AutoNF: Automated Architecture Optimization of Normalizing Flows with Unconstrained Continuous Relaxation Admitting Optimal Discrete Solution

Yu Wang[1], Ján Drgoňa[2], Jiaxin Zhang[3], Karthik Somayaji Nanjangud Suryanarayana[1], Malachi Schram[4], Frank Liu[5], Peng Li[1]

[1] University of California, Santa Barbara
[2] Pacific Northwest National Laboratory
[3] Intuit AI Research
[4] Thomas Jefferson National Accelerator Facility
[5] Oak Ridge National Laboratory
{yu95, karthi, lip}@ucsb.edu, jan.drgona@pnnl.gov, jxzhangai@gmail.com, schram@jlab.org, liufy@ornl.gov

Abstract

Normalizing flows (NF) build upon invertible neural networks and have wide applications in probabilistic modeling. Currently, building a powerful yet computationally efficient flow model relies on empirical fine-tuning over a large design space. While introducing neural architecture search (NAS) to NF is desirable, the invertibility constraint of NF brings new challenges to existing NAS methods whose application is limited to unstructured neural networks. Developing efficient NAS methods specifically for NF remains an open problem. We present AutoNF, the first automated NF architectural optimization framework. First, we present a new mixture distribution formulation that allows efficient differentiable architecture search of flow models without violating the invertibility constraint. Second, under the new formulation, we convert the original NP-hard combinatorial NF architectural optimization problem to an unconstrained continuous relaxation admitting the discrete optimal architectural solution, circumventing the loss of optimality due to binarization in architectural optimization. We evaluate AutoNF with various density estimation datasets and show its superior performance-cost trade-offs over a set of existing hand-crafted baselines.

Introduction

Normalizing flows (NF) are invertible neural networks that build upon bijections with tractable Jacobians. Flow models are promising tools since they allow for exact density estimation and sampling from complex distributions. As important kernels, NF have found applications in diverse areas such as generative models and reinforcement learning.

Utilizing NF requires careful balancing between their performance and computational cost since the optimal flow model can vary dramatically in different applications. In generative models (Dinh, Krueger, and Bengio 2015; Kingma and Dhariwal 2018), flows with the fast forward pass are preferable since forward transformations are required for every sample from the base distribution. For density estimation (Papamakarios, Pavlakou, and Murray 2017; Rippel and Adams 2013), flows with a cheap inverse prevail. For applications where a flow model is utilized as a co-trained kernel (Mazoure et al. 2020), the performance-cost trade-off is even more critical.

Unfortunately, although a variety of bijections have been proposed to build better flow models, no single bijection is optimal in both performance and computational cost. Highly expressive flows come at the cost of increased computational overhead in either the forward or the inverse pass (Durkan et al. 2019), or losing analytical invertibility (Behrmann et al. 2019); flows which are fast to compute are unable to model rich distributions and hence limited to simple applications (Rezende and Mohamed 2015).

Despite the large design space, presently flow models nowadays are hand-crafted with intensive manual fine-tuning. Hence, a systematic automated approach for optimizing NF models is highly desirable. While efficient differentiable neural architectural search (D-NAS) (Liu, Simonyan, and Yang 2019) has been developed for the optimization of unstructured neural networks, these existing methods are inapplicable to NF since they have no built-in mechanism to maintain the invertibility of the NF model under optimization; loss of invertibility in the optimization process is catastrophic since a model that is not invertible does not correspond to a properly defined normalizing flow. Furthermore, in the context of unstructured neural networks, the optimality of differential NAS may be severely comprised by binarizing the optimized relaxed continuous solution in order to obtain a legal discrete architectural solution. When unaddressed, the loss of optimality due to binarization is also expected to emerge as differentiable NAS is adapted for NF architectural optimization. The aforementioned two challenges have made efficient automated architectural optimization of NF an open problem.

To this end, we propose AutoNF, the first method that enables differentiable architecture optimization while addressing the above challenges. Our contributions are two folds:

- We propose a novel mixture distribution formulation to allow the application of efficient differentiable architectural search to the structural NF optimization while guaranteeing invertibility of the optimized flow model;
- We develop rigorous theoretical results that convert the original NP-hard combinatorial NF architectural optimization problem to an unconstrained continuous relax-

ation admitting the discrete optimal architectural solution, circumventing the loss of optimality due to binarization in architectural optimization.

Related Works

There is no prior work on automated normalizing flow architectural optimization. We provide a brief review of manually-design normalizing flows and differential neural architecture search for non-structured neural networks.

Normalizing Flows (NF): Flow models can be classified into finite multi-layer flows and continuous flow based on neural ODEs (Grathwohl et al. 2019). The finite flow family includes flows based on contractive mapping (Behrmann et al. 2019) and elemental-wise transformation (Papamakarios, Pavlakou, and Murray 2017; Kingma and Dhariwal 2018). In the latter, autoregressive flows and coupling layers are two major types, and extensive work has been proposed to improve the expressive power of both flow models. In (Huang et al. 2018), the dimension-wise scalar transformation is implemented by a sigmoid neural network, which increases the expressive power at the cost of losing analytical invertibility. In (Durkan et al. 2019), piecewise splines are used as drop-in the replacement of affine or additive transformations (Dinh, Krueger, and Bengio 2015; Dinh, Sohl-Dickstein, and Bengio 2017), which represents the current state-of-the-art (SOTA) flow model with analytical invertibility. Many recent research efforts have been devoted to increasing expressive power, albeit at the cost of more complex and expensive transformations. Nevertheless, there has been no quantitative trade-off analysis between the performance and cost among different flows.

Differentiable Neural Architecture Search (D-NAS) for Non-Structured Neural Networks: Neural architecture search (NAS) has been successful in discovering novel architectures with results better than manual designs. However, a large body of existing techniques, such as reinforcement learning (Zoph and Le 2017), genetic algorithm (Real et al. 2017; Suganuma, Ozay, and Okatani 2018; Liu et al. 2018), Monte Carlo tree search (Negrinho and Gordon 2017) or Bayesian optimization (Kandasamy et al. 2018), suffer from low efficiency in discrete optimization over a large search space. Differentiable neural architecture search (D-NAS) (Liu, Simonyan, and Yang 2019; Xie et al. 2019) allows differentiable optimization by continuous relaxation of the discrete search space and has shown superior efficiency over its counterparts. However, the application of D-NAS is limited to non-structured neural networks and the current D-NAS techniques cannot be immediately applied to NF.

Preliminaries

Normalizing Flows

Normalizing flows model complex distributions by optimizing bijections between two random variables. Given the output and input random variables $\boldsymbol{x}$ and $\boldsymbol{u}$, and their distributions $p_{\boldsymbol{x}}(\boldsymbol{x})$ and $p_{\boldsymbol{u}}(\boldsymbol{u})$, if there is a bijection T_θ that transforms $\boldsymbol{x}$ to $\boldsymbol{u}$, i.e., $\boldsymbol{u} = T_{\boldsymbol{\theta}}(\boldsymbol{x})$ [1], then the density function of $\boldsymbol{x}$ can be represented by:

$$p_{\boldsymbol{x}}(\boldsymbol{x}) = p_{\boldsymbol{u}}(T_{\boldsymbol{\theta}}(\boldsymbol{x})) \cdot |\det \boldsymbol{J}_{T_{\boldsymbol{\theta}}}(\boldsymbol{x})| \tag{1}$$

where $\det \boldsymbol{J}_{T_{\boldsymbol{\theta}}}(\boldsymbol{x})$ is the Jacobian determinant of the inverse transformation $T_{\boldsymbol{\theta}}$.

The expressive power of flow models can be improved by stacking multiple layers of bijections. For an n-layer flow model where the output and input random variables are $\boldsymbol{x_n}$ [2] and $\boldsymbol{x_0}$, respectively, n transformations $(T_1, T_2, ...T_n)$ can be applied sequentially such that $\boldsymbol{x_0} = T_1 \circ T_2 ... \circ T_n(\boldsymbol{x_n})$. The density function after i_{th} ($i \in [1, n]$) transformation is:

$$p_{\boldsymbol{x_i}}(\boldsymbol{x_i}) = p_{\boldsymbol{x_{i-1}}}(T_i(\boldsymbol{x_i})) \cdot |\det \boldsymbol{J}_{T_i}(\boldsymbol{x_i})| \tag{2}$$

Applying eq.(2) to each layer, we can obtain the density of an n-layer flow model which we denote by: $p^{\text{nf}}(\boldsymbol{x}; \boldsymbol{\theta}) = p_{T_1 ... T_n}(\boldsymbol{x}; \boldsymbol{\theta})$ in eq.(3), where $\boldsymbol{\theta}$ represents the parameters of the transformations in all layers.

$$p^{\text{nf}}(\boldsymbol{x}; \boldsymbol{\theta}) = p_{\boldsymbol{x_0}}(T_1 \dots T_n(\boldsymbol{x})) \cdot |\det \boldsymbol{J}_{T_1 ... T_n}(\boldsymbol{x})| \tag{3}$$

Architecture Search for NF: Problem Definition

The diverse existing transformations with distinct performances and computational costs result in a large design space to build optimal flow models. For an n-layer flow model, where each layer is based on one of m transformations $\{T^1, T^2, ...T^m\}$ with costs $\{C^1, C^2, ...C^m\}$, respectively, the resulted search space contains m^n flow models.

The goal of architecture optimization is to find one flow model in the m^n flow models with the best performance-cost trade-off. In this paper, the performance is evaluated by the KL divergence between the target distribution $p^*(x)$ and $p^{\text{nf}}(\boldsymbol{x}; \boldsymbol{\theta})$ (i.e., the negative log-likelihood). The computation cost of $p^{\text{nf}}(\boldsymbol{x}; \boldsymbol{\theta})$ is the sum of the costs of the all transformation layers which we denote by C^{nf}. We propose to use the weighted sum of the KL divergence and the cost as a figure of merit (FOM) to measure the performance-cost trade-off (lower the better). Mathematically, the NF architecture search problem is a discrete optimization problem over all n-layer flow models:

$$\begin{aligned} i^* &= \arg\min_i \; D_{\text{KL}}[p^*(\boldsymbol{x}) \;||\; p_i^{\text{nf}}(\boldsymbol{x}; \boldsymbol{\theta_i^*})] + \lambda \cdot C^{\text{nf}} \\ \text{s.t.} \quad \boldsymbol{\theta_i^*} &= \arg\min_{\boldsymbol{\theta_i}} \; D_{\text{KL}}[p^*(\boldsymbol{x}) \;||\; p_i^{\text{nf}}(\boldsymbol{x}; \boldsymbol{\theta_i})] + \lambda \cdot C^{\text{nf}} \\ &\text{where:} \quad i \in \{1, 2 \cdots, m^n\} \end{aligned} \tag{4}$$

In eq.(4), $p_i^{\text{nf}}(\boldsymbol{x}; \boldsymbol{\theta_i})$ demotes density function of the i_{th} discrete n-layer flow model and λ is a user-defined factor that balances the performance-cost trade-off.

Inapplicability of Existing D-NAS to NF Architecture Optimization

Discrete optimization of the combinatorial problem in eq.(4) is hard. It appears natural to consider efficient differential

[1] In this paper, for convenience, we write T as the inverse transformation that models output random variable to input random variable to avoid T^{-1} in density function transformation.

[2] In all later sections, we write $\boldsymbol{x_n}$ as $\boldsymbol{x}$ for simplicity.

NAS techniques (Liu, Simonyan, and Yang 2019) that have been developed for unstructured neural networks. D-NAS converts the discrete optimization problem into a continuous optimization problem with discrete solution constraints and solves it with approximation.

D-NAS introduces two mechanisms to efficiently solve the combinatorial problem with approximation. First, the discrete search space is relaxed to a continuous one by constructing a "supernet", which is the sum of the candidate neural network structures weighted by architecture parameters. Such a formulation enables efficient gradient-based optimization over a large discrete architectural space. Second, the final optimal discrete architecture is approximately obtained by finding the nearest binarization of the optimized continuous architectural solution.

However, we identify two issues of the existing D-NAS techniques, namely, *inapplicability* and *sub-optimality*, for NF architectural optimization.

- Inapplicability: the supernet formulation fails to work for NF since it does not maintain the invertibility of the optimized NF model during optimization. For instance, consider two one-dimensional invertible (i.e., monotonic) transformations, T_1 and T_2, which are monotonically increasing and decreasing, respectively. A weighted sum of the two invertible transformations, however, is not necessarily invertible.
- Sub-optimality: Assuming that the relaxed continuous optimal architectural solution could be found somehow, binarizing it to obtain a discrete solution as in the existing D-NAS approach may lead to a final solution that is not optimal.

Our AutoNF framework specifically addresses the two aforementioned challenges, enabling efficient differentiable architecture optimization for NF for the first time. First, we perform continuous relaxation of the search space with a novel mixture distribution formulation without violating the invertibility requirement. Second, we address the issue of loss of optimality due to binarization. We convert the original NP-hard combinatorial NF architectural optimization problem to an unconstrained continuously relaxed problem, which is much easier to solve. We rigorously prove that the solution to our continuous relaxation is globally optimal for the original discrete optimization problem.

Methods

Continuous Relaxation by Mixture Distribution

We first discuss continuous relaxation of the i_{th} layer in an n-layer flow model with m optional transformations $\{T^1, T^2, \cdots, T^m\}$ and then extend the formulation to all layers. Similar to (Liu, Simonyan, and Yang 2019), for each optional transformation T^j, we introduce an architecture parameter α_i^j to model its probability to be selected for layer i by a Softmax function:

$$w_i^j = \frac{\exp(\alpha_i^j)}{\sum_{j=1}^m \exp(\alpha_i^j)}, \tag{5}$$

which is considered the "weight" of T^j.

Continuous relaxation for NF needs to build a valid "superflow" considering all optional transformations to convert the discrete architecture optimization problem to the joint optimization of architecture parameters $\boldsymbol{\alpha}$ and model parameters of the superflow $\boldsymbol{\theta}$ in the continuous space. Applying continuous relaxation of (Liu, Simonyan, and Yang 2019) to NF amounts to constructing a weighted sum of all candidate transformations for each layer i: $T_i' = \sum_{j=1}^m w_i^j \cdot T^j$. However, the resulted T_i' is not always a bijection, hence the critical NF's invertibility property may be violated.

Instead, we construct the superflow as a mixture distribution model. Applying each T^j to transform $\boldsymbol{x_i}$ to $\boldsymbol{x_{i-1}} (i \geq 1)$ creates a new density function which we denote by $p_{T^j}(\boldsymbol{x_i})$:

$$p_{T^j}(\boldsymbol{x_i}) = p_{\boldsymbol{x_{i-1}}}(T^j(\boldsymbol{x_i})) \cdot |\det \boldsymbol{J}_{T^j}(\boldsymbol{x_i})| \tag{6}$$

All possible m density functions can be weighted summed together by w_i^j, leading to the desired mixture model $p_{\boldsymbol{x_i}}(\boldsymbol{x_i})$:

$$p_{\boldsymbol{x_i}}(\boldsymbol{x_i}) = \sum_{j=1}^m w_i^j \cdot p_{T^j}(\boldsymbol{x_i}) \quad \text{where: } \sum_j^m w_i^j = 1 \tag{7}$$

It has been shown that given the distributions of two sets of random variables, there exits an NF, i.e., an invertible function that maps one set of the random variable to the other, namely, the universal representation ability of normalizing flows (Papamakarios et al. 2021). Thus, for the proposed mixture distribution, there exists an implicit bijection T_i' from $\boldsymbol{x}_i$ to $\boldsymbol{x_{i-1}}$ that transforms $p_{\boldsymbol{x_i}}$ to $p_{\boldsymbol{x_{i-1}}}$. As a result, the proposed mixture distribution satisfies the NF's invertibility requirement.

To obtain the continuous relaxation for an n-layer flow model, we apply the mixture distribution formulation recursively to each layer as shown in Figure 1 (left), leading to a mixture distribution $p_{\text{mix}}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{a})$ of the n-layer model:

$$p_{\text{mix}}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\alpha}) = \sum_{j_n}^m w_i^{j_n} \cdots \sum_{j_1}^m w_i^{j_1} \cdot [p_{T_1^{j_1} \cdots T_n^{j_n}}(\boldsymbol{x}; \boldsymbol{\theta})], \tag{8}$$

where we denote the j_{th} transformation in the i_{th} layer and its weight by $T_i^{j_i}$ and $w_i^{j_i}$, respectively, and $p_{T_1^{j_1} \cdots T_n^{j_n}}(\boldsymbol{x}; \boldsymbol{\theta})$ is defined according to eq.(3) and represents the density of a particular n-layer flow model candidate.

Note that eq.(8) is based on m^n n-layer flow model candidates, each of which is assigned a model index k with $k = j_1 + \sum_{i=2}^n (j_i - 1) \cdot m^{i-1}$. We denote the density of the k_{th} n-layer flow model candidate and and its weight by $p_k(\boldsymbol{x}; \boldsymbol{\theta_k})$ and W_k, respectively, and rewrite eq.(8) as:

$$p_{\text{mix}}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\alpha}) = \sum_{k=1}^{m^n} W_k \cdot p_k(\boldsymbol{x}; \boldsymbol{\theta_k}) \tag{9}$$

Similarly, the cost of the superflow is defined as the weighted sum of the costs of all discrete n-layer flow models:

$$C_{\text{mix}}(\boldsymbol{\alpha}) = \sum_{k=1}^{m^n} W_k \cdot C_{p_k} \tag{10}$$

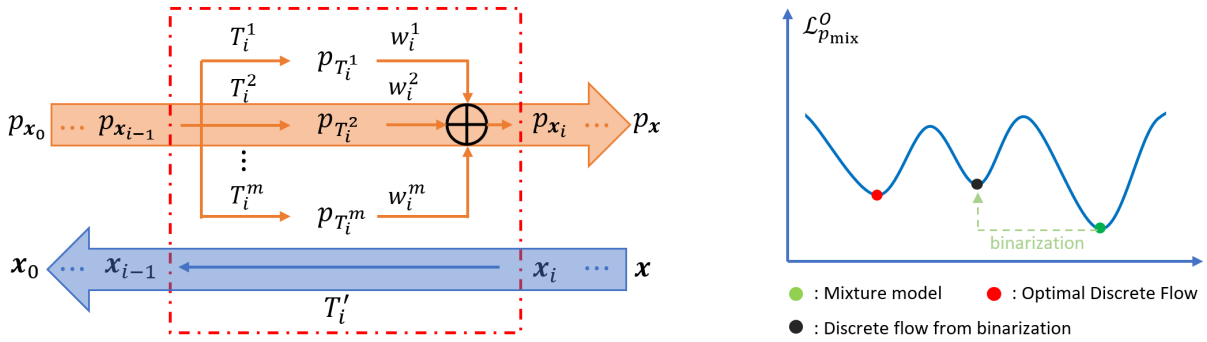


Figure 1: Left: the process to build a superflow by mixture distribution. The orange and blue arrow represent the transformation of the density functions and the transformation of the random variables, respectively. The red dashed box contains the detailed transformations for the i_{th} layer, where T_i' represents the bijection. Right: an illustrated example of the potential sub-optimality of binarized flow models. The green dot is a mixture model that minimizes the loss in OP_1. The black dot is the corresponding binarized architecture, which may be less optimal than the red dot, the true optimal discrete flow model.

We define the set that contains the densities of all discrete n-layer flow model candidates as the *feasible solution set* (S). The differentiable architecture optimization problem for NF is formulated as a constrained optimization problem ($\boldsymbol{OP_1}$):

$$\begin{aligned} &\arg\min_{\boldsymbol{\theta},\boldsymbol{\alpha}} \mathcal{L}^O_{p_{\text{mix}}}(\boldsymbol{\theta},\boldsymbol{\alpha}), \quad \text{where:} \\ &\mathcal{L}^O_{p_{\text{mix}}}(\boldsymbol{\theta},\boldsymbol{\alpha}) = -\mathbb{E}_{p^*(\boldsymbol{x})}[\log(p_{\text{mix}}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\alpha}))] + \lambda \cdot C_{\text{mix}}(\boldsymbol{\alpha}) \\ &\text{s.t.} \quad p_{\text{mix}}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\alpha}) \in S \end{aligned} \tag{11}$$

We also call OP_1 the original optimization problem.

Unconstrained Continuous Relaxation of NF D-NAS Admitting Optimal Discrete Solution

OP_1 is a difficult mixed-binary programming problem due to the discrete nature of architectural optimization. If we were to binarize the optimized continuous solution to obtain a discrete NF model as in the existing D-NAS approach (Liu, Simonyan, and Yang 2019) for unstructured neural networks, we would suffer from the resulted sub-optimality, as shown in Figure 1 (Right).

An upper bound of $\mathcal{L}^O_{p_{\text{mix}}}(\boldsymbol{\theta},\boldsymbol{\alpha})$ can be derived with Jensen's inequality, namely $\mathcal{L}^U_{p_{\text{mix}}}(\boldsymbol{\theta},\boldsymbol{\alpha})$. Interestingly, as we shall show below that the unconstrained minimization of $\mathcal{L}^U_{p_{\text{mix}}}(\boldsymbol{\theta},\boldsymbol{\alpha})$ leads to the optimal discrete architectural solution $p_z(\boldsymbol{x};\boldsymbol{\theta}^*_{\boldsymbol{z}})$ of OP_1 without any approximated binarization under the mild condition that $p_z(\boldsymbol{x};\boldsymbol{\theta}^*_{\boldsymbol{z}})$ is unique. Since the discrete flow models in the search space typically have distinct performances and computational costs, the uniqueness of $p_z(\boldsymbol{x};\boldsymbol{\theta}^*_{\boldsymbol{z}})$ is usually satisfied. Formally, we define this unconstrained optimization problem as $\boldsymbol{OP_2}$:

$$\begin{aligned} &\arg\min_{\boldsymbol{\theta},\boldsymbol{\alpha}} \mathcal{L}^U_{p_{\text{mix}}}(\boldsymbol{\theta},\boldsymbol{\alpha}), \\ &\text{where:} \ \ \mathcal{L}^U_{p_{\text{mix}}}(\boldsymbol{\theta},\boldsymbol{\alpha}) = \\ &\sum_k^{m^n} W_k \cdot (-\mathbb{E}_{p^*(\boldsymbol{x})}[\log(p_k(\boldsymbol{x};\boldsymbol{\theta_k}))]) + \lambda \cdot C_{\text{mix}}(\boldsymbol{\alpha}) \end{aligned} \tag{12}$$

Now we prove that OP_1 and OP_2 are equivalent and have the same unique optimal discrete architectural solution $p_z(\boldsymbol{x};\boldsymbol{\theta}^*_{\boldsymbol{z}})$. For convenience, we use $\mathbb{E}$ to replace $\mathbb{E}_{p^*(\boldsymbol{x})}$ and define $M_k = -\mathbb{E}[\log(p_k(\boldsymbol{x};\boldsymbol{\theta_k})] + \lambda \cdot C_{p_k}$.

Proposition 1: *For any discrete flow models in the feasible solution set S , the loss of OP_1 and OP_2 are the same, i.e., $\forall p_k(\boldsymbol{x};\boldsymbol{\theta_k}) \in S, \mathcal{L}^O_{p_{\text{mix}}} = \mathcal{L}^U_{p_{\text{mix}}}$.*

Proof: For any $p_k(\boldsymbol{x};\boldsymbol{\theta_k})$ in S, let $W_k = 1$ and $\boldsymbol{W_i} = \boldsymbol{0}$, where $i \neq k$, we have:

$$\mathcal{L}^O_{p_{\text{mix}}}(\boldsymbol{\theta_k},\boldsymbol{\alpha_k}) = M_k = \mathcal{L}^U_{p_{\text{mix}}}(\boldsymbol{\theta_k},\boldsymbol{\alpha_k}) \tag{13}$$

Theorem 1: *If the optimal solution $p_z(\boldsymbol{x};\boldsymbol{\theta}^*_{\boldsymbol{z}})$ of OP_1 is unique, OP_1 and OP_2 are equivalent and attain the same optimal solution $p_z(\boldsymbol{x};\boldsymbol{\theta}^*_{\boldsymbol{z}})$.*

Proof: $\mathcal{L}^U_{p_{\text{mix}}}(\boldsymbol{\theta},\boldsymbol{\alpha})$ can be written as a linear combination based on the cost model definition in eq.(10):

$$\mathcal{L}^U_{p_{\text{mix}}}(\boldsymbol{\theta},\boldsymbol{\alpha}) = \sum_k^{m^n} W_k \cdot (-\mathbb{E}[\log(p_k(\boldsymbol{x};\boldsymbol{\theta_k}))] + \lambda \cdot C_{p_k}) \tag{14}$$

For the k-th flow candidate, we denote the optimal model parameter that minimizes $D_{\text{KL}}[p^*(\boldsymbol{x})||p_k(\boldsymbol{x};\boldsymbol{\theta_k})]$ by $\boldsymbol{\theta}^*_{\boldsymbol{k}}$, hence:

$$-\mathbb{E}[\log(p_k(\boldsymbol{x};\boldsymbol{\theta}^*_{\boldsymbol{k}})] \leq -\mathbb{E}[\log(p_k(\boldsymbol{x};\boldsymbol{\theta_k})] \tag{15}$$

Then we have:

$$\begin{aligned} \mathcal{L}^U_{p_{\text{mix}}}(\boldsymbol{\theta},\boldsymbol{\alpha}) &= \sum_k^{m^n} W_k \cdot (-\mathbb{E}[\log(p_k(\boldsymbol{x};\boldsymbol{\theta_k}))] + \lambda \cdot C_{p_k}) \\ &\geq \sum_k^{m^n} W_k \cdot (-\mathbb{E}[\log(p_k(\boldsymbol{x};\boldsymbol{\theta}^*_{\boldsymbol{k}}))] + \lambda \cdot C_{p_k}) \end{aligned} \tag{16}$$

Based on **Proposition 1**:

$$\mathcal{L}^O_{p_{\text{mix}}}(\boldsymbol{\theta}^*_{\boldsymbol{k}},\boldsymbol{\alpha_k}) = M^*_k = \mathcal{L}^U_{p_{\text{mix}}}(\boldsymbol{\theta}^*_{\boldsymbol{k}},\boldsymbol{\alpha_k}) \tag{17}$$

By definition, the optimal discrete architectural solution $p_z(\boldsymbol{x}; \boldsymbol{\theta}_{\boldsymbol{z}}^*)$ minimizes the loss in OP_1, denoted by M_z^*. Based on eq.(17), we have:

$$\begin{aligned} M_z^* &= -\mathbb{E}[\log(p_z(\boldsymbol{x}; \boldsymbol{\theta}_{\boldsymbol{z}}^*))] + \lambda \cdot C_{p_z} \\ &< -\mathbb{E}[\log(p_k(\boldsymbol{x}; \boldsymbol{\theta}_{\boldsymbol{k}}^*)] + \lambda \cdot C_{p_k} = M_k^* \\ &\quad \forall k \in \{1, 2 \cdots m^n\},\ k \neq z \end{aligned} \tag{18}$$

Combining eq.(15) and eq.(18) leads to:

$$\begin{aligned} \mathcal{L}_{p_{\text{mix}}}^U(\boldsymbol{\theta}, \boldsymbol{\alpha}) &= \sum_k^{m^n} W_k \cdot (-\mathbb{E}[\log(p_k(\boldsymbol{x}; \boldsymbol{\theta}_{\boldsymbol{k}}))] + \lambda \cdot C_{p_k}) \\ &\geq \sum_k^{m^n} W_k \cdot (-\mathbb{E}[\log(p_k(\boldsymbol{x}; \boldsymbol{\theta}_{\boldsymbol{k}}^*))] + \lambda \cdot C_{p_k}) \\ &\geq \sum_k^{m^n} W_k \cdot (-\mathbb{E}[\log(p_z(\boldsymbol{x}; \boldsymbol{\theta}_{\boldsymbol{z}}^*))] + \lambda \cdot C_{p_z}) \\ &= -\mathbb{E}[\log(p_z(\boldsymbol{x}; \boldsymbol{\theta}_{\boldsymbol{z}}^*))] + \lambda \cdot C_{p_z} \end{aligned} \tag{19}$$

Eq.(19) indicates that the global minimizer of $\mathcal{L}_{p_{\text{mix}}}^U(\boldsymbol{\theta}, \boldsymbol{\alpha})$ is the optimal solution of OP_1, i.e., $p_z(\boldsymbol{x}; \boldsymbol{\theta}_{\boldsymbol{z}}^*)$. It is reached if and only if we have $\boldsymbol{\theta} = \boldsymbol{\theta}_{\boldsymbol{z}}$ and $W_z = 1$. ■

Decomposed Search for Deep Flow Models

The superflow consists of m^n distributions. The architectural optimization of a deep model with many layers is computationally intensive. To cope with the computational challenge, we decompose the given multiple-layer model into a set of cascading blocks with each containing a limited number of layers. We sequentially optimize these blocks based on the unconstrained problem OP_2 one at a time.

As in (Liu, Simonyan, and Yang 2019), we apply first order bi-level optimization of the model parameters $\boldsymbol{\theta}$ and the architecture parameters $\boldsymbol{\alpha}$ over the training data (X_{val}) and the validation data (X_{train}), as summarized in Algorithm 1.

Algorithm 1: AutoNF Algorithm

Input: Transformation options: $\{T^1, T^2, ...T^m\}$, Blocks: $\boldsymbol{B} = \{B_1, B_2, ...B_l\}$, Cost: C_{mix}
Output: Optimized n-layer flow model:
for $B_i \in \boldsymbol{B}$ **do**
  while *not converged* **do**
    1. $\boldsymbol{\theta}_{\boldsymbol{B_i}} = \arg\min_{\boldsymbol{\theta}_{\boldsymbol{B_i}}} \mathcal{L}_{p_{\text{mix}}}^U(X_{\text{train}}, \boldsymbol{\theta}_{\boldsymbol{B_i}}, \boldsymbol{\alpha}_{\boldsymbol{B_i}})$
    2. $\boldsymbol{\alpha}_{\boldsymbol{B_i}} = \arg\min_{\boldsymbol{\alpha}_{B_i}} \mathcal{L}_{p_{\text{mix}}}^U(X_{\text{val}}, \boldsymbol{\theta}_{\boldsymbol{B_i}}, \boldsymbol{\alpha}_{\boldsymbol{B_i}})$
  Append the optimized B_i to the flow model.

Experiments and Analysis

We evaluate AutoNF from three aspects: **1)** We demonstrate that AutoNF can discover novel architectures with superior performance-cost trade-offs than hand-crafted baselines in the existing literature. **2)** We study how different values of cost term λ affect the architecture search results. **3)** We show the benefit of solving OP_2 compared with solving OP_1 with binarization with an empirical study.

1. Performance-Cost Trade-offs of NF Models:

Transformation options: Flows with analytical invertibility are favorable since they can perform both density estimation and random sampling with a single model. We target analytically invertible flows with different performance and computational costs to better demonstrate the performance-cost trade-off between different flow models.

Flows from the prevalent autoregressive flow and coupling layer family are analytically invertible and have diverse performances and costs and are suitable for our problem. For example, Affine autoregressive flow (Papamakarios, Pavlakou, and Murray 2017) has limited expressive power but the computation cost is lower, while rational quadratic autoregressive flow (Durkan et al. 2019) has the state-of-the-art performance in autoregressive family with higher computational cost. Meanwhile, autoregressive flows are expensive in calculating the inverse but generally perform better compared with their counterparts in the coupling layer family. As a result, we choose affine autoregressive flow (MAF) and rational quadratic autoregressive flow (RQ-AF) from the autoregressive family. In coupling layers, we consider affine coupling layer (RealNVP) (Dinh, Krueger, and Bengio 2015) and rational quadratic coupling layer (RQ-C) (Durkan et al. 2019).

We follow the existing literature (Kingma and Dhariwal 2018; Durkan et al. 2019) and introduce an LU-decomposed linear transformation and a reverse permutation to pair with non-linear transformations in the selected flows for better performance. Hence, the candidate transformations for each layer of AutoNF are composited transformations.

Evaluation metrics: We introduce two kinds of costs for the candidate transformations. The forward cost is defined as the computational cost to run the forward pass of a transformation and vice versa. Forward cost needs to be considered when drawing random samples from a flow model and inverse cost dominates when the flow model is applied to density estimation tasks. Derivation of the costs can be found in details in Appendix C.

Additionally, we adopt two different performance metrics. The first metric is the negative log-likelihood (NLL) which has been widely adopted in the existing literature for density estimation. However, one drawback of NLL is that the numerical values and differences between the NLL of different flow models vary dramatically across different datasets. Evaluation of different flow models needs to be specified for a certain dataset, making an overall comparison difficult. Thus, we turn to the recently proposed "density and coverage" (Naeem et al. 2020) as the second performance metric.

"Density and coverage" evaluates the sample quality of generative models. We use density as the performance metric since it reflects the fidelity of a model and is consistent with the NLL (the lower the NLL, the higher the density). The density values of different flow models are at the same order of magnitude which allows the evaluation of flow models across datasets.

We define two kinds of the figure of merits, namely, FOM 1 and FOM 2, to evaluate the performance-cost trade-off between flow models based on the two performance metrics.

Dataset	Cost Type	Architecture	FOM 1	Density	Cost	FOM 2
POWER	Forward($\lambda = 0.7$)	RQ-AF	8.808	0.998	13.31	+696.5%
		MAF	5.842	0.974	8.66	+419.4%
		AutoNF	**1.085**	0.967	1.66	**+0%**
	Inverse($\lambda = 0.7$)	RQ-C	5.05	0.971	10.92	+192.1%
		RealNVP	3.97	0.893	8.00	+121.9%
		AutoNF	**1.58**	0.965	3.73	**+0%**
GAS	Forward($\lambda = 1$)	RQ-AF	2.632	0.755	13.31	+305.9%
		MAF	-0.21	0.588	8.66	+183.1%
		AutoNF	**-6.195**	0.608	33.10	**+0%**
	Inverse($\lambda = 1$)	RQ-C	0.13	0.769	10.92	+242.1%
		RealNVP	-0.55	0.556	8.00	+178.9%
		AutoNF	**-5.73**	0.487	2.73	**+0%**
HEPMASS	Forward($\lambda = 0.5$)	RQ-AF	23.265	0.941	13.31	+324.3%
		MAF	22.66	0.923	8.00	+155.1%
		AutoNF	**19.13**	0.896	3.10	**+0%**
	Inverse($\lambda = 0.3$)	RQ-C	21.716	0.906	10.92	+354.5%
		RealNVP	22.95	0.848	8.00	+237.7%
		AutoNF	**21.138**	0.837	2.36	**+0%**
MINIBOONE	Forward($\lambda = 1$)	RQ-AF	24.09	0.824	13.31	+147.3%
		MAF	20.11	0.797	8.66	+65.7%
		AutoNF	**16.313**	**0.867**	5.49	**+0%**
	Inverse($\lambda = 1$)	RQ-C	22.12	0.776	10.92	+125.7%
		RealNVP	21.05	0.802	8.00	+60.5%
		AutoNF	**16.97**	0.738	4.73	**+0%**
BSDS300	Forward($\lambda = 10$)	RQ-AF	-20.72	0.924	13.31	+305.5%
		MAF	-67.86	1.006	8.66	+144.5%
		AutoNF	**-121.4**	0.746	3.10	**+0%**
	Inverse($\lambda = 10$)	RQ-C	-43.45	0.861	10.92	+154.4%
		RealNVP	-69.66	0.878	8.00	+85.5%
		AutoNF	**-100.05**	**0.896**	4.36	**+0%**

Table 1: Performance-cost trade-off between architectures found by AutoNF and reference designs on five datasets

FOM 1 is the summation of the NLL and the cost of the flow model weighted by the cost term λ, i.e., the loss function value in OP_1. We define FOM 2 as "the percentage reduction of cost minus percentage increasing of density" compared with a reference model. We let the reference models for FOM 2 be the flows found by AutoNF. For both FOMs, the *lower* value indicates *better* performance-cost trade-off.

Reference designs and datasets: We utilize AutoNF to build flow models with up to 8 layers with the above 4 candidate transformations. We include identity transformation with zero cost as an additional NULL option for each layer, which is removed from the final architecture. This allows AutoNF to discover flow models with fewer layers of "real" transformations when their performances are already good. For reference designs, we follow the architectures in the existing literature and manually construct four flow models by stacking one of the candidate composited transformations for 8 layers, denoted by their literature names as MAF, RQ-AF, RealNVP and RQ-C respectively. Flows found by AutoNF are compared with reference designs on both FOMs.

We let AutoNF run on UCI density estimation datasets (Dua and Graff 2017) and BSDS300 (Martin et al. 2001) dataset. The final results are summarized in Table 1.

AutoNF can consistently find flow models with better performance-cost trade-offs over all datasets given the selected λ values and can reach up to $4.29X$ cost reduction with minor performance drop. In some test cases, AutoNF even provides a better density metric with significantly lower cost ([BSDS300, Inverse], [MINIBOONE, forward]).

2. Effect of Cost Term λ on Architecture Search: We use the same candidate transformations and build flow models of up to 8 layers based on different λ values on the POWER dataset. The cost type is set to forward. We summarize the test NLL, density, cost and FOM 2 of flows found by AutoNF in Figure 2 (Left, Middle). Especially, the reference model to calculate FOM 2 of different searched models is the flow model found by letting $\lambda = 0$.

Additionally, we plot the Pareto front with respect to the mean test NLL and the cost of the searched architectures from different λ. We also randomly sampled 10 architectures from the design space, as well as adding the baseline models for POWER dataset in Table 1 to the plot to compare with flows found by AutoNF, as shown in Figure 2 (Right).

Figure 2 (Left and Middle) shows that, when λ is set to 0, AutoNF focuses only on finding expressive flows and discovers a flow with the best performance and the highest cost. Increasing λ encourages AutoNF to select architectures with reduced cost and degraded performance. An identity flow with the lowest cost and the worst performance is produced if λ is too large. The Pareto front derived in Figure 2 (Right)

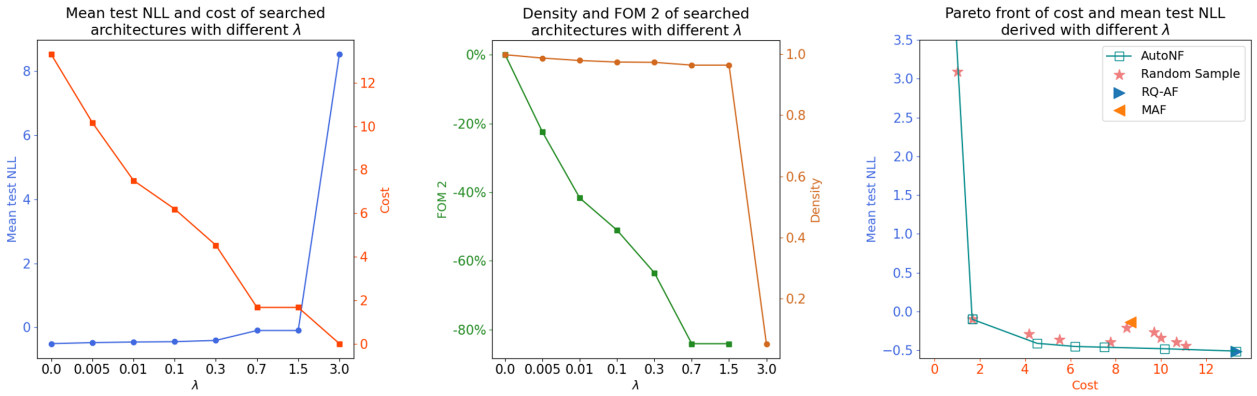
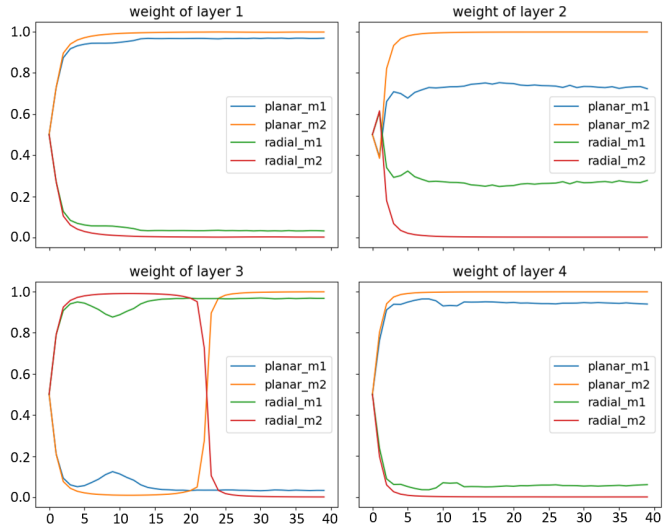
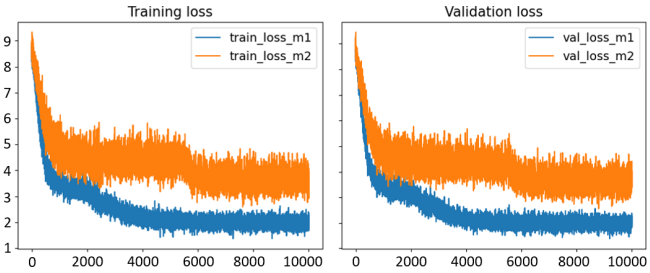


Figure 2: Left and Middle: Mean test NLL, cost, density and FOM2 of searched architectures with different λ on POWER. Right: Plot of the Pareto front derived from the searched flows and the plot of random sampled flows and baseline models.





	M1		M2	
	planar	radial	planar	radial
Final W (Layer 1)	0.968	0.032	0.998	0.002
Final W (Layer 2)	0.723	0.277	0.999	0.001
Final W (Layer 3)	0.034	0.966	0.998	0.002
Final W (Layer 4)	0.939	0.061	0.999	0.001
Searched Flow	[planar, planar, radial, planar]		[planar, planar, planar, planar]	
Test Score	4.47±0.034		3.74±0.027	

Figure 3: Left: plots of weights of transformations in each layer during the search (per 100 iteration). Top-Right: plots of training and validation loss during the search. Bottom-Right: final weights, obtained flow models, and their test scores (NLL).

indicates that tuning λ can guide AutoNF to discover flow models with different emphasis on performance or cost and achieve better performance-cost trade-offs.

3. The benefit of Solving OP_2: We search two 4-layer flow models with 2 transformation options including planar flow and radial flow (Rezende and Mohamed 2015) on the POWER dataset by two methods. The first method (M_1) solves OP_1 with binarization and the second method (M_2) directly solve OP_2. We set λ to zero for simplicity and run the search with a batch size of 512 for 10000 iterations. We summarize the results in Figure 3, with the weights of transformations in each layer, the training and validation loss during the search process, the final weights, the architectures found by two methods and their test scores (test NLL).

When optimized by M_2, the weights of the transformations in each layer converge to binary values and directly lead to a discrete flow without binarization. In contrast, with M_1, the optimization of the loss of OP_1 first leads to a mixture model. The final flow obtained by finding the nearest binarization performs worse than the flow found by M_2. This empirical study suggests that solving OP_2 is beneficial with no need to perform approximated binarization.

Discussion: Future work on AutoNF could be extended from multiple perspectives. **1)** incorporating techniques that improves vanilla D-NAS to AutoNF. For instance, (Xie et al. 2019) introduces Gumble-Softmax to optimize over the expectation of multiple sampled architectures rather than a supernet. AutoNF can possibly benefit from the formulation by sampling smaller sized superflows and improve the efficiency. **2)** RL based NAS methods in principle can be directly applied to NF and the low efficiency can be improved with proper parameter sharing (Pham et al. 2018). However, the effect of parameter sharing across large neural networks (T) in NF remains unexplored and a comparison between AutoNF and these methods would be favorable. We leave these directions to future works.

Acknowledgments

This research was supported by the U.S. Department of Energy, through the Office of Advanced Scientific Computing Research's "Data-Driven Decision Control for Complex Systems (DnC2S)" project. The research participation of the University of California at Santa Barbara was supported under award number DE-SC0021321. Pacific Northwest National Laboratory is operated by Battelle Memorial Institute for the U.S. Department of Energy under Contract No. DE-AC05-76RL01830. Oak Ridge National Laboratory is operated by UT-Battelle LLC for the U.S. Department of Energy under contract number DE-AC05-00OR22725. The Jefferson Science Associates (JSA) operates the Thomas Jefferson National Accelerator Facility for the U.S. Department of Energy under Contract No. DE-AC05-06OR23177.

The work was developed based on Pytorch (Paszke et al. 2019) and part of the code were adopted from the publicly released library: nflows (Durkan et al. 2019). The authors would also like to thank the reviewers for providing useful feedback and suggestions during the review process.

References

Behrmann, J.; Grathwohl, W.; Chen, R. T. Q.; Duvenaud, D.; and Jacobsen, J.-H. 2019. Invertible Residual Networks. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 573–582. Long Beach, California, USA: PMLR.

Dinh, L.; Krueger, D.; and Bengio, Y. 2015. NICE: Non-linear Independent Components Estimation. *CoRR*, abs/1410.8516.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2017. Density estimation using Real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.

Durkan, C.; Bekasov, A.; Murray, I.; and Papamakarios, G. 2019. Neural Spline Flows. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Grathwohl, W.; Chen, R. T. Q.; Bettencourt, J.; Sutskever, I.; and Duvenaud, D. 2019. FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Huang, C.-W.; Krueger, D.; Lacoste, A.; and Courville, A. 2018. Neural Autoregressive Flows. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 2078–2087. PMLR.

Kandasamy, K.; Neiswanger, W.; Schneider, J.; Poczos, B.; and Xing, E. P. 2018. Neural Architecture Search with Bayesian Optimisation and Optimal Transport. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative Flow with Invertible 1x1 Convolutions. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; and Kavukcuoglu, K. 2018. Hierarchical Representations for Efficient Architecture Search. In *International Conference on Learning Representations*.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*.

Martin, D.; Fowlkes, C.; Tal, D.; and Malik, J. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, 416–423 vol.2.

Mazoure, B.; Doan, T.; Durand, A.; Pineau, J.; and Hjelm, R. D. 2020. Leveraging exploration in off-policy algorithms via normalizing flows. In Kaelbling, L. P.; Kragic, D.; and Sugiura, K., eds., *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, 430–444. PMLR.

Naeem, M. F.; Oh, S. J.; Uh, Y.; Choi, Y.; and Yoo, J. 2020. Reliable Fidelity and Diversity Metrics for Generative Models. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 7176–7185. PMLR.

Negrinho, R.; and Gordon, G. 2017. DeepArchitect: Automatically Designing and Training Deep Architectures. arXiv:1704.08792.

Papamakarios, G.; Nalisnick, E.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2021. Normalizing Flows for Probabilistic Modeling and Inference. arXiv:1912.02762.

Papamakarios, G.; Pavlakou, T.; and Murray, I. 2017. Masked Autoregressive Flow for Density Estimation. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E. Z.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *CoRR*, abs/1912.01703.

Pham, H.; Guan, M.; Zoph, B.; Le, Q.; and Dean, J. 2018. Efficient Neural Architecture Search via Parameters Sharing. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80

of *Proceedings of Machine Learning Research*, 4095–4104. PMLR.

Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y. L.; Tan, J.; Le, Q.; and Kurakin, A. 2017. Large-Scale Evolution of Image Classifiers. arXiv:1703.01041.

Rezende, D.; and Mohamed, S. 2015. Variational Inference with Normalizing Flows. In Bach, F.; and Blei, D., eds., *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 1530–1538. Lille, France: PMLR.

Rippel, O.; and Adams, R. P. 2013. High-Dimensional Probability Estimation with Deep Density Models. arXiv:1302.5125.

Suganuma, M.; Ozay, M.; and Okatani, T. 2018. Exploiting the Potential of Standard Convolutional Autoencoders for Image Restoration by Evolutionary Search. arXiv:1803.00370.

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2019. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*.

Zoph, B.; and Le, Q. V. 2017. Neural Architecture Search with Reinforcement Learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.