Max-Sliced Score Matching

Safa Messaoud [1] **Kuan-Chieh Wang** [2] **Alexander Schwing** [3]

Abstract

Score matching methods have recently shown a lot of promise for estimating unormalized densities. Scaling score matching to high dimensional spaces, is however challenging as it requires an accurate estimation of the trace of a Hessian. To improve upon sliced score matching, which circumvents computation of the Hessian by projecting the scores into random directions before comparing them, we study max-sliced score matching. Max-sliced score matching benefits from its use of the most informative projection directions rather than using randomly sampled ones. We prove that max-sliced score matching is consistent and asymptotically normal. We assess these models on estimation of densities and scores for implicit distributions in Variational and Wasserstein auto-encoders.

1. Introduction

Different methods have been proposed to learn unnormalized probabilistic models, including pseudo-likelihood models (Besag, 1975), contrastive divergence (Hinton, 2002; Tieleman & Hinton, 2009; Song & Ermon, 2019b) and noise-contrastive estimation (Gutmann & Hyvärinen, 2010). Notably, score matching methods (Vincent, 2011; Song & Ermon, 2019a; Jolicoeur-Martineau et al., 2020) which minimize the distance between the derivatives of the log-density functions, *i.e.*, the scores, of the data and model distributions have received increasing interest as they circumvent computation of the partition function.

However, score matching presents an additional challenge, as it requires computation of the trace of a Hessian, which involves a number of forward and backward propagations proportional to the data dimensionality (Vincent, 2011).

To overcome this issue, Kingma & LeCun (2010); Martens et al. (2012) propose to approximate the trace of the Hessian via approximate back-propagation (Kingma & LeCun, 2010) or curvature propagation (Martens et al., 2012). However, these methods have no theoretical guarantees for the approximation error or are memory demanding even in modern back-propagation pipelines, as they either limit back-propagation to the diagonal of the Hessian (Kingma & LeCun, 2010) or require use of complex numbers (Martens et al., 2012).

More recently, denoising score matching (Vincent, 2011; Song & Ermon, 2019a; Jolicoeur-Martineau et al., 2020) was proposed to circumvent computation of the trace by perturbing the data points with a pre-specified noise distribution and using the score of the perturbed data as an estimator for the scores of the model. This procedure can however only recover the noise corrupted data and is very sensitive to the choice of the noise scales. Sliced score matching (Song et al., 2019) is another method that circumvents computation of the Hessian matrix. Instead of directly matching the high-dimensional scores, it matches their projections along random directions. Sliced score matching only involves Hessian-vector products, computed without explicitly evaluating the Hessian.

In this paper, we study max-sliced score matching, a variant of sliced score matching that finds the most informative directions for the projection of the scores. We show that those directions are the eigenvectors of the Hessian. We also show that using the Lanczos algorithm permits to compute those directions efficiently.

We assess the performance of max-sliced score matching on (1) density estimation, specifically for the class of Deep Kernel Exponential Families (DKEF) (Wenliang et al., 2019) and NICE flow models (Dinh et al., 2014), and (2) score estimation for implicit distributions in variational inference applications, specifically image generation using variational auto-encoders (VAE) (Kingma & Welling, 2014) and Wasserstein auto-encoders (WAE) (Tolstikhin et al., 2018) with implicit encoders.

2. Related Work

In the following, we review score matching in greater detail.

Score matching methods use surrogate objectives which have similar optima to the maximum likelihood objective.

[1]University of Illinois at Urbana-Champaign [2]University of Toronto [3]University of Illinois at Urbana-Champaign. Correspondence to: Safa Messaoud <messaou2@illinois.edu>.

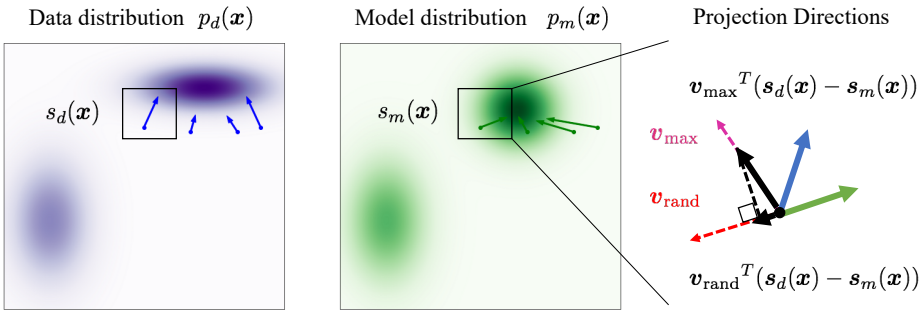

Figure 1. Visualization of the effect of slicing. $\boldsymbol{s}_d(\boldsymbol{x})$, and $\boldsymbol{s}_m(\boldsymbol{x})$ are the scores, *i.e.*, gradients of the log density of the data and model distributions respectively. Sliced score matching depends on projecting the difference in scores to a scalar. A random projection $\boldsymbol{v}_{\text{rand}}$ could make the difference in scores appear smaller than it is. Our work aims to select the projection that maximizes the difference in scores (see $\boldsymbol{v}_{\max}$). We find this to make learning more efficient.

Different variants of score matching have been used for different applications including image generation (Song et al., 2020; Song & Ermon, 2019a; Jolicoeur-Martineau et al., 2020), shape generation from point cloud data (Cai et al., 2020), waveform generation (Chen et al., 2020), and out-of-distribution image detection (Mahmood et al., 2020). Bao et al. (2020) extend score matching methods to learn general energy-based latent variable models (EBLVMs). Meng et al. (2020) propose a new divergence between distributions, which depends only on the derivatives of univariate log-conditionals (scores) of an autoregressive likelihood-based model.

Score matching (SM) (Hyvärinen, 2005) minimizes the Fisher divergence between the data distribution and the model, *i.e.*,

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \mathbb{E}_{\boldsymbol{x}\sim p_d(\boldsymbol{x})}[(\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) - \boldsymbol{s}_d(\boldsymbol{x}))^2], \tag{1}$$

with $\boldsymbol{s}_d(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log(p_d(\boldsymbol{x}))$ the score of the unknown data distribution and $\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) = \nabla_{\boldsymbol{x}} \log(p(\boldsymbol{x};\boldsymbol{\theta}))$ the score of the model distribution. By applying integration by parts, Hyvärinen (2005) shows that Eq. (1) is equivalent to

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\text{tr}(\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})) + \frac{1}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2], \tag{2}$$

with $\text{tr}(\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))$ being the trace of the Hessian $\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})$. While this formulation avoids the computation of the partition function, it introduces a new challenge, *i.e.*, computing the trace of the Hessian, which scales quadratically with the input dimensions. This often makes the application of score matching slow and impractical for real applications with high-dimensional data. Different methods have been proposed to approximate the trace. Pang et al. (2020) propose to approximate the higher order directional derivative with finite differences (FD). This however trades accuracy for speed. Another approach, *Approximate Backpropagation* (Kingma & LeCun, 2010) limits backpropagation to the diagonal of the Hessian instead of computing the full matrix. The method has no guarantees on the approximation errors. Similarly, Martens et al. (2012) propose *Curvature propagation* (CP), an estimator that introduces noise for each node in the network. This however leads to high variance and requires operating on complex numbers in neural networks, which is memory demanding.

Denoising score matching (DSM) (Vincent, 2011; Song & Ermon, 2019a) is a different method for scaling score matching, that circumvents computation of the Hessian. For this, DSM applies the original score matching loss to noise-corrupted data, *i.e.*,

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \mathbb{E}_{q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})p_d(\boldsymbol{x})}[\|\boldsymbol{s}_m(\tilde{\boldsymbol{x}};\boldsymbol{\theta}) - \nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})\|^2]. \tag{3}$$

Here, $q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ is a noise distribution. Song & Ermon (2019a) choose $q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ to be a Gaussian distribution $\mathcal{N}(\tilde{\boldsymbol{x}}|\boldsymbol{x}, \sigma^2\boldsymbol{I})$. The noisy samples in this case are generated by injecting Gaussian noise to the original samples, *i.e.*, $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}\sigma^2$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. DSM has the following limitations: (1) it can only recover the noise corrupted data distribution, (2) its performance is very sensitive to the choice of σ which is non trivial and often based on heuristics. Recently, new solutions for setting the parameters of the model have been proposed by Jolicoeur-Martineau et al. (2020). Additionally, Jolicoeur-Martineau et al. (2020) extend DSM with an adversarial loss. This led to better performance on image generation.

Sliced Score Matching (SSM) (Song et al., 2019) is another method aiming to scale the SM loss (Eq. (2)) by projecting $\boldsymbol{s}_d(\boldsymbol{x})$ and $\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})$ onto some random direction $\boldsymbol{v}$ and comparing their average difference along those random directions, *i.e.*,

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \mathbb{E}_{\boldsymbol{v}\sim p_{\boldsymbol{v}}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[(\boldsymbol{v}^T\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) - \boldsymbol{v}^T\boldsymbol{s}_d(\boldsymbol{x}))^2], \tag{4}$$

with $p_{\boldsymbol{v}}$ satisfying $\mathbb{E}_{p_{\boldsymbol{v}}}[\boldsymbol{v}\boldsymbol{v}^T] = \boldsymbol{I}$. Song et al. (2019) show, via integration by parts, that the program above is equivalent to

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{p_{\boldsymbol{v}}} \mathbb{E}_{p_d} \left[\frac{1}{2} \|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2 + \boldsymbol{v}^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) \boldsymbol{v} \right]. \quad (5)$$

Intuitively, the first term in the loss is lowest if $\boldsymbol{x}$ is a stationary point of $\log p(\boldsymbol{x};\boldsymbol{\theta})$. The second term pushes the Hessian $\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})$ toward negative definiteness, to ensure that $\log p(\boldsymbol{x};\boldsymbol{\theta})$ is concave in $\boldsymbol{x}$, *i.e.*, $\boldsymbol{x}$ is a maximum of $\log p(\boldsymbol{x};\boldsymbol{\theta})$. Note that $\boldsymbol{v}^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) \boldsymbol{v}$ with $\mathbb{E}_{p_{\boldsymbol{v}}}[\boldsymbol{v}\boldsymbol{v}^T]$ is the Hutchinson estimator (Hutchinson, 1989) of the trace of the Hessian $\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})$. The second order derivative term $\boldsymbol{v}^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) \boldsymbol{v}$ can be efficiently computed using two gradient operations for a single $\boldsymbol{v}$. If we average over M directions, $M+1$ gradient operations are required. In contrast, estimating the trace requires d differentiations. Hence, $M < d$ should be ensured to achieve an advantage of the method. SSM's main limitation is a high variance due to the small number of the projection directions. In this paper, we propose a new score matching objective, max-score matching (MSSM). It replaces the averaging over random projection directions with computation of the most informative direction.

3. Max-Sliced Score Matching (MSSM)

Given a dataset $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{|\mathcal{D}|}$ of samples $\boldsymbol{x} \sim p_d(\boldsymbol{x})$ drawn from an unknown data distribution $p_d(\boldsymbol{x})$, the goal of density estimation is to find the parameters $\boldsymbol{\theta}$ of a model distribution $p_m(\boldsymbol{x};\boldsymbol{\theta})$ which best fits the samples $\mathcal{D}$.

Energy-based models (EBMs) are frequently used for this purpose, representing the model distribution via

$$p_m(\boldsymbol{x};\boldsymbol{\theta}) = \frac{\exp -E(\boldsymbol{x};\boldsymbol{\theta})}{Z}, \quad Z = \int_{\boldsymbol{x}'} \exp -E(\boldsymbol{x}';\boldsymbol{\theta}) d\boldsymbol{x}',$$

where Z is referred to as the normalizing constant, or partition function.

There are two families of techniques for learning an EBM: by approximating the maximum likelihood gradient (see, *e.g.*, work by Du & Mordatch (2019) and references therein), or using score matching objectives. The first approach suffers from having to approximate the intractable partition function. Score matching methods circumvent this by only looking at the scores $s(\boldsymbol{x})$, *i.e.*, the gradient of the unnormalized density $s(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$. Hyvärinen (2005) proposed the Score Matching (SM) objective (Eq. (2)) for learning EBMs, but the Hessian term scales quadratically with the input dimensions. To reduce computation, Song et al. (2019) proposed Sliced Score Matching (SSM) which addresses

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \mathbb{E}_{\boldsymbol{v}\sim p_{\boldsymbol{v}}} \mathbb{E}_{\boldsymbol{x}\sim p_d} [(\boldsymbol{v}^T \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) - \boldsymbol{v}^T \boldsymbol{s}_d(\boldsymbol{x}))^2],$$

Algorithm 1 Max-Sliced Score Matching

Input: Update frequency F of $\boldsymbol{V}$; Number of computed eigenvectors K of $\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})$; Dataset $\mathcal{D}$; projection direction distribution $p_{\boldsymbol{v}}(\boldsymbol{v})$; Model distribution $p_m(\boldsymbol{x};\boldsymbol{\theta})$.
step $\leftarrow 0$
repeat
  Sample $\boldsymbol{x} \sim \mathcal{D}$
  $\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) \leftarrow \nabla_{\boldsymbol{x}} \log p_m(\boldsymbol{x};\boldsymbol{\theta})$
  $\mathcal{L}_{\text{MSSM}}(\boldsymbol{x};\boldsymbol{\theta}) \leftarrow \frac{1}{2} \|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2$
  if *(step % F) ==0* **then**
    $\boldsymbol{V} \leftarrow \text{eig}(\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))$
  end
  for $i \leftarrow 1$ *to* K **do**
    $\mathcal{L}_{\text{MSSM}} \leftarrow \mathcal{L}_{\text{MSSM}} + \frac{\hat{p}_{\boldsymbol{v}}(\boldsymbol{v}_i)}{K} \boldsymbol{v}_i^T \nabla_{\boldsymbol{x}} (\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) \boldsymbol{v}_i)$
  end
  Update $\boldsymbol{\theta}$ via Backpropagation
  step $\leftarrow$ step + 1
Return $\boldsymbol{\theta}$

where $\boldsymbol{s}_m(\cdot)$ and $\boldsymbol{s}_d(\cdot)$ are the scores of the model and data distributions respectively. Intuitively, the SSM objective applies random projections to the scores in the Fisher divergence (Eq. (1)).

We observe the random nature of the projections $\boldsymbol{v}$ to make learning inefficient. To address this we study a method which finds projections along directions that maximize the projected Fisher divergence via

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \max_{\boldsymbol{V}} \quad \mathbb{E}_{\boldsymbol{x}\sim p_d} [\|\boldsymbol{V}^T (\boldsymbol{s}_d(\boldsymbol{x}) - \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))\|^2], \\ &\text{s.t.} \quad \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{T}} = I, \end{aligned} \quad (6)$$

where $\boldsymbol{V} \in \mathbb{R}^{d\times K}$, $K \leq d$, denotes the matrix with its i-th column $\boldsymbol{v}_i \in \mathbb{R}^d$ being a projection direction. We use $\|\cdot\|$ to denote the 2-norm of a vector. The constraint $\boldsymbol{V}\boldsymbol{V}^{\boldsymbol{T}} = I$ is introduced to ensure that we make the most use of the space by requiring orthogonality. Beneficially, the constraint ensures boundedness of the loss (vectors $\boldsymbol{v}_i$ can't be scaled so that maximization reaches infinity). We refer to this objective as MSSM. In Fig. 1, we illustrate the scores $s_m(\boldsymbol{x};\boldsymbol{\theta})$, $s_d(\boldsymbol{x})$ as well as the projection directions for a Gaussian mixture model.

We show that MSSM is consistent, *i.e.*, $p_m(\boldsymbol{x};\boldsymbol{\theta})$ converges to $p_d(\boldsymbol{x})$ when Eq. (6) converges to 0. We sketch the proof in the following (see Appendix B for a full proof):

$$\begin{aligned} &\max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d} [\|\boldsymbol{V}^T (\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) - \boldsymbol{s}_d(\boldsymbol{x}))\|^2] = 0 \\ \iff &\max_{\boldsymbol{V}} \|\boldsymbol{V}^T (\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) - \boldsymbol{s}_d(\boldsymbol{x}))\|^2 = 0 \quad \forall \boldsymbol{x} \sim p_d(\boldsymbol{x}) \\ \iff &\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{s}_d(\boldsymbol{x}), \quad \forall \boldsymbol{x} \sim p_d(\boldsymbol{x}) \\ \iff &\log p_m(\boldsymbol{x};\boldsymbol{\theta}) = \log p_d(\boldsymbol{x}) + \text{Const}, \quad \forall \boldsymbol{x} \sim p_d(\boldsymbol{x}) \\ \iff &p_d(\boldsymbol{x}) = p_m(\boldsymbol{x};\boldsymbol{\theta}), \quad \forall \boldsymbol{x} \sim p_d(\boldsymbol{x}) \end{aligned}$$

Firstly, Eq. (6) can only be zero if for all real data the maximum projection of the difference between the model and groundtruth data scores are zero, forcing the model scores to match the data scores exactly. This is equivalent to matching the two normalized densities which concludes the proof.

Since $\boldsymbol{s}_d(\boldsymbol{x})$ cannot be computed (the data distribution is unknown), we follow Hyvärinen (2005); Song et al. (2019) and use integration-by-parts to obtain an objective without $\boldsymbol{s}_d(\boldsymbol{x})$ whose only difference from Eq. (6) is a constant which is independent of $\boldsymbol{s}_m(\boldsymbol{x})$:

$$\min_{\boldsymbol{\theta}} \quad \max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}}[\sum_{i=1}^{K} \boldsymbol{v}_i^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) \boldsymbol{v}_i + \frac{1}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2], \quad \text{s.t.} \quad \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{T}} = I. \tag{7}$$

We defer the proof of equivalence to Appendix C.

We argue that MSSM leads to a better maximum likelihood estimation than SSM. We show in Appendix D, that MSSM allows a better approximation of the partition function, as d vectors are needed for estimating the trace exactly, whereas 2^d vectors are needed for the sum.

Optimization for MSSM: In the following we first discuss how to address the program in Eq. (7) before elaborating on efficiency.

Claim: The solution to the inner maximization problem in Eq. (7) is the eigenvector basis of the Hessian $\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})$.

Proof: The Lagrangian of the program above is $\mathcal{L}(\boldsymbol{V};\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}}[\sum_{i=1}^{d} \boldsymbol{v}_i^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) \boldsymbol{v}_i + \frac{1}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2 - \sum_{i=1}^{d} \lambda_i(\|\boldsymbol{v}_i\|^2 - 1)]$ with $\boldsymbol{v}_i^T \boldsymbol{v}_j = 0$, $(\forall i \neq j)$. For a given $\boldsymbol{v}_i$, $\frac{\partial \mathcal{L}(\boldsymbol{v}_i;\boldsymbol{\theta})}{\partial \boldsymbol{v}_i} = 0$ is equivalent to $\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) \boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$. Hence, the direction $\boldsymbol{v}_i$ that solves the linear system corresponds to the eigenvector of the symmetric Hessian $\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})$ with the eigenvalue λ_i. The condition $\boldsymbol{v}_i^T \boldsymbol{v}_j = 0$ is satisfied since the eigenvectors are orthogonal to each other which concludes the proof. □

Unlike log-densities $\log p_d(\boldsymbol{x})$, which decrease as we move away from the data-manifold, the magnitude of the scores $\nabla_{\boldsymbol{x}} \log p_d(\boldsymbol{x})$ generally increases as we move away from the manifold. Hence, a good score estimation requires projection directions that encompass the entire space. The MSSM formulation uses a basis of d vectors to cover different directions in the d-dimensional space. As mentioned before, this is more efficient than sampling a large number of random directions to properly estimate the expectation in Eq. (4).

However, computing the eigenvectors using traditional eigenvector decomposition is very expensive and requires explicit computation and storage of the Hessian $\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})$, which defies the purpose of applying slicing/projection to Score Matching. To address this concern we study the combination of three ideas: (1) use of the Lanczos algorithm (Lanczos, 1950) with K iterations to find the K most important directions $\boldsymbol{V} \in \mathbb{R}^{d \times K}$, while adapting the Lanczos algorithm to only operate on Hessian-vector products; (2) importance sampling following a distribution $p_{\boldsymbol{v}}(\boldsymbol{v})$ to approximate the sum in Eq. (7) which reduces the number of back-propagations; and (3) computing these vectors with a frequency F instead of at every iteration. Those ideas are summarized in Alg 1 and we discuss details next.

Algorithm 2 Lanczos Algorithm for Score Matching

Input: Number of iterations K. The score $s_m(\boldsymbol{x};\boldsymbol{\theta})$. Initial $\boldsymbol{q}_0 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

Choose $\boldsymbol{r} = \boldsymbol{q}_0$
$\beta_0 = \|\boldsymbol{q}_0\|$

while $k = 1, \cdots, K$ **do**
  $\boldsymbol{q}_k = \frac{\boldsymbol{r}}{\beta_{k-1}}$
  $\boldsymbol{r} = \nabla_{\boldsymbol{x}}(\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})^T \boldsymbol{q}_k)$
  $\boldsymbol{r} = \boldsymbol{r} - \boldsymbol{q}_{k-1}\beta_{k-1}$
  $\alpha_k = \boldsymbol{q}_k^T \boldsymbol{r}$
  $\boldsymbol{r} = \boldsymbol{r} - \boldsymbol{q}_k \alpha_k$
  $\beta_k = \|\boldsymbol{r}\|^2$
end

$$\boldsymbol{T}_K = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \beta_2 & \\ & \beta_2 & \alpha_3 & \\ & & \ddots & \beta_{K-1} \\ & & \beta_{K-1} & \alpha_K \end{bmatrix}$$

$\boldsymbol{Q}_K = [\boldsymbol{q}_0, \cdots, \boldsymbol{q}_K]$
$\boldsymbol{S}_K = \text{eig}(\boldsymbol{T}_K)$

Return $\boldsymbol{V}^* = \boldsymbol{Q}_K \boldsymbol{S}_K$

Lanczos Algorithm: The Lanczos algorithm (Lanczos, 1950) computes the spectrum of a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ by first reducing it to a tridiagonal form $\boldsymbol{T}_K \in \mathbb{R}^{K \times K}$ and then computing the spectrum $\boldsymbol{S}_K \in \mathbb{R}^{K \times K}$ of that matrix instead. The computation of the spectrum of a tridiagonal matrix is very efficient. The algorithm works in steps $k \in \{1, \ldots, K\}$ and progressively builds an adapted orthonormal basis $\boldsymbol{Q}_k = [\boldsymbol{q}_0, \ldots, \boldsymbol{q}_k] \in \mathbb{R}^{d \times k}$, that satisfies at each iteration $\boldsymbol{Q}_k^T \boldsymbol{A} \boldsymbol{Q}_k = \boldsymbol{T}_k$. Each of the k iterations of the algorithm requires a single Hessian-vector multiplication, that we compute implicitly in two steps: (1) First, we compute $\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) = \nabla_{\boldsymbol{x}} \log p_m(\boldsymbol{x};\boldsymbol{\theta})$; then (2) the derivative of the vector product $\nabla_{\boldsymbol{x}}(\boldsymbol{v}^T \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))$. Overall, for K steps, $K+1$ backpropagations are needed. The Lanczos algorithm is memory efficient as it only stores three vectors at every iteration k, *i.e.*, $\boldsymbol{q}_{k-1}$, $\boldsymbol{q}_k$ and $\boldsymbol{q}_{k+1}$. The eigenvectors $\boldsymbol{V}^*$ of the matrix $\boldsymbol{A}$ are then obtained via

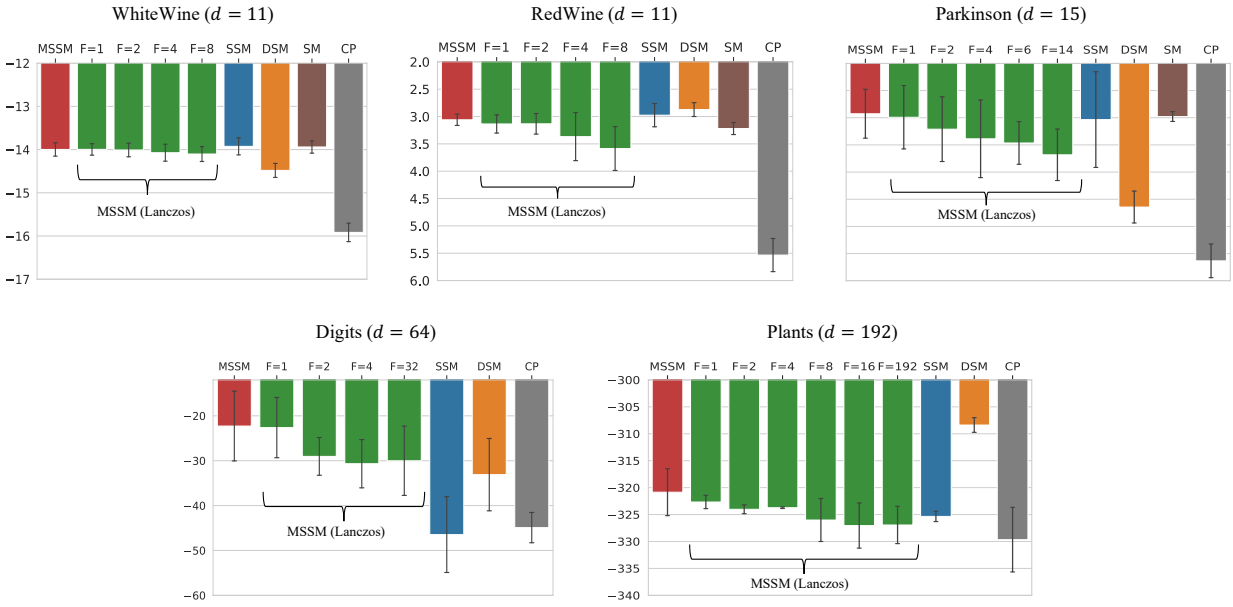

Figure 2. Log-likelihoods after training DKEF models on UCI datasets with different loss functions. Higher is better.

$\boldsymbol{V}^* = \boldsymbol{Q}_K \boldsymbol{S}_K \in \mathbb{R}^{d \times K}$, as illustrated in Alg. 2. Note that we use $\boldsymbol{V}^*$ to denote the optimal directions obtained when using K steps of the Lanczos algorithm.

Sampling: After solving for the eigenvectors $\boldsymbol{V}^*$ of $\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x}; \boldsymbol{\theta})$ obtained from Alg. 2, we use importance sampling to avoid computing the sum over the projection directions in Eq. (7), as this requires K back-propagations. Instead we address the following problem,

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\hat{p}_{\boldsymbol{v}}} \Big[\frac{\boldsymbol{v}^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x}; \boldsymbol{\theta}) \boldsymbol{v}}{\hat{p}_{\boldsymbol{v}}(\boldsymbol{v})} \Big] + \frac{1}{2} \| \boldsymbol{s}_m(\boldsymbol{x}; \boldsymbol{\theta}) \|^2 \Big]. \tag{8}$$

We choose $\hat{p}_{\boldsymbol{v}}(\boldsymbol{v})$ to be the uniform distribution over the set of the computed eigenvectors $\boldsymbol{V}^*$. Note that the distribution $\hat{p}_{\boldsymbol{v}}$ differs from the distribution $p_{\boldsymbol{v}}$ over the space of all possible vectors $\boldsymbol{v}$ which is used for sliced score matching in Eq. (4).

Overall the method requires $M \frac{K}{F}$ backpropagations. In practice, we chose $M = 1$. Here, M is the number of sampled vectors to approximate the expectation over $\boldsymbol{v}$ in Eq. (8).

4. Experiments

We study the MSSM approach on estimating (1) densities (Sec. 4.1) and (2) scores for modeling implicit distributions (Sec. 4.2). Following Song et al. (2019), we set the number of vectors M to be 1 in Eq. (8).

4.1. Density Estimation

We consider two density estimation experiments: (1) estimating the parameters of an energy-based model, *i.e.*, 'deep kernel exponential families' (DKEF) and (2) learning a deepflow model.

Baselines: We use the following baselines: (1) Score matching (SM) (Hyvärinen, 2005), (2) Sliced score matching with reduced variance (SSM) and a multivariate Rademacher distribution p_v (Song et al., 2019), (3) Denoising score matching (DSM) (Vincent, 2011), (4) Approximate backpropagation (approx. BP) (Kingma & LeCun, 2010) and (5) Curvature propagation (CP) (Martens et al., 2012). We report results of MSSM with exact eigenvector computation and with an approximate one using the Lanczos algorithm.

Metric: As evaluation metric we use maximum likelihood which directly relates to the score matching loss, as explained in Appendix D.

1) Deep Kernel Exponential Families (DKEF): DKEF parameterizes the unnormalized log-likelihood via $\log p_f(\boldsymbol{x}) = \sum_{l=1}^{L} \alpha_l k(\boldsymbol{x}, \boldsymbol{z}_l) + \log q_0(\boldsymbol{x})$, where $q_0(\boldsymbol{x})$ is a fixed function, α_l are weights, and $k(\boldsymbol{x}, \boldsymbol{z}_l)$ is a mixture of R Gaussian kernels defined on the feature space $\boldsymbol{\phi}_r$ of a deep net and evaluated at L different inducing points $\boldsymbol{z}_l$:

$$k(\boldsymbol{x}, \boldsymbol{z}_l) = \sum_{r=1}^{R} \rho_r \exp\left(\frac{-1}{2\sigma_r^2} \| \boldsymbol{\phi}_r(\boldsymbol{x}) - \boldsymbol{\phi}_r(\boldsymbol{z}_l) \|^2 \right). \tag{9}$$

The training parameters include the inducing points $\boldsymbol{z}_l$, the deep net $\boldsymbol{\phi}_r$, length scales σ_r and the non-negative mixture coefficients ρ_r. A closed-form expression for the weights α_l

Table 1. Log-likelihoods for NICE models trained on MNIST.

MLE	SSM	SSM-Staged	DSM ($\sigma = 0.1$)	DSM ($\sigma = 1.74$)	CP	Approx. BP	MSSM (Exact)	MSSM (Laczos, F=100)	MSSM (Laczos, F=200)	MSSM (Laczos, F=500)
-791	-3355	-1102	-4363	-8082	-1517	-2288	-1044	-1067	-1085	-1235

can be derived as a function of the other model parameters (see Appendix E for training details). We follow the same setup as Song et al. (2019). All models are trained with 15 different random seeds and training is stopped when the validation loss does not improve for 200 steps.

The obtained log-likelihoods on RedWine ($d = 11$), WhiteWine ($d = 11$), Parkinsons ($d = 15$), Digits ($d = 64$) and Plants ($d = 192$) (Dua & Graff, 2017) datasets are presented in Fig. 2. Note that the likelihoods are estimated using AIS (Neal, 2001), using a zero mean Gaussian proposal distribution $\mathcal{N}(0, 2I)$ and 1,000,000 samples. We observe that MSSM performance is comparable to SSM for data sets with small dimensionality, *i.e.*, RedWine, WhiteWine and Parkinsons. MSSM however scales better to the high-dimensional datasets, *i.e.*, Digits and Plants. Exact MSSM is consistently better than the Lanczos version. This is expected as we limit the number of iterations of the Lanczos algorithm to K, which may lead to smaller eigenvalues not converging. As we increase the frequency F of the eigenvector computation, MSSM (Lanczos) performance decreases. This is expected as well, since the computed eigenvectors are no longer accurate and SSM ends-up sampling more diverse directions in the space. Our MSSM models, both exact and Lanczos-based with small update frequencies F, have lower variance than SSM for RedWine, WhiteWine, Parkinsons and Digits datasets. DSM performance strongly depends on the choice of the noise rate σ, which is hard to tune. Similarly to work by Song et al. (2019), we run a grid search over a range of values. CP performance is constantly worse as it injects noise to each node in the computation graph. This is not efficient for training large deep net models. Computing the exact score matching loss becomes already prohibitively slow for the digits and plants datasets. We omit the results for approx. BP, as the log-likelihoods were smaller than -10^6 for all datasets.

2) Deep Flow Models: Since Deepflow models enable access to a tractable maximum likelihood, we train a NICE (Dinh et al., 2014) model to better understand the relationship between MSSM, SSM and MLE. For this we consider MNIST data ($d = 728$) generation. Results are reported in Tab. 1. Since score matching is a weak form of MLE, as explained in Appendix D, we obtain better maximum likelihood when applying score matching loss on the intermediate layers of the flow model (SSM-staged). When trained with the SSM loss applied only to the last layer, the log-likelihood fluctuates and the best achieved performance doesn't correspond to the one at the end of training. Indeed, Song et al. (2019) report results from the best checkpoint. To further improve upon SSM-staged, we replace the SSM loss with MSSM at the output layer. We observe that the log-likelihood training curve has a monotonically increasing behavior. We improve upon SSM-staged at the end of the training when using MSSM (exact) or MSSM with eigenvectors computed every 100 or 200 iterations using the Lanczos algorithm.

Table 2. Log-likelihoods on MNIST, estimated by AIS.

	VAE		**WAE**	
Latent Dim	8	32	8	32
ELBO	-96.87	-89.06	N/A	N/A
SSM	**-95.50**	-89.25	-98.24	-90.37
Stein	-96.71	-91.84	-99.05	-91.70
Spectral	-96.60	-94.67	-98.81	-92.55
MSSM (Exact)	-95.61	-89.26	**-97.62**	-89.53
MSSM (Laczos)	-95.75	**-88.99**	-98.25	**-89.46**

4.2. Score Estimation

We consider replacing simple explicit encoder distributions in Variational Auto-Encoders (VAE) and Wasserstein Auto-Encoders (WAE), with more expressive implicit ones that we train with an additional score matching loss. As baselines, we consider *Stein* (Li & Turner, 2018) and *Spectral* (Shi et al., 2018) score estimation techniques.

VAE with Implicit Encoders: A classical VAE is trained by minimizing the ELBO loss:

$$\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \mathbb{E}_{p_d(\boldsymbol{x})} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - \log p(\boldsymbol{z}) + \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})]. \tag{10}$$

Here, $p(\boldsymbol{z})$ is the prior distribution, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is the encoder distribution and $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is the decoder distribution. In order to compute a closed form of the loss, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is usually chosen to be a simple explicit distribution (*e.g.*, a Gaussian). This obviously limits the expressivity of the model. Extending ELBO with a score matching loss permits to model arbitrarily complex distributions $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ implicitly. This better captures the multi-modality and intricacies of common dataset. The latter is achieved by replacing the entropy term $-\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ with the term $-\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\nabla_{\boldsymbol{z}} \log q_{\boldsymbol{\phi}(\boldsymbol{z}|\boldsymbol{x})}(\boldsymbol{z}|\boldsymbol{x})^T \boldsymbol{z}]$. It can be easily checked that both terms have the same derivative with

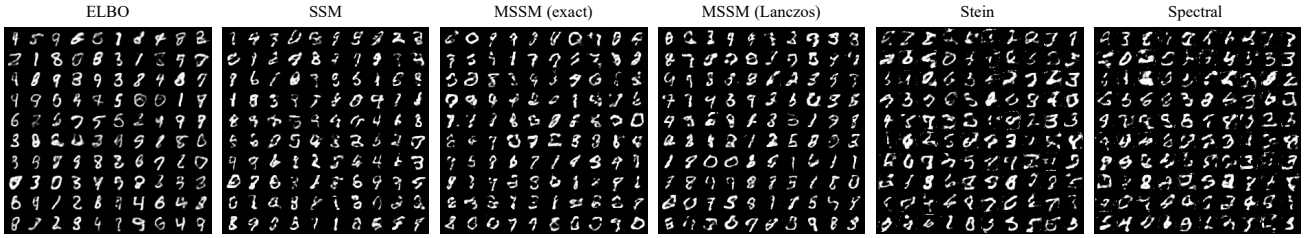

Figure 3. VAE generated samples on MNIST for a latent dimension of 32 for $\boldsymbol{z}$.

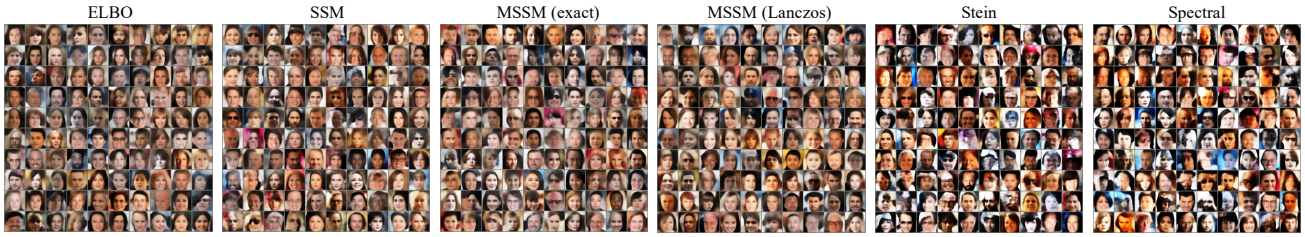

Figure 4. VAE generated samples on CelebA for a latent dimension of 32 for $\boldsymbol{z}$.

respect to ϕ, *i.e.*,

$$\nabla_{\phi}\, \mathbb{E}_{q_{\phi}(\boldsymbol{z}|\boldsymbol{x})}[\log q_{\phi}(\boldsymbol{z}|\boldsymbol{x})] = \mathbb{E}_{q_{\phi}(\boldsymbol{z}|\boldsymbol{x})}\left[\nabla_{\boldsymbol{z}} \log q_{\phi}(\boldsymbol{z}|\boldsymbol{x})^T \nabla_{\phi} \boldsymbol{z}\right].$$

The gradient $\nabla_{\boldsymbol{z}} \log q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$ is approximated with a deepnet $\boldsymbol{s}_{\psi}(\boldsymbol{z}|\boldsymbol{x})$ using a score matching loss, circumventing the computation of $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$. The objective from Eq. (10) becomes:

$$\min_{\phi,\boldsymbol{\theta},\psi} \max_{\boldsymbol{V}} \mathbb{E}_{p_d(\boldsymbol{x})}\, \mathbb{E}_{q_{\phi}(\boldsymbol{z}|\boldsymbol{x})}[-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - \log p(\boldsymbol{z}) + \boldsymbol{s}_{\psi}(\boldsymbol{z}|\boldsymbol{x})^T \boldsymbol{z} + \sum_{i=1}^{d} \boldsymbol{v}_i^T \nabla_{\boldsymbol{z}} \boldsymbol{s}_{\psi}(\boldsymbol{z}|\boldsymbol{x}) \boldsymbol{v}_i + \frac{1}{2} \|\boldsymbol{s}_{\psi}(\boldsymbol{z}|\boldsymbol{x})\|^2],$$

$$\text{s.t. } \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}.$$

We report negative log-likelihoods on MNIST, as estimated by AIS (Neal, 2001) with 1000 intermediate distributions, in Tab. 2 for latent dimensions of size 8 and 32. We present samples in Fig. 3. In Fig. 5, we present results for different Lanczos update frequencies. We observe MSSM to perform comparably to SSM. Both MSSM and SSM outperform the ELBO, which emphasizes the expressive power of an implicit encoder $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$.

In Tab. 3, we evaluate sample quality on CelebA with FID scores (Heusel et al., 2017) computed after 10k, 40k, 70k and 100k iterations respectively. We observe that MSSM and its variants all converge faster, haing low FID scores after 10k iterations already. In contrast, SSM does not perform as well early during training. Three MSSM variants outperform the ELBO loss. Samples are presented in Fig. 4.

WAE with Implicit Encoders: Unlike VAE which computes a KL-divergence between the encoder distribution $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$ of each sample and the prior distribution $p(\boldsymbol{z})$,

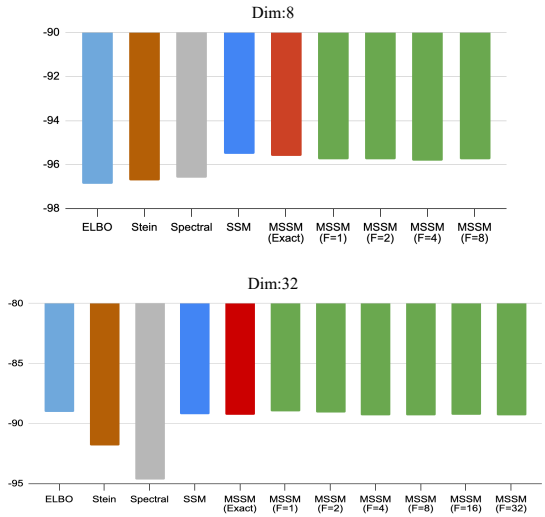

Figure 5. Log-likelihoods for VAE trained on MNIST.

WAE compares the distribution of the entire encoder space $q_{\phi}(\boldsymbol{z})$ to the prior $p(\boldsymbol{z})$. This alleviates the issue of erroneous reconstruction due to overlapping latent codes $\boldsymbol{z}$ originating from different samples $\boldsymbol{x}$. It also reduces the prior-hole problem.

Concretely, WAE minimizes the objective:

$$\min_{\boldsymbol{\theta},\phi} \mathbb{E}_{p_d(\boldsymbol{x})}\, \mathbb{E}_{q_{\phi}(\boldsymbol{z})}[c(\boldsymbol{x}, p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})) - \lambda \log p(\boldsymbol{z}) - \lambda q_{\phi}(\boldsymbol{z})],$$

composed of the reconstruction loss $c(\boldsymbol{x}, p_{\theta}(\boldsymbol{x}|\boldsymbol{z}))$ and the KL-divergence between the prior $p(\boldsymbol{z})$ and the encoder distribution $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$. We model $q_{\phi}(\boldsymbol{z})$ implicitly. Following the VAE case we approximate $\nabla_{\boldsymbol{z}} q_{\phi}(\boldsymbol{z})$ using a deepnet

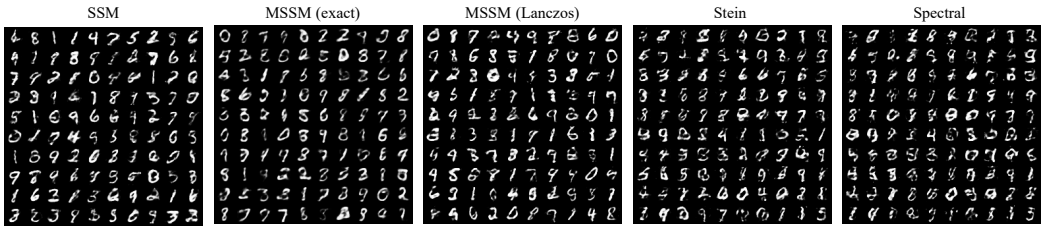

Figure 6. WAE generated samples on MNIST for a latent dimension of 32 for $\boldsymbol{z}$.

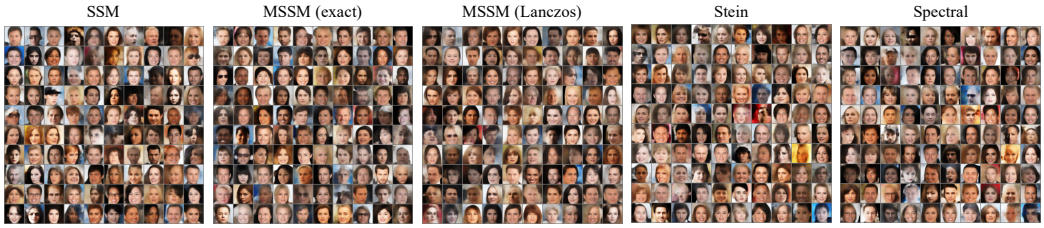

Figure 7. WAE generated samples on CelebA for a latent dimension of 32 for $\boldsymbol{z}$.

Table 3. FID scores of different methods versus number of training iterations on the CelebA dataset.

		10k	40k	70k	100k
VAE	ELBO	**96.20**	73.70	69.42	66.32
	SSM	108.52	70.28	66.52	62.50
	Stein	126.60	118.87	120.51	126.76
	Spectral	131.90	125.04	128.36	133.93
	MSSM (exact)	100.09	69.17	64.37	**60.68**
	MSSM (Lanczos, F=1)	100.26	**64.14**	65.45	60.78
	MSSM (Lanczos, F=2)	98.38	65.34	64.91	62.46
	MSSM (Lanczos, F=8)	101.15	65.34	65.78	63.76
	MSSM (Lanczos, F=16)	100.65	64.71	**62.68**	63.33
	MSSM (Lanczos, F=32)	107.24	66.30	65.31	65.91
WAE	SSM	84.11	61.09	56.23	54.33
	Stein	82.93	63.46	58.53	57.61
	Spectral	82.30	62.47	58.03	55.96
	MSSM (exact)	87.12	**60.93**	**56.17**	**53.98**
	MSSM (Lanczos, F=1)	84.08	71.68	57.29	54.03
	MSSM (Lanczos, F=2)	82.33	71.00	60.66	54.23
	MSSM (Lanczos, F=8)	82.92	73.37	59.48	54.98
	MSSM (Lanczos, F=16)	**82.21**	73.48	59.62	58.49
	MSSM (Lanczos, F=32)	83.75	70.54	59.37	59.27

$\boldsymbol{s}_{\boldsymbol{\psi}}(\boldsymbol{z})$, resulting in the objective

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\psi}} \max_{\boldsymbol{V}} \mathbb{E}_{p_d(\boldsymbol{x})} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z})}[c(\boldsymbol{x}, p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})) - \lambda \log p(\boldsymbol{z}) - \lambda \boldsymbol{s}_{\boldsymbol{\psi}}(\boldsymbol{z})\boldsymbol{z} + \boldsymbol{v}^T \nabla_{\boldsymbol{z}} \boldsymbol{s}_{\boldsymbol{\psi}}(\boldsymbol{z})\boldsymbol{v} + \frac{1}{2}\|\boldsymbol{s}_{\boldsymbol{\psi}}(\boldsymbol{z})\|^2],$$

$$\text{s.t. } \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}.$$

On MNIST, MSSM achieves the best log-likelihoods for both latent dimensions across all baselines (Tab. 2, Fig. 8), as well as the best FID scores for CelebA data (Tab. 3). We visualize generated samples obtained from training on MNIST data in Fig. 6. Samples after training on CelebA are given in Fig. 7.

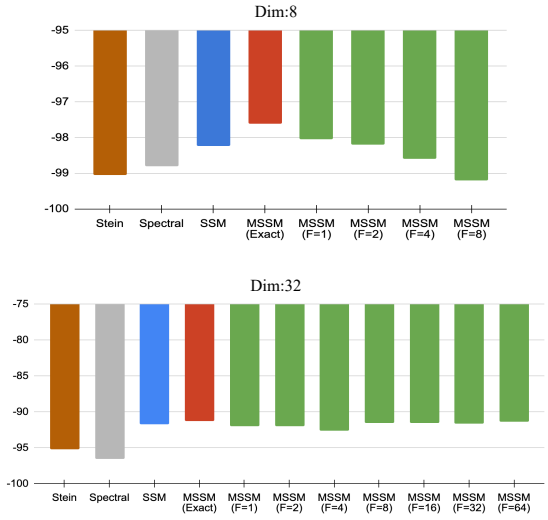

Figure 8. Log-likelihoods for WAE trained on MNIST.

5. Conclusion

We propose max-sliced score matching (MSSM) which improves upon sliced score matching (SSM) by finding the most informative projection directions for minimizing the difference between the data and model scores. We show that these directions correspond to the eigenvectors of the Hessian of the log-likelihood, which can be computed efficiently using the Lanczos algorithm, importance sampling to approximate the sum over the eigenvectors as well as a schedule for updating the eigenvectors at specific frequencies. We show improved performance on learning unnormalized statistical models and estimating scores for implicit distributions in VAEs and WAEs.

References

Bao, F., Li, C., Xu, T., Su, H., Zhu, J., and Zhang, B. Bi-level score matching for learning energy-based latent

variable models. 2020.

Besag, J. Statistical analysis of non-lattice data. *J. R. Stat. Soc*, 1975.

Cai, R., Yang, G., Averbuch-Elor, H., Hao, Z., Belongie, S., Snavely, N., and Hariharan, B. Learning gradient fields for shape generation. *arXiv preprint arXiv:2008.06520*, 2020.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Du, Y. and Mordatch, I. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Procs. AISTATS*, 2010.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 2002.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Commun. Stat. Simul. Comput.*, 1989.

Hyvärinen, A. Estimation of non-normalized statistical models by score matching. *JMLR*, 2005.

Hyvarinen, A. Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables. *IEEE Trans. Neural Netw. Learn. Syst*, 2007.

Jolicoeur-Martineau, A., Piché-Taillefer, R., Combes, R. T. d., and Mitliagkas, I. Adversarial score matching and improved sampling for image generation. 2020.

Kingma, D. P. and LeCun, Y. Regularized estimation of image statistics by score matching. In *Procs. NeurIPS*, 2010.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *Procs. ICLR*, 2014.

Lanczos, C. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. 1950.

Li, Y. and Turner, R. E. Gradient estimators for implicit models. In *Procs. ICLR*, 2018.

Mahmood, A., Oliva, J., and Styner, M. A. Multiscale score matching for out-of-distribution detection. *arXiv preprint arXiv:2010.13132*, 2020.

Martens, J., Sutskever, I., and Swersky, K. Estimating the hessian by back-propagating curvature. In *Procs. ICML*, 2012.

Meng, C., Yu, L., Song, Y., Song, J., and Ermon, S. Autoregressive score matching. 2020.

Neal, R. M. Annealed importance sampling. *Statistics and computing*, 2001.

Newey, W. K. and McFadden, D. Large sample estimation and hypothesis testing. *Handbook of econometrics*, 1994.

Pang, T., Xu, T., Li, C., Song, Y., Ermon, S., and Zhu, J. Efficient learning of generative models via finite-difference score matching. 2020.

Shi, J., Sun, S., and Zhu, J. A spectral approach to gradient estimation for implicit distributions. In *Procs. ICML*, 2018.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Procs. NeurIPS*, 2019a.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *arXiv preprint arXiv:1907.05600*, 2019b.

Song, Y., Garg, S., Shi, J., and Ermon, S. Sliced score matching: A scalable approach to density and score estimation. *UAI*, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Tieleman, T. and Hinton, G. Using fast weights to improve persistent contrastive divergence. In *Procs. ICML*, 2009.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. Wasserstein auto-encoders. In *Procs. ICLR*, 2018.

Vincent, P. A connection between score matching and denoising autoencoders. *NC*, 2011.

Wenliang, L., Sutherland, D., Strathmann, H., and Gretton, A. Learning deep kernels for exponential family densities. In *Procs. ICML*, 2019.

A. Appendix: Max-Sliced Score Matching

Sliced score matching (SSM) enabled scaling score matching (SM) to more complex problems and high-dimensional datasets. The main idea: instead of directly matching the high-dimensional scores, we match their projections along random directions, *i.e.*,

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{v}\sim p_{\boldsymbol{v}}}\, \mathbb{E}_{\boldsymbol{x}\sim p_d}[(\boldsymbol{v}^T \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) - \boldsymbol{v}^T \boldsymbol{s}_d(\boldsymbol{x}))^2]. \tag{11}$$

Here, $p_d(\boldsymbol{x})$ and $p_m(\boldsymbol{x};\boldsymbol{\theta})$ denote the data and the model distributions respectively. Moreover, $\boldsymbol{s}_d(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log p_d(\boldsymbol{x})$ and $\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x};\boldsymbol{\theta})$ are the corresponding scores. Currently, high variance is the major drawback for the SSM loss. In order to address this, we propose Max-Sliced Score Matching (MSSM), a variant of SSM that chooses the most informative projection directions $\boldsymbol{V} \in \mathbb{R}^{d\times K}$ instead of averaging over random ones. We start our derivation by applying the max-slice to the Fisher Divergence, *i.e.*,

$$\begin{aligned} \min_{\boldsymbol{\theta}} \max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\|\boldsymbol{V}^T(\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) - \boldsymbol{s}_d(\boldsymbol{x}))\|^2], \\ \text{s.t.} \quad \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{T}} = I. \end{aligned} \tag{12}$$

Since the score of the data distribution $\boldsymbol{s}_d(\boldsymbol{x})$ is unknown, our MSSM optimizes an equivalent loss that does not depend on that data score:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \max_{\boldsymbol{V}} \quad & \mathbb{E}_{\boldsymbol{x}\sim p_d}[\sum_{i=1}^{K} \boldsymbol{v}_i^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\boldsymbol{v}_i + \frac{1}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2], \\ \text{s.t.} \quad & \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{T}} = I, \end{aligned} \tag{13}$$

where $\boldsymbol{v}_i$ is the i-th column vector of matrix $\boldsymbol{V}$. Note, Eq. (12) and Eq. (13) are Eq. (6) and Eq. (7) in the main text, but we define them again here to avoid cross-referencing across documents.

In the following, we denote by $\mathcal{L}(\boldsymbol{\theta})$ the MSSM loss from Eq. (13), *i.e.*,

$$\mathcal{L}(\boldsymbol{\theta}) = \max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\|\boldsymbol{V}^T(\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) - \boldsymbol{s}_d(\boldsymbol{x}))\|^2], \tag{14}$$

$J(\boldsymbol{\theta})$ the MSSM loss from Eq. (12), *i.e.*,

$$J(\boldsymbol{\theta}) = \max_{\boldsymbol{V}} \quad \mathbb{E}_{\boldsymbol{x}\sim p_d}[\sum_{i=1}^{K} \boldsymbol{v}_i^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\boldsymbol{v}_i + \frac{1}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2], \tag{15}$$

and by $J_{N,K}(\boldsymbol{\theta})$ the loss estimator resulting from empirically evaluating $J(\boldsymbol{\theta})$ on a batch size N using K vectors $\{\boldsymbol{v}_i\}_{i=1}^K$, *i.e.*,

$$J_{N,K}(\boldsymbol{\theta}) = \max_{\boldsymbol{V}} \quad \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{K} \boldsymbol{v}_i^T \nabla_{\boldsymbol{x}^{(j)}} \boldsymbol{s}_m(\boldsymbol{x}^{(j)};\boldsymbol{\theta})\boldsymbol{v}_i + \frac{1}{2}\|\boldsymbol{s}_m(\boldsymbol{x}^{(j)};\boldsymbol{\theta})\|^2. \tag{16}$$

Here, $\boldsymbol{v}_i$ is the i-th column vector of $\boldsymbol{V}$. We also assume that $p_m(\boldsymbol{x};\boldsymbol{\theta})$ is well specified, and that $\boldsymbol{\theta}^*$ is the parameter of the data distribution, *i.e.* $p_d(\boldsymbol{x}) = p_m(\boldsymbol{x};\boldsymbol{\theta}^*)$. We denote by $\boldsymbol{\theta}$, $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\theta}}_{NK}$ the estimates resulting from minimizing $\mathcal{L}(\boldsymbol{\theta})$, $J(\boldsymbol{\theta})$ and $J_{NK}(\boldsymbol{\theta})$ under the constraint $\boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}$, respectively, *i.e.*,

$$\boldsymbol{\theta} = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \quad \text{s.t.} \quad \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}, \tag{17}$$

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \quad \text{s.t.} \quad \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}, \tag{18}$$

$$\hat{\boldsymbol{\theta}}_{NK} = \operatorname*{argmin}_{\boldsymbol{\theta}} J_{NK}(\boldsymbol{\theta}) \quad \text{s.t.} \quad \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}. \tag{19}$$

We also define $f(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{V}^*)$ as follows:

$$f(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{V}^*) = \sum_{i=1}^{K} \boldsymbol{v}_i^{*T} \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\boldsymbol{v}_i^* + \frac{1}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}^*)\|^2. \tag{20}$$

Here, $\boldsymbol{V}^*$ is the optimal basis obtained from solving the inner optimization program given in Eq. (12). Hence, $J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim p_d}[f(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{V}^*)]$ and $J_{NK}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}[f(\boldsymbol{\theta}; \boldsymbol{x}^{(i)}, \boldsymbol{V}^*)]$.

In Appendix B, we prove consistency and asymptotic-normality of the MSSM objective. In Appendix C, we derive an equivalent version of the MSSM objective that does not depend on the unknown data score $\boldsymbol{s}_d(\boldsymbol{x})$. We derive the relationship between MSSM loss and MLE loss in Appendix D. Further training details are presented in Appendix E.

B. Consistency and Asymptotic Normality of MSSM

In the following we show consistency and asymptotic normality. This is important to assess the convergence and asymptotic variance of the estimated parameters around the true parameters.

B.1. Consistency

We will show that $\hat{\boldsymbol{\theta}}_{NK}$ is consistent, *i.e.* as the sample size N increases, the sampling distribution of the estimator $\hat{\boldsymbol{\theta}}_{NK}$ becomes increasingly concentrated at the true parameter value $\boldsymbol{\theta}^*$. Formally, we will prove that for every $K \leq d$ with $K \in \mathbb{N}$,

$$\hat{\boldsymbol{\theta}}_{NK} \xrightarrow{\text{p}} \boldsymbol{\theta}^*, \quad N \to \infty.$$

Here, '$\xrightarrow{\text{p}}$' denotes convergence in probability, *i.e.* $\lim_{n\to\infty} P(|\hat{\boldsymbol{\theta}}_{NK} - \boldsymbol{\theta}^*| > \boldsymbol{\epsilon}) = 0, \forall \boldsymbol{\epsilon} > \mathbf{0}$. The proof follows two steps:

1. First, we prove that $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^*$, in **Lemma 1** and **Theorem 1**.
2. Second, in **Theorem 2**, we prove that $\hat{\boldsymbol{\theta}}_{NK} \xrightarrow{\text{p}} \hat{\boldsymbol{\theta}}$ when $N \to \infty$. For this, we show that under the assumptions: (a) the parameter space $\boldsymbol{\Theta}$ is compact, (b) $\|\boldsymbol{s}_m(\boldsymbol{x}; \boldsymbol{\theta})\| < \infty$ and (c) $\|\nabla_{\boldsymbol{x}} \log \boldsymbol{s}_m(\boldsymbol{x}; \boldsymbol{\theta})\|_\infty < \infty$, we have $|f(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{V}^*)| < \infty$. As a result, the conditions in Lemma 2.4 (Newey & McFadden, 1994) are satisfied. So, $\sup_{\boldsymbol{\theta}} |J_{NK}(\boldsymbol{\theta}) - J(\boldsymbol{\theta})| \xrightarrow{\text{p}} 0$. Thus, from Theorem 2.1 (Newey & McFadden, 1994), we conclude $\hat{\boldsymbol{\theta}}_{NK} \xrightarrow{\text{p}} \hat{\boldsymbol{\theta}}$.

Lemma 1 *Assume that $p_m(\boldsymbol{x}; \boldsymbol{\theta})$ is well specified,* i.e.*, $p_d(\boldsymbol{x}) = p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)$, and $p_m(\boldsymbol{x}; \boldsymbol{\theta}) \neq p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)$ whenever $\boldsymbol{\theta} \neq \boldsymbol{\theta}^*$. In the following, we show that, when Eq. (12) converges to 0, $\boldsymbol{\theta}$ converges to $\boldsymbol{\theta}^*$,* i.e.*,*

$$\max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\|\boldsymbol{V}^T \boldsymbol{s}_m(\boldsymbol{x}; \boldsymbol{\theta}) - \boldsymbol{V}^T \boldsymbol{s}_d(\boldsymbol{x})\|^2] = 0 \iff \boldsymbol{\theta} = \boldsymbol{\theta}^*.$$

Proof: The proof is based on the following sequence of equivalences:

$$\begin{aligned}
&\max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\|\boldsymbol{V}^T \boldsymbol{s}_m(\boldsymbol{x}; \boldsymbol{\theta}) - \boldsymbol{V}^T \boldsymbol{s}_d(\boldsymbol{x})\|^2] = 0, \quad \text{s.t. } \boldsymbol{V}\boldsymbol{V}^T = I \\
\iff &\max_{\boldsymbol{V}} (\boldsymbol{s}_m(\boldsymbol{x}; \boldsymbol{\theta}) - \boldsymbol{s}_d(\boldsymbol{x}))^T \boldsymbol{V}\boldsymbol{V}^T (\boldsymbol{s}_m(\boldsymbol{x}; \boldsymbol{\theta}) - \boldsymbol{s}_d(\boldsymbol{x})) = 0, \quad \forall \boldsymbol{x} \sim p_d(\boldsymbol{x}), \quad \text{s.t. } \boldsymbol{V}\boldsymbol{V}^T = I \\
\overset{(i)}{\iff} &\|\boldsymbol{s}_m(\boldsymbol{x}; \boldsymbol{\theta}) - \boldsymbol{s}_d(\boldsymbol{x})\|^2 = 0, \quad \forall \boldsymbol{x} \sim p_d(\boldsymbol{x}) \\
\iff &\boldsymbol{s}_m(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{s}_d(\boldsymbol{x}), \quad \forall \boldsymbol{x} \sim p_d(\boldsymbol{x}) \\
\iff &\log p_m(\boldsymbol{x}; \boldsymbol{\theta}) = \log p_d(\boldsymbol{x}) + \text{Const.}, \quad \forall \boldsymbol{x} \sim p_d(\boldsymbol{x}) \\
\overset{(ii)}{\iff} &p_d(\boldsymbol{x}) = p_m(\boldsymbol{x}; \boldsymbol{\theta}), \quad \forall \boldsymbol{x} \sim p_d(\boldsymbol{x})
\end{aligned} \tag{21}$$

Note, (i) holds because $\boldsymbol{V}\boldsymbol{V}^T = I$. Further, (ii) holds as $p_m(\boldsymbol{x}, \boldsymbol{\theta})$ and $p_d(\boldsymbol{x})$ are normalized probability density functions. Finally, $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ is a simple proof by contradiction. Suppose $\boldsymbol{\theta} \neq \boldsymbol{\theta}^*$ is true, then it implies $p_d(\boldsymbol{x}) \neq p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)$ according to our assumption. This is not true, so $\boldsymbol{\theta} = \boldsymbol{\theta}^*$.

Theorem 1 *Assume the results from Lemma 1. Given the equivalence Eq. (12) $\iff$ Eq. (13) (Appendix C), the equality $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^*$ holds.*

Proof: In Lemma 1, we proved that $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ holds. Besides, in Appendix C, we show that the programs in Eq. (12) and Eq. (13) are equivalent. This results in the equivalence $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$. As a result, $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^*$.

Theorem 2 *Suppose that the parameter space $\boldsymbol{\Theta}$ is compact, $\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\| < \infty$, $\|\nabla_{\boldsymbol{x}}\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|_\infty < \infty$, and $f(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{V}^*)$ is continuous. Then, $\hat{\boldsymbol{\theta}}_{NK} \xrightarrow{p} \hat{\boldsymbol{\theta}}$.*

Proof: We start by showing that $|f(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{V}^*)| < \infty$. Given that $\|\boldsymbol{v}_j\|^2 = 1\ \forall j \in \{1,\cdots,K\}$, then $|\boldsymbol{v}_j^{(k)}\boldsymbol{v}_j^{(l)}| < 1$, $\forall k,l \in \{1,\cdots,d\}$. Since $\|\nabla_{\boldsymbol{x}}\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\| < \infty$, we conclude that $|\boldsymbol{v}_j^{*T}\nabla_{\boldsymbol{x}}\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\boldsymbol{v}_j^*| < \infty$, $\forall j$. Besides, since $\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2 < \infty$, we conclude that $|f(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{V}^*)| < \infty$. Hence, all the assumptions for Lemma 2.4 (Newey & McFadden, 1994) are satisfied. So, $\sup_{\boldsymbol{\theta}}|J_{NK}(\boldsymbol{\theta}) - J(\boldsymbol{\theta})| \xrightarrow{\text{p}} 0$. Thus, from Theorem 2.1 (Newey & McFadden, 1994), we conclude $\hat{\boldsymbol{\theta}}_{NK} \xrightarrow{\text{p}} \hat{\boldsymbol{\theta}}$.

B.2. Asymptotic Normality

We start by introducing some notation. We denote by $\boldsymbol{H}_{ij}$ a matrix that depends on $p_m(\boldsymbol{x};\boldsymbol{\theta})$ as follows:

$$\boldsymbol{H}_{ij} = \mathbb{E}_{p_d}[(\nabla_{\boldsymbol{\theta}}[\nabla^2_{\boldsymbol{x}}\log p_m(\boldsymbol{x};\boldsymbol{\theta}^*)]_{ii} + \frac{1}{2}\nabla_{\boldsymbol{\theta}}([\nabla_{\boldsymbol{x}}\log p_m(\boldsymbol{x};\boldsymbol{\theta}^*)]_i)^2)(\nabla_{\boldsymbol{\theta}}[\nabla^2_{\boldsymbol{x}}\log p_m(\boldsymbol{x};\boldsymbol{\theta}^*)]_{jj} + \frac{1}{2}\nabla_{\boldsymbol{\theta}}([\nabla_{\boldsymbol{x}}\log p_m(\boldsymbol{x};\boldsymbol{\theta}^*)]_j)^2)^T],$$

where, $[\nabla^2_{\boldsymbol{x}}\log p_m(\boldsymbol{x};\boldsymbol{\theta})]_{ii}$ is the ii-th entry of the matrix $\nabla^2_{\boldsymbol{x}}p_m(\boldsymbol{x};\boldsymbol{\theta})$ and $[\nabla_{\boldsymbol{x}}\log p_m(\boldsymbol{x};\boldsymbol{\theta})]_i$ is the i-th entry of the vector $\nabla_{\boldsymbol{x}}\log p_m(\boldsymbol{x};\boldsymbol{\theta})$. Also, we denote $\nabla_{\boldsymbol{x}}\log p_m(\boldsymbol{x};\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}$ and $J(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}$ as $\nabla_{\boldsymbol{x}}\log p_m(\boldsymbol{x};\boldsymbol{\theta}^*)$, and $J(\boldsymbol{\theta}^*)$, respectively.

In the following, we show that $\hat{\boldsymbol{\theta}}_{NK}$ is asymptotically normally distributed around the true parameter $\boldsymbol{\theta}^*$ when the batch size N becomes very large, *i.e.*,

$$\sqrt{N}(\hat{\boldsymbol{\theta}}_{NK} - \boldsymbol{\theta}^*) \xrightarrow{d} \mathcal{N}(0, \boldsymbol{\Sigma}).$$

Here '$\xrightarrow{d}$' denotes a convergence in distribution and $\boldsymbol{\Sigma}$ is the covariance matrix. Specifically, we prove the theorem:

Theorem 3 *If (1) $\boldsymbol{\Theta}$ is compact, (2) $\hat{\boldsymbol{\theta}}_{NK} \xrightarrow{p} \boldsymbol{\theta}^*$, (3) $\log p_m(\boldsymbol{x};\boldsymbol{\theta})$ is twice continuously differentiable with respect to $\boldsymbol{\theta}$, (4) $J(\boldsymbol{\theta}^*)$ is invertible at $\boldsymbol{\theta}^*$, (5) $\|\nabla^2_{\boldsymbol{\theta}}[\nabla^2_{\boldsymbol{x}}\log p_m(\boldsymbol{x};\boldsymbol{\theta})]_{ii}\|_\infty < \infty$, $\forall i \in \{1,\cdots,d\}$ and (6) $\|\nabla^2_{\boldsymbol{\theta}}\|\nabla_{\boldsymbol{x}}\log p_m(\boldsymbol{x};\boldsymbol{\theta})\|^2\|_\infty < \infty$, then,*

$$\sqrt{N}(\hat{\boldsymbol{\theta}}_{NK} - \boldsymbol{\theta}^*) \xrightarrow{d} \mathcal{N}\Big(\mathbf{0}, [\nabla^2_{\boldsymbol{\theta}}J(\boldsymbol{\theta}^*)]^{-1}\Big(\sum_{i,j=1}^{d}\boldsymbol{H}_{ij}\Big)[\nabla^2_{\boldsymbol{\theta}}J(\boldsymbol{\theta}^*)]^{-1}\Big).$$

The proof is based on Theorem 3.1 (Newey & McFadden, 1994). To apply it, we first prove:

1. $\sqrt{N}\nabla_{\boldsymbol{\theta}}J_{NK}(\boldsymbol{\theta}^*) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \sum_{i,j=1}^{d}\boldsymbol{H}_{ij})$, in **Lemma 2**.
2. $\sup_{\boldsymbol{\theta}}|\nabla^2_{\boldsymbol{\theta}}J_{NK}(\boldsymbol{x};\boldsymbol{\theta}) - \nabla^2_{\boldsymbol{\theta}}J(\boldsymbol{x};\boldsymbol{\theta})| \xrightarrow{p} \mathbf{0}$ in **Lemma 3**. Note that the absolute value is applied element wise to all the elements of the matrix.

Lemma 2 *If $\log p_m(\boldsymbol{x};\boldsymbol{\theta})$ is twice continuously differentiable with respect to $\boldsymbol{\theta}$, then*

$$\sqrt{N}\nabla_{\boldsymbol{\theta}}J_{NK}(\boldsymbol{\theta}^*) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \sum_{i,j=1}^{d}\boldsymbol{H}_{ij}).$$

Proof: The central limit theorem gives,

$$\sqrt{N}\sum_{i=1}^{K}\nabla_{\boldsymbol{\theta}}f(\boldsymbol{\theta}^*;\boldsymbol{x},\boldsymbol{V}^*) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \text{Var}[\nabla_{\boldsymbol{\theta}}f(\boldsymbol{\theta}^*;\boldsymbol{x},\boldsymbol{V}^*)]). \tag{22}$$

Note that $\mathbb{E}[\nabla_{\boldsymbol{\theta}}f(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{V})] = 0$ as $\boldsymbol{\theta}^*$ is the optimal parameter for $f(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{V}^*)$. We compute $\text{Var}[\nabla_{\boldsymbol{\theta}}f(\boldsymbol{\theta}^*;\boldsymbol{x},\boldsymbol{V}^*)]$ as

follows:

$$
\begin{aligned}
&\text{Var}[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \boldsymbol{x}, \boldsymbol{V}^*)] \\
&= \mathbb{E}_{p_d}[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \boldsymbol{x}, \boldsymbol{V}^*) \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \boldsymbol{x}, \boldsymbol{V}^*)^T] \\
&= \mathbb{E}_{p_d}[(\sum_{i=1}^{K} \nabla_{\boldsymbol{\theta}} \boldsymbol{v}_i^T \nabla_{\boldsymbol{x}}^2 \boldsymbol{p}_m(\boldsymbol{x}; \boldsymbol{\theta}^*) \boldsymbol{v}_i + \frac{1}{2} \nabla_{\boldsymbol{\theta}} \|\nabla_{\boldsymbol{x}} \boldsymbol{p}_m(\boldsymbol{x}; \boldsymbol{\theta}^*)\|^2)(\sum_{j=1}^{K} \nabla_{\boldsymbol{\theta}} \boldsymbol{v}_j^T \nabla_{\boldsymbol{x}}^2 \boldsymbol{p}_m(\boldsymbol{x}; \boldsymbol{\theta}^*) \boldsymbol{v}_j + \frac{1}{2} \nabla_{\boldsymbol{\theta}} \|\nabla_{\boldsymbol{x}} \boldsymbol{p}_m(\boldsymbol{x}; \boldsymbol{\theta}^*)\|^2)^T] \\
&= \mathbb{E}_{p_d}[(\sum_{i=1}^{d} \nabla_{\boldsymbol{\theta}} [\nabla_{\boldsymbol{x}}^2 \log p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)]_i + \frac{1}{2} \nabla_{\boldsymbol{\theta}} ([\nabla_{\boldsymbol{x}} \log p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)]_i)^2)(\sum_{j=1}^{d} \nabla_{\boldsymbol{\theta}} [\nabla_{\boldsymbol{x}}^2 \log p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)]_j + \frac{1}{2} \nabla_{\boldsymbol{\theta}} ([\nabla_{\boldsymbol{x}} \log p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)]_j)^2)^T] \\
&= \sum_{i,j}^{d} H_{ij}
\end{aligned}
$$

Hence, $\sqrt{N} \nabla_{\boldsymbol{\theta}} J_{NK}(\boldsymbol{\theta}^*) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \sum_{i,j=1}^{K} \boldsymbol{H}_{ij})$.□

Lemma 3 *Assuming $\boldsymbol{\Theta}$ is compact and $p_m(\boldsymbol{x}; \boldsymbol{\theta})$ being twice continuously differentiable with respect to $\boldsymbol{\theta}$, then*

$$
\sup_{\boldsymbol{\theta}} |\nabla_{\boldsymbol{\theta}}^2 J_{NK}(\boldsymbol{\theta}^*) - \nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*)| \xrightarrow{p} \mathbf{0}. \tag{23}
$$

Proof: We write down the second order derivative of $f(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{V}^*)$ as:

$$
\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}^*; \boldsymbol{x}, \boldsymbol{V}^*) = \sum_{i=1}^{K} \nabla_{\boldsymbol{\theta}}^2 [\boldsymbol{v}_i^{*T} \nabla_{\boldsymbol{x}}^2 \log p_m(\boldsymbol{x}; \boldsymbol{\theta}^*) \boldsymbol{v}_i^*] + \frac{1}{2} \nabla_{\boldsymbol{\theta}}^2 \|\nabla_{\boldsymbol{x}} \log p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)\|^2 \tag{24}
$$

By assumption, we have $\|\nabla_{\boldsymbol{\theta}}^2 [\nabla_{\boldsymbol{x}}^2 \log p_m(\boldsymbol{x}; \boldsymbol{\theta})]_{ii}\|_\infty < \infty$ and $\|\nabla_{\boldsymbol{\theta}}^2 \|\nabla_{\boldsymbol{x}} \log p_m(\boldsymbol{x}; \boldsymbol{\theta})\|^2\|_\infty < \infty$. Hence, $\|\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}^*; \boldsymbol{x}, \boldsymbol{V}^*)\| < \infty$. The assumptions for Lemma 2.4 (Newey & McFadden, 1994) are satisfied. So,

$$
\sup_{\boldsymbol{\theta}} |\frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}^*; \boldsymbol{x}, \boldsymbol{V}^*) - \mathbb{E}_{\boldsymbol{x} \sim p_d} [\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}^*; \boldsymbol{x}, \boldsymbol{V}^*)]| \xrightarrow{p} \mathbf{0}
$$

which is equivalent to:

$$
\sup_{\boldsymbol{\theta}} |\nabla_{\boldsymbol{\theta}}^2 J_{NK}(\boldsymbol{\theta}^*) - \nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*)| \xrightarrow{p} \mathbf{0}.\square
$$

Combining **Theorem 3** assumptions with **Lemma 2** and **Lemma 3** results, satisfies all the assumptions of Theorem 3.1 (Newey & McFadden, 1994). Hence,

$$
\sqrt{N}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*) \to \mathcal{N}\Big(\mathbf{0}, (\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*))^{-1} \left(\sum_{i,j=1}^{d} \boldsymbol{H}_{ij} \right) (\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*))^{-1} \Big).\square \tag{25}
$$

Remark: As a result, the asymptotic variance of MSSM is the same as SM's asymptotic variance and smaller than the one for SSM. Proof are presented in Appendix B.4 by Song et al. (2019):

- **SSM** with $p_{\boldsymbol{v}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$:

$$
\sqrt{N}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*) \to \mathcal{N}\Big(\mathbf{0}, (\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*; p_{\boldsymbol{v}}))^{-1} \left(\sum_{i,j=1}^{d} H_{ij} + \frac{2}{M} \sum_{i=1}^{d} H_{ii} + \frac{2}{M} \sum_{\substack{i,j=1 \\ i \neq j}}^{d} W_{ij} \right) (\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*; p_{\boldsymbol{v}}))^{-1} \Big) \tag{26}
$$

  Here,

$$
\begin{aligned}
W_{ij} = \mathbb{E}_{p_d}[(\nabla_{\boldsymbol{\theta}} [\nabla_{\boldsymbol{x}}^2 \log p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)]_{ij} + \frac{1}{2} \nabla_{\boldsymbol{\theta}} ([\nabla_{\boldsymbol{x}} \log p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)]_i [\nabla_{\boldsymbol{x}} \log p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)]_j))(\nabla_{\boldsymbol{\theta}} [\nabla_{\boldsymbol{x}}^2 \log p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)]_{ij} \\
+ \frac{1}{2} \nabla_{\boldsymbol{\theta}} [\nabla_{\boldsymbol{x}} \log p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)]_i [\nabla_{\boldsymbol{x}} \log p_m(\boldsymbol{x}; \boldsymbol{\theta}^*)]_j)^T].
\end{aligned} \tag{27}
$$

- **SSM** with $p_{\boldsymbol{v}}$ follow a Rademacher distribution:

$$\sqrt{N}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*) \to \mathcal{N}\Big(\mathbf{0}, (\nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^*; p_{\boldsymbol{v}}))^{-1} \left(\sum_{i,j=1}^{d} H_{ij} + \frac{2}{M} \sum_{\substack{i,j=1 \\ i \neq j}}^{d} W_{ij} \right) (\nabla^2_{\boldsymbol{\theta}} J(\theta^*; p_{\boldsymbol{v}}))^{-1} \Big) \tag{28}$$

- **SM** (Eq. (5)):

$$\sqrt{N}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*) \to \mathcal{N}\Big(\mathbf{0}, (\nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^*; p_{\boldsymbol{v}}))^{-1} \left(\sum_{i,j=1}^{d} H_{ij} \right) (\nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^*; p_{\boldsymbol{v}}))^{-1} \Big) \tag{29}$$

C. Proof: MSSM objective (Eq. (12) $\iff$ Eq. (13))

We assume that the model's score function $\boldsymbol{s}_m(\boldsymbol{x}, \boldsymbol{\theta})$ and the data score function $\boldsymbol{s}_d(\boldsymbol{x})$ are differentiable and satisfy $\mathbb{E}_{\boldsymbol{x}\sim p_d}[\|\boldsymbol{s}_d(\boldsymbol{x})\|^2] < \infty$ and $\mathbb{E}_{\boldsymbol{x}\sim p_d}[\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2] < \infty$. Besides, we assume that the boundary condition holds, *i.e.*, $\lim_{x\to+\infty} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})p_d(\boldsymbol{x}) = 0, \forall \boldsymbol{\theta}$. In the following, we prove that the MSSM objective

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \max_{\boldsymbol{V}} \quad \mathbb{E}_{\boldsymbol{x}\sim p_d}[\|\boldsymbol{V}^T(\boldsymbol{s}_d(\boldsymbol{x}) - \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))\|^2], \\ &\text{s.t.} \quad \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{T}} = I, \end{aligned} \tag{30}$$

is equivalent to:

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \max_{\boldsymbol{V}} \quad \mathbb{E}_{\boldsymbol{x}\sim p_d}[\boldsymbol{V}^T\nabla_{\boldsymbol{x}}\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\boldsymbol{V} + \frac{1}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2], \\ &\text{s.t.} \qquad \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}. \end{aligned} \tag{31}$$

Proof: We show how to derive a loss that doesn't depend on the unknown distribution $p_d(\boldsymbol{x})$ using: (1) the constraint $\boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}$ and (2) Integration by parts (cf. Lemma 4 by Hyvärinen (2005)):

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\frac{1}{2}\|\boldsymbol{V}^T\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{w}) - \boldsymbol{V}^T\boldsymbol{s}_d(\boldsymbol{x})\|^2], \quad \text{s.t.} \quad \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I} \\ &\iff \min_{\boldsymbol{\theta}} \max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\frac{1}{2}\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})^T\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) + \boldsymbol{s}_d(\boldsymbol{x})^T\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{s}_d(\boldsymbol{x}) - 2(\boldsymbol{V}^T\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))^T(\boldsymbol{V}^T\boldsymbol{s}_d(\boldsymbol{x}))], \quad \text{s.t.} \quad \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I} \\ &\overset{(i)}{\iff} \min_{\boldsymbol{\theta}} \max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\frac{1}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2 + \frac{1}{2}\|\boldsymbol{s}_d(\boldsymbol{x})\|^2 - (\boldsymbol{V}^T\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))^T(\boldsymbol{V}^T\boldsymbol{s}_d(\boldsymbol{x}))], \quad \text{s.t.} \quad \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I} \\ &\overset{(ii)}{\iff} \min_{\boldsymbol{\theta}} \max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\frac{1}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2 + \frac{1}{2}\|\boldsymbol{s}_d(\boldsymbol{x})\|^2 + \sum_{j=1}^{K}\boldsymbol{v}_j^T\nabla_{\boldsymbol{x}}\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\boldsymbol{v}_j], \quad \text{s.t.} \quad \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I} \\ &\overset{(iii)}{\iff} \min_{\boldsymbol{\theta}} \max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\frac{1}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2 + \sum_{j=1}^{K}\boldsymbol{v}_j^T\nabla_{\boldsymbol{x}}\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\boldsymbol{v}_j], \quad \text{s.t.} \quad \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I} \end{aligned}$$

Note, (i) is due to the fact that $\boldsymbol{V}\boldsymbol{V}^T = I$. Further, (iii) is due to the fact that $\boldsymbol{s}_d(\boldsymbol{x})$ does not depend on either $\boldsymbol{\theta}$ or $\boldsymbol{V}$. Moreover, (ii) is due to the integration by parts trick, as we show in the following:

$$\begin{aligned} &\max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\sum_{j=1}^{K}(\boldsymbol{v}_j^T\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))(\boldsymbol{v}_j^T\boldsymbol{s}_d(\boldsymbol{x}))] \\ &\iff \max_{\boldsymbol{V}} \sum_{j=1}^{K}\int_{\boldsymbol{x}} p_d(\boldsymbol{x})(\boldsymbol{v}_j^T\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))(\boldsymbol{v}_j^T\boldsymbol{s}_d(\boldsymbol{x}))\boldsymbol{d}\boldsymbol{x} \\ &\iff \max_{\boldsymbol{V}} \sum_{j=1}^{K}\int_{\boldsymbol{x}} p_d(\boldsymbol{x})(\boldsymbol{v}_j^T\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))(\boldsymbol{v}_j^T\nabla_{\boldsymbol{x}}\log p_d(\boldsymbol{x}))\boldsymbol{d}\boldsymbol{x} \\ &\iff \max_{\boldsymbol{V}} \sum_{j=1}^{K}\int_{\boldsymbol{x}} (\boldsymbol{v}_j^T\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))(\boldsymbol{v}_j^T\nabla_{\boldsymbol{x}}p_d(\boldsymbol{x}))\boldsymbol{d}\boldsymbol{x} \end{aligned}$$

$$
\begin{aligned}
&\iff \max_{\boldsymbol{V}} \sum_{j=1}^{K} \int_{\boldsymbol{x}} (\boldsymbol{v}_j^T \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))(\boldsymbol{v}_j^T \nabla_{\boldsymbol{x}} p_d(\boldsymbol{x}))d\boldsymbol{x} \\
&\iff \max_{\boldsymbol{V}} \sum_{j=1}^{K} \sum_{i=1}^{d} \int_{\boldsymbol{x}} (\boldsymbol{v}_j^T \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))(\boldsymbol{v}_j^{(i)} \frac{\partial p_d(\boldsymbol{x})}{\partial \boldsymbol{x}_i})d\boldsymbol{x} \\
&\iff \max_{\boldsymbol{V}} \sum_{j=1}^{K} \sum_{i=1}^{d} \int_{\boldsymbol{x}} (\boldsymbol{v}_j^T \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))(\boldsymbol{v}_j^{(i)} \frac{\partial p_d(\boldsymbol{x})}{\partial \boldsymbol{x}_i})d\boldsymbol{x} \\
&\overset{(iv)}{\iff} \max_{\boldsymbol{V}} \sum_{j=1}^{K} \sum_{i=1}^{d} \Big[(\boldsymbol{v}_j^T \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))(\boldsymbol{v}_j^{(i)} p_d(\boldsymbol{x}))\Big]_{-\infty}^{+\infty} - \int_{\boldsymbol{x}} \boldsymbol{v}_j^{(i)} p_d(\boldsymbol{x}) \boldsymbol{v}_j^T \frac{\partial \boldsymbol{s}_m(\boldsymbol{x})}{\partial \boldsymbol{x}_i} d\boldsymbol{x} \\
&\overset{(v)}{\iff} \max_{\boldsymbol{V}} \sum_{j=1}^{K} \sum_{i=1}^{d} - \int_{\boldsymbol{x}} \boldsymbol{v}_j^{(i)} p_d(\boldsymbol{x}) \boldsymbol{v}_j^T \frac{\partial \boldsymbol{s}_m(\boldsymbol{x})}{\partial \boldsymbol{x}_i} d\boldsymbol{x} \\
&\iff \max_{\boldsymbol{V}} - \sum_{j=1}^{K} \int_{\boldsymbol{x}} p_d(\boldsymbol{x}) \boldsymbol{v}_j^T \nabla_{\boldsymbol{x}} s_m(\boldsymbol{x}) \boldsymbol{v}_j d\boldsymbol{x} \\
&\iff \max_{\boldsymbol{V}} - \sum_{j=1}^{K} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\boldsymbol{v}_j^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x}) \boldsymbol{v}_j]
\end{aligned}
$$

Note, (iv) is due to partial integration. Further, (v) is the boundary condition, *i.e.*, $p_d(\boldsymbol{x})$ vanishes at infinity. Hence,

$$
\max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\sum_{j=1}^{K} (\boldsymbol{v}_j^T \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))(\boldsymbol{v}_j^T \boldsymbol{s}_d(\boldsymbol{x}))] \iff \max_{\boldsymbol{V}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[- \sum_{j=1}^{K} \boldsymbol{v}_j^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) \boldsymbol{v}_j]. \tag{32}
$$

D. Proof: MSSM relationship to MLE

As explained by Hyvarinen (2007), the objective: $\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim p_d}[\text{tr}(\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})) + \frac{1}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2$, can be derived using the 2nd-order Taylor approximation of the contrastive divergence loss:

$$
L_{\text{CD}}(\boldsymbol{x};\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim p_d(\boldsymbol{x})}[\log p_m(\boldsymbol{x};\boldsymbol{\theta}) - \mathbb{E}_{\boldsymbol{x}'|\boldsymbol{x}}[\log p_m(\boldsymbol{x}';\boldsymbol{\theta})]], \tag{33}
$$

with a 1-step Langevin Monte-Carlo as the sampler from the model distribution $p_m(\boldsymbol{x};\boldsymbol{\theta})$.

The Langevin Monte-Carlo sampler generates samples via:

$$
\boldsymbol{x}' = \boldsymbol{x} + \frac{\mu}{2} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) + \sqrt{\mu}\boldsymbol{\sigma}, \quad \boldsymbol{\sigma} \sim \mathcal{N}(0, I). \tag{34}
$$

The Taylor expansion gives:

$$
\begin{aligned}
\log p_m(\boldsymbol{x}';\boldsymbol{\theta}) &= \log p_m(\boldsymbol{x};\boldsymbol{\theta}) + \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})^T(\boldsymbol{x}-\boldsymbol{x}') + (\boldsymbol{x}'-\boldsymbol{x})^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})(\boldsymbol{x}'-\boldsymbol{x}) \\
&= \log p_m(\boldsymbol{x};\boldsymbol{\theta}) + \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})^T \left(\frac{\mu}{2}\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) + \sqrt{\mu}\boldsymbol{\sigma}\right) + \left(\frac{\mu}{2}\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) + \sqrt{\mu}\boldsymbol{\sigma}\right)^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) \left(\frac{\mu}{2}\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) + \sqrt{\mu}\boldsymbol{\sigma}\right).
\end{aligned}
$$

Leveraging the fact that $\mathbb{E}[\boldsymbol{\sigma}] = 0$ and $\mathbb{E}[\boldsymbol{\sigma}\boldsymbol{\sigma}^T] = I$, we obtain:

$$
\mathbb{E}_{\boldsymbol{x}'|\boldsymbol{x}}[p_m(\boldsymbol{x}';\boldsymbol{\theta})] = \mathbb{E}_{\boldsymbol{x}\sim p_d(\boldsymbol{x})}[\log p_m(\boldsymbol{x};\boldsymbol{\theta}) + \frac{\mu}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2 + \mu\,\text{tr}(\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})) + \frac{\mu^2}{4} \boldsymbol{s}_m^T(\boldsymbol{x};\boldsymbol{\theta}) \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})].
$$

Hence, Eq. (33) becomes:

$$
\begin{aligned}
L_{\text{CD}}(\boldsymbol{x};\boldsymbol{\theta}) &= -\mathbb{E}_{\boldsymbol{x}\sim p_d(\boldsymbol{x})}[\frac{\mu}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2 + \mu\,\text{tr}(\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})) + \frac{\mu^2}{4} \boldsymbol{s}_m^T(\boldsymbol{x};\boldsymbol{\theta}) \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}) \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})] \\
&= -\mathbb{E}_{\boldsymbol{x}\sim p_d(\boldsymbol{x})}[\frac{\mu}{2}\|\boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\|^2 + \mu\,\text{tr}(\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})) + \mathcal{O}(\mu^2)].
\end{aligned}
$$

SSM approximates the trace operator via the Hutchinson's trick, *i.e.*, the trace operator $\text{tr}(\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta}))$ is replaced with $\mathbb{E}_{p_{\boldsymbol{v}}}[\boldsymbol{v}^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\boldsymbol{v}]$, such that $\mathbb{E}_{p_{\boldsymbol{v}}}[\boldsymbol{v}\boldsymbol{v}^T] = \boldsymbol{I}$. In case the projection directions follow a Rademacher distribution, *i.e.*, $\boldsymbol{v} \in \{-1, 1\}^d$, 2^d vectors are needed to exactly estimate the trace operator. However for MSSM, the equality $\text{tr}(\nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})) = \sum_{i=1}^{d} \boldsymbol{v}^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_m(\boldsymbol{x};\boldsymbol{\theta})\boldsymbol{v}$ holds when only considering d vectors.

E. Training Details

We closely follow the training setup of Song et al. (2019) for the different applications. In the following, we derive a closed form expression for the weight parameters $\boldsymbol{\alpha} = \{\alpha_l\}_{l=1}^{L}$ of the **Kernel Exponential Families** (DKEF) model, in case of the MSSM loss.

DKEF parameterizes the unnormalized log-likelihood via $\log p_f(\boldsymbol{x}) = \sum_{l=1}^{L} \alpha_l k(\boldsymbol{x}, \boldsymbol{z}_l) + \log q_0(\boldsymbol{x})$, where $q_0(\boldsymbol{x})$ is a fixed function, and $k(\boldsymbol{x}, \boldsymbol{z}_l)$ is a mixture of R Gaussian kernels defined on the feature space $\boldsymbol{\phi}_r$ of a deep net and evaluated at L different inducing points $\boldsymbol{z}_l$:

$$k(\boldsymbol{x}, \boldsymbol{z}_l) = \sum_{r=1}^{R} \rho_r \exp\left(\frac{-1}{2\sigma_r^2} \|\boldsymbol{\phi}_r(\boldsymbol{x}) - \boldsymbol{\phi}_r(\boldsymbol{z}_l)\|^2\right). \tag{35}$$

The kernel parameters $\{\rho_r\}_{r=1}^{R}$,$\{\sigma_r\}_{r=1}^{R}$, $\{\boldsymbol{\phi}_r\}_{r=1}^{R}$ and the inducing points $\{\boldsymbol{z}_l\}_{l=1}^{L}$ are learned via gradient descent. We prove that for fixed ρ_r, σ_r, $\boldsymbol{\phi}_r$, $\boldsymbol{z}_l$ and $\boldsymbol{V}^*$, finding $\boldsymbol{\alpha}$ that minimizes the program,

$$\min_{\boldsymbol{\alpha}} \underbrace{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} \boldsymbol{v}_j^{*T} \nabla^2_{\boldsymbol{x}^{(i)}} \log p_f(\boldsymbol{x}^{(i)}) \boldsymbol{v}_j^*}_{J_1} + \underbrace{\frac{1}{N}\sum_{i=1}^{N} \frac{1}{2} \|\nabla_{\boldsymbol{x}} \log p_f(\boldsymbol{x}^{(i)})\|^2}_{J_2} + \frac{1}{2}\lambda_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|^2, \tag{36}$$

is equivalent to solving a linear system:

$$(\boldsymbol{G} + \lambda_{\boldsymbol{\alpha}} \boldsymbol{I})\boldsymbol{\alpha} = -\boldsymbol{b} - \boldsymbol{c}, \tag{37}$$

with $\lambda_{\boldsymbol{\alpha}}$ being a regularization parameters, and $\boldsymbol{G} \in \mathbb{R}^{L \times L}$ and $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^L$ are defined as follows:

$$\begin{aligned}
G_{l,l'} &= \frac{1}{N}\sum_{i=1}^{N} (\nabla_{\boldsymbol{x}^{(i)}} k(\boldsymbol{x}^{(i)}, \boldsymbol{z}_l))^T (\nabla_{\boldsymbol{x}^{(i)}} k(\boldsymbol{x}^{(i)}, \boldsymbol{z}_{l'})), \quad \forall l, l' \in [1, L] \\
b_l &= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} \boldsymbol{v}_j^{*T} \nabla^2_{\boldsymbol{x}^{(i)}} k(\boldsymbol{x}^{(i)}, \boldsymbol{z}_l) \boldsymbol{v}_j^*, \quad \forall l \in [1, L] \\
c_l &= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} \nabla_{\boldsymbol{x}^{(i)}} q_0(\boldsymbol{x}^{(i)})^T k(\boldsymbol{x}^{(i)}, \boldsymbol{z}), \quad \forall l \in [1, L].
\end{aligned} \tag{38}$$

Note, N is the batch size and K is the number of eigenvectors.

Proof:
In the following we express the quadratic (J_1) and squared norm (J_2) terms of the loss in Eq. (36) as functions of $\boldsymbol{\alpha}$, $\boldsymbol{G}$, $\boldsymbol{b}$ and $\boldsymbol{c}$. Starting with J_1:

$$\begin{aligned}
\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} \boldsymbol{v}_j^{*T} \nabla^2_{\boldsymbol{x}} \log p_f(\boldsymbol{x}) \boldsymbol{v}_j^* &= \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{L}\sum_{j=1}^{K} \alpha_l \boldsymbol{v}_j^{*T} \nabla_{\boldsymbol{x}^{(i)}} k(\boldsymbol{x}^{(i)}, \boldsymbol{z}_l) \boldsymbol{v}_j^* + \text{Const} \\
&= \sum_{l=1}^{L} \alpha_l b_l + \text{Const} \\
&= \boldsymbol{\alpha}^T \boldsymbol{b} + \text{Const}
\end{aligned} \tag{39}$$

Next, we express J_2 as a function of $\boldsymbol{\alpha}$, $\boldsymbol{G}$ and $\boldsymbol{c}$:

$$
\begin{aligned}
&\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\|\nabla_{\boldsymbol{x}}\log p_f(\boldsymbol{x})\|^2 \\
&=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\|\sum_{l=1}^{L}\alpha_l\nabla_{\boldsymbol{x}^{(i)}}k(\boldsymbol{x}^{(i)},\boldsymbol{z}_l)+\log q_0(\boldsymbol{x}^{(i)})\|^2 \\
&=\frac{1}{2N}\sum_{i=1}^{N}\Big(\sum_{l,l'}^{L}\alpha_l\alpha_{l'}\nabla_{\boldsymbol{x}^{(i)}}k(\boldsymbol{x}^{(i)},\boldsymbol{z}_l)^T\nabla_{\boldsymbol{x}^{(i)}}k(\boldsymbol{x}^{(i)},\boldsymbol{z}_{l'})+2\sum_{l=1}^{L}\alpha_l\nabla_{\boldsymbol{x}^{(i)}}k(\boldsymbol{x}^{(i)},\boldsymbol{z}_l)^T\nabla_{\boldsymbol{x}^{(i)}}q_0(\boldsymbol{x}^{(i)})\Big)+\text{Const} \\
&=\frac{1}{2}\sum_{l,l'}^{L}\alpha_l\alpha_{l'}G_{l,l'}+\sum_{l=1}^{L}\alpha_l c_l+\text{Const} \\
&=\frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{G}\boldsymbol{\alpha}+\boldsymbol{\alpha}^T\boldsymbol{c}+\text{Const}
\end{aligned}
\tag{40}
$$

Hence, the optimization problem in Eq. (36) becomes:

$$
\begin{aligned}
\boldsymbol{\alpha}&=\operatorname*{argmin}_{\boldsymbol{\alpha}}\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}[\boldsymbol{v}_j^{*T}\nabla^2_{\boldsymbol{x}^{(i)}}\log p_f(\boldsymbol{x}^{(i)})\boldsymbol{v}_j^*+\frac{1}{2}\|\nabla_{\boldsymbol{x}}\log p_f(\boldsymbol{x}^{(i)})\|^2]+\frac{1}{2}\lambda_{\boldsymbol{\alpha}}\|\boldsymbol{\alpha}\|^2 \\
&=\operatorname*{argmin}_{\boldsymbol{\alpha}}\frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{G}\boldsymbol{\alpha}+\boldsymbol{\alpha}^T\boldsymbol{b}+\boldsymbol{\alpha}^T\boldsymbol{c}+\frac{1}{2}\lambda_{\boldsymbol{\alpha}}\|\boldsymbol{\alpha}\|^2 \\
&=-(\boldsymbol{G}+\lambda_{\boldsymbol{\alpha}}\boldsymbol{I})^{-1}(\boldsymbol{b}+\boldsymbol{c}).
\end{aligned}
\tag{41}
$$