does provide competitive performance but it might be harder to train than a unimodal policy.

The true strengths of normalizing flow policies as we explained earlier are robustness and multi-modality and the paper illustrates them with a few experiments. When considering robustness to noisy observations we find that, when compared with unimodal PPO, NFP is able to learn faster on a number of tasks which shows its ability to navigate more complex environments and reward landscapes. As for the ability to learn multi-modal policies, experiments show that normalizing flows policies are extremely flexible and able to commit to multiple goals while a unimodal gaussian is forced to commit to only one.

3.1.2 Boosting Trust Region Policy Optimization by Normalizing Flows Policy

In a similar vein to the previous paper, Tang and Agrawal [17] extended their work on the possible applications of normalizing flows. We remain in the same framework of improving policy gradient methods with a better choice of policy distributions and we take a special interest in the Trust Region Policy Optimization algorithm (TRPO) [12]. As the name suggests, TRPO defines a trust region for the policy updates to avoid taking large steps which significantly degrade the policy and the progress of the training. This trust region is defined by introducing a KL divergence constraint on the new policy such that it must verify :

$$\mathbb{E}_s[\mathbb{KL}(\pi_\theta(.|s)||\pi_{\theta_{new}}(.|s)] < \epsilon$$

where π_θ is the policy distribution parameterized by θ.

Tang and Agrawal argue that such constraint can "put a very stringent restriction on the new policy, making it either hard to bypass locally optimal solutions or slow down the learning process". The paper illustrates this with several examples comparing the factorized gaussian policy and the normalizing flows policy under KL constraints. They show that normalizing policies have larger variance and by extent a larger effective support on the sample space which allows for more efficient exploration. Tang and Agrawal also highlight two strengths of the normalizing flows policy: its ability to capture multi-modal behaviour as seen in the previous paper and its ability to learn correlated actions. In their experiments, normalizing flows are able to better take into account such complexities compared to both the factorized gaussian and the Gaussian Mixture Modal (GMM) as illustrated by figure 2 from their paper. On locomotion benchmarks, they also show that normalizing flows outperform the factorized gaussian policy as well as GMM on a wide range of tasks. The experiments highlight how the gaussian policy can get stuck in a locally optimal solution while the normalizing flows policy is able to bypass this and continue to improve consistently especially in tasks with highly complex dynamics.

Just like in their previous work, Tang and Agrawal chose the real NVP [2] implementation of normalizing flows. Both papers use the same architecture for the normalizing flows policy which we will briefly examine. The main ideas have already been detailed in section 2 when we presented Real NVP but let's look at how they are incorporated from a reinforcement learning point of view.

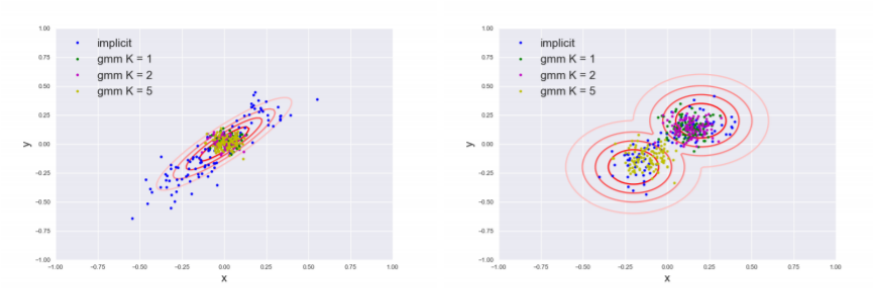
We start by representing the stochastic policy in the general case with the function f_θ where θ incorporates state information s and ϵ is an independent source noise with a simple probability distribution ρ_0. In state s, action a is sampled as follows :

$$a = f_\theta(s, \epsilon), \epsilon \sim \rho_0(.)$$

For a normalizing flows policy, we introduce a series of invertible mappings $g_{\theta_i}(.), 1 \leq i \leq K$ each with parameter θ_i to transform source noise ρ_0 to a more complex distribution. These mappings are non-linear neural transformations as seen for Real NVP in section 2. As a result, we obtain the target sample x such that :

$$x = g_{\theta_K} \circ g_{\theta_{K-1}} \circ \cdots \circ g_{\theta_1}(\epsilon)$$

In order to combine state information, another neural network L_{θ_s} is introduced to embed state s. The output vector $L_{\theta_s}(s)$ can be inserted between any two layers and a choice was made to introduce



(a) Correlated Actions (b) Bimodal Rewards

Figure 2: Expressiveness of normalizing flows policy: (a) Bandit problem with reward $r(a) = -a^T \Sigma^{-1} a$. The maximum entropy optimal policy is a Gaussian distribution with Σ as its covariance (red contours). normalizing flows policy (blue) can capture such covariance while Gaussian cannot (green). (b) Bandit problem with multimodal reward (red contours the reward landscape). normalizing flows policy can capture the multimodality (blue) while Gaussian cannot (green).

it after the first transformation such that :

$$a = g_{\theta_K} \circ g_{\theta_{K-1}} \circ \cdots \circ (L_{\theta_s}(s) + g_{\theta_1}(\epsilon))$$

The paper implements a chain of $K = 4$ transformations with 4-layer neural networks.

3.1.3 Latent Space Policies for Hierarchical Reinforcement Learning

In [3], Harnooja et al. delve into the possibility of defining a hierarchical structure in reinforcement learning settings with each hierarchy capturing a level of abstraction or representation in a similar fashion to computer vision. The idea is that incorporating deep hierarchical representations may "enable reasoning and decision making at different levels of abstract". The paper explores a hierarchical framework where each layer corresponds to a policy with internal latent variables. The problem is shown to be equivalent to an inference problem for a particular probabilistic graphical model. Moreover, it optimizes the maximum entropy objective by maximizing both the cumulative reward and the entropy of the policy.

The hierarchical RL framework proposed in the paper defines latent variable policies as action distributions $\pi(a_t|s_t, h_t)$ conditioned on two factors: the state s_t and a latent variable h_t with prior $p(h_t)$. By integrating this base policy into the transition structure of the MDP, we are able to define a higher-level of actions h_t through the process of marginalization out the actions a_t. This process can be repeated multiple times to introduce several policy layers that we refer to as sub-policies.

Normalizing flows are introduced when considering the choice of such sub-policies. These distributions must verify three properties. They need to be tractable in order to efficiently marginalize out the latent variables. They need to be expressive to avoid "suppressing the information flow from the higher levels to the lower levels". Lastly, they need to be deterministic since their output represents a part of the environment for higher levels. The strength of normalizing flows is highlighted since they satisfy all three constraints. To summarize, the normalizing flows framework allows us to define a hierarchical policy $f = f_0 \circ f_1 \circ \cdots \circ f_{K-1}$ where K is the level of hierarchy and for each layer i we have $h_t^{(i)} = f_i(h_t^{(i+1)}; s_t)$ with f_i an invertible transformation.

As with previous papers, the transformations are implemented following the real NVP architecture which provides an efficient and stable way to compute the Jacobians and to invert the transformations. The experiments conducted in the paper focus on evaluating the latent space hierarchical framework so we can't isolate the effect of normalizing flows but the paper contains this interesting statement : "experimental evaluation shows that [the normalizing flows] approach can attain state-of-the-art results on a number of continuous control benchmark tasks by itself, independently of its applicability to hierarchical RL".

3.2 Randomized Value Functions via Multiplicative Normalizing Flows

After examining the role that normalizing flows can play in defining more expressive tractable policies with highly desirable properties such as robustness and multi-modality, we now dive into a different application of this approach. This time, we will cast our reinforcement learning problem into a variational inference framework where the posteriors are value functions which we will represent using normalizing flows.

The paper addresses the problem of efficient exploration in high dimensional environments and builds on the work of Osband et al. [9] who introduced randomized value functions. Their idea was to maintain a posterior distribution over value functions instead of a point estimate in order to favor the exploration of uncertain values. This earlier work used Bayesian linear regression as a function approximator to learn the value functions and the goal of this paper is to improve upon it by using deep neural networks as approximators. In order for the problem to be tractable, we must introduce approximations such as the ones used in variational inference. As discussed in section 2, a major limiting factor in our variational inference approach is the mean field approximation and the use of fully factorized Gaussians to represent the posteriors over network parameters w. In this context, normalizing flows are used to transform an initial simple distribution over latent variable z_0, for example a factorized gaussian $q(z_0) = \prod_j \mathcal{N}(\mu_{z_{0,j}}, \sigma^2_{z_{0,j}})$ into a more complex distribution by applying a sequence of K transformations f_k. The paper also follows the already studied approach of multiplicative normalizing flows proposed by Louizos and Welling [8] for variational inference where the random variable z is introduced as a multiplicative noise modeled via normalizing flows. As a result, the posterior over weights is defined in the following way.

$$q_\phi(w|z) = \prod_{i=1}^{D_h} \prod_{j=1}^{D_w} \prod_{k=1}^{D_f} \mathcal{N}(z_k \mu_{ijk}, \sigma^2_{ijk}) \tag{15}$$

where D_h, D_w, D_f are the height, width and number of filters and z is sampled according to the resulting normalizing flows distribution.
An auxiliary posterior distribution parameterized by inverse normalizing flows is also introduced to further lower bound the intractable KL divergence term.

Finally, for the choice of normalizing flows, Touati et al. use the Inverse Autoregressive Flows version. When combining it with Deep Q-networks, they use normalizing flows of length 2 with 50 hidden units then later with Deep Deterministic Policy Gradient [14], they use flows of length 1 with 16 hidden units. They conduct experiments to evaluate the performance of DQN and DDPG combined with MNF on several benchmarks and compare it with other algorithms which use Bayesian parameter updates or ϵ-greedy/noisy exploration. They are able to show that MNF-DQN learns faster on toy tasks such as n-Chain and Mountain Car. It also has a competitive performance on Atari games but there is not much difference between MNF-DQN and the best performing baseline. What we can conclude however is that its actual strength might be its ability to adapt and consistently perform well on all types of environments while other strategies fail in certain situations. This can be attributed to the expressiveness of normalizing flows and their ability to learn complex posteriors.

4 Conclusion

In this review, we have presented the normalizing flows framework and given an overview of its recent applications in RL. We have seen that normalizing flows can have desirable properties such as robustness and expressiveness which makes them an interesting choice for policy or value function distributions. But this doesn't mean that normalizing flows should become the default approach, as the Gaussian distribution still remains powerful in a wide range of cases. These advantages must also be weighed against the complexity introduced in training. We note that the normalizing flows technique is still relatively new to the field of reinforcement learning and would benefit from further work on the effects of the resulting distributions. Furthermore, we noticed that researchers have so far focused on early architectures like Real NVP but other advanced designs have proven to be more powerful in other areas and might be worth exploring in the context of reinforcement learning.

References

[1] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. *CoRR*, abs/1410.8516, 2014.

[2] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *CoRR*, abs/1605.08803, 2016.

[3] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. *CoRR*, abs/1804.02808, 2018.

[4] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *CoRR*, abs/1702.08165, 2017.

[5] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233, November 1999.

[6] Diederik P. Kingma, Tim Salimans, and Max Welling. Improving variational inference with inverse autoregressive flow. *CoRR*, abs/1606.04934, 2016.

[7] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems 31*, pages 10236–10245. Curran Associates, Inc., 2018.

[8] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *ICML*, 2017.

[9] Ian Osband, Daniel Russo, Zheng Wen, and Benjamin Van Roy. Deep exploration via randomized value functions. *CoRR*, abs/1703.07608, 2017.

[10] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems 30*, pages 2338–2347. Curran Associates, Inc., 2017.

[11] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

[12] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[14] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, and Schrittwieser et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[15] Esteban Tabak and Cristina V. Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2 2013.

[16] Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Commun. Math. Sci.*, 8(1):217–233, 03 2010.

[17] Yunhao Tang and Shipra Agrawal. Boosting trust region policy optimization by normalizing flows policy. *CoRR*, abs/1809.10326, 2018.

[18] Yunhao Tang and Shipra Agrawal. Implicit policy for reinforcement learning. *CoRR*, abs/1806.06798, 2018.

[19] Ahmed Touati, Harsh Satija, Joshua Romoff, Joelle Pineau, and Pascal Vincent. Randomized value functions via multiplicative normalizing flows. *CoRR*, abs/1806.02315, 2018.

[20] Rianne van den Berg, Leonard Hasenclever, Jakub M. Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. In *UAI*, pages 393–402. AUAI Press, 2018.