RealMVP: A Change of Variables Method For Rectangular Matrix-Vector Products

Edmond Cunningham
University of Massachusetts

Madalina Fiterau
University of Massachusetts

Abstract

Rectangular matrix-vector products (MVPs) are used extensively throughout machine learning and are fundamental to neural networks such as multi-layer perceptrons. However, rectangular MVPs are notably missing not used as normalizing flow transforms. This paper identifies this methodological gap and plugs it with a tall and wide MVP change of variables formula. Our theory builds up to a scalable algorithm that envelops existing dimensionality increasing flow methods such as augmented flows (Huang et al., 2020). We show that tall MVPs are closely related to the stochastic inverse of wide MVPs and empirically demonstrate that they improve density estimation over existing dimension changing methods.

1 Introduction

Despite the prominence of rectangular matrix-vector products (MVPs) in neural network architectures, rectangular MVPs have yet to be analyzed in the context of density estimation. MVPs are prevalent in machine learning because they can learn useful linear combinations of inputs. Additionally, the input and output dimensionality can impact on the kind of representations that are learned. For example, it may be convenient to map to a higher dimension in order to perform richer computations (Dupont et al., 2019; Zhang et al., 2020) or map to a lower dimension in order to learn low dimensional structure. However these properties are inherently at odds with architectures that require invertibility, such as normalizing flows.

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

Normalizing flows (Dinh et al., 2014; Rezende and Mohamed, 2015; Papamakarios et al., 2019) transform simple probability distributions into complex ones using bijective functions whose parameters are trained for maximum likelihood exactly using the probability change of variables formula. One such transformation is the square matrix vector product, $x = Az$. As long as A is square and has full rank, there is a one-to-one mapping between x and z, so we can apply the change of variable formula to compute the density of x or z. However if A is rectangular, there is no longer a bijective map between every x and z because their dimensionalities will differ.

We cannot map across dimensions with deterministic functions in the context of normalizing flows because there will no longer be a bijective map between the entire input and output space. To see why this happens, consider a normalizing flow that uses a tall MVP to map its low dimensional latent state to a high dimensional data point. The tall MVP will only map to a hyperplane in the high dimensional data space rather than the entire output space. This is problematic because the model cannot assign a density to a point it cannot generate, so it will almost surely not be a valid model for real world datasets that can lie anywhere in the high dimensional space. By a similar argument, a flow that uses a wide MVP to map its high dimensional latent space to a low dimensional data point will have a subspace of latent vectors that map to the same data point. This is also an issue because the change of variable formula expects a single inverse associated with a data point, but the wide MVP gives us an infinite number of possible inverses. As we can see, the direct use of rectangular MVPs is fundamentally incompatible with normalizing flows. However, we can alleviate these issues by explicitly modeling these failure cases.

In this paper we discuss how to incorporate rectangular matrix vector products into normalizing flows. Given a ”tall” vector t and ”short” vector s, we show that to relate t and s with a matrix vector product it is necessary to also model a noise random variable, ϵ, that is placed in an orthogonal direction to the MVP in order to bypass the incompatibility between rectangular MVPs and normalizing flows. In particular, we examine the equation $t = As + U_{\perp}\epsilon$ where $s, \epsilon \sim p_s(s)p_\epsilon(\epsilon|s)$ and $U_{\perp}\epsilon$ is orthogonal to As for all s, ϵ. We show that the problems of representing data with a tall MVP and wide MVP and closely related and are in fact an instance the recently proposed SurVAE Flows (Nielsen et al., 2020). Furthermore, we introduce a novel algorithm called the RealMVP that affords the use of rectangular MVPs that scales well to high dimensions. We review related work in section 2 and review preliminary details of our approach in section 3. Section 4 introduces our main theory that establishes a change of variables formula for rectangular MVPs and connects tall MVPs with additive orthogonal noise to wide MVPs. In section 5 we slightly modify the methodology of section 4 to circumvent computational issues and arrive at the RealMVP. Finally, in section 6 we demonstrate that our algorithm for rectangular MVPs outperforms the existing approach for rectangular MVPs.

In summary, our contributions are as follows:

- A theory that relates wide MVPs and tall MVPs in the context of probabilistic transformations (Fig.1).
- A scalable algorithm, called the RealMVP, to use rectangular MVPs for density estimation in existing machine learning libraries (Fig.2).

2 Related Work

Recent work in normalizing flows has focused on extending their capabilities to non-bijective functions (Nielsen et al., 2020), dimension changing (Huang et al., 2020; Cunningham et al., 2020) and generally bypassing the topological constraints of bijectivity (Cornish et al., 2020). Our work lies at the intersection of these methods.

SurVAE (Nielsen et al., 2020) introduced a framework to consider surjective functions as probabilistic transformations. A key insight to their work is that surjective functions can be stochastically inverted. This stochastic inverse is constructed so that its samples can always be deterministically reconstructed to the same value. The paper considered absolute value, maximum, sort, rounding, slice and ReLU surjections. We introduce a new kind of surjective transformation based on rectangular matrix multiplication - a tall MVP with orthogonal noise is the stochastic inverse of of a wide MVP with its pseudo-inverse matrix.

Injective functions can be composed with normalizing flows to build flows whose image is a manifold in the data space (Gemici et al., 2016; Brehmer and Cranmer, 2020; Kumar et al., 2020). These flows have no probability density defined for off-manifold data but instead have a density defined over their image. Given a sequence of injective and bijective functions $f_1, \dots, f_K$ and base distribution $p_z(z)$, the probability density function of $x = f_1 \circ \cdots \circ f_K(z) = f_{1:K}(z)$ is given by the equation (Gemici et al., 2016)

$$p(f_{1:K}(z)) = p_z(f_{1:K}(z))|\frac{df_{1:K}(z)}{dx}^T\frac{df_{1:K}(z)}{dx}|^{-\frac{1}{2}} \quad (1)$$

In contrast to the change of variable formula for bijective functions, the Jacobian determinant term does not decompose as the product of Jacobian determinants for each f_k - a computational challenge noted by (Brehmer and Cranmer, 2020; Kumar et al., 2020). Our proposed method does not suffer from this computational challenge and instead maintains a log likelihood contribution that decomposes for every flow layer specifically because we consider a model that is valid over the entire data space. In section 4.1 of the appendix, we outline an argument for why the dirac delta term in Eq.12 accounts for the difference between the two formulations.

The authors of (Cunningham et al., 2020) also considered tall MVPs with additive noise in a decomposable way, however they arrived at their objective from properties of the Gaussian distribution instead of a direct change of variables formula. Furthermore, they considered full rank noise instead of orthogonal noise which meant that their objective was not closed form in general.

Augmented flows (Huang et al., 2020) introduced a method to increase the dimensionality of vectors by padding data with noise. Our method generalizes their coupling based encoding transformation. When the matrix A in the methodology section of our paper is equal to $\begin{bmatrix} I \\ 0 \end{bmatrix}$, the RealMVP reduces to vector padding as presented in augmented flows. After the first encoding transformation, the rest of the augmented flows network uses a flow over the augmented data space. Although our method can be decomposed as the composition of an augmented flow layer followed by a square MVP, this composed approach requires storing the entire square matrix which can become computationally intractable in high dimensions. (Dupont et al., 2019) and (Zhang et al., 2020) proposed a similar method but in the context of neural ODEs (Chen et al., 2018) and residual flows (Chen et al.,

2019) and focused on using padding to overcome the topological limitations of bijectivty.

Continuously indexed normalizing flows (Cornish et al., 2020) show that in order to learn a target distribution with a different topology than that of the base distribution, the transformation cannot only use bijective functions. To do this in practice, the authors propose using bijective functions that are indexed with a continuous random variable with a practical algorithm that resembles that of the augmented flows paper. Like the continuously indexed flow, the wide MVP uses an auxiliary variable to augment a deterministic transformation, however our method does not use a bijective function but instead a surjective function.

3 Preliminaries

3.1 Properties involving the image of a tall matrix

We first review some identities involving the orthonormal basis for the image of a tall matrix and its orthogonal complement. An intuitive reference can be found at (Ratliff, 2014).

Let $A \in \mathbb{R}^{N\times M}, N > M$ be a tall matrix with full rank [1]. Its singular value decomposition is written as:

$$A = \begin{bmatrix} U_{\parallel} & U_{\perp} \end{bmatrix} \begin{bmatrix} S \\ 0 \end{bmatrix} V^T$$

$\begin{bmatrix} U_{\parallel} & U_{\perp} \end{bmatrix}$ and V are orthogonal matrices with sizes $\mathbb{R}^{N\times N}$ and $\mathbb{R}^{M\times M}$ respectively. The M columns of $U_{\parallel}$ span the image of A while the $N-M$ columns of $U_{\perp}$ span the orthogonal complement of the image of A.

The pseudo inverse of A is defined as

$$A^{+} = (A^T A)^{-1} A^T \tag{2}$$

and satisfies $A^{+}A = I$. The pseudo inverse exists and is unique for every matrix A with full rank (Penrose, 1955). We can also relate $U_{\parallel}$ and $U_{\perp}$ with A^{+}:

$$\begin{aligned} U_{\parallel} &= A(A^T A)^{\frac{-1}{2}}, \quad U_{\parallel}U_{\parallel}^T = AA^{+} \\ U_{\perp}U_{\perp}^T &= I - AA^{+} \end{aligned}$$

To keep notation clean later in this document, we will use $A^{\perp}$ to denote $U_{\perp}U_{\perp}^T$:

$$A^{\perp} := U_{\perp}U_{\perp}^T \tag{3}$$

[1]We will assume all matrices have full rank unless stated otherwise.

We further simplify the notation in this paper by writing wide matrices as the pseudo inverse of a tall matrix. Specifically, we consider wide matrices $A^{+} \in \mathbb{R}^{M\times N}, M < N$ whose singular value decomposition is:

$$A^{+} = V \begin{bmatrix} S^{-1} & 0 \end{bmatrix} \begin{bmatrix} U_{\parallel}^T \\ U_{\perp}^T \end{bmatrix}$$

In this setting, $U_{\parallel}$ and $U_{\perp}$ take a different interpretation - $U_{\perp}$ has columns that span the nullspace of A^{+} and $U_{\parallel}$ spans its orthogonal complement. This observation will play a vital role in the probability density function of wide MVPs.

3.2 Selected properties of the dirac delta function

The dirac delta function (S.N. Gurbatov, 2011) is used to develop key steps in our derivations. In this section we review the properties that are used in this paper.

Consider a random variable $z \sim p_z(z), \quad z \in \mathbb{R}^M$ and a function $f: \mathbb{R}^M \to \mathbb{R}^N$. Then the random variable $x = f(z)$ has the probability density function (Au and Tam, 1999):

$$p_x(x) = \int p_z(z)\delta(x - f(z))dz \tag{4}$$

Eq.4 serves as the starting point for our derivations. Although it is a valid equation, it is not practical because the integrand takes the values 0 or ∞. We can go from Eq.4 to a practical formula using the sifting property of the dirac delta function:

$$\int f(z)\delta(z-a)dz = f(a) \tag{5}$$

The general strategy we take in our derivations is to identify integrals of delta functions and make their form amenable to the sifting property. We can do this trivially when $\dim(x) = \dim(z)$ and f is invertible to get the standard change of variables formula, but is not as straight forward when $\dim(x) \neq \dim(z)$. The following three identities can help us get there:

$$\delta(Px) = \delta(x)|P|^{-1}, \quad P \in \mathbb{R}^{N\times N} \tag{6}$$

$$\int \delta(x - f(z))dx = 1, \quad \forall f(z) \tag{7}$$

$$\delta(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = \delta(x_1)\delta(x_2) \tag{8}$$

Eq.6 and Eq.7 are combined with Eq.8 to form key steps in the derivations for tall and wide MVPs respectively.

3.3 Factored Representation

We will refer to the trivial way to represent rectangular MVPs using a square and vector padding as the "factored" method. This method uses the combination of vector padding followed by a square MVP to simulate a rectangular MVP.

$$\begin{bmatrix} A \end{bmatrix} \text{ vs } \begin{bmatrix} & M & \end{bmatrix} \begin{bmatrix} I \\ \\ 0 \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} & A^{+} & \end{bmatrix} \text{ vs } \begin{bmatrix} I & & 0 \end{bmatrix} \begin{bmatrix} & & \\ & M^{-1} & \\ & & \end{bmatrix} \tag{10}$$

Although square MVPs are a standard part of normalizing flows and vector padding in normalizing flows can be performed using Augmented Flows (Huang et al., 2020), we will see in this paper that it can be beneficial to learn A directly.

4 Rectangular MVPs for Normalizing Flows

We will start by discussing tall MVPs and their shortcomings. To remedy their issues, we will add noise that is orthogonal their image, which will naturally lead to wide MVPs. Full derivations can be found in section 3 of the appendix.

4.1 Tall MVP

The probability density function of a tall matrix vector product is known (Dyrholm, 2004) but is only useful in practice when data is guaranteed to lie on its image.

Theorem 1. *Let $s \in \mathbb{R}^M$, $t \in \mathbb{R}^N$, $N > M$* [2] *and $A \in \mathbb{R}^{N\times M}$ be defined as in section 3. The probability density function of a tall MVP, $t = As$, is:*

$$t = As, \quad s \sim p_s(s) \tag{11}$$
$$p_t(t) = \delta(U_\perp^T t) p_s(A^+ t) |A^T A|^{-\frac{1}{2}} \tag{12}$$

Eq.12 has two notable components: $p_s(A^+t)|A^TA|^{-\frac{1}{2}}$ and $\delta(U_\perp^T t)$. The first term, $p_s(A^+t)|A^TA|^{-\frac{1}{2}}$, is the standard change of variables formula for a tall matrix when t is restricted to the image of A (Gemici et al., 2016). The other term, $\delta(U_\perp^T t)$, is a dirac delta function centered at the orthogonal component of t on the image of A.

[2]We use the variable names s and t instead of z and x to avoid confusion about the name of the data variable. We let t denote a data vector when when we use a tall MVP and s when we are using a wide MVP to reflect that data will be the "tall" or "short" compared to the latent variable.

Intuitively, the dirac delta term ensures that t lies on the image of A because it is 0 otherwise. In addition, it also ensures that $p_t(t)$ integrates to 1 over $\mathbb{R}^N$. However, the term cannot be used in practice because it is equal to ∞ when $t \in \text{Im}(A)$. We alleviate this issue by convolving $p_t(t)$ with some distribution in order to apply the sifting property of the dirac delta function (S.N. Gurbatov, 2011).

4.2 Tall MVP with additive orthogonal noise

We can remove the dirac delta term in Eq.12 by considering the sum of a tall matrix vector product with orthogonal noise. There are two parts to this formulation - the noise should be additive because the probability density of the summed vector is the convolution of density of each summand, and orthogonal to the image of A so that the convolution yields a closed form equation.

Theorem 2. *Let s, t and A be defined as before. Consider a random variable $\epsilon \in \mathbb{R}^{N-M}$. The vector $U_\perp \epsilon$ will be orthogonal to the image of A. The probability density function of a tall MVP with additive orthogonal noise, $t = As + U_\perp\epsilon$, is:*

$$t = As + U_\perp \epsilon, \quad s \sim p_s(s), \quad \epsilon \sim p_\epsilon(\epsilon|s) \tag{13}$$
$$p_t(t) = p_s(A^+t) p_\epsilon(U_\perp^T t | A^+ t) |A^T A|^{-\frac{1}{2}} \tag{14}$$

The theorem can be proven in a similar manner to Thm.1 (see the appendix section 3.2). This result might seem trivial - if we had started with the vector $\begin{bmatrix} s \\ \epsilon \end{bmatrix} \sim p_s(s)p_\epsilon(\epsilon|s)$, we could have computed the density of $t = \begin{bmatrix} U_\parallel & U_\perp \end{bmatrix} \begin{bmatrix} SV^T & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} s \\ \epsilon \end{bmatrix}$ using the well-known square MVP change of variable formula. The distinction is important for two reasons. First, explicitly considering s and ϵ has the intuitive explanation that if t represents noisy data that is generated around a hyperplane, then the generative model $t = As + U_\perp\epsilon$ suitably models these assumptions. Second, for t and s that are related as $s = A^+t$, the problem of inferring t from s requires identifying an ϵ so that $t = As + U_\perp\epsilon$. In other words, the tall MVP with additive orthogonal noise is the stochastic inverse of the wide MVP.

4.3 Wide MVP

Recall that it is sufficient to represent any wide matrix as the pseudo inverse of some tall matrix. Consider $s = A^+t$. The linear map A^+ is surjective, so there is a space of possible t that yields the same value of s.

Theorem 3. *Let s, t and A be defined as before. The probability density function of a wide MVP, $s = A^+t$,*

Method	Tall	Wide	Δ Log Likelihood	Runtime	Memory
SVD	$t = As + U_\perp \epsilon, \quad \epsilon \sim p_\epsilon(\epsilon\|s)$	$\begin{bmatrix} s \\ \epsilon \end{bmatrix} = \begin{bmatrix} A^+ \\ U_\perp^T \end{bmatrix} t$	$\log p_\epsilon(\epsilon\|s) - \frac{1}{2}\|A^T A\|$	$O(N^3)$	$O(NM)$
Factored	$t = M \begin{bmatrix} s \\ \epsilon \end{bmatrix}, \quad \epsilon \sim p_\epsilon(\epsilon\|s)$	$\begin{bmatrix} s \\ \epsilon \end{bmatrix} = M^{-1} t$	$\log p_\epsilon(\epsilon\|s) - \log \|M\|$	$O(N)/O(N^2)$	$O(N^2)$
RealMVP	$t = As + \gamma_\perp, \quad \gamma_\perp = A^\perp \gamma,$ $\gamma \sim N(\gamma\|\mu(s), \Sigma(s))$	$\begin{bmatrix} s \\ \gamma_\perp \end{bmatrix} = \begin{bmatrix} A^+ \\ A^\perp \end{bmatrix} t$	$\log Z(\mu(s) - \gamma_\perp\|A, \Sigma(s))$	$O(M^3)$	$O(NM)$

Figure 1: Overview of the algorithms presented in this paper. All three approaches define an invertible map between t and $(s, \epsilon/\gamma_\perp)$ so that it is possible to exactly recover the log-likelihood contribution. The SVD method is described in section 4.2, the factored in section 3.3 and RealMVP in section 5. The SVD and factored approaches use $\epsilon \in \mathbb{R}^{N-M}$ to model the noise orthogonal to As while the scalable method uses $\gamma_\perp \in \mathbb{R}^N$. When $M << N$, the RealMVP can run faster and cost less memory than the SVD and factored methods while still relating t and s through a rectangular MVP and providing a closed form likelihood contribution.

is

$$s = A^+ t, \quad t \sim p_t(t) \tag{15}$$

$$p_s(s) = \int p_t(As + U_\perp \epsilon) d\epsilon |A^T A|^{\frac{1}{2}} \tag{16}$$

Proof.

$$\begin{aligned}
p_s(s) &= \int p_t(t)\delta(s - A^+ t)dt \\
&= \int p_t(t)\delta(s - A^+ t) \underbrace{\int \delta(\epsilon - U_\perp^T t) d\epsilon}_{1} dt \qquad (17) \\
&= \int\int p_t(t)\delta(\begin{bmatrix} s \\ \epsilon \end{bmatrix} - \underbrace{\begin{bmatrix} A^+ \\ U_\perp^T \end{bmatrix}}_{R} t) d\epsilon dt \\
&= \int\int p_t(t)\delta(\underbrace{\begin{bmatrix} A & U_\perp \end{bmatrix}}_{R^{-1}} \begin{bmatrix} s \\ \epsilon \end{bmatrix} - t) \underbrace{|A^T A|^{\frac{1}{2}}}_{|R|^{-1}} d\epsilon dt \qquad (18) \\
&= \int p_t(As + U_\perp \epsilon) d\epsilon |A^T A|^{\frac{1}{2}}
\end{aligned}$$

□

The critical step in the proof is Eq.17 - it builds a new dirac delta term that is amenable to the sifting property of the dirac delta function in Eq.18.

Intuitively, Eq.16 says that the probability density of a short variable is proportional to the average density of all the tall vectors that could have generated it. In fact, this set of tall vectors is equal to $\{t : t = As + U_\perp \epsilon, \forall \epsilon\}$, which is exactly the image of a tall MVP where s is fixed.

Another way to understand the role of $U_\perp \epsilon$ is to connect it to the nullspace of A^+. For any ϵ, it is true that $A^+ U_\perp \epsilon = 0$. So given an inverse of s (which we choose to be As), we can construct a different inverse by simply adding $U_\perp \epsilon$ because $s = A^+(As + U_\perp \epsilon)$. This second interpretation directly connects our method to SurVAE Flows (Nielsen et al., 2020). The tall MVP has a deterministic inverse, $s = A^+ t$, while the wide MVP has a *stochastic* inverse, $t = As + U_\perp \epsilon$, where ϵ is drawn from some distribution $q(\epsilon|s)$. $q(\epsilon|s)$ can be introduced as an importance sampling distribution to help evaluate a lower bound on $\log p_s(s)$ (Jordan et al., 1998):

$$\log p_s(s) \geq \int q(\epsilon|s) \log \frac{p_t(As + U_\perp \epsilon)}{q(\epsilon|s)} d\epsilon |A^T A|^{\frac{1}{2}} \tag{19}$$

5 The RealMVP

The equations presented in the previous section require the ability to compute and potentially back-propagate through $U_\perp$ during training. This may be expensive or not available coding libraries for machine learning because computing $U_\perp$ involves computing the SVD of A. Ideally we should have expressions involving only simple computations with A. We can achieve this by considering a slightly different kind of orthogonal noise.

Recall that $A^\perp := U_\perp U_\perp^T = I - AA^+$. $A^\perp$ is not only easy to compute, but matrix multiplication with $A^\perp$ will project onto a space orthogonal to the image of A. For these reasons, we replace the orthogonal noise from Eq.13, $U_\perp \epsilon$ with a new noise variable, $A^\perp \gamma$ where $\gamma \in \mathbb{R}^N$ is drawn from a Gaussian with parametrized mean and covariance $\mathcal{N}(\gamma|\mu(s), \Sigma(s))$. The resulting structural equation for t is:

$$\begin{gathered}
\gamma \sim \mathcal{N}(\gamma|\mu(s), \Sigma(s)), \quad s \sim p_s(s) \\
t = As + \underbrace{A^\perp \gamma}_{\gamma_\perp},
\end{gathered} \tag{20}$$

A major difference between Eq.13 and Eq.20 is the choice of orthogonal noise. Previously we sampled

a RealMVP for tall matrix vector products

```
1: Input $t$, $A, \theta$
2: $s \leftarrow A^+t$                                  // Pseudo-inverse
3: $\mu, \Sigma \leftarrow \mathrm{NN}(s, \theta)$      // Features distribution
4: $\gamma_\perp \leftarrow t - AA^+t$                  // Get orthogonal features
5: $L \leftarrow \log Z(\mu - \gamma_\perp|A, \Sigma)$  // Likelihood contr.
6: return $s$, $L$
```

b RealMVP for wide matrix vector products

```
1: Input $s$, $A, \theta$
2: $\mu, \Sigma \leftarrow \mathrm{NN}(s, \theta)$       // Features distribution
3: $\gamma \sim \mathcal{N}(\mu, \Sigma)$                // Sample features
4: $\gamma_\perp \leftarrow \gamma - AA^+\gamma$         // Orthogonalize features
5: $t \leftarrow As + \gamma_\perp$                      // Generate output
6: $L \leftarrow -\log Z(\mu - \gamma_\perp|A, \Sigma)$  // Likelihood contr.
7: return $t$, $L$
```

Figure 2: RealMVP algorithm. The similarities between the tall (Fig.2a) and wide (Fig.2b) algorithms are highlighted by the colors. Each algorithm is based on the structural equation, $t = As + \gamma_\perp$ where $\gamma_\perp = A^\perp\gamma$ and $\gamma \sim N(\gamma|\mu(z), \Sigma(z))$ (see section 5 for a full discussion). The likelihood contribution of each algorithm depends on the orthogonal component and a learned distribution over this orthogonal space. These algorithms correspond to Eq.22 and Eq.26 respectively.

$\epsilon \in \mathbb{R}^{N-M}$ from an unconstrained distribution $p_\epsilon(\epsilon|s)$ to construct the vector $U_\perp\epsilon$ which is orthogonal to the image of A. Here we draw $\gamma \in \mathbb{R}^N$ from a constrained distribution $\mathcal{N}(\gamma|\mu(s), \Sigma(s))$, where $\mu(s)$ and $\Sigma(s)$ can be deep neural networks, in order to construct the vector $A^\perp\gamma$.

The choice to have γ lie in $\mathbb{R}^N$ is motivated by the fact MVPs with $A^\perp$ are significantly easier to compute than with $U_\perp$, and the choice of having its prior distribution be a parametrized Gaussian is so that the math for the marginal densities of t and s considerably simplify. This simplification is a result of the following definition:

Definition 4. *Let $x \in \mathbb{R}^N, z \in \mathbb{R}^M$, $A \in \mathbb{R}^{N\times M}$ and let $\mathcal{N}(x|\mu, \Sigma)$ denote the probability density function of a Gaussian centered at μ with covariance Σ. We construct the function $Z(x|A, \Sigma)$ to be equal to the following:*

$$\begin{aligned} Z(x|A, \Sigma) &:= \int \mathcal{N}(x|Az, \Sigma)dz \qquad (21)\\ &= \frac{\mathcal{N}(x|0, \Sigma)}{\mathcal{N}(h|0, J)}|J|^{-1} \\ J &= A^T\Sigma^{-1}A, \quad h = A^T\Sigma^{-1}x \end{aligned}$$

Refer to appendix section 3.4 for the derivation.

5.1 Tall MVP with additive orthogonal noise

Next, we derive the probability density function for the RealMVP.

Theorem 5. *Let s, t and A be defined as before. If $t = As + A^\perp\gamma$, $\quad s \sim p_s(s)$, $\quad \gamma \sim \mathcal{N}(\gamma|\mu(s), \Sigma(s))$, then the probability density function of t is*

$$p_t(t) = p_s(A^+t)Z(\mu(A^+t) - A^\perp t|A, \Sigma(A^+t)) \qquad (22)$$

Proof.

$$\begin{aligned} p_t(t) &= p_s(A^+t)p_{U_\perp^T\gamma}(U_\perp^T t|A^+t)|A^TA|^{\frac{-1}{2}} \\ &= p_s(A^+t)\int p_\gamma(A^\perp t + U_\parallel r|A^+t)dr|A^TA|^{\frac{-1}{2}} \\ &= p_s(A^+t)\int p_\gamma(A^\perp t + As|A^+t)ds \\ &= p_s(A^+t)\int \mathcal{N}(\mu(A^+t) - A^\perp t|As, \Sigma(A^+t))ds \\ &= p_s(A^+t)Z(\mu(A^+t) - A^\perp t|A, \Sigma(A^+t)) \end{aligned}$$

□

This new pdf does not depend on $U_\perp$ and is still available in closed form. In practice, we can define $p_s(s)$ using a normalizing flow and $(\mu(s), \Sigma(s))$ using an unconstrained neural network. $p_s(s)$ will define learn the distribution on the image of A while $\mathcal{N}(\mu(s), \Sigma(s))$ will learn the distribution over the orthogonal component of the image at a given point. The final learning algorithm in Fig.2a breaks a tall vector into a parallel and an orthogonal component, passes the parallel component back through the rest of the normalizing flow, and evaluates the likelihood of the orthogonal component.

5.2 Wide MVP

The RealMVP can also be used for wide matrix vector products, however the derivation of the density is a bit more involved. We begin with two lemmas that are proved in appendix section 3.6 and 3.7.

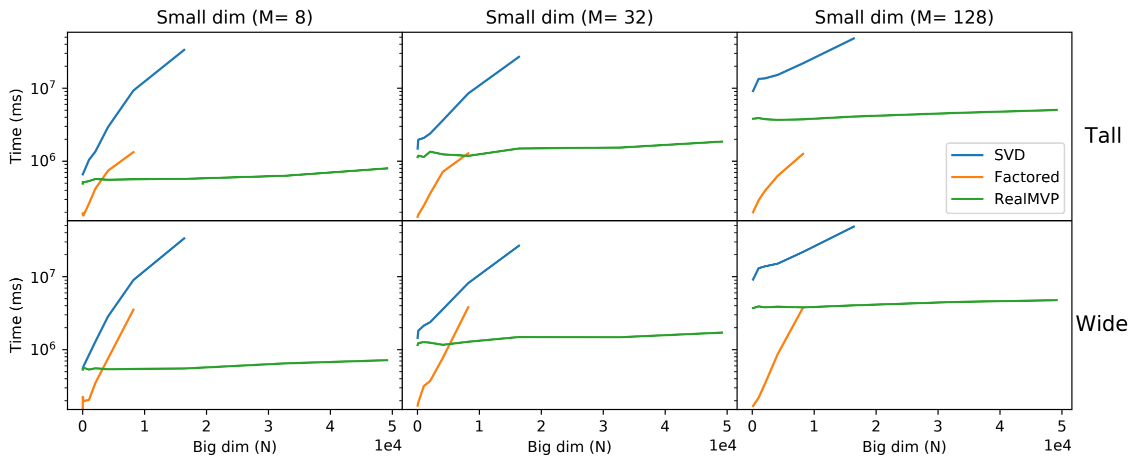

Figure 3: Runtime comparison between the SVD, factored and RealMVP algorithms for increasing dimensions N. Each plot shows how execution time (milliseconds) varies with the dimensionality of the tall vector (N). We repeat this experiment over three different short vector dimensionalities (M) and over both the tall and wide computations. See Fig.1 for an overview of the different algorithms. Each experiment is run on roughly as large of a problem as possible before our hardware was not able to allocate enough memory to complete the tall and wide computations. Our experiments were run on one NVIDIA 1080Ti GPU with 11GB of memory. The SVD and factored algorithms are not able to run in problems where $N > 20,000$ while the RealMVP continues to work even when $N = 50,000$.

Lemma 6. *Let* $B = \begin{bmatrix} A^+ \\ A^\perp \end{bmatrix}$. *Then* $B^+ = \begin{bmatrix} A \\ A^\perp \end{bmatrix}$ *and* $|B^TB| = |A^TA|^{-1}$.

Lemma 7. *Let* $B = \begin{bmatrix} A^+ \\ A^\perp \end{bmatrix}$. *Consider the left singular vectors of* B *that are orthogonal to the image of* B, $U_\perp(B)$. *Then there exists an orthogonal matrix,* Q, *such that* $U_\perp(B) = \begin{bmatrix} 0 \\ U_\| \end{bmatrix} Q$.

With these lemmas we begin the derivation of the scalable wide MVP algorithm. We start by deriving the change of variables formula for a wide MVP in a different form than Eq.16.

Lemma 8. *Let* s, t *and* A *be defined as before. The probability density function of* $s = A^+t$, *where* $t \sim p_t(t)$ *is*

$$p_s(s) = \int \delta(U_\|^T\gamma_\perp)p_t(Ax + A^\perp\gamma_\perp)d\gamma_\perp|A^TA|^{\frac{1}{2}} \quad (23)$$

Proof.

$$\begin{aligned} p_s(s) &= \int p_t(t)\delta(s - A^+t)dt \\ &= \int p_t(t)\delta(s - A^+t)\int \delta(\gamma_\perp - A^\perp t)d\gamma_\perp dt \\ &= \int\int p_t(t)\delta(\begin{bmatrix} s \\ \gamma_\perp \end{bmatrix} - \underbrace{\begin{bmatrix} A^+ \\ A^\perp \end{bmatrix}}_{B} t)d\gamma_\perp dt \\ &= \int \delta(U_\perp^T(B)\begin{bmatrix} s \\ \gamma_\perp \end{bmatrix})p_t(B^+\begin{bmatrix} s \\ \gamma_\perp \end{bmatrix})d\gamma_\perp|B^TB|^{\frac{-1}{2}} \\ &= \int \delta(U_\|^T\gamma_\perp)p_t(As + A^\perp\gamma_\perp)d\gamma_\perp|A^TA|^{\frac{1}{2}} \end{aligned}$$

□

Eq.23 is not outright useful - it is an intractable integral with an integrand that takes the value 0 of ∞. However, we can introduce a specially constructed importance sampler, $q(\gamma_\perp|s)$, in order to cancel the delta term. We prove the following lemma in appendix section 3.9:

Lemma 9. *Let* $\gamma_\perp = A^\perp\gamma, \quad \gamma \sim N(\gamma|\mu, \Sigma)$. *Then*

$$p_{\gamma_\perp}(\gamma_\perp) = \delta(U_\|^T\gamma_\perp)Z(\mu - \gamma_\perp|A, \Sigma)|A^TA|^{\frac{1}{2}} \quad (24)$$

Using lemmas 8 and 9, we can easily derive the final pdf.

Theorem 10. *Let* s, t *and* A *be defined as before. Also let* $q(\gamma_\perp|s)$ *be the pdf of* $\gamma_\perp$ *where* $\gamma_\perp = A^\perp\gamma, \quad \gamma \sim$

	Circles	Swiss roll	Grid
Factored	1.32	3.71	2.95
RealMVP	**1.31**	**3.60**	**2.85**

(a) Directly learning a rectangular matrix instead of a large square matrix like in Eq.9 yields better density estimation results. The table shows negative log likelihood over the test set for synthetic 2d datasets (lower is better).

Multiscale type	Bits/dimension ↓
Baseline	4.67
Factored	3.88
RealMVP	**3.72**

(b) We interpret the multiscale architecture (Dinh et al., 2017) as a pixel-wise tall MVP and find that learning a tall MVP. The table shows bits per dimension over the test set of the CIFAR-10 (Krizhevsky) dataset (lower is better).

Figure 4: Density estimation comparison learning rectangular MVPs with our approach (Full) and an existing method (Factored). See Eq.9 and Eq.10 for the comparison between the full and factored parametrization of rectangular matrices.

$N(\gamma|\mu(s),\Sigma(s))$. *Then the probability density function of* $s = A^{+}t$ *is*

$$p_s(s) = \int q(\gamma_\perp|s)\frac{p_t(As+\gamma_\perp)}{Z(\mu(s)-\gamma_\perp|A,\Sigma(s))}d\gamma_\perp \qquad (25)$$

Proof.

$$\begin{aligned} p_s(s) &= \int \delta(U_{\parallel}^T\gamma_\perp)p_t(Ax+A^{\perp}\gamma_\perp)d\gamma_\perp|A^TA|^{\frac{1}{2}} \\ &= \int \frac{q(\gamma_\perp|s)}{q(\gamma_\perp|s)}\delta(U_{\parallel}^T\gamma_\perp)p_t(Ax+A^{\perp}\gamma_\perp)d\gamma_\perp|A^TA|^{\frac{1}{2}} \\ &= \int q(\gamma_\perp|s)\frac{p_t(As+A^{\perp}\gamma_\perp)}{Z(\mu(s)-\gamma_\perp|A,\Sigma(s))}d\gamma_\perp \\ &= \int q(\gamma_\perp|s)\frac{p_t(As+\gamma_\perp)}{Z(\mu(s)-\gamma_\perp|A,\Sigma(s))}d\gamma_\perp \end{aligned}$$

□

The log of Eq.25 can be optimized using the ELBO decomposition (Jordan et al., 1998) of $\log p_s(s)$.

Corollary 11. *The ELBO of* $\log p_s(s)$ *is*

$$\log p_s(s) \geq \mathbb{E}_{q(\gamma_\perp|s)}[\log \frac{p_t(As+\gamma_\perp)}{Z(\mu(s)-\gamma_\perp|A,\Sigma(s))}] \qquad (26)$$

We can train using Eq.26. The algorithm, shown in Fig.2b, constructs a tall vector by projecting the input short vector onto the image of A and by sampling its orthogonal component using the importance sampler.

6 Experiments

The goal of our experiments is to demonstrate that learning the full parametrization of a rectangular matrix outperforms the factored parametrization. We show this for wide MVPs (Eq.10) on 2D toy datasets and tall MVPs on the CIFAR-10 (Krizhevsky) dataset. All of our code is written using the JAX (Bradbury et al., 2018) python library.

6.1 Algorithm Scalability

Our first experiment compares the speed of the SVD, factored and RealMVP algorithms across increasing dimensions. The results in Fig.3 look at the runtime of each algorithm in milliseconds over various input and output dimensionalities for both the tall and wide MVPs. Each algorithm was run until either the tall or wide algorithms could not be performed due to memory constraints (this experiment was run on one NVIDIA 1080Ti GPU with 11 GB of memory). For these experiments, we set the noise distribution to be a Gaussian whose parameters are a linear combination of ϵ/γ to provide a fair comparison of the different methods.

We find that the SVD approach is the slowest algorithm, factored is fastest in small dimensions and RealMVP is the fastest (and only tractable algorithm) in high dimensions. Even though the SVD algorithm has the same memory complexity as RealMVP, in practice the computation of the SVD itself may require more than $O(MN)$ memory which causes the method to fail relatively quickly. Similarly, the factored algorithm fails quickly due to memory constraints. RealMVP, on the other hand, scales well with N. These results indicate that RealMVP is a viable method to low dimensional global structure to high dimensional problems, such as image modeling, by using a tall MVP to connect a high dimensional and low dimensional flow.

6.2 Wide matrix

We compare the full and factored parametrization of a wide MVP on toy 2d datasets. The model starts with a wide MVP that uses either the RealMVP or factored parametrization to increase the dimensionality of the data to $\mathbb{R}^4$

$$\mathbb{R}^2 \to \text{Wide MVP} \to \mathbb{R}^4$$

The remainder of the flow has more three layers before the unit Gaussian prior - a mixture of logistics nonlinearity (Ho et al., 2019) with 8 mixture components, a square MVP, and another mixture of logistics with 8 mixture components (see section 5 of the appendix for more details on the experimental setup). Over all three datasets, the full parametrization of the wide MVP yielded better density estimation results.

6.3 Tall matrix

Our tall MVP tests replace the factor step of the multiscale architecture (Dinh et al., 2017) for image models. The multiscale architecture can be thought of as a flow layer that pads gaussian noise to an input using a tall MVP with orthogonal noise. With this view in mind, we use tall MVPs to reduce the dimensionality of an input by half. We test the CIFAR-10 dataset with three tall MVPs:

$$\begin{aligned}\mathbb{R}^{32\times32\times3} &\to \text{Tall MVP} \to \mathbb{R}^{16\times16\times6}\\ &\to \text{Tall MVP} \to \mathbb{R}^{8\times8\times12}\\ &\to \text{Tall MVP} \to \mathbb{R}^{4\times4\times24}\end{aligned}$$

The MVPs act on each pixel along the channel dimension, like a one by one convolution (Kingma and Dhariwal, 2018), so that the computations remained tractable. In addition to the RealMVP and factored parametrization, we also include the regular multiscale factorization for comparison. The results in table 4b show that the use of tall MVPs to reduce dimensionality outperforms the baseline method of Gaussianizing half of the dimensions, and that RealMVP outperforms the factored parametrization.

7 Conclusion

We presented a novel method for using rectangular matrix vector products as probabilistic transformations. We showed that tall MVPs with additive orthogonal noise are the stochastic inverse of wide MVPs and derived a learning algorithm that can be easily implemented. As future work, we hope to investigate efficient parametrizations of A that allows for fast computation of A^+.

References

Chi Au and Judy Tam. Transforming Variables Using the Dirac Generalized Function. *The American Statistician*, 53:270–272, 1999.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *arXiv:2003.13913 [cs, stat]*, June 2020. URL http://arxiv.org/abs/2003.13913. arXiv: 2003.13913.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural Ordinary Differential Equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 6571–6583. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Tian Qi Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 9913–9923, 2019. URL http://papers.nips.cc/paper/9183-residual-flows-for-invertible-generative-modeling.

Rob Cornish, Anthony L. Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows. In *arXiv:1909.13833 [cs, stat]*, Vienna, Austria, August 2020. URL http://arxiv.org/abs/1909.13833. arXiv: 1909.13833.

Edmond Cunningham, Renos Zabounidis, Abhinav Agrawal, Ina Fiterau, and Daniel Sheldon. Normalizing Flows Across Dimensions. page 18, July 2020.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 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, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented Neural ODEs. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 3140–3150. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/21be9a4bd4f81549a9d1d241981cec3c-Paper.pdf.

Mads Dyrholm. Some Matrix Results, 2004. URL http://www.machlea.com/mads/misc-papers/madrix.pdf.

Mevlana C. Gemici, Danilo Rezende, and Shakir Mohamed. Normalizing Flows on Riemannian Manifolds. *arXiv:1611.02304 [cs, math, stat]*, November 2016. URL http://arxiv.org/abs/1611.02304.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2722–2730. PMLR, 2019. URL http://proceedings.mlr.press/v97/ho19a.html.

Chin-Wei Huang, Laurent Dinh, and Aaron Courville. Augmented Normalizing Flows: Bridging the Gap Between Generative Flows and Latent Variable Models. *arXiv:2002.07101 [cs, stat]*, February 2020. URL http://arxiv.org/abs/2002.07101.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An Introduction to Variational Methods for Graphical Models. In Michael I. Jordan, editor, *Learning in Graphical Models*. Springer Netherlands, Dordrecht, 1998. ISBN 978-94-010-6104-9 978-94-011-5014-9. doi: 10.1007/978-94-011-5014-9_5. URL http://link.springer.com/10.1007/978-94-011-5014-9_5.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 10236–10245, 2018. URL http://papers.nips.cc/paper/8224-glow-generative-flow-with-invertible-1x1-convolutions.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. page 60.

Abhishek Kumar, Ben Poole, and Kevin Murphy. Regularized Autoencoders via Relaxed Injective Probability Flow. In *AISTATS*, 2020.

Didrik Nielsen, Priyank Jaini, Emiel Hoogeboom, Ole Winther, and Max Welling. SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows. *arXiv:2007.02731 [cs, stat]*, July 2020. URL http://arxiv.org/abs/2007.02731. arXiv: 2007.02731.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *arXiv:1912.02762 [cs, stat]*, December 2019. URL http://arxiv.org/abs/1912.02762.

Roger Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge University Press, 1955.

Nathan Ratliff. *Multivariate Calculus II: The geometry of smooth maps*, 2014. URL https://ipvs.informatik.uni-stuttgart.de/mlr/wp-content/uploads/2014/12/mathematics_for_intelligent_systems_lecture6_notes.pdf.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1530–1538. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/rezende15.html.

O.V. Rudenko S.N. Gurbatov. Appendix fundamental properties of generalized functions. In O.V. Rudenko S.N. Gurbatov, editor, *Waves and structures in nonlinear nondis*, pages 441 – 468. 2011. ISBN 978-7-04-031695-7. URL http://academic.hep.com.cn/skld/CN/chapter/chapter296957.shtml.

Han Zhang, Xi Gao, Jacob Unterman, and Tom Arodz. Approximation Capabilities of Neural ODEs and Invertible Residual Networks. page 10, Vienna, Austria, 2020.