A Generative Deep Recurrent Model for Exchangeable Data

Iryna Korshunova ♥ [1] Jonas Degrave ♥ † [1] Ferenc Huszár [2] Yarin Gal [3] Arthur Gretton ♠ [4] Joni Dambre ♠ [1]

Abstract

We present a novel model architecture which leverages deep learning tools to perform exact Bayesian inference on sets of high dimensional, complex observations. Our model is provably exchangeable, meaning that the joint distribution over observations is invariant under permutation: this property lies at the heart of Bayesian inference. The model does not require variational approximations to train, and new samples can be generated conditional on previous samples, with cost linear in the size of the conditioning set. The advantages of our architecture are demonstrated on learning tasks requiring generalisation from short observed sequences while modelling sequence variability, such as conditional image generation, few-shot learning, set completion, and anomaly detection.

1. Introduction

We address the problem of modelling unordered sets of objects that have some characteristic in common. Set modelling has been a recent focus in machine learning, both due to relevant application domains and to efficiency gains when dealing with groups of objects (Szabo et al., 2016; Zaheer et al., 2017; Edwards & Storkey, 2017). The relevant concept in statistics is the notion of an exchangeable sequence of random variables – a sequence where any re-ordering of the elements is equally likely. To fulfil this definition, subsequent observations must behave like previous ones, which implies that we can make predictions about the future. This property allows the formulation of some machine learning problems in terms of modelling exchangeable data. For instance, one can think of a few-shot concept learning as learning to complete short exchangeable sequences (Lake et al., 2015). A related example comes from a generative image modelling field, where we might want to generate images that are in some ways similar to the ones from a given set. At present, however, there are few flexible and provably exchangeable deep generative models to solve this problem.

♥♠Equal contribution †Now at DeepMind [1]Ghent University, Belgium [2]Twitter [3]University of Oxford, UK [4]Gatsby Computational Neuroscience Unit, University College London, UK. Correspondence to: Iryna Korshunova <iryna.korshunova@ugent.be>.

Formally, a finite or infinite sequence of random variables $x_1, x_2, x_3, \ldots$ is said to be exchangeable if for all n and all permutations π

$$p(x_1, \ldots, x_n) = p(x_{\pi(1)}, \ldots, x_{\pi(n)}), \tag{1}$$

i.e. the joint probability remains the same under any permutation of the sequence. If random variables in the sequence are independent and identically distributed (i.i.d), then it is easy to see that the sequence is exchangeable. The converse is false: exchangeable random variables can be correlated. One example is a sequence of Gaussian random variables $x_1, \ldots, x_n$, which jointly have a multivariate normal distribution $\mathcal{N}_n(\mathbf{0}, \mathbf{\Sigma})$ with the same variance and covariance for all the dimensions (Aldous et al., 1985):

$$\Sigma_{ii} = 1 \text{ and } \Sigma_{ij, i \neq j} = \rho, \text{ with } 0 \leq \rho < 1 \tag{2}$$

The concept of exchangeability is intimately related to Bayesian statistics. De Finetti's theorem states that every exchangeable process (infinite sequence of random variables) is a mixture of i.i.d. processes,

$$p(x_1, \ldots, x_n) = \int p(\theta) \prod_{i=1}^{n} p(x_i|\theta) d\theta, \tag{3}$$

where θ is some parameter vector (finite or infinite dimensional) conditioned on which the random variables are i.i.d (Aldous et al., 1985). In the Gaussian example with covariance defined in Eq. 2, one can prove that $x_1, \ldots, x_n$ are i.i.d. with $x_i \sim \mathcal{N}(\theta, 1-\rho)$ conditioned on $\theta \sim \mathcal{N}(0, \rho)$.

In terms of predictive distributions $p(x_n|x_{1:n-1})$, the stochastic process in Eq. 3 can be written as

$$p(x_n|x_{1:n-1}) = \int p(x_n|\theta) p(\theta|x_{1:n-1}) d\theta, \tag{4}$$

by conditioning both sides on $x_{1:n-1}$. Eq. 4 is exactly the posterior predictive distribution, where we marginalise the likelihood of x_n given θ with respect to the posterior distribution of θ. From this follows one possible interpretation of the de Finetti's theorem: learning to fit an exchangeable model to sequences of data is implicitly the same as learning to reason about the hidden variables behind the data.

One strategy for defining models of exchangeable sequences is through explicit Bayesian modelling: one defines a prior $p(\theta)$, a likelihood $p(x_i|\theta)$ and calculates the posterior in Eq. 3 directly. Here, the key difficulty is the intractability of the posterior and the predictive distribution $p(x_n|x_{1:n-1})$. Both of these expressions require integrating over the parameter θ, so we might end up having to use approximations. This could violate the exchangeability property and make explicit Bayesian modelling difficult.

On the other hand, we do not have to explicitly represent the posterior to ensure exchangeability. One could define a predictive distribution $p(x_n|x_{1:n-1})$ directly, and as long as the process is exchangeable, it is consistent with Bayesian reasoning. The key difficulty here is defining an easy-to-calculate $p(x_n|x_{1:n-1})$ which satisfies exchangeability. For example, it is not clear how to train or modify an ordinary recurrent neural network (RNN) to model exchangeable data. In our opinion, the main challenge is to ensure that a hidden state contains information about all previous inputs $x_{1:n}$ regardless of sequence length.

In this paper, we propose a novel architecture which combines features of both the above approaches, which we will refer to as BRUNO: Bayesian RecUrrent Neural mOdel. Our model is *provably exchangeable*, and makes use of deep features learned from observations so as to model complex data types such as images. To achieve this, we construct a *bijective* mapping between random variables $x_i \in \mathcal{X}$ in the observation space and features $z_i \in \mathcal{Z}$, and explicitly define an exchangeable model for the sequences $z_1, z_2, z_3, \dots,$ where we know an analytic form of $p(z_n|z_{1:n-1})$ without explicitly computing the integral in Eq. 4.

Using BRUNO, we are able to generate samples conditioned on the input sequence by sampling directly from $p(x_n|x_{1:n-1})$. The latter is also tractable to evaluate, i.e has linear complexity in the number of data points. In respect of model training, evaluating the predictive distribution requires a single pass through the neural network that implements $\mathcal{X} \mapsto \mathcal{Z}$ mapping. The model can be learned straightforwardly, since $p(x_n|x_{1:n-1})$ is differentiable with respect to the model parameters.

The paper is structured as follows. In Section 2 we will look at two methods selected to highlight the relation of our work with previous approaches to modelling exchangeable data. Section 3 will describe BRUNO, along with necessary background information. In Section 4, we will use our model for conditional image generation, few-shot learning, set expansion and set anomaly detection. Our code will be made available in a final version of the paper.

2. Related work

Bayesian sets (Ghahramani & Heller, 2006) aim to model exchangeable sequences of binary random variables by analytically computing the integrals in Eq. 3, 4. This is made possible by using a Bernoulli distribution for the likelihood and a beta distribution for the prior. To apply this method to other types of data, e.g. images, one needs to engineer a set of binary features (Heller & Ghahramani, 2006). In that case, there is usually no one-to-one mapping between the input space $\mathcal{X}$ and the features space $\mathcal{Z}$: in consequence, it is not possible to draw samples from $p(x_n|x_{1:n-1})$. Unlike Bayesian sets, our approach does have a bijective transformation, which makes our model generative, and guarantees that inference in $\mathcal{Z}$ is equivalent to inference in space $\mathcal{X}$. In Section 4.3 we will see how our model compares to Bayesian sets for the task of content-based image retrieval.

The Neural Statistician (Edwards & Storkey, 2017) is an extension of a variational autoencoder model (Kingma & Welling, 2014; Rezende et al., 2014) applied to datasets. In addition to learning an approximate inference network over the latent variable $\boldsymbol{z}_i$ for every $\boldsymbol{x}_i$ in the set, approximate inference is also implemented over a latent variable $\boldsymbol{c}$ – a context that is global to the dataset. The architecture for the inference network $q(\boldsymbol{c}|\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ maps every $\boldsymbol{x}_i$ into a feature vector and applies a mean pooling operation across these representations. The resulting vector is then used to produce parameters of a Gaussian distribution over $\boldsymbol{c}$. Mean pooling makes $q(\boldsymbol{c}|\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ invariant under permutations of the inputs. In addition to the inference networks, the neural statistician also has a generative component $p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n|\boldsymbol{c})$ which assumes that $\boldsymbol{x}_i$'s are independent given $\boldsymbol{c}$. Here, it is easy to see that $\boldsymbol{c}$ plays the role of θ from Eq. 3. In the neural statistician, it is intractable to compute $p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$, so its variational lower bound is used instead. In our model, we perform an implicit inference over θ and can exactly compute predictive distributions and the marginal likelihood. Despite these differences, both neural statistician and BRUNO can be applied in similar settings, namely few-shot learning and conditional image generation, albeit with some restrictions, as we will see in Section 4.

3. Method

We begin this section with an overview of the mathematical tools needed to construct our model: first the Student-t process (Shah et al., 2014); and then the Real NVP – a deep, stably invertible and learnable neural network architecture for density estimation (Dinh et al., 2017). We next propose BRUNO, wherein we combine an exchangeable Student-t process with the Real NVP, and derive recurrent equations for the predictive distribution such that our model can be trained as an RNN. Our approach is illustrated in Fig. 1.

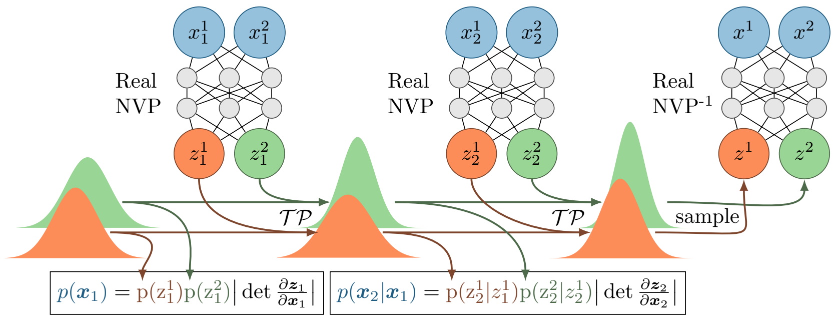
Real NVP
Real NVP
Real NVP-1
x_1^1 x_1^2 z_1^1 z_1^2
x_2^1 x_2^2 z_2^1 z_2^2
x^1 x^2 z^1 z^2
$\mathcal{TP}$
$\mathcal{TP}$
sample
$p(\boldsymbol{x}_1) = \mathrm{p}(z_1^1)\mathrm{p}(z_1^2)\left|\det \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{x}_1}\right|$
$p(\boldsymbol{x}_2|\boldsymbol{x}_1) = \mathrm{p}(z_2^1|z_1^1)\mathrm{p}(z_2^2|z_2^1)\left|\det \frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{x}_2}\right|$

Figure 1. A schematic of the BRUNO model. It depicts how Bayesian thinking can lead to an RNN-like computational graph in which Real NVP is a bijective feature extractor and the recurrence is represented by Bayesian updates of an exchangeable Student-t process.

3.1. Student-t processes

The Student-t process ($\mathcal{TP}$) is the most general elliptically symmetric process with an analytically representable density (Shah et al., 2014). Our choice of $\mathcal{TP}$s in preference to the more commonly used Gaussian processes ($\mathcal{GP}$s) is motivated by some non-trivial differences in their behaviour for the task we set out to solve. In what follows, we provide the background and definition of $\mathcal{TP}$s.

Let us assume that $\boldsymbol{z} = (z_1, \dots z_n) \in \mathbb{R}^n$ follows a multivariate Student-t distribution $MVT_n(\nu, \boldsymbol{\mu}, \boldsymbol{K})$ with degrees of freedom $\nu \in \mathbb{R}_+ \setminus [0, 2]$, mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and a positive definite $n \times n$ covariance matrix $\boldsymbol{K}$. Its density is given by

$$p(\boldsymbol{z}) = \frac{\Gamma(\frac{\nu+n}{2})}{((\nu-2)\pi)^{n/2}\Gamma(\nu/2)}|\boldsymbol{K}|^{-1/2} \times \left(1 + \frac{(\boldsymbol{z}-\boldsymbol{\mu})^T\boldsymbol{K}^{-1}(\boldsymbol{z}-\boldsymbol{\mu})}{\nu-2}\right)^{-\frac{\nu+n}{2}}. \tag{5}$$

For our problem, we are interested in computing a conditional distribution. Suppose we can partition $\boldsymbol{z}$ into two consecutive parts $\boldsymbol{z}_a \in \mathbb{R}^{n_a}$ and $\boldsymbol{z}_b \in \mathbb{R}^{n_b}$, such that

$$\begin{bmatrix}\boldsymbol{z}_a\\ \boldsymbol{z}_b\end{bmatrix} \sim MVT_n\left(\nu, \begin{bmatrix}\boldsymbol{\mu}_a\\ \boldsymbol{\mu}_b\end{bmatrix}, \begin{bmatrix}\boldsymbol{K}_{aa} & \boldsymbol{K}_{ab}\\ \boldsymbol{K}_{ba} & \boldsymbol{K}_{bb}\end{bmatrix}\right). \tag{6}$$

Then conditional distribution $p(\boldsymbol{z}_b|\boldsymbol{z}_a)$ is given by

$$\begin{aligned} p(\boldsymbol{z}_b|\boldsymbol{z}_a) &= MVT_{n_b}\left(\nu + n_a, \tilde{\boldsymbol{\mu}}_{\boldsymbol{b}}, \frac{\nu+\beta_a-2}{\nu+n_a-2}\tilde{\boldsymbol{K}}_{bb}\right),\\ \tilde{\boldsymbol{\mu}}_{\boldsymbol{b}} &= \boldsymbol{K}_{ba}\boldsymbol{K}_{aa}^{-1}(\boldsymbol{z}_a - \boldsymbol{\mu}_a) + \boldsymbol{\mu}_b\\ \beta_a &= (\boldsymbol{z}_a - \boldsymbol{\mu}_a)^T\boldsymbol{K}_{aa}^{-1}(\boldsymbol{z}_a - \boldsymbol{\mu}_a)\\ \tilde{\boldsymbol{K}}_{bb} &= \boldsymbol{K}_{bb} - \boldsymbol{K}_{ba}\boldsymbol{K}_{aa}^{-1}\boldsymbol{K}_{ab}. \end{aligned} \tag{7}$$

In the general case, when one needs to invert the covariance matrix, the complexity of computing $p(\boldsymbol{z}_b|\boldsymbol{z}_a)$ is $\mathcal{O}(n_a^3)$. These computations become infeasible for large datasets, which is a known bottleneck for $\mathcal{GP}$s and $\mathcal{TP}$s (Rasmussen & Williams, 2005). In Section 3.3, we will show that exchangeable processes do not have this issue.

The parameter ν, representing the degrees of freedom, has a large impact on the behaviour of $\mathcal{TP}$s. It controls how heavy-tailed the t-distribution is: as ν increases, the tails get lighter and t-distribution gets closer to the Gaussian. From Eq. 7, we can see that as ν or n_a tends to infinity, the predictive distribution tends to the one from a $\mathcal{GP}$. Thus, for small ν and n_a, a $\mathcal{TP}$ would give less certain predictions than its corresponding $\mathcal{GP}$.

A second feature of the $\mathcal{TP}$ is the scaling of the predictive variance with a β coefficient, which explicitly depends on the values of the conditioning observations. From Eq. 7, the value of β is precisely the Hotelling statistic for the vector $\boldsymbol{z}_a$, and has a $\chi^2_{n_a}$ distribution with mean n_a in the event that $\boldsymbol{z}_a \sim \mathcal{N}_{n_a}(\boldsymbol{\mu}_a, \boldsymbol{K}_{aa})$. Looking at the weight ${}^{(\nu+\beta_a-2)}/_{(\nu+n_a-2)}$, we see that the variance of $p(\boldsymbol{z}_a|\boldsymbol{z}_b)$ is increased over the Gaussian default when $\beta_a > n_a$, and is reduced otherwise. In other words, when the samples are dispersed more than they would be under the Gaussian distribution, the predictive uncertainty is increased compared with the Gaussian case. It is helpful in understanding these two properties to recall that the multivariate Student-t distribution can be thought of as a Gaussian distribution with an inverse Wishart prior on the covariance (Shah et al., 2014). We return to this point in our experiments of Section 4.4, where this prior ensures stability when the covariance parameters of the $\mathcal{TP}$ model are fit from data, as compared with a $\mathcal{GP}$ model.

3.2. Real NVP

Real NVP (Dinh et al., 2017) is a member of the normalising flows family of models, where some density in the input space $\mathcal{X}$ is transformed into a desired probability distribution in space $\mathcal{Z}$ through a sequence of invertible mappings (Rezende & Mohamed, 2015). Specifically, Real NVP proposes a design for a bijective function $f : \mathcal{X} \mapsto \mathcal{Z}$ with $\mathcal{X} = \mathbb{R}^D$ and $\mathcal{Z} = \mathbb{R}^D$ such that **1)** the inverse is easy to evaluate, i.e. the cost of computing $\boldsymbol{x} = f^{-1}(\boldsymbol{z})$ is the same as for the forward mapping and **2)** computing the Jacobian determinant takes linear time in the number of dimensions D. Additionally, Real NVP assumes a simple distribution for $\boldsymbol{z}$, e.g. an isotropic Gaussian, so one can use a change of variables formula to evaluate $p(\boldsymbol{x})$:

$$p(\boldsymbol{x}) = p(\boldsymbol{z}) \left| \det \left(\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \right) \right| . \tag{8}$$

The main building block of Real NVP is a coupling layer. It implements a mapping $\mathcal{X} \mapsto \mathcal{Y}$ that transforms half of its inputs while copying the other half directly to the output:

$$\begin{cases} \boldsymbol{y}^{1:d} = \boldsymbol{x}^{1:d} \\ \boldsymbol{y}^{d+1:D} = \boldsymbol{x}^{d+1:D} \odot \exp(s(\boldsymbol{x}^{1:d})) + t(\boldsymbol{x}^{1:d}), \end{cases} \tag{9}$$

where $\odot$ is an elementwise product, s (scale) and t (translation) are arbitrarily complex functions.

One can show that the coupling layer is a bijective, easily invertible mapping with a triangular Jacobian and composition of such layers preserves these properties. To obtain a highly nonlinear mapping $f(\boldsymbol{x})$, one needs to stack coupling layers $\mathcal{X} \mapsto \mathcal{Y}_1 \mapsto \mathcal{Y}_2 \ldots \mapsto Z$ while alternating the dimensions that are being copied to the output.

To make good use of modelling densities, the Real NVP has to treat its inputs as instances of a continuous random variable (Theis et al., 2016). To do so, integer pixel values in $\boldsymbol{x}$ are dequantised by adding uniform noise $\boldsymbol{u} \in [0,1)^D$. The values $\boldsymbol{x} + \boldsymbol{u} \in [0,256)^D$ are then rescaled to a $[0,1)$ interval and transformed with an elementwise function: $f(x) = \text{logit}(\alpha + (1-2\alpha)x)$ with some small α. The latter ensures that when we take the inverse mapping, which is what we do when generating samples, the outputs always lie within $(\frac{-\alpha}{1-2\alpha}, \frac{1-\alpha}{1-2\alpha})$.

3.3. BRUNO: the exchangeable sequence model

We now combine the Bayesian and deep learning tools from the previous sections and present our model for exchangeable sequences.

Assume we are given an exchangeable sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, where every element is a D-dimensional vector: $\boldsymbol{x}_i = (x_i^1, \ldots x_i^D)$. We apply a Real NVP transformation to every $\boldsymbol{x}_i$, which results in an exchangeable sequence in the latent space: $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$, where $\boldsymbol{z}_i \in \mathbb{R}^D$. The proof that the latter sequence is exchangeable is given in Section A of the appendix.

We make the following assumptions about the latents:

A1: dimensions $\{z^d\}_{d=1,\ldots,D}$ are independent, so $p(\boldsymbol{z}) = \prod_{d=1}^D p(z^d)$

A2: for every dimension d, we have the following assumption:

$$(z_1^d, \ldots z_n^d) \sim MVT_n(\nu^d, \mu^d \mathbf{1}, \boldsymbol{K}^d) \tag{10}$$

with parameters:

- degrees of freedom $\nu^d \in \mathbb{R}_+ \setminus [0,2]$
- mean $\mu^d \mathbf{1}$ is a $1 \times n$ dimensional vector of ones multiplied by the scalar $\mu^d \in \mathbb{R}$
- $n \times n$ covariance matrix $\boldsymbol{K}^d$:

$$\boldsymbol{K}_{ij}^d = \begin{cases} v^d & i = j \\ \rho^d & i \neq j \end{cases} \tag{11}$$

  with $0 \leq \rho^d < v^d$ to make sure that $\boldsymbol{K}^d$ is a positive-definite matrix that complies with covariance properties of exchangeable sequences (Aldous et al., 1985).

The structure of the covariance matrix in Eq. 11 and having the same mean for every n, guarantees that the sequence $z_1^d, z_2^d \ldots z_n^d$ is exchangeable.

Because the covariance matrix is simple, we can derive recurrent updates for parameters of $p(z_{n+1}^d | z_{1:n}^d)$. Using the recurrence is a lot more efficient compared to the closed-form expressions in Eq. 7 since we want to compute the predictive distribution for every step n.

We start from a prior Student-t distribution for $p(z_1)$ with parameters $\mu_1 = \mu$, $v_1 = v$, $\nu_1 = \nu$, $\beta_1 = 0$. Here, we will drop the dimension index d to simplify the notation. A detailed derivation of the following results is given in Section B of the appendix.

To compute the degrees of freedom, mean and variance of $p(z_{n+1}|z_{1:n})$ for every n, we begin with the recurrent relations

$$\begin{aligned} \nu_{n+1} &= \nu_n + 1 \\ \mu_{n+1} &= (1-d_n)\mu_n + d_n z_n \\ v_{n+1} &= (1-d_n)v_n + d_n(v-\rho), \end{aligned} \tag{12}$$

where $d_n = \frac{\rho}{v + \rho(n-1)}$. Note that the $\mathcal{GP}$ recursions simply use the latter two equations, i.e. if we were to assume that $(z_1^d, \ldots z_n^d) \sim \mathcal{N}_n(\mu^d \mathbf{1}, \boldsymbol{K}^d)$.

For $\mathcal{TP}$s, however, we also need to compute β – a data-dependent term that scales the covariance matrix as in Eq. 7. To update β, we introduce recurrent expressions for the auxiliary variables,

$$\begin{aligned} \tilde{z}_i &= z_i - \mu \\ a_n &= \frac{v + \rho(n-2)}{(v-\rho)(v+\rho(n-1))} \\ b_n &= \frac{-\rho}{(v-\rho)(v+\rho(n-1))} \\ \beta_{n+1} &= \beta_n + (a_n - b_n)\tilde{z}_n^2 + b_n(\sum_{i=1}^n \tilde{z}_i)^2 - b_{n-1}(\sum_{i=1}^{n-1} \tilde{z}_i)^2 \end{aligned}$$

Such recurrent updates are not the only advantage of an exchangeable covariance structure. Another valuable property is that the computational complexity of making predictions in exchangeable $\mathcal{GP}$s or $\mathcal{TP}$s scales linearly with the number of observations, i.e. $\mathcal{O}(n)$ instead of a general $\mathcal{O}(n^3)$ case where one needs to compute an inverse covariance matrix.

So far, we have constructed an exchangeable Student-t process in the latent space $\mathcal{Z}$. By coupling it with a bijective Real NVP mapping, we get an exchangeable process in space $\mathcal{X}$. Although we do not have an explicit analytic form of the transitions in $\mathcal{X}$, we still can sample from this process and evaluate the predictive distribution via the change of variables formula in Eq. 8.

3.4. Training

Having an easy-to-evaluate autoregressive distribution $p(\boldsymbol{x}_{n+1}|\boldsymbol{x}_{1:n})$ allows us to use a training scheme that is common for RNNs, i.e. maximise the likelihood of the next element in the sequence at every step. Thus, our objective function for sequences of a fixed length N can be written as $\sum_{n=0}^{N-1} \log p(\boldsymbol{x}_{n+1}|\boldsymbol{x}_{1:n})$, which is equivalent to maximising $\log p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)$. By using backpropagation through time, we update the parameters of the Real NVP model and also learn the parameters of the prior Student-t distribution. For the latter, we have 4 trainable parameters per dimension: degrees of freedom ν^d, mean μ^d, variance v^d and covariance ρ^d, which must satisfy the constraints from Eq. 10.

4. Experiments

In this section, we will consider a few problems that fit naturally into the framework of modelling exchangeable data. We chose to work with sequences of images, so the results are easy to analyse; yet our model does not make any image-specific assumptions, and our conclusions generalise to other types of data. Specifically, we used a general-purpose Real NVP coupling layer as proposed by Papamakarios et al. (2017). In contrast to the original Real NVP model, which uses convolutional architecture for scaling and translation functions in Eq. 9, a general implementation has s and t composed from fully connected layers. More details on the architecture and its training are given in Section C of the appendix.

In all our experiments, the models are trained on about 13M MNIST (LeCun et al., 1998) image sequences of length 32. We form each sequence by uniformly sampling a digit and then selecting 32 random images of that digit. This scheme implies that a model is trained to implicitly infer a class label that is global to a sequence. In what follows, we will see how this property can be used in many tasks.

4.1. Conditional image generation

We first consider a problem of generating samples conditionally on a set of images, which reduces to sampling from a predictive distribution $p(\boldsymbol{x}_{n+1}|\boldsymbol{x}_{1:n})$. This is different from a general Bayesian approach, where one needs to infer the posterior over some meaningful latent variable and then 'decode' it.

To draw samples from $p(\boldsymbol{x}_{n+1}|\boldsymbol{x}_{1:n})$, we first need to sample $\boldsymbol{z} \sim p(\boldsymbol{z}_{n+1}|\boldsymbol{z}_{1:n})$ and then compute the inverse Real NVP mapping: $\boldsymbol{x} = f^{-1}(\boldsymbol{z})$. Since we assumed that dimensions of $\boldsymbol{z}$ are independent, we can sample each z^d from a univariate Student-t distribution with parameters ν^d, μ^d and v^d. To do so, we modified Bailey's polar t-distribution generation method (Bailey, 1994) to be computationally efficient for GPU. Its algorithm is given in the appendix.

In Figure 2A, we show samples from the prior distribution $p(\boldsymbol{x}_1)$ and conditional samples from a predictive distribution $p(\boldsymbol{x}_{n+1}|\boldsymbol{x}_{1:n})$ at steps $n = 1, \dots, 15$. The input sequence is constructed from MNIST test images that were not used during training. More examples with longer sequences are shown in the appendix.

To better understand how BRUNO behaves, we test it on special types of input sequences that were not seen during training. For example, Figure 2B illustrates the case when the same image is used throughout the sequence (correlations between digits stronger than expected from training). Here, the recursive $\mathcal{TP}$ updates cause the variability of the samples to reduce as the models gets more inputs. This property does not hold for the neural statistician model (Edwards & Storkey, 2017), discussed in Section 2. As mentioned earlier, the neural statistician computes the approximate posterior $q(\boldsymbol{c}|\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ and then uses its mean to sample $\boldsymbol{x}$ from a conditional model $p(\boldsymbol{x}|\boldsymbol{c}_{mean})$. This scheme does not account for the variability in the inputs as a consequence of applying mean pooling over the features of $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ when computing $q(\boldsymbol{c}|\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$. Thus, when all x_i's are the same, the neural statistician would still sample different instances from the class specified by x_i.

Another example which cannot be handled by models that enforce exchangeability by using pooling operations is given in Figure 2C. Here, the input images come from two classes, and despite the model not being trained to deal with this case, it generates images of both digits.

To see if the model can generalise to more difficult types of images, we trained it on Fashion MNIST (Xiao et al., 2017) – a direct drop-in replacement for MNIST. Samples from this model are given in Fig. 3.

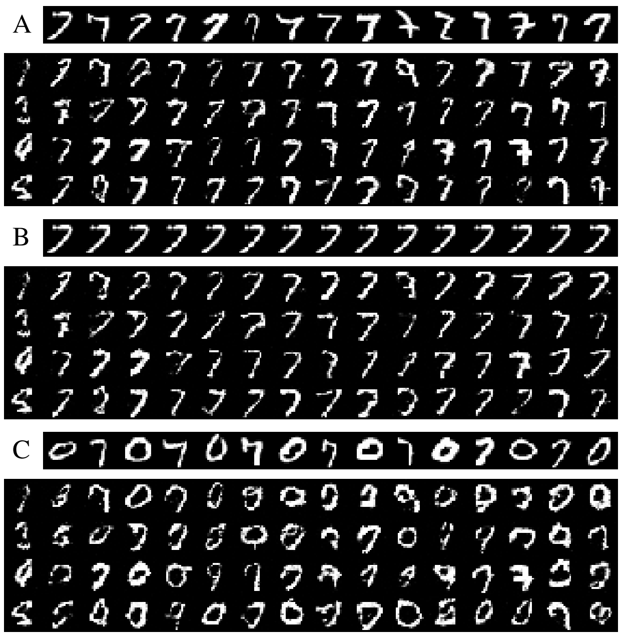

Figure 2. Each subplot shows an input sequence in the top row and samples in the bottom 4 rows. Every column of the subplots contains 4 samples from the predictive distribution conditioned on the input images up to and including that column. That is, the 1st column shows samples from the prior when no input image is given; the 2nd column shows samples conditioned on the 1st input image; the 3rd column shows samples conditioned on the first two images, etc. *Top (A)*: input sequence contains test images of one digit (the same setting as during training). *Middle (B)*: the same image is used as an input at every step. We see that samples have low variability in their appearance. *Bottom (C)*: input images come from two classes. This is reflected in the samples, though the model was not trained on such sequences. Note that our model uses fully-connected layers, so the quality of its samples is lower than for convolutional models.

4.2. Few-shot learning

Once we have conditional probabilities from BRUNO, we can classify examples based on their similarity to sequences of images coming from certain classes. This setup is particularly interesting when the model did not see those classes during training. When each sequence contains only a few elements, this is equivalent to a few-shot learning problem. We will address one-shot and few-shot learning tasks in this section after we provide some important details on how we evaluate the predictive probabilities.

When we care exclusively about comparing conditional densities of $\boldsymbol{x}_{n+1}$ under different sequences $\boldsymbol{x}_{1:n}$, we can compare densities in the latent space $\mathcal{Z}$ instead. This is because the Jacobian from the change of variable formula in Eq. 8 does not depend on the sequence we condition on. Also, evaluating predictive densities for all D dimensions can be redundant because for most of them, the predictive

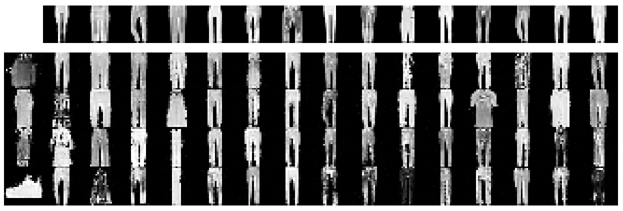

Figure 3. Similarly to Fig. 2, we show conditional samples and samples from the prior in the first column. The model was trained on Fashion MNIST sequences.

distribution is not notably different from the prior. We can find the irrelevant dimensions by inspecting the correlations ρ^d/v^d and discarding the dimensions where the correlations are negligibly small. This both helps to reduce the noise in the predictions and speed up the computations. Henceforth, we will assume that

$$p(\boldsymbol{z}|\boldsymbol{z}_{1:n}) = \prod_{\substack{d=1, \\ \rho^d/v^d > \epsilon}}^{D} p(z^d|z^d_{1:n}), \tag{13}$$

In our experiments, we used $\epsilon = 0.01$, which leaves us with 14 instead of MNIST's 784 dimensions. Section E of the appendix includes a plot of ϵ versus the number of the remaining dimensions from which we can see that any ϵ within $[0.005, 0.1]$ interval is a good cut-off point between retaining all of the dimensions and having around a dozen of the most correlated ones.

For the following experiment, we trained the model only on even MNIST digits (30 508 training images).[1] We then test its ability to classify the unseen odd digits. We construct sequences of n images of odd digits from the train subset of MNIST and perform a 5-way classification on 5074 images of odd digits from the test set. In other words, for every test image $\boldsymbol{x}$, which represents an odd digit, we compute $p(\boldsymbol{z}|\boldsymbol{z}_{1:n}^{C=i})$ for each class $i = \{1, 3, 5, 7, 9\}$. Taking a maximum over i will give us a class label of $\boldsymbol{x}$. We repeat this process multiple trials, where each time we select different conditioning images. In Table 1, we report an average accuracy over 100 trials for different n (number of shots).

With a non-convolutional Real NVP architecture, we are not able to directly compare our performance to the state-of-the-art methods, which is usually done on the OMNIGLOT dataset (Lake et al., 2015). Instead, we considered a k-nearest neighbours (k-NN) baseline where matching is done on the features from a pretrained classifier. This simple strategy is known to achieve excellent results across many tasks, and for a few-shot learning problem it can be close to the state-of-the-art performance especially for a 5-shot

[1]The original model was overfitting when training on half of the training data, so we reduced the model size by a half, i.e. 512 units in every dense layer instead on 1024.

Table 1. Classification accuracy for a few-shot learning task with models trained on even digits and tested on odd. The k-NN method uses features from a pretrained classifier.

METHOD	1-SHOT	5-SHOT
K-NN (features)	0.607	0.779
BRUNO	0.606	0.788

case (Vinyals et al., 2016). Our classifier was an MLP with 2 layers of 256 hidden units plus a 5-way softmax for the last layer. We did not use convolutions in order to make it comparable to our Real NVP implementation with densely connected layers. We trained the classifier on even digits and then used features from its pre-softmax layer for nearest neighbour matching with $k = 1$ and Euclidean distance. This experiment demonstrates that BRUNO performs well in one- and few-shot learning tasks.

We next address the problem of generation unseen images: this is much harder, as we can see from Fig. 4. Taking into account that our model is not convolutional and it lacks knowledge of image priors, we believe these samples are reasonable.

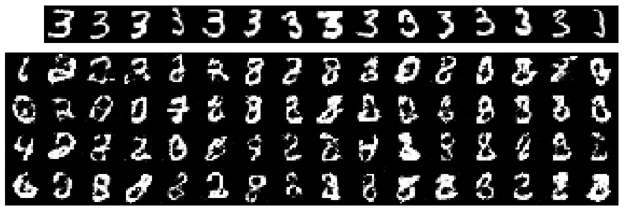

Figure 4. As in Fig. 2, we show conditional samples. The model was trained on even digits, yet it can tolerably generate digit ‘3’.

4.3. Clustering on demand and set anomaly detection

Similar to Bayesian Sets (Ghahramani & Heller, 2006), we consider a problem of clustering on demand: completing a cluster conditioning on the query set containing some elements of that cluster. When dealing with images, this can be seen as a particular case of the Content-Based Image Retrieval task (Heller & Ghahramani, 2006).

To rank a data point $\boldsymbol{x}$ on how well it fits the query set $\boldsymbol{x}_{1:n}$, Bayesian sets use the probabilistic score:

$$\text{score}(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{x}_{1:n})}{p(\boldsymbol{x})}. \tag{14}$$

In BRUNO, we can evaluate the above score as a ratio of the corresponding densities in the latent space, since the Jacobian cancels out when the change of variables formula is used in the numerator and the denominator. As previously, we will use Eq. 13 to compute $p(\boldsymbol{z}|\boldsymbol{z}_{1:n})$ and similarly $p(\boldsymbol{z})$.

Bayesian sets are designed to operate on binary feature vectors, so Heller and Ghahramani (2006) proposed a way to binarise handcrafted texture and colour features in an informative way. In our reimplementation we binarised features from a pretrained classifier which we used in the previous section. As before, all the models were trained on images of even digits. We form queries from images of odd digits coming from the training subset of MNIST and rank images of odd digits from the MNIST test set.

Table 2. Label ranking average precision scores for the task of content-based image retrieval. Models were trained on even MNIST digits and tested on odd. Both nearest neighbours and Bayesian sets use features from a pretrained classifier.

METHOD	$n=1$	$n=5$
NEAREST NEIGHBOURS (features)	0.604	0.714
BAYESIAN SETS (binarised features)	0.606	0.819
BRUNO	0.593	0.768

We evaluate the performance using a label ranking average precision (LRAP) score, which is higher when we give better ranks to the images with the same class label as the ones in the query set (Pedregosa et al., 2011). In Table 2 we report a mean LRAP score averaged over 20 trials for each query set of n images from a certain odd digit. Here, we also make a comparison with a nearest neighbours analogy, where test images are ranked according to their Euclidean distance to the query image in the feature space of the pretrained classifier. When the query contains multiple images, we will use a minimum over distances to all of the query images. Note that the rank equals the inverse of this distance.

We find that our method performs on par with the two other algorithms. It requires more computation than Bayesian sets, but less than the distance ranking method. The performance of these algorithms greatly depends on the number of images in the query set. In the most common use case of a single image query, the results closely match those of the two other algorithms. For $n = 5$, we outperform nearest neighbours, but do less well than Bayesian sets.

Online anomaly detection for exchangeable data is another application where we can use Bayesian scoring. In Fig. 5, we give a typical example of how the score evolves as the model gets more data points and how it behaves in the presence of inputs that do not conform with the majority of the sequence. We observe that our model can detect anomalies in a stream of incoming data.

4.4. Failure of $\mathcal{GP}$-based models

To compare the behaviour of $\mathcal{TP}$s versus $\mathcal{GP}$s, we trained a model where Student-t predictive probabilities were replaced by those from a Gaussian process. The samples from this model are shown in Fig. 6. Notice that prior samples are nonsensical. This is explained by the mismatch between

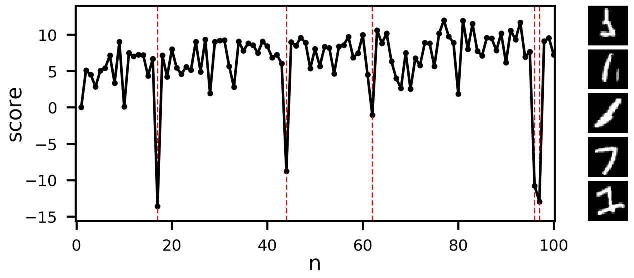

Figure 5. Evolution of the score as the model sees more images of digit '1'. The obvious outliers at steps marked with vertical lines are plotted on the right in order from top to bottom. Note that the model was trained only on images of even digits.

the prior distribution we sample from and the actual distribution for some of the latent dimensions, as shown in Fig. 7 (right). In about a third of cases, the learned means μ^d are far from the actual feature means of the Real NVP mapping. We found that for these dimensions, parameter ρ^d (the covariance between z_i^d and z_j^d for $\forall i, j$) is close to the variance v^d. Referring back to Eq. 12, the updates at $n = 1$ are: $\mu_{n+1}^d \approx z_n^d$ and $v_{n+1}^d \approx 0$. In other words, the predictive variance collapses, and the mean takes the value of the first observation z_1^d. In subsequent iterations, the predictive mean becomes a running average of the inputs, and the variance remains negligible. As a result, prior parameters μ^d and v^d are ignored during model training and can be arbitrarily far from the parameters of the actual distribution.

The better performance of the $\mathcal{TP}$ can be understood by recalling its interpretation as a $\mathcal{GP}$ with an inverse Wishart covariance prior (Shah et al., 2014), which corrects this pathology of the $\mathcal{GP}$ maximum likelihood solution. In practical terms, and referring back to Eq. 7, the number of degrees of freedom of the t-distribution is lower when there are fewer conditioning samples, and the predictive variance is greater when the conditioning samples are more broadly dispersed than expected under the prior Gaussian distribution. This means that initial inputs in a sequence cannot dominate the subsequent predictions, which in turn forces the prior means μ^d to carry information on the distribution. Typical behaviour for the $\mathcal{TP}$ is shown in Fig. 7 (left): in particular, we never encounter a severe mismatch between the assumed and the actual distributions.

5. Discussion and conclusion

In this paper, we have introduced BRUNO, a new technique combining deep learning and Student-t processes for modelling exchangeable data. With this architecture, we may carry out implicit inference, which avoids the need to compute explicit posteriors and reduces the high computational cost often associated with explicit Bayesian inference. Based on our experiments, BRUNO shows promise

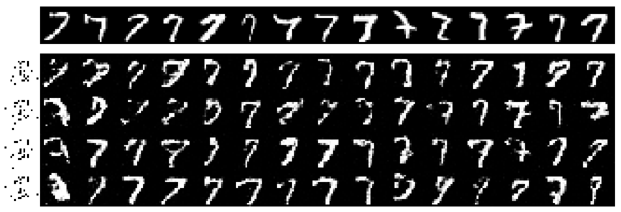

Figure 6. As in Fig. 2, we show samples from the prior and conditional distributions. Here, instead of a $\mathcal{TP}$ process, we used a $\mathcal{GP}$. While conditional samples are on par to those from Fig. 2A, samples from the prior do not resemble MNIST images.

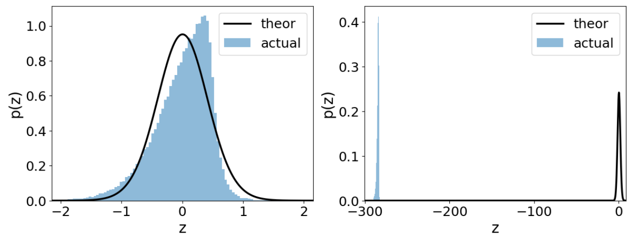

Figure 7. Both plots show the assumed prior density $p(z^d)$ and the histogram of the actual distribution of the d-th latent variable. The latter was obtained by encoding all training MNIST images with Real NVP. *On the left*: typical example of the prior distributions from a $\mathcal{TP}$-based model. *On the right*: an example from a $\mathcal{GP}$-based model, where about a third of the latent dimensions exhibit similar behaviour. When using $\mathcal{TP}$s, we found no such cases.

for applications such as conditional image generation, few-shot learning, content-based image retrieval and online set anomaly detection. This combination of deep learning tools with practical, exact Bayesian inference will be useful for future research in this field, leading to scalable and more data efficient models.

A limiting factor when modelling image data was the use of densely connected layers inside the Real NVP architecture. Convolutional layers might improve the sample quality and performance on images. By not using image-specific assumptions, however, our model can be used with other data modalities as well.

Our current method requires a supervised training phase in order to discover how images within the sequences are correlated. Specifically, we construct the training data such that parameter θ from Eq. 3 has a certain meaning. In future work, we propose to extend our method such that we update the covariance parameters ρ over the course of a sequence. This way, we are not limited to discovering correlations that the model has seen during the training phase. We further propose to define an exchangeable model able to handle two sequences of inputs where each x_i is coupled with y_i. For example, in case of supervised learning, pairs can be formed by the image and its label. It would be interesting to construct an exchangeable process where one can efficiently get predictive probabilities $p(y_{n+1}|x_{n+1}, x_{1:n}, y_{1:n})$.

Acknowledgements

We would like to thank Lucas Theis for suggesting to use Real NVP and many other things we have learned from him, Conrado Miranda and Frederic Godin for their comments on the paper, Wittawat Jitkrittum for useful discussions and Lionel Pigou for setting up the hardware.

References

Aldous, D.J., Hennequin, P.L., Ibragimov, I.A., and Jacod, J. *Ecole d'Ete de Probabilites de Saint-Flour XIII, 1983*. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 1985. ISBN 9783540152033.

Bailey, R. W. Polar generation of random variates with the t-distribution. *Math. Comp.*, 62(206):779–781, 1994. ISSN 0025-5718.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

Edwards, H. and Storkey, A. Towards a neural statistician. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

Ghahramani, Z. and Heller, K. A. Bayesian sets. In Weiss, Y., Schölkopf, B., and Platt, J. C. (eds.), *Advances in Neural Information Processing Systems 18*, pp. 435–442. MIT Press, 2006.

Heller, K. A. and Ghahramani, Z. A simple bayesian framework for content-based image retrieval. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2110–2117, 2006.

Kingma, D. P and Welling, M. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 2015.

LeCun, Y., Cortes, C., and Burges, C. JC. The MNIST database of handwritten digits, 1998.

Papamakarios, G., Murray, I., and Pavlakou, T. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems 30*, pp. 2335–2344. 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1530–1538, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 1278–1286, 2014.

Shah, A., Wilson, A. G., and Ghahramani, Z. Student-t processes as alternatives to gaussian processes. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, pp. 877–885, 2014.

Szabo, Z., Sriperumbudur, B., Poczos, B., and Gretton, A. Learning theory for distribution regression. *Journal of Machine Learning Research*, 17(152), 2016.

Theis, L., van den Oord, A., and Bethge, M. A note on the evaluation of generative models. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29*, pp. 3630–3638. 2016.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint*, abs/1708.07747, 2017.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in Neural Information Processing Systems 30*, pp. 3394–3404. 2017.

Supplementary Material

A. Proofs

Lemma 1

Given two exchangeable sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ of random variables, where x_i is independent from y_j for $\forall i, j$, the concatenated sequence $\boldsymbol{x}^\frown \boldsymbol{y} = ((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n))$ is exchangeable as well.

Proof. For any permutation π, as both sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ are exchangeable we have:

$$\begin{aligned}&p(x_1, x_2, \ldots, x_n)p(y_1, y_2, \ldots, y_n)\\&= p(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})p(y_{\pi(1)}, y_{\pi(2)}, \ldots, y_{\pi(n)}),\end{aligned}$$

Independence between elements in $\boldsymbol{x}$ and $\boldsymbol{y}$ allows to write it as a joint distribution:

$$\begin{aligned}&p((x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n))\\&= p((x_{\pi(1)}, y_{\pi(1)}), (x_{\pi(2)}, y_{\pi(2)}), \ldots, (x_{\pi(n)}, y_{\pi(n)})),\end{aligned}$$

and thus the sequence $\boldsymbol{x}^\frown \boldsymbol{y}$ is exchangeable.

Lemma 2

Given an exchangeable sequence $(x_1, x_2, \ldots, x_n)$ of random variables $x_i \in \mathcal{X}$ and a bijective mapping $f : \mathcal{X} \mapsto \mathcal{Z}$, the sequence $(f(x_1), f(x_2), \ldots, f(x_n))$ is exchangeable.

Proof. Consider a vector function $\boldsymbol{g} : \mathbb{R}^n \mapsto \mathbb{R}^n$ such that $(x_1, \ldots, x_n) \mapsto (z_1 = f(x_1), \ldots, z_n = f(x_n))$. A change of variable formula gives:

$$p(x_1, x_2, \ldots, x_n) = p(z_1, z_2, \ldots, z_n) \left|\det \boldsymbol{J}\right|,$$

where $\det \boldsymbol{J} = \prod_{i=1}^n \frac{\partial f(x_i)}{\partial x_i}$ is the determinant of the Jacobian of $\boldsymbol{g}$. Since both the joint probability of $(x_1, x_2, \ldots, x_n)$ and the $|\det \boldsymbol{J}|$ are invariant to the permutation of sequence entries, so must be $p(z_1, z_2, \ldots, z_n)$. This proves that $(z_1, z_2, \ldots, z_n)$ is exchangeable.

B. Derivation of recurrent Bayesian updates for exchangeable Student-t processes

We assume that $\boldsymbol{x} = (x_1, x_2, \ldots x_n) \in \mathbb{R}^n$ follows a multivariate Student-t distribution $MVT_n(\nu, \boldsymbol{\mu}, \boldsymbol{K})$ with degrees of freedom $\nu \in \mathbb{R}_+ \setminus [0, 2]$, mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and a positive definite $n \times n$ covariance matrix $\boldsymbol{K}$. Its density is given by:

$$\begin{aligned}p(\boldsymbol{x}) = &\frac{\Gamma(\frac{\nu+n}{2})}{((\nu-2)\pi)^{n/2}\Gamma(\nu/2)}|\boldsymbol{K}|^{-1/2}\\&\times (1 + \frac{(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{K}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{\nu-2})^{-\frac{\nu+n}{2}}\end{aligned} \tag{15}$$

Note that this parameterization of the multivariate Student-t distribution as defined by Shah et al. (2014) is slightly different from the commonly used one. We used this parametrization as it makes the formulas easier.

If we partition $\boldsymbol{x}$ into two consecutive parts $\boldsymbol{x}_a \in \mathbb{R}^{n_a}$ and $\boldsymbol{x}_b \in \mathbb{R}^{n_b}$:

$$\begin{bmatrix}\boldsymbol{x}_a\\\boldsymbol{x}_b\end{bmatrix} \sim MVT_n\left(\nu, \begin{bmatrix}\boldsymbol{\mu}_a\\\boldsymbol{\mu}_b\end{bmatrix}, \begin{bmatrix}\boldsymbol{K}_{aa} & \boldsymbol{K}_{ab}\\\boldsymbol{K}_{ba} & \boldsymbol{K}_{bb}\end{bmatrix}\right)$$

the conditional distribution $p(\boldsymbol{x}_b|\boldsymbol{x}_a)$ is given by:

$$p(\boldsymbol{x}_b|\boldsymbol{x}_a) = MVT_{n_b}(\nu + n_a, \tilde{\boldsymbol{\mu}}_{\boldsymbol{b}}, \frac{\nu + \beta_a - 2}{\nu + n_a - 2}\tilde{\boldsymbol{K}}_{bb}),$$

where

$$\begin{aligned}\tilde{\boldsymbol{\mu}}_{\boldsymbol{b}} &= \boldsymbol{K}_{ba}\boldsymbol{K}_{aa}^{-1}(\boldsymbol{x}_a - \boldsymbol{\mu}_a) + \boldsymbol{\mu}_b\\\beta_a &= (\boldsymbol{x}_a - \boldsymbol{\mu}_a)^T\boldsymbol{K}_{aa}^{-1}(\boldsymbol{x}_a - \boldsymbol{\mu}_a)\\\tilde{\boldsymbol{K}}_{bb} &= \boldsymbol{K}_{bb} - \boldsymbol{K}_{ba}\boldsymbol{K}_{aa}^{-1}\boldsymbol{K}_{ab}\end{aligned}$$

Derivation of this result is given in the appendix of Shah et al. (2014). Let us now simplify these equations for the case of exchangeable sequences with the following covariance structure:

$$\boldsymbol{K} = \begin{pmatrix}v & \rho & \cdots & \rho\\\rho & v & \cdots & \rho\\\vdots & \vdots & \ddots & \vdots\\\rho & \rho & \cdots & v\end{pmatrix}$$

In our problem, we are interested in doing one-step predictions, i.e. computing a univariate density $p(x_{n+1}|x_{1:n})$ with parameters ν_{n+1}, μ_{n+1}, v_{n+1}. Therefore, we can take: $\boldsymbol{x} \in \mathbb{R}^{n+1}$, $n_b = 1$ and $n_a = n$. In this case, $\boldsymbol{K}_{aa} = \boldsymbol{K}_{1:n,1:n}$, $\boldsymbol{K}_{ab} = \boldsymbol{K}_{1:n,n+1}$, $\boldsymbol{K}_{ba} = \boldsymbol{K}_{n+1,1:n}$ and $\boldsymbol{K}_{bb} = \boldsymbol{K}_{n+1,n+1} = v$

Computing the parameters of the predictive distribution requires the inverse of $\boldsymbol{K}_{aa}$, which we can find using the Sherman-Morrison formula:

$$\boldsymbol{K}_{aa}^{-1} = (\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T)^{-1} = \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}}$$

with

$$\boldsymbol{A} = \begin{pmatrix} v-\rho & 0 & \cdots & 0 \\ 0 & v-\rho & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & v-\rho \end{pmatrix},$$

$$\boldsymbol{u} = \begin{pmatrix} \rho \\ \rho \\ \vdots \\ \rho \end{pmatrix}, \boldsymbol{v} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

After a few steps, the inverse of $\boldsymbol{K}_{aa}$ is:

$$\boldsymbol{K}_{aa}^{-1} = \begin{pmatrix} a_n & b_n & \cdots & b_n \\ b_n & a_n & \cdots & b_n \\ \vdots & \vdots & \ddots & \vdots \\ b_n & b_n & \cdots & a_n \end{pmatrix}$$

with

$$a_n = \frac{v + \rho(n-2)}{(v-\rho)(v+\rho(n-1))}$$
$$b_n = \frac{-\rho}{(v-\rho)(v+\rho(n-1))}$$

Note that entries of $\boldsymbol{K}_{aa}^{-1}$ explicitly depend on n.

Equations for the mean and variance of the predictive distribution require the following term:

$$\boldsymbol{K}_{ba}\boldsymbol{K}_{aa}^{-1} = \begin{pmatrix} \rho & \rho & \cdots & \rho \end{pmatrix} \boldsymbol{K}_{aa}^{-1} = \left\{ \frac{\rho}{v + \rho(n-1)} \right\}_{1:n},$$

which is a $1 \times n$ vector.

With this in mind, it is easy to derive the following recurrence:

$$d_n = \frac{\rho}{v + \rho(n-1)}$$

$$\mu_{n+1} = (1 - d_n)\mu_n + d_n x_n$$

$$v_{n+1} = (1 - d_n)v_n + d_n(\rho - v)$$

Finally, let us derive recurrent equations for $\beta_{n+1} = (\boldsymbol{x}_a - \boldsymbol{\mu}_a)^T K_{aa}^{-1}(\boldsymbol{x}_a - \boldsymbol{\mu}_a)$

Let $\tilde{\boldsymbol{x}} = \boldsymbol{x}_a - \boldsymbol{\mu}_a$, then:

$$\begin{aligned}\beta_{n+1} &= \tilde{\boldsymbol{x}}^T K_{aa}^{-1} \tilde{\boldsymbol{x}} \\ &= (a_n\tilde{x}_1 + b_n\sum_{i\neq 1}^{n} \tilde{x}_i, a_n\tilde{x}_2 \\ &+ b_n \sum_{i\neq 2}^{n} \tilde{x}_i, \ldots, a_n\tilde{x}_n + b_n \sum_{i\neq n} \tilde{x}_i)^T(\tilde{x}_1, \tilde{x}_2, \ldots \tilde{x}_n) \\ &= (a_n - b_n)\sum_{i=1}^{n} \tilde{x}_i^2 + b_n(\sum_{i=1}^{n} \tilde{x}_i)^2\end{aligned}$$

Similarly, β_n from $p(x_n|x_{1:n-1})$ is:

$$\beta_n = (a_{n-1} - b_{n-1})\sum_{i=1}^{n-1} \tilde{x}_i^2 + b_{n-1}(\sum_{i=1}^{n-1} \tilde{x}_i)^2$$

$$\begin{aligned}\beta_{n+1} &= (a_n - b_n)(\sum_{i=1}^{n-1} \tilde{x}_i^2 + \tilde{x}_n^2) + b_n(\sum_{i=1}^{n} \tilde{x}_i)^2 \\ &= (a_n - b_n)\frac{\beta_n - b_{n-1}(\sum_{i=1}^{n-1}\tilde{x}_i)^2}{a_{n-1} - b_{n-1}} \\ &+ (a_n - b_n)\tilde{x}_n^2 + b_n(\sum_{i=1}^{n} \tilde{x}_i)^2\end{aligned}$$

It is easy to show that $\frac{a_n - b_n}{a_{n-1} - b_{n-1}} = 1$, so β_{n+1} can be written recursively as:

$$s_{n+1} = s_n + \tilde{x}_n$$
$$\beta_{n+1} = \beta_n + (a_n - b_n)\tilde{x}_n^2 + b_n(s_{n+1}^2 - s_n^2)$$

with $s_0 = 0$.

C. Implementation details

We used a general implementation of the Real NVP coupling layer similarly to Papamakarios et al. (2017). Namely, where scaling and translation functions s and t from are fully-connected neural networks. The networks s and t share the parameters in the first two dense layers with 1024 hidden units and ELU nonlinearity (Clevert et al., 2016)(we found however, that models with ReLU are easier to train, but those perform a bit worse in terms of the likelihood). Their output layers are different: s ends with a dense layer with $\tanh$ and t ends with a dense layer without a nonlinearity. We stacked 6 coupling layers with alternating the the indices of the transformed dimensions between odd and even as described by Dinh et al. (2014). For the first layer, which implements a logit transformation of the inputs, we used $\alpha = 10^{-6}$.

The models are trained using Adam (Kingma & Ba, 2015) with an intial learning rate of 10^{-3}, which we halve every

25 000 iterations. Initial parameters of the Student-t distribution are updated with a 10 times smaller learning rate. In total, we trained for 200 000 iterations using a minibatch size of 64 and a sequence length of 32. We made sure that models do not overfit by tracking a validation loss on a set of 10 000 examples when training on MNIST.

We noticed that training models on longer sequences results in a better test performance. Our default sequence length was 32. Having the same models trained on sequences of 8 images, but with batch size increased 4 times, the results were worse. Namely, the model trained on 32-image sequences yields an average negative log-likelihood of 862 nats when tested with a sequence length of 8, while the model trained with sequence length of 8, has only 983 nats. Both models were evaluated on a set of 10 000 sequences of MNIST test images. This can be due to intial predictions being fairly uncertain, so that the learning signal from the beginning of the sequence is less strong. This also holds for classical RNNs, where the predictions in the beginning are less accurate due to the lack of context.

D. Sampling from a Student-t distribution

Algorithm 1 Efficient sampling on GPU from a univariate Student-t distribution with mean μ, variance v and degrees of freedom ν

```
function sample($\mu, v, \nu$)
    $a, b \leftarrow \mathcal{U}(0, 1)$
    $c \leftarrow \min(a, b)$
    $r \leftarrow \max(a, b)$
    $\alpha \leftarrow \frac{2\pi c}{r}$
    $t \leftarrow \cos(\alpha)\sqrt{(\nu/r^2)(r^{-4/\nu} - 1)}$
    $\sigma \leftarrow \sqrt{v\left(\frac{\nu-2}{\nu}\right)}$
    return $\mu + \sigma t$
end function
```

E. Plots

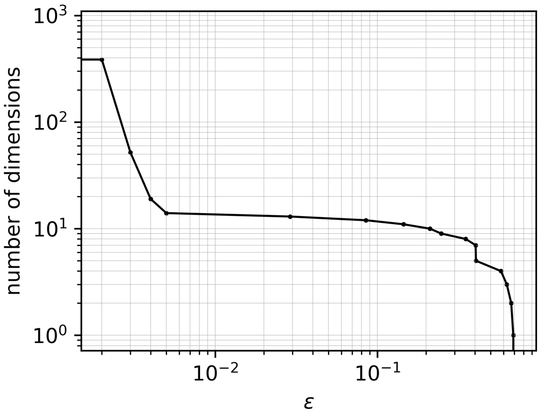

Figure E.1. Number of dimensions where $\rho^d/v^d > \epsilon$ plotted on a double logarithmic scale. Note that a low number of dimensions are responsible for most of the correlations, and that ϵ can be chosen robustly.

F. Model samples

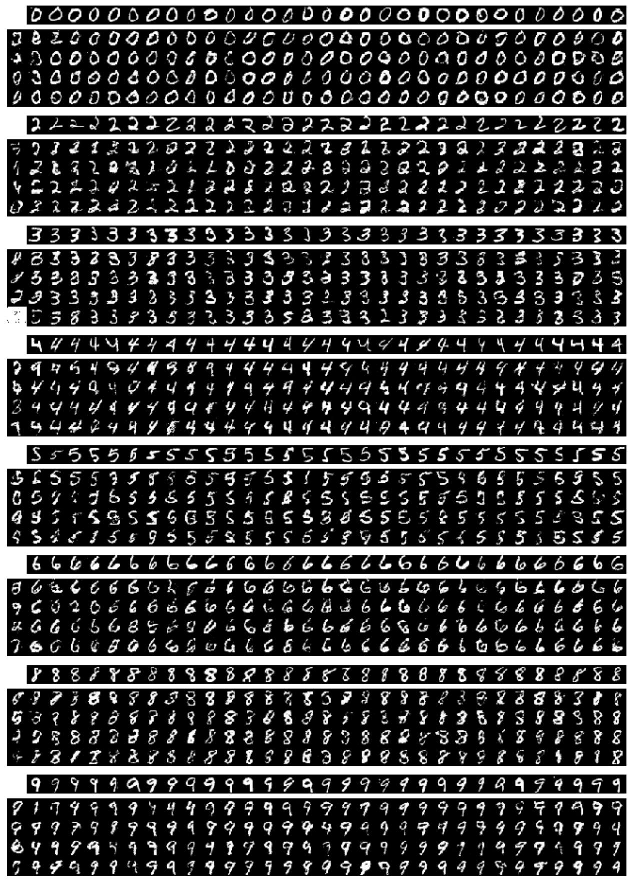

Figure F.1. Model samples as in Fig. 2A of the main text.

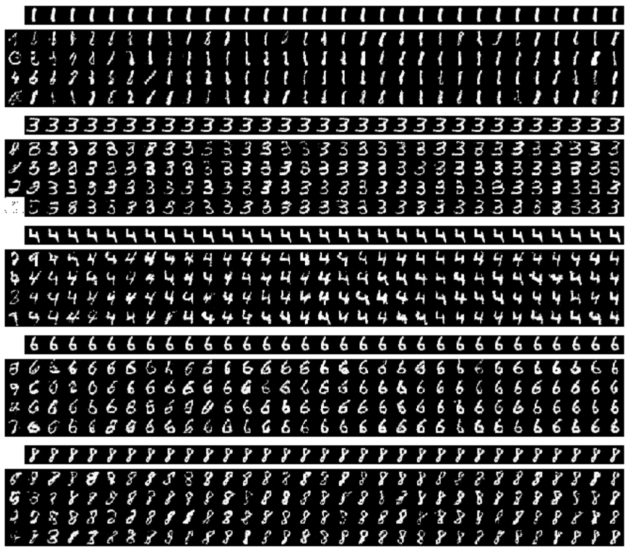
Figure F.2. Model samples, when the input image is repeated across the sequence (additive random noise is different).

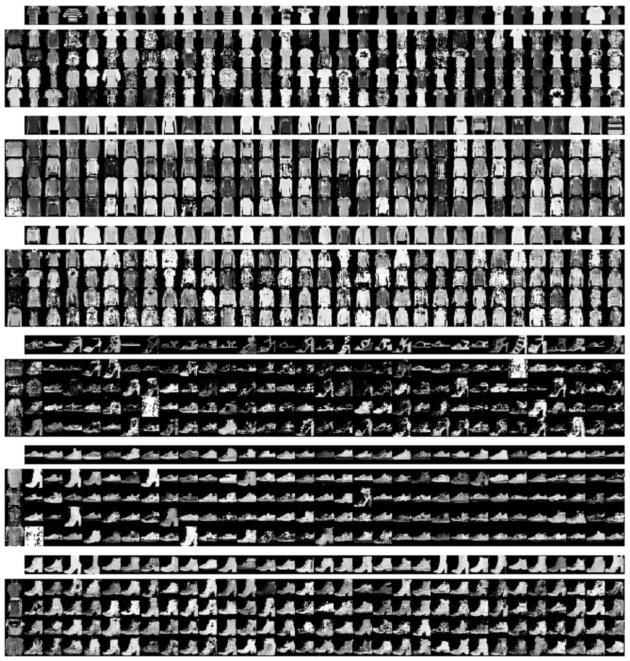
Figure F.4. Samples from the model trained on Fashion MNIST.

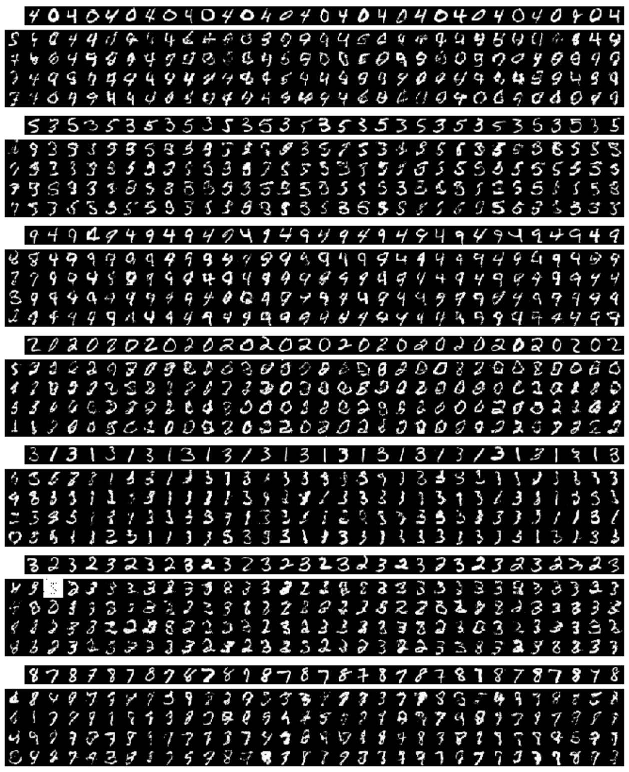
Figure F.3. Model samples when the input sequence is composed from images of two digits.

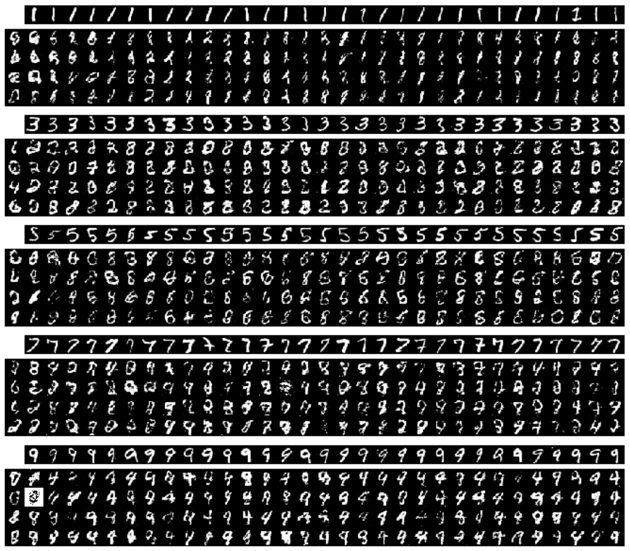
Figure F.5. Samples from the model trained only on even digits.

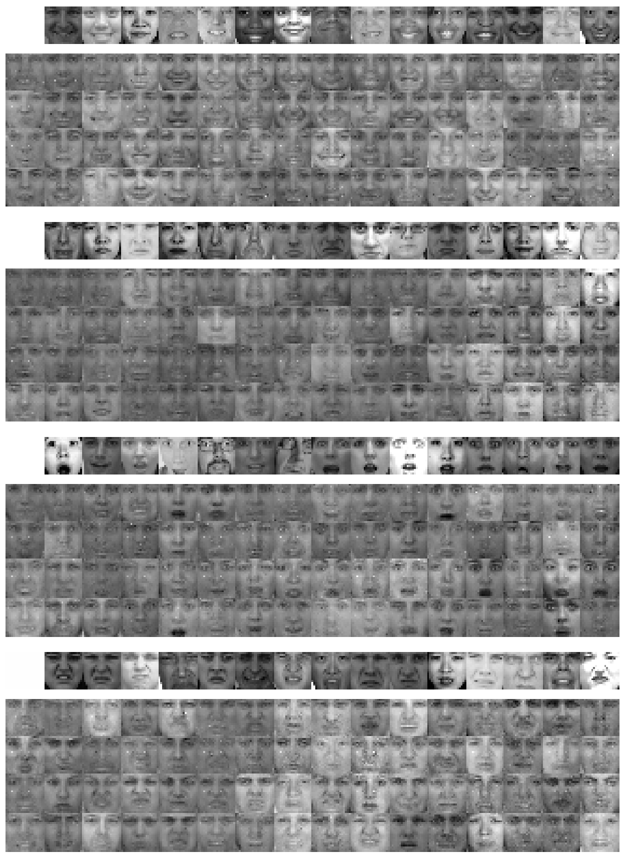

Figure F.6. Samples from the model trained on the labelled part of the TFD (Susskind et al., 2010) except for the images used here as the inputs. Images have the dimensionality of 48×48, so we modified our default model architecture. The training dataset is small (4066 images over 6 classes) so models are prone to overfitting. However, our model is still able to infer the emotion class from the sequence of the unseen images and sample faces from that class.

References

Clevert, D., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (ELUs). In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

Dinh, L., Krueger, D., and Bengio, Y. NICE: non-linear independent components estimation. *arXiv preprint*, abs/1410.8516, 2014.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

Papamakarios, G., Murray, I., and Pavlakou, T. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems 30*, pp. 2335–2344. 2017.

Shah, A., Wilson, A. G., and Ghahramani, Z. Student-t processes as alternatives to gaussian processes. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, pp. 877–885, 2014.

Susskind, J., Anderson, A., and Hinton, G.E. The Toronto face dataset., 2010.