AUTM Flow: Atomic Unrestricted Time Machine for Monotonic Normalizing Flows

Difeng Cai[1] **Yuliang Ji**[1] **Huan He**[2] **Qiang Ye**[3] **Yuanzhe Xi**[1]

[1]Department of Mathematics, Emory University, Atlanta, GA, USA
[2]Department of Computer Science, Emory University, Atlanta, GA, USA
[3]Department of Mathematics, University of Kentucky, Lexington, KY, USA

Abstract

Nonlinear monotone transformations are used extensively in normalizing flows to construct invertible triangular mappings from simple distributions to complex ones. In existing literature, monotonicity is usually enforced by restricting function classes or model parameters and the inverse transformation is often approximated by root-finding algorithms as a closed-form inverse is unavailable. In this paper, we introduce a new integral-based approach termed: Atomic Unrestricted Time Machine (AUTM), equipped with unrestricted integrands and easy-to-compute explicit inverse. AUTM offers a versatile and efficient way to the design of normalizing flows with explicit inverse and unrestricted function classes or parameters. Theoretically, we present a constructive proof that AUTM is universal: all monotonic normalizing flows can be viewed as limits of AUTM flows. We provide a concrete example to show how to approximate any given monotonic normalizing flow using AUTM flows with guaranteed convergence. The result implies that AUTM can be used to transform an existing flow into a new one equipped with explicit inverse and unrestricted parameters. The performance of the new approach is evaluated on high dimensional density estimation, variational inference and image generation. Experiments demonstrate superior speed and memory efficiency of AUTM.

1 INTRODUCTION

Generative models aim to learn a latent distribution from given samples and then generate new data from the learned distribution. There are several kinds of generative models, including generative adversarial networks (GANs) [Goodfellow et al., 2014], variational autoencoders (VAE) [Kingma and Welling, 2013], and normalizing flows [Rezende and Mohamed, 2015], etc. Unlike GANs and VAE, normalizing flows offer a tractable and efficient way for exact density estimation and sampling. Applications of normalizing flows include image generation [Kingma and Dhariwal, 2018, Ho et al., 2019], noise modelling [Abdelrahman Abdelhamed, 2019], and reinforcement learning [Mazoure et al., 2019], et al.

Two challenges in normalizing flows are the computation of Jacobian determinant and the inverse transformation. Different architectures have been proposed to address those issues. Neural ODE [Chen et al., 2018] and Free-form Jacobian of Reversible Dynamics (FFJORD) [Grathwohl et al., 2019] pioneered the way of modeling the transformation as a dynamical system. The inverse can be easily computed by reversing the dynamics in time, but Jacobian determinant is hard to compute and the use of neural network to model the dynamics often leads to high computational cost. To simplify the Jacobian computation, most flows employ monotone triangular mappings so that the Jacobian is triangular. Two such architectures are autoregressive flows and coupling flows. Examples of monotone mappings used in those flows include 1) **Special function classes** such as affine function [Dinh et al., 2014, 2017, Kingma and Dhariwal, 2018]), rational function [Ziegler and Rush, 2019], logistic mixture [Ho et al., 2019], splines [Durkan et al., 2019a,b]; 2) **Neural networks** [Huang et al., 2018, De Cao et al., 2020]; and 3) **Integral of positive functions** [Jaini et al., 2019, Wehenkel and Louppe, 2019].

To ensure monotonicity, methods using “special function classes" such as splines or sigmoid function σ (see Table 1) have to impose constraints on model parameters, which often impede the expressive power of the transformation as well as the efficiency of training. For example, during optimization, updates like $\theta = \theta - \gamma\nabla_\theta\mathcal{L}$ can potentially make the parameter θ fall out of the prescribed range and modifications are needed to guarantee the monotonicity under the new update. Integral-based methods rely on the simple fact that the function $q(x) = c + \int_0^x g(x)dx$ is always increasing as long as the integrand g is globally *positive*. For example, g is modeled as positive polynomials in sum-of-squares (SOS) polynomial flow by [Jaini et al., 2019] and as positive neural networks in unconstrained monotonic neural networks (UMNN) by [Wehenkel and Louppe, 2019]. Due to the flexibility offered by an integral form, these methods allow *unrestricted* model parameters as compared to other methods. It was shown in [Wehenkel and Louppe, 2019] that the method requires fewer parameters than straightforward neural network-based methods in [Huang et al., 2018, De Cao et al., 2020] and can scale to high dimensional datasets. However, unlike Neural ODE and FFJORD, these integral-based monotone mappings do not possess an explicit inverse formula and one has to resort to root-finding algorithms to compute the inverse transformation. This leads to increased computational cost because in *each* iteration of root-finding, one has to compute an integral of a complicated function. As discussed in [Wehenkel and Louppe, 2019], a judicious choice of quadrature rule is needed.

Contributions In this paper, we propose a new integral-based monotone triangular flow with proven universal approximation property for monotonic flows. The major contributions include the following.

- **Unrestricted model classes and parameters.** The proposed integral-based transformation is strictly increasing with *no* constraint on the integrand except being Lipschitz continuous.
- **Explicit inverse.** The inverse formula is explicitly given and compatible with fast root-finding methods for numerical inversion.
- **Universality.** The proposed model is *universal* in the sense that *any* monotonic normalizing flow is a limit of the proposed flows.

2 BACKGROUND

Normalizing flows are a class of generative models that aim to find a bijective mapping f such that f^{-1} "normalizes" the complex distribution into a tractable base distribution (Gaussian, for example). Once the "normalizing" mapping is found, generating new data points boils down to simply sampling from the base distribution and applying the forward transformation f to the samples. This makes normalizing flows a popular choice in density estimation, variational inference, image generation, etc.

Let $Y \in \mathbb{R}^D$ be a random variable with a possibly complicated probability density function $p_Y(y)$ and $X \in \mathbb{R}^D$ be a random variable with a well-studied probability density function $p_X(x)$. Assume that there is an invertible (vector-valued) function $f : \mathbb{R}^D \to \mathbb{R}^D$ that transforms the "base" variable X to Y, i.e. $Y = f(X)$. Then according to the change of variables formula, the probability density functions p_Y and p_X satisfy

$$p_Y(y) = p_X(x) \left|\det J_{f^{-1}}(y)\right| = p_X(x) \left|\det J_f(x)\right|^{-1}, \tag{1}$$

where $J_f(x)$ denotes the Jacobian of f evaluated at x.

The success of normalizing flows hinges on many factors, including **1:** the expressive power of the class of bijective transformations; **2:** the efficiency in computing the transformation and its inverse; **3:** the efficiency in computing the Jacobian determinant.

A popular architectural design to address those points is to employ an invertible triangular transformation, whose Jacobian is triangular and inversion can be computed in an entrywise fashion. Two representative triangular normalizing flows are autoregressive flows and coupling flows. Throughout the paper, we use $x_{1:k}$ to denote the vector $(x_1, \dots, x_k)$.

2.1 AUTOREGRESSIVE FLOWS

Autoregressive flows choose $f : \mathbb{R}^D \to \mathbb{R}^D$ to be an autoregressive mapping:

$$\begin{aligned} f(x;\theta) = (&q_1(x_1;\theta_1), q_2(x_2;\theta_2(x_1)), \dots, \\ &q_k(x_k;\theta_k(x_{1:k-1})), \dots, q_D(x_D;\theta_D(x_{1:D-1}))), \end{aligned} \tag{2}$$

where $x = (x_1, \dots, x_D)$ and each q_k, termed *transformer*, is a bijection parametrized by the so-called *conditioner* $\theta_k(x_{1:k-1})$. The Jacobian of f is a lower triangular matrix. To guarantee the invertibility of f, q_k is chosen to be a monotone function of x_k. Since q_k is usually a nonlinear neural network, an analytic inverse is not available and the inverse is computed by root-finding algorithms.

2.2 COUPLING FLOWS

Coupling flows first partition the input vector $x = (x_1, \dots, x_D)$ into two parts $x_{1:d}$ and $x_{d+1:D}$ and then apply the following transformation:

$$y_{1:d} = x_{1:d}, \quad y_{d+1:D} = q(x_{d+1:D};\theta(x_{1:d})), \tag{3}$$

where the parameter $\theta(x_{1:d})$ is an arbitrary function of $x_{1:d}$ and the scalar *coupling function* q is applied entrywisely, i.e., $q(x_{d+1:D}) = (q(x_{d+1}), \dots, q(x_D))$. The transformation in (3) from x to y is called a *coupling layer*. In normalizing flows, multiple coupling layers are composed to obtain a more complex transformation with the role of the two mappings in (3) swapped in alternating layers. The Jacobian of (3) is a lower triangular matrix with a 2-by-2 block structure corresponding to the partition of x. In [Dinh et al., 2017], the coupling function q is chosen as an affine function: $q(z) = z \cdot \exp(\alpha(x_{1:d})) + \beta(x_{1:d})$, where α and β are arbitrary functions, and the resulting flow is termed *affine coupling flow*.

Table 1: Comparison of different normalizing flow architectures. `E.I.` stands for explicit inverse, `U.P.` for unrestricted parameters, `U.F.` for unrestricted function representations.

Method	monotone map	`E.I`	`U.P.`	`U.F.`
Real NVP [Dinh et al., 2017]	affine function	yes	yes	no
Glow [Kingma and Dhariwal, 2018]	affine function	yes	yes	no
Flow++ [Ho et al., 2019]	$\alpha\sigma^{-1}\left(\sum_{i=1}^{r} c_i\sigma\left(\frac{x-a_i}{b_i}\right)\right)+\beta$	no	no	no
NSF [Durkan et al., 2019b]	rational-quadratic spline	yes	no	no
SOS [Jaini et al., 2019]	$\int_0^x \sum_{i=1}^{L} p_i(x)^2 dx + c$	no	yes	no
UMNN [Wehenkel and Louppe, 2019]	$\int_0^x f(x)dx + \beta \ \ (f > 0)$	no	yes	no
AUTM (new)	$x + \int_0^1 g(v(t), t)dt$	yes	yes	yes

3 ATOMIC UNRESTRICTED TIME MACHINE (AUTM) FLOWS

Lots of efforts have been made in recent years to construct a coupling function or transformer $q(x)$ that is strictly monotone (thus invertible) as well as expressive enough. As shown in Table 1, sophisticated machinery is used to improve the expressive power and meanwhile ensure the monotonicity (invertibility) of q, which usually requires restricting the form of q or the model parameters, e.g. in [Ziegler and Rush, 2019, Ho et al., 2019, Durkan et al., 2019a,b]. Moreover, since q is a complicated nonlinear function, an analytic format of q^{-1} is generally not available and thus numerical root-finding algorithms are often used to compute the inverse transformation. It is natural to ask whether *there exists a family of universal monotone functions with analytic inverses and unrestricted model parameters or representations?*

We propose a new approach to construct a monotone $q(x)$ based on integration with respect to a free *latent* variable. The introduction of the latent variable enables the use of unconstrained transformations and renders exceptional flexibility for manipulating the transformation and its inverse. The resulting coupling flows and autoregressive flows can be inverted easily using the inverse formula and the Jacobian is triangular.

We define $q : \mathbb{R} \to \mathbb{R}$ through a latent function $v(t)$ by

$$q : x \to y = v(1), \quad v(t) = x + \int_0^t g(v(t), t)dt. \tag{4}$$

where $g(v, t)$ is uniformly Lipschitz continuous in v and continuous in t $(0 \leq t \leq 1)$. Equivalently, $v(t)$ satisfies $v'(t) = g(v(t), t)$ and $v(0) = x$. So the transformation from x to y can be viewed as an evolution of the latent dynamic $v(t)$. Note that the integral in (4) is with respect to t instead of x and the integrand does *not* have to be positive. Moreover, we can easily find the inverse transform as

$$q^{-1} : y \to x = v(0), \ v(t) = y + \int_1^t g(v(t), t)dt. \tag{5}$$

An explicit inverse formula brings significant computational speedups as compared to existing integral-based methods that rely on numerical root-finding algorithms. Compared with other coupling functions/transformers, there is no assumption on g other than Lipschitz continuity. We will show later in Section 4 that the transformation in (4) is strictly increasing and is general enough to approximate *any* given continuously increasing map.

AUTM We term the mapping q in (4) an "Atomic Unrestricted Time Machine (AUTM)". "**Atomic**" means that (i) q is always *univariate* and *scalar-valued*; (ii) $g(v, t)$ can be as simple as an affine function in v and does *not* have to be a deep neural network to achieve good performance; (iii) the computation of the transformation as well as Lipschitz constant is *lightweight*; (iv) the model can be easily incorporated into existing normalizing flow architectures. In fact, we will see later in Section 4 that *any* monotonic normalizing flow is a limit of AUTM flows. "**Unrestricted**" means that there is no constraint on parameters or function forms in the model. "**Time Machine**" refers to the fact that the model is automatically invertible and the computation of inverse is essentially a reverse of integral limits. Thanks to the "atomic" property, AUTM can be easily incorporated into triangular flow architectures such as coupling flows and autoregressive flows.

AUTM coupling flows Given a D dimensional input $x = (x_1, \ldots, x_D)$ and $d < D$, the AUTM coupling layer $f : \mathbb{R}^D \to \mathbb{R}^D$ is defined as follows.

$$y_{1:d} = x_{1:d}, \quad y_{d+1:D} = q(x_{d+1:D}; \theta(x_{1:d})), \tag{6}$$

where q is the AUTM map defined in (4). Let f_* denote the AUTM coupling layer with q applied to $x_{1:d}$ instead of

$x_{d+1:D}$, i.e.

$$y_{1:d} = q(x_{1:d}; \theta(x_{d+1:D})), \quad y_{d+1:D} = x_{d+1:D}. \qquad (7)$$

The AUTM coupling flow is defined by stacking f and f_* in a multi-layer fashion as shown in Figure 1(left). Since the inverse q^{-1} is given in (4), the inverse transformation f^{-1} or f_*^{-1} is readily available.

AUTM autoregressive flows An autoregressive flow is composed of autoregressive mappings, similar to coupling layers in coupling flows. The AUTM autoregressive mapping on $\mathbb{R}^D$ is defined as

$$f(x;\theta) = (q_1(x_1;\theta_1), \ldots, q_D(x_D;\theta_D(x_{1:D-1}))), \qquad (8)$$

where each $q_k(x_k;\theta_k(x_{1:k-1}))$ is an AUTM map with unrestricted conditioner $\theta_k(x_{1:k-1})$. See Figure 1(right) for an illustration. The inverse mapping f^{-1} can be computed rapidly by first computing q_1^{-1} (which gives x_1) and then $q_2^{-1}, q_3^{-1}, \ldots, q_D^{-1}$, where each q_k^{-1} is explicitly given by (5). The Jacobian of f in (8) is lower triangular.

Jacobian determinant and log-density The Jacobian of AUTM flow is lower triangular. It will be shown in Theorem 1 that the derivative of the mapping $q(x)$ is given by $q'(x) = \exp\left(\int_0^1 \frac{\partial g}{\partial v}(v(t), t)dt\right)$. Thus one can immediately derive the Jacobian determinant of AUTM flow. If coupling layer is used, the Jacobian determinant is

$$\exp\left(\int_0^1 \sum_{k=d+1}^{D} \frac{\partial g}{\partial v}(v_k(t), t; \theta(x_{1:d}))dt\right)$$

If autoregressive layer is used, the Jacobian determinant is

$$\exp\left(\int_0^1 \sum_{k=1}^{D} \frac{\partial g}{\partial v}(v_k(t), t; \theta(x_{1:k-1}))dt\right).$$

From the above formulas and (1), the change of log-density of an AUTM flow follows immediately. If coupling layer is used, then

$$\log p_Y(y) = \log p_X(x) - \int_0^1 \sum_{k=d+1}^{D} \frac{\partial g}{\partial v}(v_k(t), t; \theta(x_{1:d}))dt.$$

If autoregressive layer is used, then

$$\log p_Y(y) = \log p_X(x) - \int_0^1 \sum_{k=1}^{D} \frac{\partial g}{\partial v}(v_k(t), t; \theta(x_{1:k-1}))dt.$$

4 MONOTONICITY AND UNIVERSALITY OF AUTM FLOWS

In this section, we present several key results on AUTM flows, including monotonicity and universality. The proofs can be found in Appendix A. The derivative of $q(x)$ in (4) is explicitly available.

Theorem 1 (Derivative)**.** *Let $q(x)$ be defined in* (4)*. Then $q'(x) = \exp\left(\int_0^1 \frac{\partial g}{\partial v}(v(t), t)dt\right)$.*

Theorem 2 (Monotonicity)**.** *The mapping $q(x)$ defined in* (4) *is invertible and strictly increasing.*

The expressive power of AUTM is summarized in the following theorems, which state that one can approximate *any* monotone continuous transformation with a family of AUTM transformations.

Theorem 3 (AUTM as a universal monotone mapping)**.** *Let $\mathcal{C}$ be the space of continuous functions on $\mathbb{R}$ with compact-open topology and let $\mathcal{M} \subset \mathcal{C}$ be the cone of (strictly) increasing continuous functions. Then the set of AUTM bijections*

$$\mathcal{Q} = \{q(x) \text{ in } (4) : v(0) = x \in \mathbb{R}\}$$

is dense in $\mathcal{M}$.

Theorem 4 (AUTM as a universal flow)**.** *For any coupling or autoregressive flow $F = F_1 \circ F_2 \circ \cdots \circ F_p$ from $\mathbb{R}^D$ to $\mathbb{R}^D$, where each F_k is a triangular monotone transformation, there exists a family of AUTM flows $\{T_s\}_{s>0} = \{T_{s,1} \circ T_{s,2} \circ \cdots \circ T_{s,p}\}_{s>0}$ such that T_s converges to F pointwisely and compactly as $s \to 0$. In fact, there exists a family such that the convergence rate is $O(e^{-\frac{1}{s}})$ as $s \to 0$.*

Theorem 3 and Theorem 4 imply that all coupling flows and autoregressive flows can be approximated arbitrarily well by AUTM flows. In the following, we present an explicit construction of such a family of AUTM flows that converge to an arbitrarily given monotonic normalizing flow. The convergence result provides a link between the proposed AUTM flows and existing monotonic normalizing flows and illustrates the representation power of AUTM. Notice that every AUTM flow has explicit inverse, so the universality result in this section shows that we can approximate any *monotonic* flow by a flow with *explicit* inverse.

Universal AUTM flows. Let $\phi(x)$ be an arbitrary increasing continuous function on $\mathbb{R}$. Define a family of AUTM bijections parametrized by $s > 0$ as follows:

$$q_s(x) = x + \int_0^1 \phi(v_s(te^{-\frac{1}{s}})) - v_s(te^{-\frac{1}{s}})dt, \qquad (9)$$

where $v_s(t) = x + \int_0^t \phi(v_s(ze^{-\frac{1}{s}})) - v_s(ze^{-\frac{1}{s}})dz$. Then it can be shown that (see Appendix A - Proof of Theorem 3): $q_s|_K$ converges to $\phi|_K$ uniformly on any compact set $K \subset \mathbb{R}$ as $s \to 0$ and the convergence rate is $O(e^{-\frac{1}{s}})$. Based on q_s, one can construct a family of AUTM coupling flows or autoregressive flows that converge to the given flow based on ϕ.

In fact, the family of AUTM flows in (9) is just one particular family of universal AUTM flows with more general

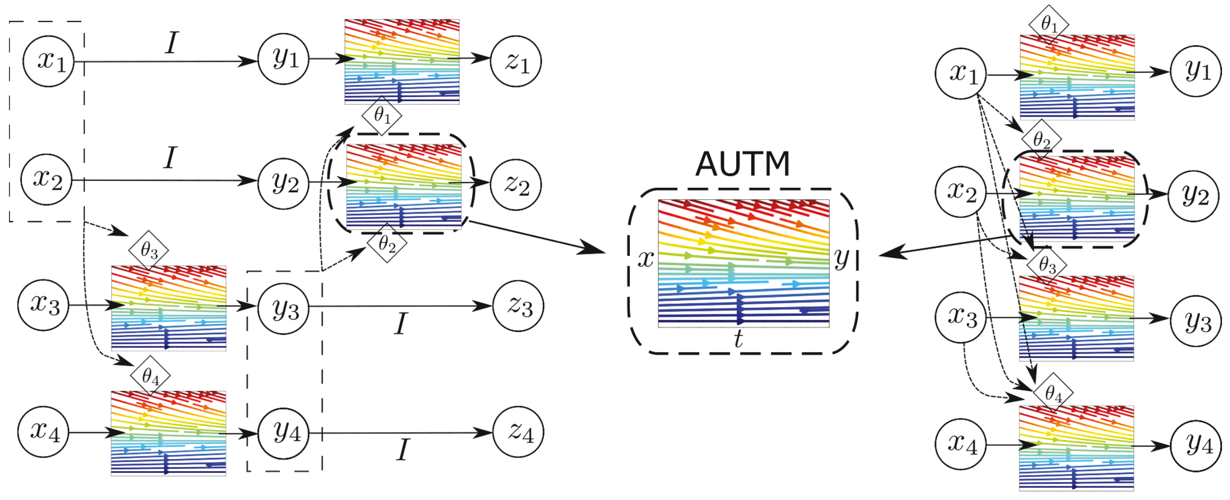

Figure 1: Left to right: AUTM coupling flow (2 layers), AUTM map, AUTM autoregressive flow (1 layer).

integrands. This is formalized in the theorem below regarding universal AUTM flows that generalize (9). The proof is given in Appendix A.

Theorem 5. *For $s > 0$, let $\kappa_s \in C([0,1])$ be a positive function such that*

$$\int_0^1 \kappa_s(t)dt = 1 \text{ and } \int_0^{e^{-\frac{1}{s}}} \kappa_s(t)dt \to 0 \text{ as } s \to 0. \quad (10)$$

Given any increasing continuous function $\phi(x)$, we define q_s as follows $q_s(x) = x + \int_0^1 g_s(v_s, t)dt$, where $g_s(v,t) = \kappa_s(t)[\phi(v(te^{-\frac{1}{s}})) - v(te^{-\frac{1}{s}})]$ and $v_s(t) = x + \int_0^t g_s(v_s, z)dz$. Then as $s \to 0$, $q_s|_K$ converges to $\phi|_K$ uniformly for any compact set $K \subset \mathbb{R}$.

Remark 1. The proof of Theorem 3 in Appendix A indicates that it may be sufficient to choose $g(v,t)$ as a function that is explicit in v only. In fact, it is shown in the proof that, after a scaling of the time variable, the equation for the approximant v_s is autonomous. Thus we expect good approximation power if $g(v,t) = \frac{dv}{dt}$ is explicit in v only.

5 RELATED WORK

Integral-based methods: SOS and UMNN. Existing integral-based methods like [Jaini et al., 2019, Wehenkel and Louppe, 2019] require the integrand to be positive and the inverse transformation is *not* analytically available. When computing the inverse transformation, AUTM only requires evaluating *one* integral while above methods require evaluating k different integrals for k iterations in the root-finding algorithm. Moreover, [Wehenkel and Louppe, 2019] models the integrand as a positive neural network while AUTM allows using general function classes with better computational efficiency than neural networks.

Neural ODE and FFJORD. Neural ODE [Chen et al., 2018] and Free-form Jacobian of Reversible Dynamics (FFJORD) [Grathwohl et al., 2019] use a dynamical system to model the transformation, in which a *multivariate* neural network is used to model the *vector-valued* dynamics. The use of the integral representation in AUTM to enable the analytic inverse transformation is inspired by Neural ODE and FFJORD. AUTM differs from FFJORD in three aspects. Firstly, FFJORD is not computationally-efficient because it relies on the the neural network to model the dynamics and the Jacobian is a fully dense matrix whose log-determinant can not be computed easily. AUTM, similar to other integral-based flows, employs (1) a coupling or autoregressive structure so that the Jacobian is a triangular matrix and the log-density is explicitly given; (2) decoupled entrywise transformations in each layer. The integrand g in AUTM is specified by the user and can be as simple as polynomials while still achieve competitive results. Secondly, it is rigorously proved that AUTM admits universal approximation property, while it is unclear weather FFJORD admits universality. Thirdly, AUTM is a monotonic flow while no monotonicity result for FFJORD can be found. Overall, AUTM benefits from the triangular Jacobian and decoupled transformations with simple integrands and is thus much more computationally efficient than FFJORD. It is also possible to incorporate hierarchical structures in Cai et al. [2018], Erlandson et al. [2020] to further improve the efficiency of AUTM, which will be pursued in a later date.

Other models. Affine coupling flows [Dinh et al., 2014, 2017] use an affine coupling function so that the inverse is trivial to compute. Recent developments consist in using *nonlinear* monotone functions to improve the expressiveness of affine coupling functions. To guarantee invertibility, unlike integral-based models, many architectures e.g. [Ziegler and Rush, 2019, Durkan et al., 2019b, Huang et al., 2018, De Cao et al., 2020] restrict model parameters, which limit the expressive power of the model and the training efficiency. Moreover, computation of the inverse transformation usually requires numerical root-finding methods since a tractable analytic inverse is often *not* available. The AUTM framework circumvents those issues by using an integral representation with respect to a *latent* variable. The inverse is explicit, regardless of the choice of model classes or parameters. This enables the rigorous justification of the universality of AUTM flows.

6 EXPERIMENT

In this section, we present experiments to evaluate our model. In Section 6.1, we perform density estimation on five tabular datasets and compare with other methods. In Section 6.2, we train our model on the CIFAR10 and ImageNet32 datasets for image generation. Experiments are conducted on either Nvidia 3080 GPU or Nvidia V100 GPU. All experimental details are provided in Appendix B.

For image datasets, we model $g(v, t)$ in (4) as a quadratic polynomial in v. For density estimation, we consider three different choices of $g(v, t)$ in (4): $g(v, t) = av + b + cv^2$, $g(v, t) = av + b + cv^3$ and $g(v, t) = av + b + c\sigma(v)$, where σ denotes the sigmoid function (g is chosen to be explicit only in v due to *Remark 1*). Note that this is different from many existing methods that rely on deep neural networks to model the core function in the model, such as UMNN [Wehenkel and Louppe, 2019] for the positive integrand, NAF [Huang et al., 2018] and BNAF [De Cao et al., 2020] for the autoregressive mapping, neural ODEs [Chen et al., 2018] and FFJORD [Grathwohl et al., 2019] for the entire dynamical system. We show in the following that, compared to the state-of-the-art models, our proposed AUTM model achieves excellent performance with simple choices of g. More importantly, for high-dimensional image datasets like ImageNet32, AUTM model requires significantly less model parameters compared to other models.

6.1 DENSITY ESTIMATION

Data sets and baselines. We first evaluate our method on four datasets from the UCI machine-learning repository [Dheeru and Taniskidou, 2017]: POWER, GAS. HEPMASS, MINIBOONE, and also the BSDS300 dataset, which are all preprocessed by [Papamakarios et al., 2017]. We compare our method to several existing normalizing flow models, including Real NVP [Dinh et al., 2017], Glow [Kingma and Dhariwal, 2018], RQ-NSF [Durkan et al., 2019b]), CP-FLOW [Huang et al., 2021], FFJORD [Grathwohl et al., 2019], UMNN [Wehenkel and Louppe, 2019] and autoregressive models such as MAF [Papamakarios et al., 2017], MADE [Germain et al., 2015] and BNAF [De Cao et al., 2020].

Model configuration and training. We use 10 (or 5) masked AUTM layers and set the hidden dimensions 40 times (or 10 times) the dimension of the input. We apply a random permutation of the elements of each output vector, as the masked linear coupling layer, so that a different set of elements is considered at each layer, which is a widely used technique [De Cao et al., 2020], [Dinh et al., 2017], [Papamakarios et al., 2017]. We use Adam as the optimizer and select hyperparameters after an extensive grid search.

Results. We report average negative log-likelihood estimates on the test sets in Table 2. It can be observed that AUTM consistently outperforms Real NVP, Glow, MADE, MAF, CP-Flow. On MINIBONDE dataset, our models perform better than all other models except BNAF. On POWER, HEPMASS, BSDS300 dataset, one of the AUTM models performs best among all baseline models. On GAS dataset, our results are competitive at either top 2 or top 3 spot with a tiny gap from the best.

6.2 EXPERIMENT ON IMAGE DATASET

Data sets and baselines. We then evaluate our method on the CIFAR10 [Krizhevsky, 2009] and ImageNet32 [Oord et al., 2016] datasets. Unlike density estimation tasks, image datasets are large-scale and high-dimensional. As a result, there are only a limited number of models available for image tasks. We calculate bits per dim and compare with other normalizing flow models including Real NVP [Dinh et al., 2017], Glow [Kingma and Dhariwal, 2018], Flow++ [Ho et al., 2019], NSF [Durkan et al., 2019b]. The results of bits per dim for each model are given in Table 3. We also show sampled images by using AUTM in Figure 2.

Model configuration and training. We use 14 AUTM coupling layers with 8 residual blocks for each layer (cf. [Kaiming et al., 2016]) in our model. Each residual block has three convolution layers with 128 channels. Our method is trained for 2500 epochs with batch size 64 for CIFAR10, and 50 epochs with batch size 64 for ImageNet32 dataset.

Results. From Table 3, we can find our method outperforms all baselines with the exception of Flow++ [Ho et al., 2019] on CIFAR10 dataset, which uses a variational dequantization technique. In addition, Table 3 shows AUTM is not sensitive to the size of the dataset when we transfer from CIFAR10 to ImageNet32. Comparing the number of parameters used for each method, we find that AUTM yields the best performance with much fewer parameters compared to other models for ImageNet32. In particular, the number of model parameters of Flow++ increases dramatically as we move from CIFAR10 to ImageNet32 while *affine* models like Real NVP and Glow only have a moderate increase in the number of parameters. This is reasonable since Flow++ is the only *nonlinear* model other than AUTM. It demonstrates that AUTM, as a *nonlinear* model, yields

Table 2: Average test negative log-likelihood (in nats) of tabular datasets (lower is better). Numbers in the parenthesis are standard deviations. Average/standard deviation is computed by 3 runs. The best performance for each dataset is highlighted in boldface.

Model	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
Real NVP[Dinh et al., 2017]	-0.17(0.01)	-8.33(0.14)	18.71(0.02)	13.55(0.49)	-153.28(1.78)
Glow[Kingma and Dhariwal, 2018]	-0.17(0.01)	-8.15(0.40)	18.92(0.08)	11.35(0.07)	-155.07(0.03)
FFJORD[Grathwohl et al., 2019]	-0.46(0.01)	-8.59(0.12)	14.92(0.08)	10.43(0.04)	-157.40(0.19)
UMNN[Wehenkel and Louppe, 2019]	-0.63(0.01)	-10.89(0.70)	**13.99(0.21)**	9.67(0.13)	**-157.98(0.01)**
MADE[Germain et al., 2015]	3.08(0.03)	-3.56(0.04)	20.98(0.02)	15.59(0.50)	-148.85(0.28)
MAF[Papamakarios et al., 2017]	-0.24(0.01)	-10.08(0.02)	17.70(0.02)	11.75(0.44)	-155.69(0.28)
CP-Flow[Huang et al., 2021]	-0.52(0.01)	-10.36(0.03)	16.93(0.08)	10.58(0.07)	-154.99(0.08)
BNAF[De Cao et al., 2020]	-0.61(0.01)	-12.06(0.09)	14.71(0.38)	**8.95(0.07)**	-157.36(0.03)
RQ-NSF (C)[Durkan et al., 2019b]	**-0.64(0.01)**	**-13.09(0.02)**	14.75(0.03)	9.67(0.47)	-157.54(0.28)
AUTM: $g(v,t) = av + b + cv^2$	-0.63(0.03)	-12.24(0.04)	14.62(0.30)	9.16(0.18)	-157.45(0.05)
AUTM: $g(v,t) = av + b + cv^3$	**-0.64(0.01)**	-12.37(0.06)	14.76(0.25)	9.33(0.10)	-157.54(0.10)
AUTM: $g(v,t) = av + b + c\sigma(v)$	-0.61(0.02)	-12.03(0.06)	14.94(0.33)	9.29(0.20)	-157.28(0.14)

better efficiency and robustness in terms of parameter use.

Table 3: Results of BPD (bits per dim) on CIFAR10 and ImageNet32 datasets. Results in brackets indicate the model using variational dequantization.

Model	CIFAR10 BPD	CIFAR10 parameters	ImageNet32 BPD	ImageNet32 parameters
Real NVP	3.49	44.0M	4.28	66.1M
Glow	3.35	44.7M	4.09	67.1M
Flow++	(3.08)	31.4M	(3.86)	169.0M
RQ-NSF (C)	3.38	11.8M	-	-
Our Method	3.29	35.5M	3.80	35.5M

6.3 NUMERICAL INVERSION OF AUTM

Existing normalizing flow models with no explicit inverse usually employ bisection to compute the inverse transformation. For AUTM, more options are available to compute the inverse mapping, such as fixed point iteration, which offers faster convergence than bisection. We compare the performance of bisection and fixed point iteration for AUTM by considering a toy example where function g in (5) is chosen as a specific quadratic polynomial in v and the input variable x is randomly chosen from the unit interval. We use the discretized version (five-point trapezoidal rule) of the inverse formula in (5) as the initial guess for fixed point iteration. Table 4 shows that this leads to significantly fewer (around 50%) iteration steps than bisection to achieve the same solution accuracy.

Table 4: Number of steps (averaged over 1000 random input) of root-finding method to reach a certain error tolerance

Error tolerance	1e-3	1e-4	1e-5	1e-6
Iteration Method	4.565	6.642	8.658	10.831
Binary Search	8.967	12.398	15.668	19.073

6.4 COMPARISON OF FFJORD AND AUTM

Next, we test the runtime of FFJORD and AUTM on four datasets from the UCI machine-learning repository POWER, GAS. HEPMASS, MINIBOONE, all preprocessed by [Papamakarios et al., 2017]. For AUTM, we choose $g(v,t) = av + b + cv^3$. We define the target negative log-likelihood (target NLL) as the NLL achieved by FFJORD after training for 12 hours. The time for each method to reach the target NLL is reported in Table 5. It demonstrates that AUTM is significantly more efficient than FFJORD. This is attributed to the structural advantages of AUTM. Firstly, AUTM transforms the input vector $x \in \mathbb{R}^D$ in an entrywise fashion where the ith entry is a univariate function of x_i. In FFJORD, the transformation of x is characterized by a neural network where each output dimension is a nonlinear multivariate function of $x = (x_1, \ldots, x_D)$. Secondly, due to the aforementioned structural differences, AUTM has a triangular Jacobian while FFJORD has a dense Jacobian that induces difficulty in computing the log-determinant accurately. Thirdly, the integrand g in AUTM can be chosen

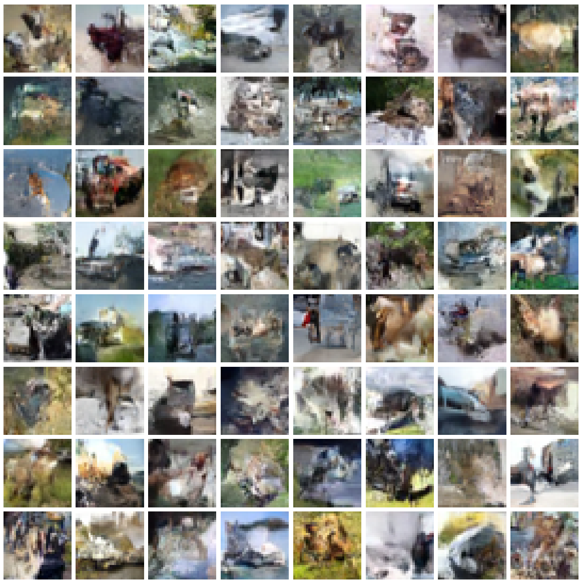
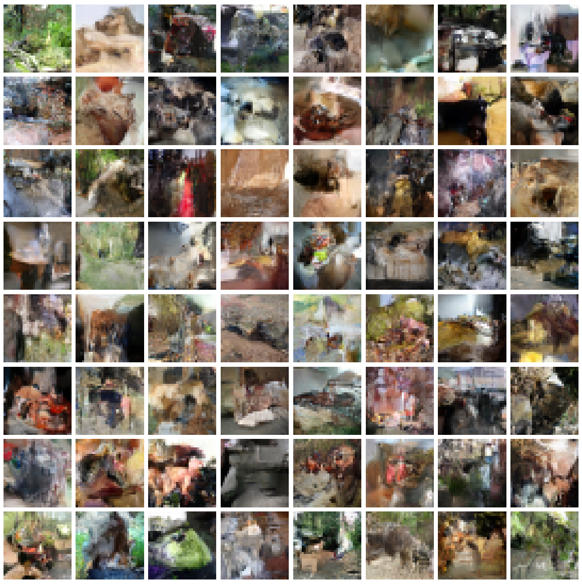
Figure 2: **Left**: Samples generated by using a pretrained model on CIFAR10 dataset. **Right**: Samples generated by using a pretrained model on ImageNet32 dataset.

as a simple function, for example, a quadratic function in v. In FFJORD, the integrand needs to be a neural network with D input variables $x_1, \ldots, x_D$. To evaluate the integral of such a complicated integrand accurately, a large number of quadrature nodes are needed, which will increase the cost in both forward and backward transformations. Additionally, AUTM enables the use of user-defined integrand g, which will be beneficial if prior information of the transformation to be learned is available.

Table 5: Runtime for FFJORD and AUTM to reach the target negative log-likelihood for each dataset.

Dataset	Target NLL	FFJORD	AUTM
POWER	0.23	12hr	6.92min
GAS	-5.24	12hr	3.67min
HEPMASS	21.85	12hr	7.40min
MINIBOONE	11.29	12hr	1.75min

7 SUMMARY

We have introduced a new nonlinear monotonic triangular flow called AUTM. AUTM leverages the explicit inverse formula used in FFJORD and the triangular Jacobian in coupling and autoregressive flows. Compared to FFJORD, AUTM demonstrates much better computational efficiency thanks to the triangular Jacobian structure, decoupled input dimensions, simple representation of the scalar integrand. Compared to other monotonic flows, AUTM has unrestricted parameters and more convenient computation of the inverse transformation. Theoretically, we have proved that AUTM is a universal approximator for any monotonic normalizing flow. The performance is demonstrated by comparison to the state-of-the-art models in density estimation and image generation. As a nonlinear monotonic flow, AUTM is able to achieve competitive performance on high-dimensional image datasets.

Author Contributions

D. Cai conceived the idea, wrote the paper and proofs. Y. Ji and H. He created the code. Q. Ye and Y. Xi organized the experiments and figures.

Acknowledgements

D. Cai and Y. Xi are supported by NSF awards OAC 2003720 and DMS 2038118. Y. Xi is also supported by DMS 2208412. Q. Ye and H. He are supported by NSF awards DMS 1821144 and IIS 1838200, respectively.

References

Michael S. Brown Abdelrahman Abdelhamed, Marcus A. Brubaker. Noise flow: Noise modeling with conditional normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3165––3173., 2019.

D. Cai, E. Chow, L. Erlandson, Y. Saad, and Y. Xi. Smash: Structured matrix approximation by separation and hierarchy. *Numerical Linear Algebra with Applications*, 25 (6):e2204, 2018.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, pages 6571–6583, 2018.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Block neural autoregressive flow. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 1263–1273, 2020.

D. Dheeru and E. Karra Taniskidou. Uci machine learning repository. In *URL http://archive.ics.uci.edu/ml.*, 2017.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Cubic-spline flows. *arXiv preprint arXiv:1906.02145*, 2019a.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, pages 7509–7520, 2019b.

L. Erlandson, D. Cai, Y. Xi, and E. Chow. Accelerating parallel hierarchical matrix-vector products via data-driven sampling. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 749–758. IEEE, 2020.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 881–889, 2015.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations*, 2019.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2722–2730, 2019.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087. PMLR, 2018.

Chin-Wei Huang, Ricky T. Q. Chen, Christos Tsirigotis, and Aaron Courville. Convex potential flows: Universal probability distributions with optimal transport and convex optimization, 2021.

Priyank Jaini, Kira A. Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 3009–3018, 2019.

He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31, pages 10215–10224, 2018.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Bogdan Mazoure, Thang Doan, Audrey Durand, R Devon Hjelm, and Joelle Pineau. Leveraging exploration in off-policy algorithms via normalizing flows. In *3rd Conference on Robot Learning*, 2019.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1747–1756, 2016.

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

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Zachary Ziegler and Alexander Rush. Latent normalizing flows for discrete sequences. In *International Conference on Machine Learning*, pages 7673–7682. PMLR, 2019.