Gaussian Process Deep Belief Networks: A Smooth Generative Model of Shape with Uncertainty Propagation

Alessandro Di Martino [1], Erik Bodin [2], Carl Henrik Ek [2], and Neill D. F. Campbell [1]

[1] University of Bath, Department of Computer Science
[2] University of Bristol, Department of Computer Science

Abstract. The shape of an object is an important characteristic for many vision problems such as segmentation, detection and tracking. Being independent of appearance, it is possible to generalize to a large range of objects from only small amounts of data. However, shapes represented as silhouette images are challenging to model due to complicated likelihood functions leading to intractable posteriors. In this paper we present a generative model of shapes which provides a low dimensional latent encoding which importantly resides on a smooth manifold with respect to the silhouette images. The proposed model propagates uncertainty in a principled manner allowing it to learn from small amounts of data and providing predictions with associated uncertainty. We provide experiments that show how our proposed model provides favorable quantitative results compared with the state-of-the-art while simultaneously providing a representation that resides on a low-dimensional interpretable manifold.

Keywords: Shape Models · Unsupervised Learning · Gaussian Processes · Deep Belief Networks.

1 Introduction

The space of silhouette images is challenging to work with as it is not smooth in terms of a representation as pixels. A transformation that we would consider semantically smooth might correspond to a drastic change in pixel values. Our goal is to learn a smooth low dimensional representation of silhouette images such that images can be generated in a natural manner. Further, as data is at a premium, we want to learn a fully probabilistic model that allows us to propagate uncertainty throughout the generative process. This will allow us to learn from *smaller amounts of data* and also associate a quantified uncertainty to its predictions. This uncertainty allows the model to be used as a building block in larger models.

The results of our model challenge the current trend in unsupervised learning towards maximum likelihood training of increasingly large parametric models with increasingly large datasets. We demonstrate that by propagating uncertainty throughout the model, our approach outperforms two standard generative deep learning models, a Variational Auto-Encoder (VAE [15]) and a Generative Adversarial Network (InfoGAN [5]) with comparable architectures and can achieve similar performance with far smaller training datasets.

In our work we revisit a few classic machine learning models with complementary properties. On the one hand, parametric models such as Restricted Boltzmann Machines (RBMs) [25] are particularly interesting as they are stochastic, generative and can be stacked easily into *deeper* models such as deep belief networks (DBNs); these can be trained in a greedy fashion, layer by layer [13]. RBMs can approximate a probability distribution on visible units. DBNs, in addition, learn deep representations by composing features learned by the lower layers, yielding progressively more abstract and flexible representations at higher layers and often leading to more expressive and efficient models compared to shallow ones [2].

However, DBNs suffer from a number of limitations. Firstly, they do not guarantee a smooth representation in the learned latent space. Secondly, the classic contrastive divergence algorithm used for greedy training is slow and can place limitations on architectures. Finally, a DBN does not provide any explicit generative process from a manifold, as the standard way to sample from a DBN is to start from a training example and perform iterations of Gibbs sampling.

The Gaussian Process Latent Variable Model (GPLVM) [17] combines a Gaussian process (GP) prior with a likelihood function in order to learn a representation. By specifying a prior that encourages smooth functions a smooth latent representation can be recovered. However, to make inference tractable the likelihood is also chosen to be Gaussian which does not reflect the statistics of natural images. Further, even though the mapping from the latent space is non-linear the posterior is linear in the observed space. This makes the GPLVM unsuitable for modelling images. To circumvent this one can compose hierarchies of GPs [6], however, these models are inherently difficult to train.

The characteristics of the DBN and GPLVM can be considered complementary, where the DBN excels the GPLVM fails and vice versa. Unfortunately, combining the two models into a single one by simply stacking a GPLVM on top of a DBN would not preserve uncertainty propagation. Furthermore, this would pose a challenge to training (while the GPLVM is a non-parametric model trained by optimizing an objective function, a DBN is a parametric model, with non-differentiable Bernoulli units, and is trained with contrastive divergence). Another important challenge is learning from very little data. The ability to learn from a small dataset expands the applicability of a model to domains where there is a lack of available data or where collection of data is costly or time-consuming.

In this paper we address these challenges and present the following contributions:

1. A model (which we call GPDBN) that combines the properties of a smooth, interpretable manifold for synthesis with a data specific likelihood function (a deep structure) capable of decomposing images into an efficient representation while propagating uncertainty throughout the model in a principled manner.
2. We train the model end to end using back propagation with the same complexity as a standard feed-forward neural network by minimising a single objective function.
3. We also show that the model is able to learn from very little data, outperforming current generative deep learning models, as well as scaling linearly to larger datasets by the use of mini-batching.

	Non-Gaussian Likelihood	Explicit Smooth Low-Dim Manifold	Fully Generative	Propagates Uncertainty
GPLVM [17]		✓	✓	✓
GPLVMDT [22]		✓	✓	✓
DBN [13]	✓			✓
SBM [10]	✓			✓
VAE [15]	✓	∼	✓	
InfoGAN [5]	✓	∼	✓	
ShapeOdds [8]	✓		✓	✓
This work	✓	✓	✓	✓

Table 1. Summary of properties of related models.

2 Related Work

Modelling of shape is important for many computer vision tasks. It is beyond the scope of this paper to make a complete review of the topic, we refer the reader to the comprehensive work of Taylor *et al*. [7]. In our work we focus on recent unsupervised statistical models that operate directly on the pixel domain. Interest in these models was revived by the Shape Boltzmann Machine (SBM) work of Eslami et al. [10] and they have been shown to be useful for a variety of vision applications [9, 16, 28]. These deep models can also be readily extended into the 3D domain, *e.g*., by recent work on 3D ShapeNets [31]. Detailed analysis of the DBN, GPLVM and SBM is provided in § 3.

Desirable Properties Table 1 highlights the desirable properties of the most closely related previous works. We have identified four advantageous properties: (i) It is well known that pixel silhouettes are not well modelled by a Gaussian likelihood. (ii) The utility of an unsupervised shape model is well described by the properties of its latent representation. Ensuring a smooth manifold opens up a number of applications to data in the pixel domain that previously required custom representations, *e.g*., interactive drawing [29]. (iii) A fully generative model ensures that there is a well defined space that can be sampled as well as interpreted; *e.g*., dynamics models can be defined in such a space to perform tracking [20, 22]. (iv) Correctly propagating uncertainty is vital to perform data efficient learning, for example when data is scarce or expensive to obtain.

Auto-Encoders The VAE model by Kingma and Welling [15] performs a variational approximation of a generative model with a non-Gaussian likelihood through a feed-forward or Multi-Layer Perceptron (MLP) network. In addition, it uses MLP networks to encode the variational parameters (in a similar manner to [18]). While this model provides a generative mapping, the feed-forward (decoder) network fails to propagate uncertainty from the latent space. Furthermore, the independent prior on the latent space does not promote a smooth manifold; any smoothness arises as a by-product of the MLP encoding network. This characteristic depends on the MLP architecture and is not directly parametrised. The key limitation of the VAE for our purposes is the lack of uncertainty propagation that results in poor results with limited training data.

The guided, non-parametric autoencoder model of Snoek et al. [26] appears similar, however, there are a number of important differences. They use label information (supervision) to guide a latent space learning process for an autoencoder; this is not a pure unsupervised learning task and we do not have label information available to us.

Furthermore, as with the VAE, uncertainty is not propagated from the latent manifold to the output space due to the use of the feed-forward network to the output.

InfoGAN Another prominent generative model in unsupervised learning is the Generative Adversarial Network (GAN) [11]. The model learns an implicit generator distribution using a minimax game between a deep generator network, which transforms a noise variable to a sample, and a deep discriminator network, which is used to classify between samples from the generator distribution and the true data distribution. One issue common with GAN models is that they do not provide a smooth latent manifold for synthesis nor uncertainty in their estimates (like the VAE). From the plethora of different variations of GANs models available in the literature we have chosen to include in our comparisons the InfoGAN model [5], since it also considers the goal of interpretable latent representations (by maximising the mutual information between a subset of GAN's noise variables and observations).

ShapeOdds The recent ShapeOdds work of Elhabian and Whitaker [8] confers state-of-the-art performance and captures many of the desired properties including a generative probabilistic model that propagates uncertainty. The approach taken is quite different to ours as they specify a detailed probabilistic model including a Gaussian Markov Random Field (MRF) with individual Bernoulli random variables for the pixel lattice. In contrast, our model is more flexible, we allow the network to learn the structure from the data directly but ensure that we still maintain uncertainty quantification throughout. We would also argue that the specific form of the low dimensional manifold we generate is desirable with its guaranteed smoothness that makes the latent space readily interpretable. This provides the tradeoff between the two models. We expect the ShapeOdds model to perform very well at generalisation due to the inclusion of the MRF prior. In contrast, our model will be more data dependent in this respect (weaker prior assumptions on the nature of images), however, it provides a generative space that is highly interpretable and easy to work with. We identify that a topic for further work would be to combine our smooth priors with the likelihood model of ShapeOdds.

GPLVM Representations A possible workaround to the problem of non-Gaussian likelihoods is to perform a deterministic transformation to a domain where the data is approximately Gaussian. This has been successful for domains where, for example, the shape can be represented in a new geometric representation away from pixels, *e.g.*, parametric curves [3, 23]. However, this is application dependent and not suitable for arbitrary pixel based silhouettes considered here. A common approach that retains the pixel grid is to transform it into a level-set problem via the distance transform, *e.g.*, [22]. This can improve results in some settings, however, the uncertainty is not correctly preserved and therefore not correctly captured in predictions. We denote this model GPLVMDT in our comparisons.

3 Background

3.1 Deep Belief Networks

RBM The restricted Boltzmann machine (RBM), or Harmonium, [25] is a generative stochastic neural network that learns a probability distribution over a vector of random

variables. The RBM is when stacked the basic the basic component of a deep belief network. The graphical model of the RBM is an undirected bipartite graph, consisting of a set of visible random variables (or units): $\boldsymbol{v}$, and a set of hidden units $\boldsymbol{h}$ (Fig. 1(a)). Typically, all variables are binary (Bernoulli), taking on values from $\{0, 1\}$.

The RBM model specifies a probability distribution over both the visible and hidden variables jointly as

$$P(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h})\right) \tag{1}$$

which defines a Gibbs distribution with energy function

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h} - \boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} \,, \tag{2}$$

where $\boldsymbol{W}$, $\boldsymbol{b}$, $\boldsymbol{c}$ are the parameters of the model: $\boldsymbol{W}$ as a linear weight matrix and $(\boldsymbol{b}, \boldsymbol{c})$ are bias vectors for the visible and hidden units respectively. The normalising constant Z is the, computationally intractable, sum over all possible random vectors $\boldsymbol{v}$ and $\boldsymbol{h}$.

The bipartite structure of the model (*i.e.*, the graph has no visible-visible or hidden-hidden connections, as shown in Fig. 1(a)), affords efficient Gibbs sampling from the visible units given the hidden variables (or vice versa). The conditional distribution of the hidden units given the visible ones, and vice versa, factorize as each set of variables are conditionally independent given the other:

$$P(\boldsymbol{h} \,|\, \boldsymbol{v}) = \textstyle\prod_{j=1}^{H} P(h_j \,|\, \boldsymbol{v}), \;\; P(\boldsymbol{v} \,|\, \boldsymbol{h}) = \textstyle\prod_{i=1}^{V} P(v_i \,|\, \boldsymbol{h}) \,. \tag{3}$$

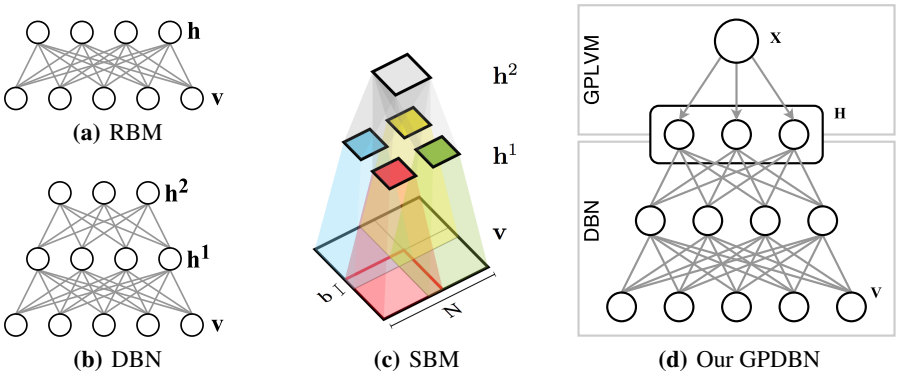
Replacing binary units with Gaussian units can be performed by modifying the energy function [12]. Unfortunately, parameter learning is difficult since direct calculation of the gradients of the log likelihood w.r.t. the parameters requires the intractable computation of the normalising constant Z. In *current* practice, the approximate maximum-likelihood contrastive divergence algorithm is used [4].

DBN When multiple layers of RBMs are stacked on top of each other they form a deep belief network (Fig. 1(b)). Hinton *et al*. [13] demonstrated that a DBN can be trained in a greedy fashion, layer by layer. Essentially, the samples (activations) from the hidden units of a trained layer are used as the data to train the next layer in the stack.

Sampling Sampling from an RBM proceeds by conditioning on some input data and performing a Gibbs sample for the hidden units. Subsequently, a Gibbs sample can be drawn for the visible units by conditioning the hidden units on this sample. This process is then repeated for a number of cycles. Since a DBN is a stack of RBMs, this process has to be repeated for all layers; the output of one layer becomes the input to condition on for the next layer. In this way, an input data point can be propagated up and down the network.

Limitations Although a DBN is good at learning low-dimensional stochastic representations of high-dimensional data, it has three key drawbacks that we will address by combining the strengths of the DBN with a flexible non-parametric model in § 4:

1. It lacks a directed generative sampling process from a well defined latent representation. In order to generate a sample one must condition on some input data and propagate it through the network back and forth until a sample from the lowest layer is obtained.



(a) RBM **(b)** DBN **(c)** SBM **(d)** Our GPDBN

Fig. 1. Graphical representations of the (a) RBM, (b) DBN and (c) SBM. (d) A graphical representation of our proposed GPDBN model where $\boldsymbol{X}$ represents the latent variables, $\boldsymbol{H}$ the Gaussian activations (10), and $\boldsymbol{V}$ the observed (data) space. *(The SBM figure is taken from [10].)*

2. There is no explicit representation of the uncertainty, instead this only arises implicitly through the propagation of point estimates (samples) at each layer.
3. A side effect of the conditional independence assumption of (3) is that the correlations between the hidden units of the top layer of a DBN are not captured because each latent dimension is independent. Most importantly, a DBN does not, therefore, give any guarantee about learning a smooth latent space.

3.2 GPLVM

The Gaussian Process Latent Variable Model (GPLVM) [17] learns a generative representation by placing a Gaussian process (GP) prior over the mapping from the latent to the observed data. This approach has the benefit that it is very easy to ensure a smooth mapping from the latent representations to the observed data. Further, due to the principled uncertainty propagation of the GP, all predictions will have an associated uncertainty.

In specific, each observed datapoint $\boldsymbol{y}_n$, $n \in [1, N]$, is assumed to be generated by a latent location $\boldsymbol{x}_n$ through a mapping f. Due to the marginalising property of a Gaussian, the predictive posterior over function values $\boldsymbol{f}^*$ at a test location $\boldsymbol{x}^*$ can be reached in closed form as,

$$p(\boldsymbol{f}^* \mid \boldsymbol{Y}, \boldsymbol{x}^*, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{m}_{\mathrm{GP}}, \sigma^2_{\mathrm{GP}}) \tag{4}$$

$$\boldsymbol{m}_{\mathrm{GP}} = k(\boldsymbol{x}^*, \boldsymbol{X})[k(\boldsymbol{X}, \boldsymbol{X})]^{-1}\boldsymbol{Y} \tag{5}$$

$$\sigma^2_{\mathrm{GP}} = k(\boldsymbol{x}^*, \boldsymbol{x}^*) - k(\boldsymbol{x}^*, \boldsymbol{X})[k(\boldsymbol{X}, \boldsymbol{X})]^{-1}k(\boldsymbol{X}, \boldsymbol{x}^*)\,, \tag{6}$$

where $k(\cdot, \cdot)$ is the covariance function specifying the Gaussian process and $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top$. We used the common *squared exponential* kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \alpha^2 \exp\left(-\frac{1}{2\ell^2}\left\|\boldsymbol{x} - \boldsymbol{x}'\right\|^2\right)\,, \tag{7}$$

with hyperparameters α^2 (signal variance) and ℓ (lengthscale), to ensure a smooth manifold. Importantly, even though the function f can be non-linear, the relationship between the predicted mean (5) and the training data $\boldsymbol{Y}$ is linear. Due to this linearity, a GPLVM is inherently not suitable for modeling image data.

3.3 Shape Boltzmann Machine

The Shape Boltzmann Machine (SBM) [10] is a specific architecture of the Boltzmann machine. It consists of three layers: a rectangular layer of $N \times M$ visible units $\boldsymbol{v}$, and two layers of latent variables: $\boldsymbol{h}^1$ and $\boldsymbol{h}^2$. Each hidden unit in $\boldsymbol{h}^1$ is connected only to one of the four subsets of visible units of $\boldsymbol{v}$ (Fig. 1(c)). Each subset forms a rectangular patch and the weights of each patch (except the biases) are shared so that a patch effectively behaves as a local receptive field. To avoid boundary inconsistencies, the patches are slightly overlapped (in Fig. 1(c), the overlap has size b). Layer $\boldsymbol{h}^2$ is fully connected to $\boldsymbol{h}^1$.

While the SBM offers improved generalization over a DBN with the same number of parameters, the SBM has a fixed structure which is not easily extended to more layers or patches. In contrast, a DBN, as a stack of simple RBMs, has a more generic and flexible structure which can be adapted easily and combined with other models. Furthermore, like the DBN, the SBM lacks of a proper generative process.

4 The GPDBN Model

In our model, we connect a DBN and GPLVM so that the data space of the GPLVM corresponds the latent space of the DBN (Fig. 1(d)) to obtain a model that can be optimized by minimizing a single objective function.

New Concrete Layers The uppermost hidden layer of the DBN has Gaussian units to interface with the Gaussian likelihood of the GPLVM. In the lower layers, we replace the standard binary units with a *Concrete distribution* [21]. This is a continuous relaxation to discrete random variables, in our case, to the Bernoulli distribution. This allows us to draw low bias samples, in an analogous manner to the reparameterization trick [15], using a function that is differentiable with respect to the model parameters,

$$\text{Concrete}\,(p,u) = \text{Sigmoid}\big(\tfrac{1}{\lambda}\big(\log p - \log(1-p) + \log u - \log(1-u)\big)\big)\ , \quad (8)$$

where p is the parameter of a Bernoulli distribution, λ is a scaling factor, which we fix to 0.1, and u is a uniform sample from $[0,1]$.

Learning Given a dataset $\mathcal{D} = \{\boldsymbol{t}_n\}_{n=1}^N$, we train the model end-to-end by minimizing the following objective function jointly with respect to all the parameters and the matrix of latent points $\boldsymbol{X}$ (omitted from the notation to avoid clutter):

$$L = \sum_{n=1}^{N} \underbrace{(\boldsymbol{t}_n \log(\boldsymbol{s}_n) + (1-\boldsymbol{t}_n)\log(1-\boldsymbol{s}_n))}_{\text{data term}} + \tfrac{1}{2}\underbrace{\text{Tr}\big[\boldsymbol{K}^{-1}\boldsymbol{H}\boldsymbol{H}^\top\big]}_{\text{joint term}} + \tfrac{D}{2}\underbrace{\log|\boldsymbol{K}|}_{\substack{\text{complexity}\\ \text{term}}} + \underbrace{||\boldsymbol{X}||^2}_{\substack{\text{prior}\\ \text{term}}}\ . \quad (9)$$

Here, $\boldsymbol{t}_n$ is a training datapoint, $\boldsymbol{s}_n$ is a sample from the model, $\boldsymbol{K} = k(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}_N$ is the covariance matrix of the latent points and D is the number of Gaussian units in the uppermost DBN layer (equal to the dimension of the GPLVM output space). We use a standard Gaussian as the prior on $\boldsymbol{X}$. The variance of the noise parameter is σ^2 and $\boldsymbol{I}_N$ is an $N \times N$ identity matrix.

To join the two models, the $N \times D$ matrix of activations $\boldsymbol{H}$, from the Gaussian units, is defined as:

$$\boldsymbol{H} = \boldsymbol{A} + \boldsymbol{\sigma}^{\mathrm{GP}} \otimes \boldsymbol{\sigma}^{\mathrm{DBN}} \odot \boldsymbol{\mathcal{E}} \,, \tag{10}$$

where $\boldsymbol{A} = [\boldsymbol{m}_1^{\mathrm{GP}}, \ldots, \boldsymbol{m}_N^{\mathrm{GP}}]^\top$ is a matrix in which each row is the mean output of the Gaussian units corresponding to each input training datapoint. This is combined with $\boldsymbol{\sigma}^{\mathrm{GP}}$, the $N \times 1$ vector of predictive standard deviations from the GPLVM , and $\boldsymbol{\sigma}^{\mathrm{DBN}}$, the $1 \times D$ vector of standard deviation parameters of the Gaussian units. Note that $\otimes$ is an outer product, and $\odot$ is an element-wise product.

The $\boldsymbol{H}$ matrix represents the observed data for the GPLVM and is updated at each training iteration by sampling $\boldsymbol{\mathcal{E}}$ a different $N \times D$ matrix of independent Gaussian noise, $\mathcal{E}_{n,d} \sim \mathcal{N}(0, 1)$. This is a second application of the reparameterization trick. At each iteration, $\boldsymbol{H}$ is always normalized, to match our zero mean GP assumption, by subtracting its column-wise mean and dividing by $\boldsymbol{\sigma}^{\mathrm{DBN}}$.

Minibatches The objective (9) can be evaluated on an uniformly drawn subset of data $\{\boldsymbol{t}_b\}_{b=1}^B$ yielding an estimator for the full objective,

$$\begin{aligned} L_{\text{batched}} \simeq \tfrac{N}{B} \textstyle\sum_{b=1}^{B} \left(\boldsymbol{t}_b \log(\boldsymbol{s}_b) + (1 - \boldsymbol{t}_b) \log(1 - \boldsymbol{s}_b)\right) + \tfrac{N}{2B} \mathrm{Tr}\left[\boldsymbol{K}_B^{-1} \boldsymbol{H}_B \boldsymbol{H}_B^\top\right] \\ + \tfrac{ND}{2B} \log |\boldsymbol{K}_B| + \tfrac{N}{B} ||\boldsymbol{X}_B||^2 \,, \end{aligned} \tag{11}$$

where $\boldsymbol{H}_B$ and $\boldsymbol{K}_B$ corresponds to $\boldsymbol{H}$ and $\boldsymbol{K}$ evaluated on the subset $\boldsymbol{X}_B$ of $\boldsymbol{X}$. Using this estimator the model can be optimised using mini-batching to scale linearly to larger datasets. We note that the matrix inversion does introduce bias into the estimator; empirical results suggest this is small and removing it is a topic for future work.

Scaling via Convolutional Architecture When defining the likelihood directly over the pixels, the fully-connected conditional independence of the RBM layers limits scalability in terms of image size. This can be circumvented by adding convolution and deconvolution steps to replace the dense matrix product in (2) in the lower layers.

Sampling A sample $\boldsymbol{s}_n$ from the model is drawn by first generating a hidden sample $\boldsymbol{h}_n$ from latent point $\boldsymbol{x}_n$:

$$\boldsymbol{h}_n(\boldsymbol{x}) = (\boldsymbol{m}_n^{\mathrm{GP}} + \sigma_n^{\mathrm{GP}} \times \boldsymbol{\epsilon}_n) \odot \boldsymbol{\sigma}^{\mathrm{DBN}} + \boldsymbol{h}_\mu \,, \tag{12}$$

using $\boldsymbol{m}_n^{\mathrm{GP}}$ and σ_n^{GP} as the predictive mean and standard deviation of the GPLVM given latent point $\boldsymbol{x}_n$. This is combined with a sample $\boldsymbol{\epsilon}_n$, a $1 \times D$ vector of spherical Gaussian noise. The term $\boldsymbol{h}_\mu$ is the mean vector that is subtracted from $\boldsymbol{H}$ in the normalization step. The sample $\boldsymbol{h}_n$ is then propagated down through the DBN, sampling layer-by-layer, to give an output sample $\boldsymbol{s}_n$.

Prediction and Projection Since we have a simple sampling process, we can propagate uncertainty for our predictions by taking the empirical mean of a set of J samples from the model as $\boldsymbol{s}_* = \frac{1}{J}\sum_j^J \boldsymbol{s}_j\big(\boldsymbol{h}_j(\boldsymbol{x}_*)\big)$ for the latent location $\boldsymbol{x}^*$. Since we can efficiently take gradients through the sampling process, we can project new observations into the latent space by minimizing the reprojection error w.r.t. the latent locations for predictions from a set of random starting locations in the manifold.

Interpretation We note that the objective (9) consists of terms in contrast with each other. The first encodes a *data* term that ensures the observed data is well represented by the model. The third provides a *complexity* term that encourages a simple (low complexity) latent space $\boldsymbol{X}$ through the covariance matrix $\boldsymbol{K}$ to prevent overfitting.

The second term "glues" the two models together by ensuring that the covariance matrix $\boldsymbol{K}$ is a good model of the covariance of the Gaussian units at the top of the DBN. This in turn, ensures that the DBN learns an appropriate network to give sensible Gaussian activations rather than the unconstrained binary activations from a normal DBN. The last term encodes a *prior* which encourages the latent points to stay close to the origin.

The applications of the reparamerization trick ensures that efficient, low variance samples can be taken during training with gradients propagated throughout all parts of the network. The use of sampling and stochastic networks allows uncertainty to be propagated down through the entire model as well to ensure uncertainty is well quantified both at training and test time.

5 Experiments

In keeping with previous work, we evaluated our models in terms of four experiments: (i) *Synthesis*, that is, generating samples that are plausible. (ii) *Representation and Generalisation*, demonstrating the ability to capture the variability of the silhouettes away from the training data. (iii) *Smoothness*, evaluating the quality of the learned latent space through interpolation; smooth trajectories in the latent space should produce smooth variations in the silhouette space. (iv) *Scaling*, evaluating how the model performs with respect to the size of the training dataset.

Our Models In the comparisons, our main model (which we will refer to as GPDBN) consists of a three-layer DBN plus a GPLVM layer connected as described in § 4. From the bottom (observed) to the top (hidden) layer the architecture consists of 200 (Concrete units), 100 (Concrete) and 50 (Gaussian). The connected GPLVM layer has only 2 latent dimensions for easy visualisation. The model is optimized jointly as described in § 4. Our second model, GPSBM, is similar to the GPDBN where the three-layer DBN has been replaced with an SBM architecture of [10] with hidden Concrete units in the bottom layer and hidden Gaussian units at the top. We implemented all our models in the TensorFlow [1] framework and trained using the Adam optimizer [14].

Baselines For comparison, we compared our models to size baselines: (i) A vanilla GPLVM with 2 latent dimensions. (ii) GPLVMDT, a GPLVM operating on a signed distance function representation in a similar manner to [22]; samples are obtained by thresholding through the hyperbolic tangent function. (iii) The state-of-the-art ShapeOdds

model [8]. (iv) A DBN with binary units and the same architecture as our GPDBN. (v) The SBM [10] model with binary units (trained layer by layer with contrastive divergence like the DBN) with the same architecture as our GPSBM. (vi) The VAE [15] model with the same architecture as our GPDBN (mirrored for the decoder) and 2 latent dimensions. (vii) An InfoGAN [5] with same architecture as the VAE and GPDBN (mirrored for the discriminator) and 2 latent dimensions of structured noise.

Datasets In keeping with previous work, we trained the models on the Weizmann horse dataset [24], which consists of 328 binary silhouettes of horses facing left. The limited number of training samples and the high variability in the position of heads, tails, and legs make this dataset difficult. We also trained the models on 300 binary images from the Caltech101 dataset of motorbikes facing right [19]. All images in both datasets have been cropped and normalized to 32×32 pixels. The test datasets consisted of the challenging held-out data from [10]; an additional 14 horses and 9 motorbikes not contained in the training datasets.

Synthesis Fig. 2(a), shows the manifold learned by the GPDBN on the Weizmann horse dataset. Each blue point on the manifold represents the latent location corresponding to a training datapoint. The heat map is given by the log predictive variance (6) that encodes uncertainty in the latent space. The model is more likely to generate valid shapes from any location in the bright regions (*i.e.*, low variance regions).

Unlike GP based models, a standard DBN (or the SBM) does not learn such a generative manifold. This implies, first of all, that a DBN does not allow us to sample "from the top" in a direct manner. Instead we must provide a test image to the visible units and condition on it before propagating it up and down the network for a few iterations to obtain a sample. Secondly, like the VAE and InfoGAN, a DBN does not provide information about how plausible a generated sample is.

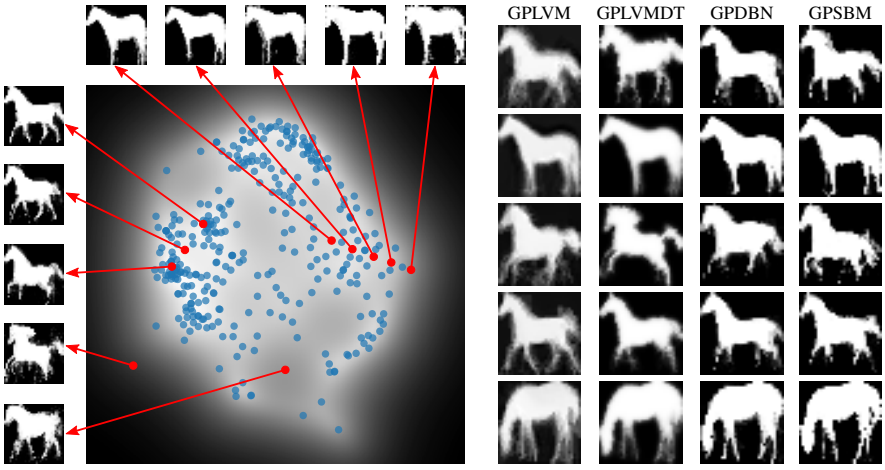
A smooth generative manifold, such the one learned by our model in Fig. 2(a) is informative as it gives us an indication about where to sample from to get plausible silhouettes. Fig. 2(b) compares silhouettes generated by the models that allow sampling from the manifold.[3] We note that the GPLVM and GPLVMDT produce blurry images since the shapes present interpolation artifacts from the Gaussian likelihood. In contrast, the results from both the GPDBN and GPSBM are sharper.

Representation and Generalisation In the recent literature on shape modelling, quantitative results are reported in terms of the distance between the test data not seen by the model and the most likely prediction under the model. For the models that can be sampled from, this amounts to finding the location on the manifold that most closely represents the test input (discussed for our model in § 4). For the models that learn an explicit manifold we find the closest silhouette to a test silhouette $\boldsymbol{t}^*$ by minimising the following objective with respect to a latent location $\boldsymbol{x}^*$ on the manifold:

$$L_{\text{proj}}(\boldsymbol{x}^*) = \tfrac{1}{P} \textstyle\sum_{i=1}^{V} \left(\boldsymbol{t}^* \log(\boldsymbol{s}_i) + (1 - \boldsymbol{t}^*) \log(1 - \boldsymbol{s}_i)\right) + \gamma \times \log(\sigma^2(\boldsymbol{x}^*)) \,, \quad (13)$$

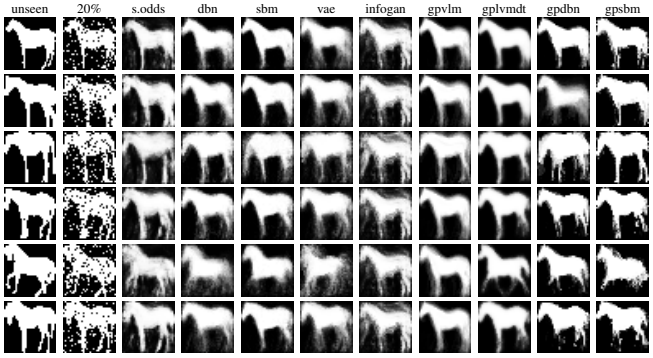
where we use V samples to evaluate the cross entropy to the test silhouette. The second term is the log predictive variance of the latent location $\boldsymbol{x}^*$ (as defined in Eq.(6)), this

[3] When we show generated silhouettes from any model, we actually show grayscale images denoting pixel-wise probabilities of turning white rather than binary samples.



(a) GPDBN horse manifold. **(b)** Qualitative comparison of samples.

Fig. 2. (a) Manifold learned by the GPDBN model on the Weizmann horse dataset. Moving over the manifold changes the pose of the horse with smooth paths in the manifold producing smooth transitions in silhouette pose. The heat map encodes the predictive variance of the model with darker regions indicating higher uncertainty and lower confidence in the silhouette estimates. (b) Qualitative comparison of silhouettes generated from low variance manifold areas by each of the models (images manually ordered by visual similarity).



(a) Example results for projection onto manifold (20% noise).

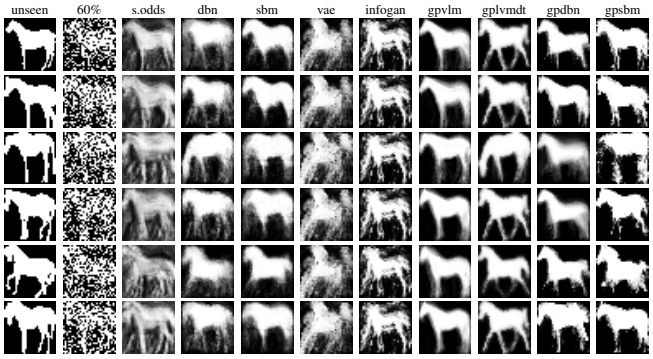
Method	SSIM
ShapeOdds	0.43 ± 0.06
DBN	0.43 ± 0.10
SBM	0.54 ± 0.11
VAE	0.36 ± 0.08
InfoGAN	0.27 ± 0.06
GPLVM	0.48 ± 0.07
GPLVMDT	0.54 ± 0.09
GPDBN	0.54 ± 0.12
GPSBM	0.59 ± 0.08

(b) SSIM score (higher is better).

Fig. 3. Manifold projection from corrupted observations. (a) Test silhouettes (first column) are corrupted with 20% salt and pepper noise (second column). The remaining columns show estimated silhouettes from each model. (b) Mean and standard deviation of the SSIM score between silhouettes from each model against the original test data without noise.

encourages the model to generate plausible silhouettes from the manifold. The scaling factor γ ensures that the two term have approximatively the same scale.

Samples for a DBN (or SBM) are usually generated by conditioning on an observed sample and propagating it through the network for several cycles, as described in § 3.1, with Gibbs samples taken after a burn in period. In our experiments, we fixed the condi-



(a) Example results for projection onto manifold (60% noise).

Method	SSIM
ShapeOdds	0.25 ± 0.04
DBN	0.27 ± 0.05
SBM	0.35 ± 0.05
VAE	0.11 ± 0.02
InfoGAN	0.18 ± 0.03
GPLVM	0.44 ± 0.07
GPLVMDT	0.37 ± 0.03
GPDBN	0.42 ± 0.10
GPSBM	0.51 ± 0.09

(b) SSIM score (higher is better).

Fig. 4. Manifold projection from corrupted observations. (a) Test silhouettes (first column) are corrupted with 60% salt and pepper noise (second column). The remaining columns show estimated silhouettes from each model. (b) Mean and standard deviation of the SSIM score between silhouettes from each model against the original test data without noise.

tioning on the test datapoint and averaged the results of a number of propagated samples through the model to prevent the sample chain from drifting away from the test data.

Projection under Noise To provide a challenging evaluation, we take unseen test data, corrupt it with noise and ask each models to find their most likely silhouette. Simply asking to reconstruct the test data would not be a sufficient evaluation since an identity mapping would be able to perform this task. Instead, we need the model to demonstrate that it can reject data that should not be in the trained model (the noise). In Fig. 3(b), we report the results for our proposed model and the baseline methods. We use the Structured Similarity (SSIM) [30] metric (range [0,1] with high values better) with a small window size of 3 to perform quantitative evaluations since it is known to outperform both cross-entropy and MSE as a perceptual metric. A random sample of corresponding silhouettes for the horse dataset are provided in Fig. 3(a). We also test our model in a more challenging environment, Fig. 4, where test data has been corrupted by significant noise. The quantitative comparisons shown that our GPDBN and GPSBM models have captured a high quality probabilistic estimate of the data manifold while still preserving interpretability.

Interpolation Test We trained a GPDBN, VAE and InfoGAN models on a 30 image dataset (which we call *stars* dataset) generated from a *known* 1-dimensional manifold using a simple script. The full dataset is displayed in the top row of Fig. 5. The deterministically generated dataset allows us to determine quantitatively whether interpolations in the latent space are representative of the true data distribution. The middle rows of Fig. 5 show the model outputs for the interpolation between two latent points corresponding to a four-pointed *star* (leftmost sample) and a *square* (rightmost sample). The uncertainty information of the GPDBN allows us to go from one point to the other passing through low-variance regions by following a geodesic [27]. We can see that the GPDBN produces smoothly varying shapes of high quality that reflect the true manifold. In contrast, the VAE and InfoGAN results do not smoothly follow the true

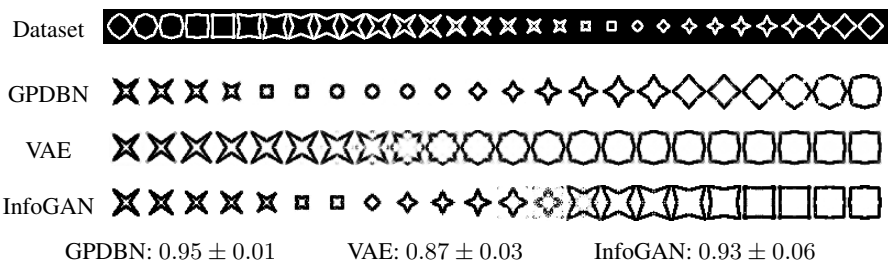


Fig. 5. Example results of the interpolation test between two training points from the stars dataset. The top row shows the geodesic interpolation generated by the GPDBN. The middle and the last rows are the linear interpolation generated by the VAE and InfoGAN respectively. The bottom row provides the mean and standard deviation of the SSIM score over 10 interpolation experiments. (In this picture black and white are inverted respect to the training dataset.)

Fig. 6. Graph showing the SSIM score of the output of the GPDBN, InfoGAN and VAE models against the test data without noise as the training dataset size increases from 100 to 10000 points. A higher score is better.

manifold and contain some erroneous interpolants that are not part of the true distribution; this is supported by the quantitative results that measure the quality of the samples to the true data using SSIM. The ability to exploit variance information in the GPDBN is clearly an advantage over the VAE and InfoGAN where the absence of direct access to the latent predictive posterior distribution prevents easy access to geodesics. Further demonstrations of the smoothness are available in supplementary material.

Scaling Experiments In Fig. 6 we compare the performance of the GPDBN, InfoGAN and VAE models as the size of the training dataset increases; here we use the standard MNIST digit dataset. We used a 10-dimensional latent space for all of the three models to account for the larger quantity of data. Similarly to the experiments in Figs. 3 and 4, we took 30 random images from the MNIST test data, add 20% salt-and-pepper noise, and calculated the SSIM score between the output of the models and the test data without noise. We plotted the score against dataset size (in log scale). We can see that the GPDBN model is able to capture a high quality model of the data manifold even from small datasets; for example, it achieves the same quality as a VAE trained on 10,000 images using only 100. We argue that the propagation of uncertainty throughout

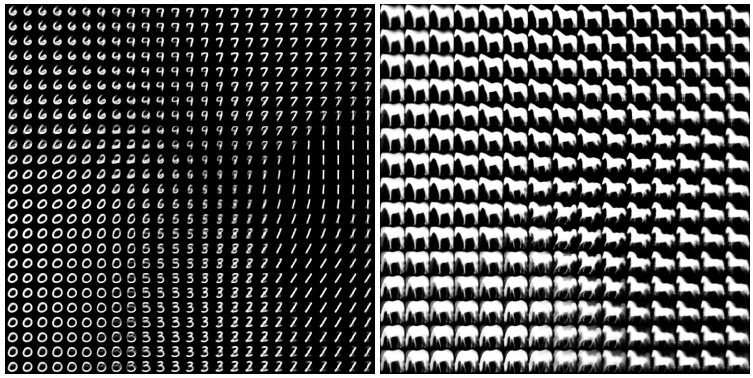


Fig. 7. The model can be made to scale to a large number of datapoints by optimizing the objective using mini-batching (11). Furthermore, scalability in the size of images can be obtained by adding upscaling and convolutions in the lowermost layer. Left: GPDBN trained on MNIST comprised of 60,000 28x28 images. Right: GPDBN trained on Weizmann Horses comprised of 328 300x300 images.

the model provides the advantage over both the VAE and InfoGAN which are both trained with only maximum likelihood approaches.

In Fig. 7 we provide results that demonstrate that our approach also overcomes scaling issues normally present in GP models and DBNs. Firstly, we show training on the 60,000 MNIST images via our proposed mini-batching approach. In addition, we also show the manifold for higher resolution images from the horse dataset (300×300). By using convolutional architectures, we can scale the number of parameters in an identical manner to convolutional feed-forward networks and our concrete layers allow us to train from random weight initialisation using back propagation without the need to use slow contrastive divergence. With both these approaches we still maintain our full uncertainty model so the same model can perform well with small and large datasets.

6 Conclusion

We have presented the GPDBN, a model that combines the properties of a smooth, interpretable low-dimensional latent representation with a data specific non-Gaussian likelihood function (for silhouette images). The model fully propagates and captures uncertainty in its estimates, it is trained end to end with the same complexity as a standard feed-forward neural network by minimising a single objective function, and is able to learn from very little data as well as scaling to larger datasets linearly by using mini-batching. We have shown both quantitatively and qualitatively that our model performs on par with the best shape models while at the same time introducing a smooth and low-dimensional latent representation with associated uncertainty that facilitates easy synthesis of data.

Acknowledgements This work was supported by the EPSRC CAMERA (EP/M023281/1) grant and the Royal Society.

References

1. Abadi, M., et al.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015), software available from tensorflow.org
2. Bengio, Y., LeCun, Y., et al.: Scaling learning algorithms towards AI. Large-scale kernel machines **34**(5), 1–41 (2007)
3. Campbell, N.D.F., Kautz, J.: Learning a manifold of fonts. ACM Transactions on Graphics (SIGGRAPH) **33**(4), 91 (2014)
4. Carreira-Perpiñán, M.Á., Hinton, G.E.: On Contrastive Divergence Learning. In: Cowell, R.G., Ghahramani, Z. (eds.) Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005. Society for Artificial Intelligence and Statistics (2005)
5. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. pp. 2172–2180 (2016)
6. Damianou, A.C., Lawrence, N.D.: Deep Gaussian Processes. In: Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013. JMLR Workshop and Conference Proceedings, vol. 31, pp. 207–215. JMLR.org (2013)
7. Davies, R., Twining, C., Taylor, C.: Statistical models of shape: Optimisation and evaluation. Springer Science & Business Media (2008)
8. Elhabian, S.Y., Whitaker, R.T.: ShapeOdds: Variational Bayesian Learning of Generative Shape Models. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 2185–2196. IEEE Computer Society (2017)
9. Eslami, S.M.A., Williams, C.K.I.: A Generative Model for Parts-based Object Segmentation. In: Bartlett, P.L., Pereira, F.C.N., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. pp. 100–107 (2012)
10. Eslami, S.A., Heess, N., Williams, C.K., Winn, J.: The shape boltzmann machine: a strong model of object shape. International Journal of Computer Vision **107**(2), 155–176 (2014)
11. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative Adversarial Nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada. pp. 2672–2680 (2014)
12. Hinton, G.E.: A Practical Guide to Training Restricted Boltzmann Machines. In: Montavon, G., Orr, G.B., Müller, K. (eds.) Neural Networks: Tricks of the Trade - Second Edition, Lecture Notes in Computer Science, vol. 7700, pp. 599–619. Springer (2012)
13. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. science **313**(5786), 504–507 (2006)
14. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: International Conference on Learning Representations (ICLR) (2014)

15. Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes. In: International Conference on Learning Representations (ICLR) (2013)
16. Kirillov, A., Gavrikov, M., Lobacheva, E., Osokin, A., Vetrov, D.P.: Deep Part-Based Generative Shape Model with Latent Variables. In: Wilson, R.C., Hancock, E.R., Smith, W.A.P. (eds.) Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016. BMVA Press (2016)
17. Lawrence, N.D.: Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. Journal of Machine Learning Research **6**, 1783–1816 (2005)
18. Lawrence, N.D., Candela, J.Q.: Local distance preservation in the GP-LVM through back constraints. In: Cohen, W.W., Moore, A. (eds.) Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006. ACM International Conference Proceeding Series, vol. 148, pp. 513–520. ACM (2006)
19. Li, F., Fergus, R., Perona, P.: Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2004, Washington, DC, USA, June 27 - July 2, 2004. p. 178. IEEE Computer Society (2004)
20. Li, W., Viola, F., Starck, J., Brostow, G.J., Campbell, N.D.F.: Roto++: Accelerating Professional Rotoscoping using Shape Manifolds. ACM Transactions on Graphics (SIGGRAPH) **35**(4), 62 (2016)
21. Maddison, C.J., Mnih, A., Teh, Y.W.: The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. vol. abs/1611.00712 (2016)
22. Prisacariu, V.A., Reid, I.D.: PWP3D: Real-time segmentation and tracking of 3D objects. International journal of computer vision **98**(3), 335–354 (2012)
23. Prisacariu, V.A., Reid, I.D.: Nonlinear shape manifolds as shape priors in level set segmentation and tracking. In: The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011. pp. 2185–2192. IEEE Computer Society (2011)
24. Roy, A., Todorovic, S.: Combining Bottom-Up, Top-Down, and Smoothness Cues for Weakly Supervised Image Segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 7282–7291. IEEE Computer Society (2017)
25. Smolensky, P.: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1. chap. Information Processing in Dynamical Systems: Foundations of Harmony Theory, pp. 194–281 (1986)
26. Snoek, J., Adams, R.P., Larochelle, H.: Nonparametric guidance of autoencoder representations using label information. Journal of Machine Learning Research **13**, 2567–2588 (2012)
27. Tosi, A., Hauberg, S., Vellido, A., Lawrence, N.D.: Metrics for Probabilistic Geometries. In: Zhang, N.L., Tian, J. (eds.) Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, UAI 2014, Quebec City, Quebec, Canada, July 23-27, 2014. pp. 800–808. AUAI Press (2014)
28. Tsogkas, S., Kokkinos, I., Papandreou, G., Vedaldi, A.: Semantic Part Segmentation with Deep Learning. arXiv preprint **arXiv:1505.02438** (2015)
29. Turmukhambetov, D., Campbell, N.D.F., Goldman, D.B., Kautz, J.: Interactive Sketch-Driven Image Synthesis. Computer Graphics Forum **34**(8), 130–142 (2015)
30. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Processing **13**(4), 600–612 (2004)
31. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. pp. 1912–1920. IEEE Computer Society (2015)