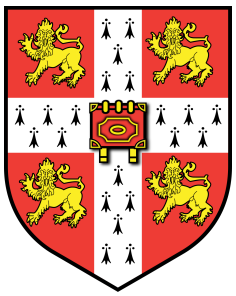
Learning Skew in GJS regularisation of VAEs.

Candidate Number: 8237W
Supervisor: Pietro Liò
Word Count: 4999

May 17, 2021



Department of Physics

University of Cambridge

Abstract

Variational Autoencoders (VAEs) learn latent representations of data via the maximisation of the Evidence Lower Bound (ELBo) which consists of a log model likelihood term and a regularisation term. Typically the regularisation term is chosen as the reverse Kullback-Leibler divergence (KL) which is known to have a zero-forcing property. It is not clear that constraining the latent distribution in such a way as to encourage this property will always maximise the information capacity of the latent space however, and in particular the zero-avoidance property of the forward KL may in some cases be more suitable. The Geometric Jensen-Shannon (GJS) divergence [1] has been proposed as an alternative regularisation due to its ability to smoothly interpolate between the forward and reverse KL, and enforce properties on the latent space that are intermediate between zero-forcing and zero-avoiding. Furthermore, it has been shown that by selecting an optimal alpha parameter VAEs using the GJS divergence can out perform standard VAE models.

Here an extension of the GJS divergence is presented which has an additional trainable skew parameter α. This new divergence was trained on four datasets and it was found that it was possible to consistently learn an optimal skew value independently of the initialisation value of α. Moreover, this optimal skew value consistently corresponded to the point in alpha space which produced a minimisation of the reconstruction loss of the VAE. To achieve accurate reconstructions, the generative performance of the GJS with trainable skew tended to suffer however, although this was not consistently observed across datasets.

In this report these results are presented in detail, in addition to a discussion as to why the learning skew can produce better performance than the standard VAE with reverse KL regularisation, and suggestions of the direction of further work.

Impacts of Covid-19

The impact of Covid-19 has been minimal as the project is computational.

1 Acknowledgements

I would like to thank my supervisor Pietro Liò for all his insightful comments and feedback throughout the completion of this project. I would also like to thank my day-to-day supervisors Nikola Simidjievski and Jacob Deasy for their constant help and support running my experiments.

I would finally like to thank my family and friends, and especially Úna, for their love and support throughout the completion of this work.

Contents

2 Introduction

At the heart of modern science is the philosophy of *Empiricism* - the notion that the source of all knowledge is observation. Yet Empiricism alone does not constitute a complete Philosophy Of Science [2], and the quantification of uncertainty is a vital component of any modern scientific work. To this end, Bayesian statistics plays an important role in many areas of Physics, particularly in data-intensive disciplines such as Particle Physics [3] and Astronomy [4], and therefore developments in methods to solve Bayesian inference problems are of direct relevance to the field of Physics in general.

Unsupervised Machine Learning, UML, deals with estimating probability distributions of phenomena given only observations of that phenomena. The observations are referred to as the *dataset* of the problem, and in general these observations can take any form. For any given problem, UML aims to uncover structure in the dataset in the form of a probability distribution without any prior knowledge of this structure.

In particular, in the UML framework data is taken to be generated from some probability distribution:

$$\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}) \tag{1}$$

It is assumed that the observed data $\mathbf{x}$ is dependent on some unobserved variables $\mathbf{z}$, the latent variables, and that the probability distribution from which data is drawn can be expressed as the marginal probability over these latent variables:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x},\, \mathbf{z}) \mathbf{dz} \tag{2}$$

The UML task is to estimate the parameters $\boldsymbol{\theta}$ of the joint distribution $p_{\boldsymbol{\theta}}(\mathbf{x},\, \mathbf{z})$. Ideally the learnt approximation to $p_{\boldsymbol{\theta}}(\mathbf{x},\, \mathbf{z})$ should be found in as flexible a form as possible so that Bayes' rule can be applied allowing various posterior probabilities to be computed. For example, if the task is to classify some observed data, Bayes' rule can be applied as:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\, \mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} \tag{3}$$

Conversely, samples in $\mathbf{x}$ can be drawn as:

$$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\, \mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z})} \tag{4}$$

Equations (3) and (4) show the motivation for defining the UML problem as one of estimating distributions involving latent variables: if the prior probability distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$ can be determined, then not only is a convenient distribution for calculating posterior probabilities obtained, but, in addition, information about the process generating the observed data itself is contained in the latent distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$.

Unfortunately, computing probability distributions in the UML framework is often difficult due to intractability of the above integral (2). Therefore a large body of work has been developed to efficiently estimate the marginal data distribution. One especially convenient way to rewrite the marginal probability is using a lower bound given by:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \left[\int p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \mathbf{dz} \right] \tag{5}$$

$$= \log \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z})} \left[p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \right] \tag{6}$$

$$\geq \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \right] \tag{7}$$

Where from (6) to (7) Jensen's inequality has been used. Expression (7) can be rewritten, using Bayes' rules from (4), as:

$$\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \right] = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z})} \left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z})} \right] = \mathcal{L}_{\boldsymbol{\theta}}(\mathbf{x}) \tag{8}$$

Expression (8) is a lower bound on the log model probability, meaning that this expression can be used rather than the exact form of log model probability in any optimisation problem. In the context of VAEs, expression (8) can be written as the *Evidence Lower Bound (ELBo)*.

A number of methods have been proposed to efficiently evaluate expression (8). For example, the *Expectation Maximisation Algorithm* [5, 6] iteratively optimises the ELBo by maximising $\mathcal{L}$ with respect to $p_{\boldsymbol{\theta}}(\mathbf{z})$ with constant $\boldsymbol{\theta}$, and then maximising with respect to $\boldsymbol{\theta}$ with constant $p_{\boldsymbol{\theta}}(\mathbf{z})$ in turn. On the other hand, *Variational Inference (VI)* [7, 8] employs a similar set-up to EM, but used calculus of variations to pose a constrained optimisation problem.

More recently, breakthroughs have been made due to the development of the *Variational Autoencoder (VAE)* framework which combines principles of VI with deep neural networks [9, 10]. Development of the VAE model is currently an active area of research, and a number of extensions to the original VAE framework have produced notable performance improvements [11, 12, 13, 14].

One recent extension to the VAE model demonstrated that using a modified VAE learning objective, in which the *Kullback-Leibler divergence (KL)* is replaced by the *Geometric Jensen-Shannon divergence (GJS)* as the *regularisation term*, can lead to performance improvements [1]. In this project an extension of the GJS VAE is presented in which the skew parameter of the GJS divergence α is a trainable parameter.

The primary contributions of this work are:

1. It is shown that for a given dataset, it is possible to consistently learn an optimal value of skew independently of the initialisation of α.
2. VAEs with trainable skew GJS produce best in class reconstruction performance, however this is at the cost of generative ability.
3. It is shown that these characteristics of the GJS divergence are demonstrable both in benchmark datasets as well as in a real world case study, on tasks of breast-cancer subtyping and prognosis.

This report has the following sections. First, in Section 3 the VAE framework is outlined, and the GJS with trainable skew is motivated. Section 4 outlines the experiments carried out, and Section 5 presents results. Section 6 discusses some interpretations of GJS with trainable skew and Section 7 concludes.

3 Background

3.1 The VAE Framework

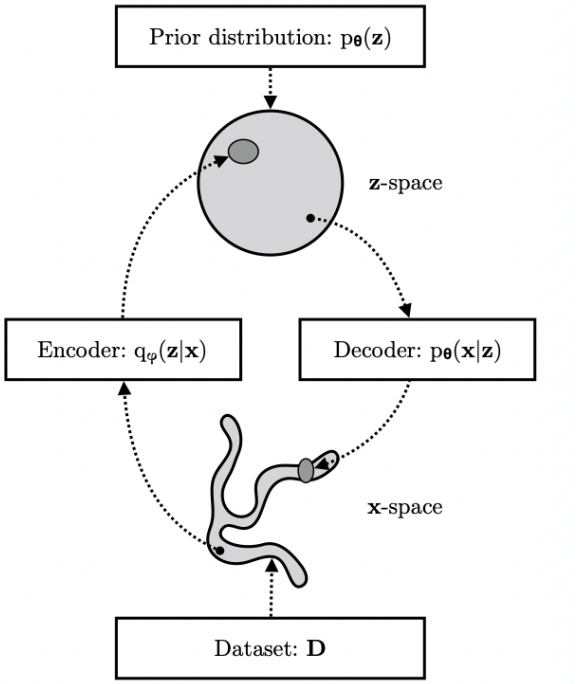


Figure 1: Encoding and decoding [15]

The VAE model is an autoencoding framework that maps between the 'coding' space and the model input and output space probabilistically. In particular, the VAE model consists of three probability distributions: $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, the probabilistic encoder which maps from the data space $\mathbf{x}$ to the latent space $\mathbf{z}$, $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ the probabilistic decoder which maps in the opposite direction, and $p_{\boldsymbol{\theta}}(\mathbf{z})$, the prior on the latent space. $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is also referred to as the *inference model*, as the latent space representation of a sample $\mathbf{x}$ can be used for downstream tasks such as classification or regression, and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ the *generative model*, as given a value of $\mathbf{z}$, a sample in the data space $\mathbf{x}$ can be generated by passing through the decoder. In particular, samples in $\mathbf{x}$ can be drawn as:

$$\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z}) \tag{9}$$

In the VAE framework the probability distributions $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ are parameterised by neural networks, and therefore the optimisation task is to compute the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ which maximise the probability distributions given the data $\mathbf{x}$. Generally, the parameterisation of the prior distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$ is chosen to be constant during training, and taken as a multivariate, diagonal Gaussian distribution over the latent space.

In order to implement the maximisation of the VAE probability distributions, it is convenient to express the model probabilities in terms of a lower bound:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \int p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})\mathbf{dz} \tag{10}$$

$$= \log \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\mathbf{dz} \tag{11}$$

$$= \log \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \tag{12}$$

$$\geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \tag{13}$$

From (13) we then have the definition of the *Evidence Lower Bound (ELBo)* as:

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x},\ \mathbf{z}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \tag{14}$$

As mentioned in Section 2, the ELBo is a more convenient metric to use as an objective in a variational optimisation. To see why this is the case, (14) can be rewritten in a number of forms which elucidate more clearly what is being optimised when the ELBo is maximised. For example, the ELBo is equivalent to the negative Helmholtz free energy of the model [16, 17]:

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x},\ \mathbf{z}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \tag{15}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \tag{16}$$

$$= E - H \tag{17}$$

$$= -F \tag{18}$$

i.e. maximisation of the ELBo corresponds to the minimisation of the model free energy. From the perspective of implementing the ELBo in practice however, a more convenient form is given in [9]:

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x},\ \mathbf{z}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \tag{19}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \tag{20}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - KL(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z})) \tag{21}$$

which is the form generally used in the optimisation of the encoder and decoder.

Optimisation of the ELBo Given that $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ are parameterised by neural networks, standard error *Backpropagation* [18, 19] and *Stochastic Gradient Descent, SGD,* [20] methods can be used to optimise $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. In particular, this consists of iterating each 'weight' term in the neural networks as:

$$w_{l,t+1} = w_{l,t} - \gamma \frac{1}{B} \sum_{i}^{B} \nabla_w Q(x_i, w_{l,t}) \tag{22}$$

where w is a general weight parameter in $\boldsymbol{\theta}$ or $\boldsymbol{\phi}$, B is the number of samples used to calculate the gradient in the iteration, and $\nabla_w Q(x_i, w_{l,t})$ is the gradient of the loss backpropagated to the l^{th} weight in the neural network.

Notice that in (22) the gradient of the loss, $\nabla_w Q(x_i, w_{l,t})$ is calculated using only a sample of the data, rather the complete data. The reason for this is computational, and it is only a valid procedure when gradients of a sample of the data is an unbiased estimate of the full gradient.

Therefore, unbiased estimates of the ELBo objective must exist if it is to be used as a valid optimisation objective, which is the case when the *reparameterisation trick* is used as shown in [9, 10]. In particular, we require:

$$\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) + \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \tag{23}$$

The gradient with respect to the generative parameters $\boldsymbol{\theta}$ can be written:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \tag{24}$$

However, a similar equality does not hold for the recognition parameters $\boldsymbol{\phi}$:

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \neq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \nabla_{\boldsymbol{\phi}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \tag{25}$$

To use gradient descent [21] to train VAEs, an expressions for the rhs of (25) is required. This can be obtained using the *Stochastic Backpropagation* rules as derived in [10]. This consists of using the reparameterisation trick, in which the latent space is resampled as:

$$\epsilon \sim g(\epsilon) \tag{26}$$

$$\mathbf{z}' = \boldsymbol{z} + \epsilon \tag{27}$$

$$g(\epsilon) = \mathcal{N}(0, I) \tag{28}$$

This then allows for unbiased estimated of the gradient (25) to be written as:

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) &= \nabla_{\boldsymbol{\phi}} \mathbb{E}_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) && (29)\\
&= \nabla_{\boldsymbol{\phi}} \int \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) && (30)\\
&= \nabla_{\boldsymbol{\phi}} \int \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}') && (31)\\
&= \int \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \nabla_{\boldsymbol{\phi}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}') && (32)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \nabla_{\boldsymbol{\phi}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}') && (33)
\end{aligned}$$

Equation (33) therefore allows for efficient calculation of the unbiased gradient of (21).

What does the ELBo optimise? The first term on the rhs of (21) is the negative reconstruction loss and the second term is a constraint between the inference model and the prior, referred to as the *regularisation*. The regularisation is minimised during training, meaning that it tends to enforce certain properties onto the latent representation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. These enforced properties help prevent over-fitting, and can also be useful in encouraging the VAE to learn interpretable latent space distributions [22, 11]. Constraining the latent space does not necessarily lead to better VAE models however, and the relationship between the regularisation and reconstruction terms of the ELBo is complex.

In particular, the minimisation of each term in the ELBO individually can have the effect of negatively impacting the minimisation of the other. For instance, the KL term is non-negative for all values of $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{z})$, and equal to zero only when $p_{\boldsymbol{\theta}}(\mathbf{z}) = q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. However, as the posterior approaches the prior, the information content of the posterior decreases, negatively impacting the reconstruction probabilities $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Conversely, if a powerful decoder is used in the model, $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ can become independent of $\mathbf{z}$, and good reconstruction performance can achieved without using the latent representation [23, 24].

In both these cases the latent space is not encoding information about the input, which is in opposition to the motivation of using a VAE model in the first place: one of the primary features of VAEs is the latent space distribution which we would like to be as informative as possible. Therefore, one key challenge for improving VAE models is how to simultaneously constrain the latent space, while at the same time ensuring that the quality of the inference and generative models does not suffer.

3.2 Extensions to the VAE

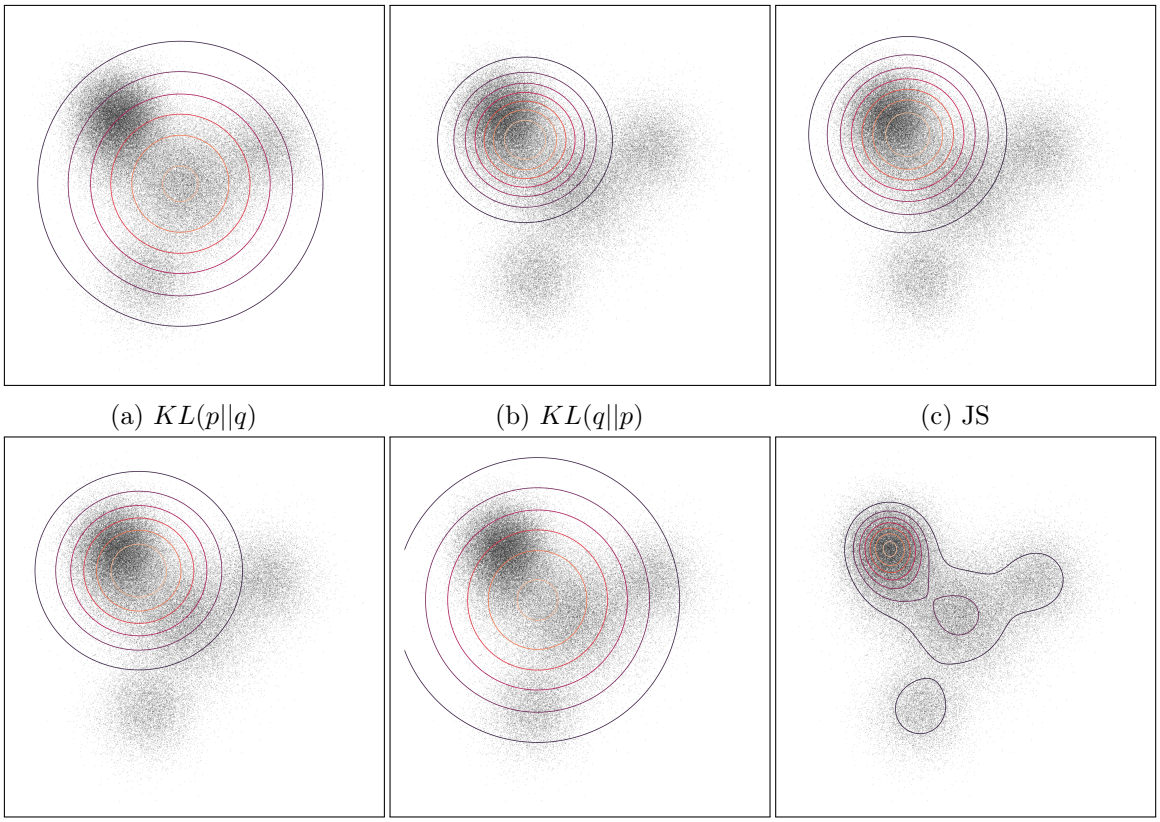
A number of modifications to the original VAE framework have been suggested. For example, multiple works have proposed using a modified form for the posterior. These include VAEs that use autoregressive flows as the posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ [25], which allows for more complex forms of probability density in $\mathbf{z}$, and VAEs with inverse autoregressive flows [26], which extends [25] allowing for higher dimensional latent spaces to be used. Autoregressive models have also been implemented as the prior $p_{\boldsymbol{\theta}}(\mathbf{z})$ and posterior $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ [23].

Alternatively, other works have proposed that by modifying just the prior, a better representation in the latent space can be achieved. These include using a mixture of Gaussians prior [27], using a multimodal prior [28], estimating the prior as a mixture of approximate posteriors [29].

Instead of modifying the prior, using different regularisations in the ELBo have been proposed to improve the representation learnt by the latent space. Most notably, by increasing the weight of the regularisation term relative to the reconstruction term, the β-VAE [11], was able to achieve better *disentanglement* in the latent space. An extension of this VAE variant, β-VAE with controlled capacity increase [22], which gradually increases β during training, has also been shown to improve upon [11].

Meanwhile, more recently it has been demonstrated that by replacing the reverse KL regularisation with a linear combination of reverse and forward KLs, better VAE performance can be achieved [1]. In particular, in the original VAE framework the regularisation term is the reverse KL between the prior and the posterior:

$$KL(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{z})) = \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z})} \mathbf{dz} \tag{34}$$



(a) $KL(p||q)$ (b) $KL(q||p)$ (c) JS

(d) GJS, $\alpha = 0.5$ (e) GJS learnt alpha. (f) Actual distribution.

Figure 2: Fitted bivariate Guassian distribution optimised by minimising various divergences, plotted over samples from the actual Gaussian mixture distribution.

Given that the KL between any two distributions P and Q is greater than or equal to zero, and zero only at $P = Q$, this constraint pushes the posterior to have a form approaching that of the prior. The KL divergence is not symmetric however, meaning that we could alternatively regularise using the forward KL:

$$KL(p_{\boldsymbol{\theta}}(\mathbf{z}) \parallel q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) = \int p_{\boldsymbol{\theta}}(\mathbf{z}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \mathbf{dz} \tag{35}$$

The forward and reverse KL have different properties, meaning that the divergence chosen as the regularisation plays a crucial role in how VAEs learn. In particular, the reverse KL is known to have *zero-forcing* and *mode matching* properties, while the forward KL has *mean matching* and *zero-avoidance* properties [30, 31]. Figure 2 shows how these properties impact the form of the learnt distribution in the case of fitting a bivariate Gaussian to a multimodal mixture of Gaussians by minimising the divergence between the actual distribution and learnt distribution.

In particular, while the reverse KL tends to focus the learnt probability density on strong modes in the mixture distribution, the forward KL generally learns a more spread density, centred on the mean of the mixture. Optimisation using the forward and reverse KL clearly leads to different characteristics, which may be more or less useful depending on the case at hand. For instance, if sampling from the latent is important, the reverse KL optimisation may be more appropriate. On the other hand, if the task is to map from the data space to

the latent space for a dimensionality reduction task, the reverse KL may not be so good, as we would like to maintain variation in the data when mapping from $\mathbf{x}$ to $\mathbf{z}$.

This raises the question of whether some intermediate divergence exists, the optimisation of which produces a hybrid of the properties of the forward and reverse KL. The *Geometric Jensen-Shannon, GJS* divergence [1] does exactly this by using a symmetrised version of the KL in which the mixture distribution is a geometric average.

This divergence is given by:

$$GJS_\alpha(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{z})) = (1-\alpha)KL(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \parallel G_\alpha) + \alpha KL(p_{\boldsymbol{\theta}}(\mathbf{z}) \parallel G_\alpha) \tag{36}$$

Similarly, the dual of (36) is:

$$GJS^*_\alpha(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{z})) = (1-\alpha)KL(G_\alpha \parallel q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) + \alpha KL(G_\alpha \parallel p_{\boldsymbol{\theta}}(\mathbf{z})) \tag{37}$$

Where the geometric distribution is:

$$G_\alpha = q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})^\alpha p_{\boldsymbol{\theta}}(\mathbf{z})^{1-\alpha} \tag{38}$$

Given that this distribution is the sum of two KL terms, an exact form exists for this divergence whenever the prior and posterior are parameterised as Gaussians, meaning that it can be used in the standard VAE model.

In particular, the KL between two multivariate Normal distributions, $\mathcal{N}_1(\mu_1, \Sigma_1)$ and $\mathcal{N}_2(\mu_2, \Sigma_2)$, is:

$$KL(\mathcal{N}_1(\mu_1, \Sigma_1) \parallel \mathcal{N}_2(\mu_2, \Sigma_2)) = \frac{1}{2}\left(tr\left(\Sigma_2^{-1}\Sigma_1\right) + \ln\left[\frac{|\Sigma_2|}{|\Sigma_1|}\right] + (\mu_1-\mu_1)^T\Sigma_2^T(\mu_1-\mu_1) - n\right) \tag{39}$$

With this analytic form of the GJS divergence in hand, a new optimisation objective for the VAE can then be defined by posing a constrained optimisation problem [11, 1]:

$$\max_{\phi,\theta} \mathbb{E}_{p_{\mathcal{GJS}}(x)}\left[\log \mathbb{E}_{q_\phi(z|x)}\left[p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right]\right] \qquad \text{subject to } \; GJS(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z})) < \varepsilon, \tag{40}$$

Equation (40) can then be written as a Lagrangian, using *Karush–Kuhn–Tucker (KKT)* conditions [32, 33], to obtain a modified ELBO:

$$\mathcal{F}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x},\, \mathbf{z}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - \lambda(GJS(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z})) - \epsilon) \tag{41}$$

This objective can be used to train VAE models using the same algorithm with reparameterisation trick as the regular VAE.

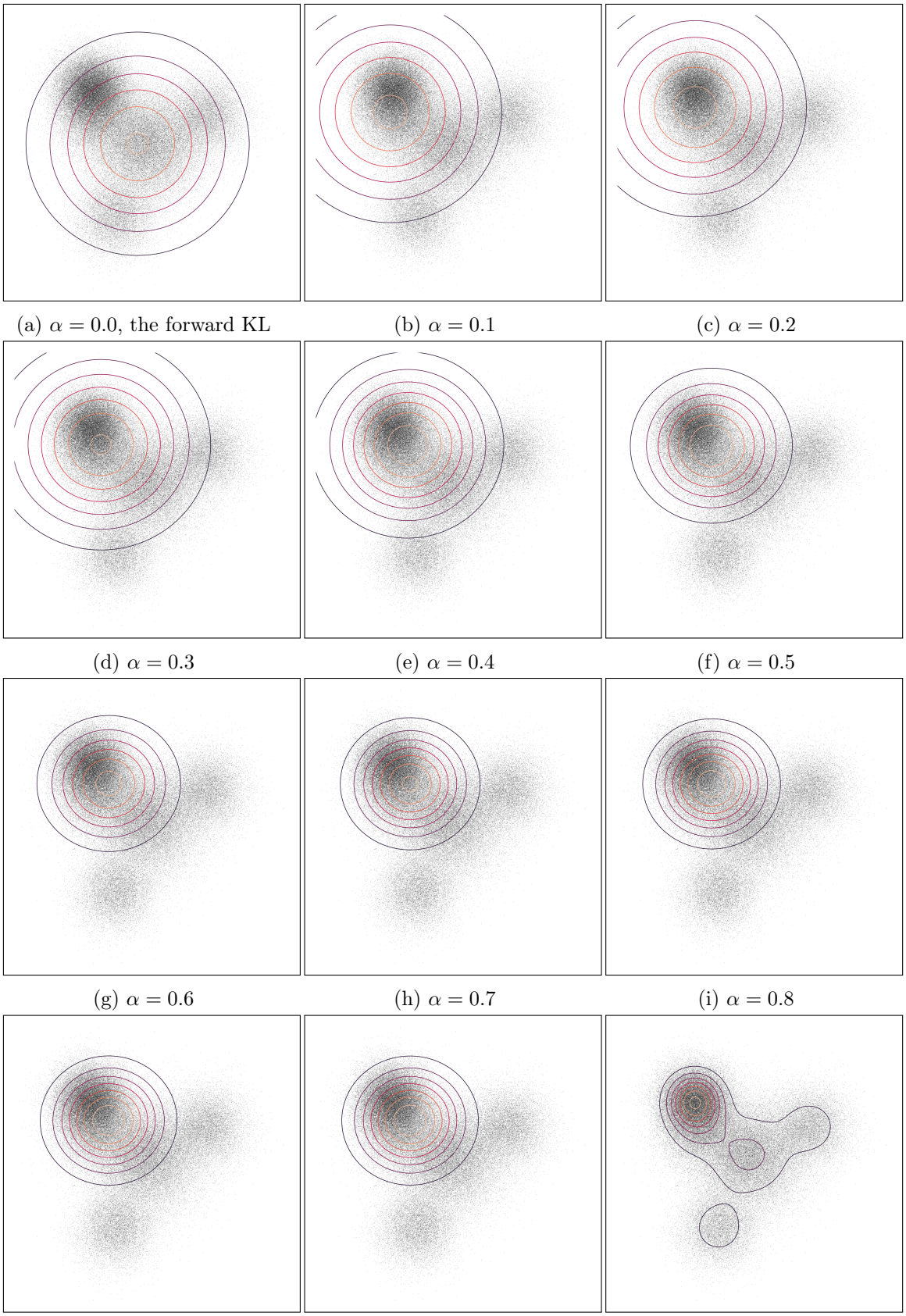
3.3 The GJS Skew Parameter

The GJS divergence is useful because, by varying the skew parameter α, this divergence smoothly interpolates between the forward and reverse KL. Figure 3 shows how this

parameter can be used to vary the properties of a single bivariate Gaussian approximating a multimodal mixture of Gaussians, and how values of the skew parameter between zero and one produce properties intermediate between those of the forward and reverse KL.

It has been demonstrated that when the regularisation of a VAE is taken as the GJS, it is possible to find values of the skew parameter which outperform the traditionally used reverse KL on reconstruction tasks [1]. Moreover, it was shown the point of alpha for best reconstruction was dependent on the dataset, meaning that the GJS divergence is a more flexible VAE regularisation which can adjust the properties it enforces specifically to the task at hand.

However, to find the point of optimal skew, expensive grid searches over the space of alpha were previously required. This raises the question of whether it is possible to automatically discover the optimal alpha during training, and the extent to whether this can be answered in the affirmative or negative is the basis of this project.



(a) $\alpha = 0.0$, the forward KL (b) $\alpha = 0.1$ (c) $\alpha = 0.2$

(d) $\alpha = 0.3$ (e) $\alpha = 0.4$ (f) $\alpha = 0.5$

(g) $\alpha = 0.6$ (h) $\alpha = 0.7$ (i) $\alpha = 0.8$

(j) $\alpha = 0.9$ (k) $\alpha = 1.0$, the reverse KL (l) Actual Distribution.

Figure 3: Learnt Gaussian distribution for the Geometric Jensen-Shannon divergence for varying alpha value.

4 Methods

This project aims to extend the work carried out in [1] by investigating whether the skew parameter of the GJS divergence can be automatically learnt during training. Therefore, in the same guise as [1], this work is primarily concerned with investigating the *relative* performance improvements using GJS with trainable skew rather than any absolute measure of performance. In particular, the primary benchmarks for comparison are the VAE with reverse KL regularisation as in [9, 10], VAE with forward KL, and VAE with Maximum Mean Discrepancy regularisation [12]. These variations of the VAE are chosen for comparison as they modify the ELBo in the same way as the proposed GJS, i.e. only by the changing form of the regularisation.

Reconstruction loss and generative log model probability are the two metrics which were used to characterise the models investigated in this project. Unless otherwise stated, the reconstruction loss used was Mean Squared Error, and the log model probability was estimated by drawing samples from the trained VAE and comparing these samples with the actual probability distribution in the input space using Binary Cross Entropy.

4.1 Datasets

The following benchmark datasets were investigated, and implemented in PyTorch [34] using the decoder and encoder from [11], adapting the code from [1]:

- MNIST [35] - contains 28 x 28 images of handwritten digits, with an equal balance of each digit between 0-9.
- FashionMNIST [36] - a dataset which aims to emulate the properties of MNIST, but with the more complex example of images of pieces of clothing. There are again 10 distinct classes, each of which appears in the data set with the same frequency.
- dSprites [37] - a synthetic dataset consisting of 64 x 64 images of white shapes on a black background which is produced by five ground truth factors - shape, scale, orientation, x position and y position. The dataset has 737,280 samples in total.

In addition to these canonical datasets, a further real-world example was investigated, the METABRIC cancer database [38]. This dataset consists of the copy number, gene expression and clinical observation data for 1,992 breast cancer patients. For this dataset, in addition to the reconstruction metric, a downstream tasks important for this data domain was also used to characterise model performance.

In particular, Random Forest [39] and SVM [40, 41] classifiers were trained on the latent space representation of data on the task of predicting *distance relapse*, which is a binary variable indicating whether a patient's illness relapsed after treatment. It would clearly be very useful to predict this variable accurately, but given its complex dependence on multiple data types, it is a non-trivial task to combine all the available medical data into a single predictive model. However, by using a VAE as a pre-processing step to integrate different data types, it has been shown that improved accuracy of predictive models can be achieved [42].

In this project a number of data integration tasks of the METABRIC dataset are investigated. In particular, the different integration combinations investigated were:

1. Clinical and RNA gene expression
2. Clinical and CNA expression
3. RNA and CNA expression

These components of METABRIC have the following data types:

1. Clinical data: mixture of continuous and categorical
2. CNA expressions: ordinal
3. RNA expressions: continuous

Two structures of encoder and decoder pairs were examined on METABRIC in this project, the 'CNC' and 'X' VAEs from [42]. The CNC VAE consisted of concatenating the raw input data to produce a single input vector $\mathbf{x}$. This method requires a single measure of the reconstruction loss to be applied to the output vector from the VAE, which is a blunt method of measuring reconstruction given that the data types concatenated to produce the input would individually be more suited to different loss measures to one another. The X flavour of VAE represents a more nuanced approach in this respect by passing a tuple of vectors to the VAE. This allows specific loss functions to be applied to each vector in the output tuple, which should help the VAE to learn.

Therefore, six different data integration and VAE structure combinations were used to compare the GJS divergence performance in this project. This part of the project is a particularly useful extension to the benchmark datasets as:

1. Allows the GJS properties to be tested on data other than image datasets.
2. Allows for the performance of GJS to be demonstrated on a real-world task that is arguably more difficult than the standard datasets.

Therefore, any trends from this section of the project should be somewhat more indicative of the general potential for GJS to be used in practical applications.

In the following section key results from the experiments carried out are presented. In particular, for the standard datasets, alpha convergence, reconstruction performance and generative ability are shown, while for METABRIC, alpha convergence and classification performance are presented.

5 Results

5.1 Alpha Convergence

Divergence	MNIST	FashionMNIST	dSprites
GJS	0.120 ± 0.0007	0.16 ± 0.009	0.06 ± 0.007
Dual GJS	0.360 ± 0.0005	0.38 ± 0.001	0.390 ± 0.0005

Table 1: Converged values of alpha after 100 epochs of training.

It was found that for all datasets investigated, it was possible to consistently learn an optimal value of the skew parameter independent of the initialisation of α. Moreover, the value converged to was dependent on the dataset. Figure 4 shows how the value of alpha progresses during the first epochs of training for MNIST, and Tables 1 and 2 contain the values to which alpha converged for all the datasets investigated. Additional plots of alpha convergence for all the datasets investigated can be found in Appendix A.

Given that convergence of the skew parameter to a stationary point independent of initialisation could consistently be observed across datasets, characterisation of the GJS divergence in alpha space was a natural next step in this project. In particular, the shape of the GJS divergence and total VAE loss with respect to alpha is crucial in determining why it is possible to train the skew at all. Two experiments were carried out in order to investigate this characterisation:

1. Training with fixed alpha, for sweeping alpha 0 to 1.0 in 0.01 intervals, and recording how the total VAE loss iterates during training for each value of alpha.
2. Training with learning alpha, and at each epoch in training, sweeping through the space of alpha and recording how the divergence changes.

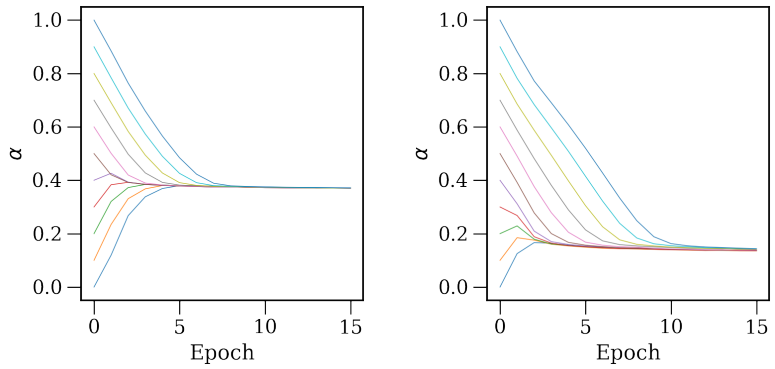
Figure 5 shows the former of these experiments, and Figure 6 the latter.

Notably, these methods of characterisation both indicated that there are minima in the objective measured with respect to alpha, for both the regular and dual GJS. Specifically, the first measure, which was only investigated on MNIST and FashionMNIST due to limited availability of compute time, was found to show minima that aligned with the converged values of alpha. On the other hand, the shape of the divergence during training was found to be clearly convex, which intuitively explains why it is possible to train alpha - in gradient descent, the value of alpha can simply iterate toward the minimum of the convex function.

Appendix B contains additional plots showing the total VAE loss experiment, while Appendix C contains additional convex plots of the divergence for additional datasets.

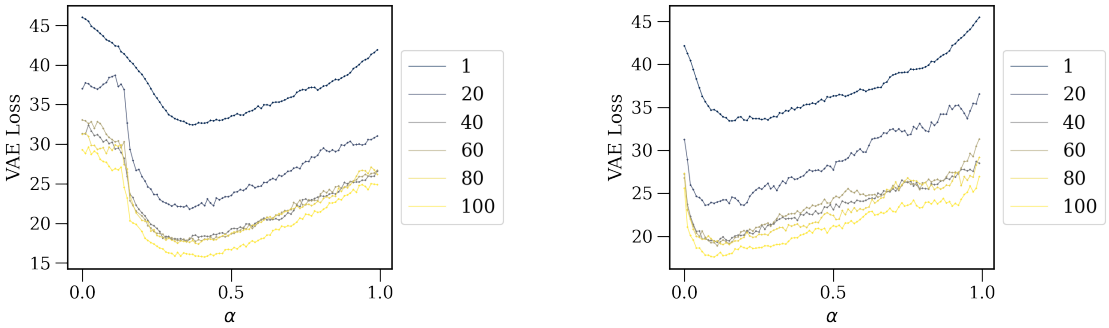
Divergence	Clin + RNA		Clin + CNA		RNA + CNA	
	X	CNC	X	CNC	X	CNC
GJS	0.42 ± 0.01	0.47 ± 0.005	0.32 ± 0.02	0.31 ± 0.02	0.33 ± 0.02	0.40 ± 0.03
Dual GJS	0.48 ± 0.002	0.49 ± 0.004	0.43 ± 0.004	0.43 ± 0.001	0.42 ± 0.006	0.44 ± 0.006

Table 2: Converged values on METABRIC.



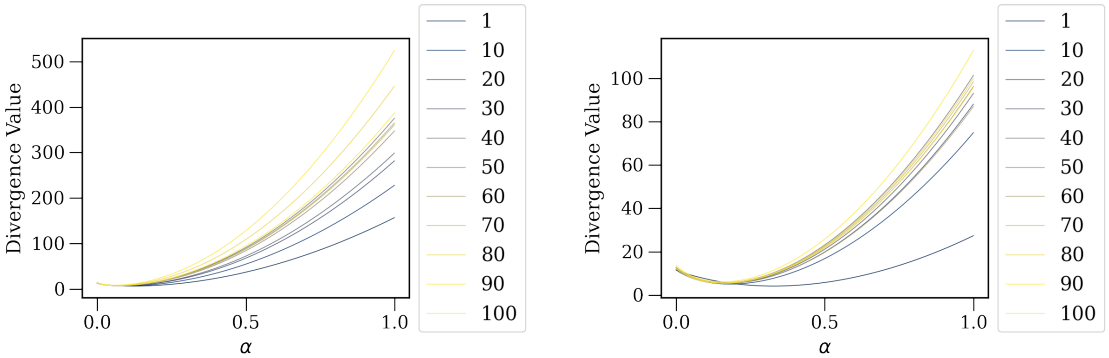
(a) Dual GJS, MNIST (b) GJS, MNIST

Figure 4: Alpha convergence on MNIST.



(a) Dual GJS, MNIST (b) GJS, MNIST

Figure 5: Total VAE loss versus skew during training of GJS VAE with constant alpha, MNIST.



(a) Dual GJS, MNIST (b) GJS, MNIST

Figure 6: Divergence value vs skew during training, for FashionMNIST. Data was generated by training with trainable alpha and at each epoch sweeping through alpha in increments of 0.01.

5.2 Reconstruction Performance

Demonstrating whether learning skew in GJS regularisation is actually beneficial for training VAE models was a key line of enquiry in this project. In particular, although learning skew is is a relatively straightforward adjustment to the original VAE framework, requiring the learning of only one additional hyper-parameter, for this divergence to of practical use, it must show performance improvements over the reverse KL.

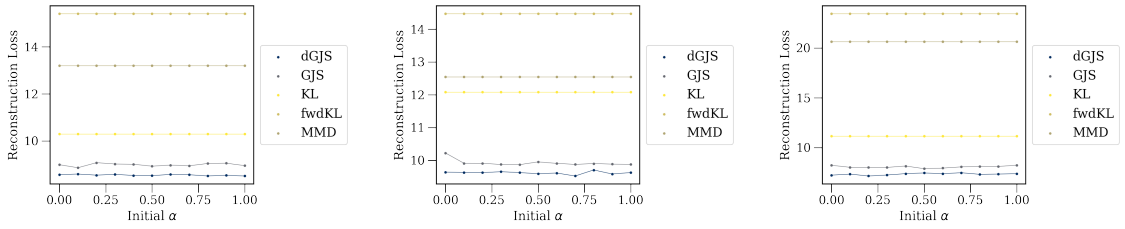
Two metrics of VAE performance were investigated in this project, the first of which was the reconstruction loss of the model. This is a measure of how well the VAE represents the data by comparing a data sample in $\mathbf{x}$ with a data sample that has been passed through the encoder and then decoder sequentially, and ideally the reconstruction loss should be as low as possible.

It was found that, in general, training VAEs using GJS with trainable skew led to better reconstruction performance. This was observed universally in the case of the benchmark datasets, as shown in figure 7, while for METABRIC better reconstruction performance was observed for the GJS in a majority tasks. Table 3 shows this performance, and it was found that the dual GJS gave best in class performance on seven out of the nine datasets investigated.

Additional plots can be found in Appendix D.

Divergence	**MNIST**	**FashionMNIST**	**dSprites**	**Clin + RNA**		**Clin + CNA**		**RNA + CNA**	
				X	CNC	X	CNC	X	CNC
GJS	0.86	0.82	0.72	1.05	1.17	0.98	0.99	0.99	1.30
Dual GJS	**0.83**	**0.80**	**0.66**	**1.001**	1.13	**0.97**	**0.98**	**0.98**	1.20
MMD	1.28	12.55	1.85	⋮	⋮	⋮	⋮	⋮	⋮
Forward KL	1.15	1.20	2.10	1.14	1.16	1.02	1.02	1.04	1.20
Reverse KL	1.0	1.0	1.0	**1.0**	**1.0**	1.0	1.0	1.0	**1.0**

Table 3: Reconstruction loss relative reverse KL.



(a) MNIST (b) FashionMNIST (c) dSprites

Figure 7: Reconstruction benchmark datasets.

5.3 Generative Ability

A second metric to measure how well VAEs capture information about the input data is the generative ability of the decoder model, and Table 4 shows these results.

Divergence	MNIST	FashionMNIST	dSprites
GJS	−105	**-242**	−47
Dual GJS	−109	245	−46
Forward KL	−127	−257	−35
Reverse KL	**-103**	**-242**	−28
MMD	−115	−247	**-21**

Table 4: Generative performance in terms of $\log p_{\boldsymbol{\theta}}(\mathbf{x})$.

On FashionMNIST it was found that the dual GJS was competitive with the reverse KL, however this was not found to be replicated across other datasets, and in particular, the performance of the reverse KL in this task was far superior in the other two datasets investigated.

5.4 Real-world case study

Given the performance improvements demonstrated using GJS with trainable skew in the benchmark datasets, a real-world use case of VAEs was also used to further investigate the potential benefits of using this divergence in practice.

Table 5 shows the predictive accuracy of Random Forest and Support Vector Machine classifiers trained on predicting the patient outcome 'distance relapse' using a low dimensional input produced using VAE with various divergences and with PCA.

It was found that the GJS was best or tied best on seven out of the eleven tasks investigated. Although this consistent performance benefit is encouraging, the magnitude of these improvements was only of the order of tenths of a percent.

		Clin + RNA		Clin + CNA		RNA + CNA	
	Divergence	**RF**	**SVM**	**RF**	**SVM**	**RF**	**SVM**
X VAE	GJS	69.27 ± 3.39	$\mathbf{70.00 \pm 0.9}$	$\mathbf{70.24 \pm 2.72}$	$\mathbf{70.80 \pm 1.62}$	68.51 ± 3.01	70.96 ± 1.20
	dGJS	69.43 ± 3.7	69.78 ± 1.02	69.72 ± 2.77	70.28 ± 1.8	$\mathbf{68.92 \pm 3.06}$	$\mathbf{71.0 \pm 1.13}$
	Reverse KL	69.31 ± 0.65	$\mathbf{70.85 \pm 0.0}$	70.12 ± 0.39	$\mathbf{70.85 \pm 0.0}$	67.50 ± 0.86	70.85 ± 0.0
	Forward KL	$\mathbf{69.84} \pm 0.84$	69.43 ± 0.0	70.10 ± 0.59	70.04 ± 0.0	69.35 ± 0.84	70.65 ± 0.0
CNC VAE	GJS	69.01 ± 3.15	70.65 ± 0.0	69.78 ± 3.22	70.91 ± 1.55	68.83 ± 3.65	70.91 ± 1.13
	dGJS	$\mathbf{70.06} \pm 3.86$	70.65 ± 0.0	$\mathbf{70.30} \pm 2.40$	$\mathbf{71.09 \pm 1.52}$	69.25 ± 2.99	$\mathbf{71.16 \pm 1.26}$
	Reverse KL	68.72 ± 0.87	70.65 ± 0.0	69.60 ± 0.96	70.45 ± 0.0	$\mathbf{69.37 \pm 1.09}$	70.85 ± 0.0
	Forward KL	69.47 ± 0.63	70.65 ± 0.0	69.31 ± 0.57	$\mathbf{71.26 \pm 0.0}$	68.10 ± 0.63	70.65 ± 0.0
PCA		71.13 ± 0.67	70.85 ± 0.18	68.83 ± 0.93	70.32 ± 0.23	69.01 ± 0.68	70.02 ± 0.13

Table 5: Accuracy of binary classifier. GJS values are quoted are the average of 11 models trained with varying initial alpha. Errors quoted with respect to 10 different seed values.

6 Discussion

6.1 Why does alpha converge?

The form of the GJS divergence in alpha space is a sum of two terms that are quadratic in alpha [1]. In particular, by writing out equation (36) in terms of integrals we have:

$$GJS_\alpha(q \parallel p) = (1-\alpha)\int q \log \frac{q}{q^\alpha p^{1-\alpha}} dz + \alpha \int p \log \frac{p}{q^\alpha p^{1-\alpha}} dz \tag{42}$$

$$= (1-\alpha)\int q \log \frac{q^{1-\alpha}}{p^{1-\alpha}} dz + \alpha \int p \log \frac{p^\alpha}{q^\alpha} dz \tag{43}$$

$$= (1-\alpha)^2 \int q \log \frac{q}{p} dz + \alpha^2 \int p \log \frac{p}{q} dz \tag{44}$$

$$= (1-\alpha)^2 KL(q \parallel p) + \alpha^2 KL(p \parallel q) \tag{45}$$

In a similar fashion we can also write the dual GJS as:

$$GJS^*_\alpha(q \parallel p) = (1-\alpha)^2 \int q^\alpha p^{1-\alpha} \log \frac{p}{q} dz + \alpha^2 \int q^\alpha p^{1-\alpha} \log \frac{q}{p} dz \tag{46}$$

It can be shown that there is a single minimum in the GJS with respect to alpha, as shown in proposition 6.1.

Proposition 6.1. *The stationary point of* JS^{G_α} *with respect to* α *occurs at*

$$\alpha^* = \frac{\mathrm{KL}(q \parallel p)}{\mathrm{KL}(q \parallel p) - \mathrm{KL}(p \parallel q)} \tag{47}$$

and is a global minimum.

Proof. Differentiating (42) with respect to α gives

$$\frac{dGJS_\alpha}{d\alpha} = 2(1-\alpha)\mathrm{KL}(q \parallel p) + 2\alpha \mathrm{KL}(p \parallel q) = 0, \tag{48}$$

and rearranging gives equation (47), before differentiating again

$$\frac{d^2 GJS_\alpha}{d\alpha^2} = 2\mathrm{KL}(q \parallel p) + 2\mathrm{KL}(p \parallel q) \geq 0, \tag{49}$$

demonstrates the global minimum property as KL divergence is always positive. □

Given the more complex form of the dual GJS it is more difficult to formally derive the point of minimum in this divergence. However, the results presented in section 5.1, and plots in Appendix C, which indicate that the dual GJS is convex with respect to alpha, are strongly suggestive that there is also a single minima in alpha for this divergence.

Therefore, we have an analytical motivation for why alpha converges: the learnt value of alpha is the minimum of a convex function.

6.2 VAEs in practice

The reverse KL has been regularisation of choice in VAEs since their inception [9, 10]. Furthermore, some of the most important extensions to the VAE framework utilise KL regularisations. These include, the *Stein Variational Family* of InfoVAE objectives [12], NVAE [13], which optimises a KL term on each of the latent variable layers in the deep hierarchical encoder, and Child's recent deep VAE [14], which similarly uses KL as regularisation on each latent layer. Deep VAEs in particular have been shown to produce state-of-the-art results on CIFAR-10 [43] and ImageNet [44]. KL regularisation is also important in reinforcement learning, where the learning of *World Models* can be implemented using *KL Balancing* [45].

Given that in the present work, the consistent out-performance of KL by GJS with trainable skew has been demonstrated, there is therefore significant scope to improve the above mentioned state-of-the-art results by using this divergence. This is particularly true for the latter mentioned case, as in reinforcement learning the important metric of performance is reconstruction loss.

Before GJS with trainable skew can be deployed more broadly however, there are still a number of areas of work. Primarily, further analytical analysis of the dual GJS is required to prove that it has a minimum in alpha space, which would prove that skew is always trainable. Analysis using Bregman divergences [46] is one potential path to showing this, given the close relation of the TVO to the GJS objective.

Additionally, a more thorough theoretical understanding of VAEs with GJS is also required. In particular, a number of theoretical works on the ELBo could be extended to analyse the GJS. This includes [47], which, by posing the optimisation of the ELBo as one with constraints in the data space rather than the latent space, offers a more robust way to balance data compression and reconstruction, and sheds light on the mechanisms leading to poor reconstructions and latent representations in VAEs. An analysis of the GJS using the GECO algorithm proposed in this paper may lead to insights about why the GJS gives better reconstruction.

A related theoretical analysis of the ELBo is offered by [48], in which the trade-off between reconstruction and compression in VAEs in analysed using *rate-distortion theory*. In this work it is noted that the ELBo alone cannot distinguish between models that make use of latent variables and those that do not, and an alternative optimisation procedure is proposed that codifies the objective in terms of the *Rate* and *Distortion*. It is shown that this alternative procedure is a generalisation of the β-VAE, and that it achieves better latent representations. An analysis of the GJS using this framework would be beneficial in determining how GJS learns latent representations.

7 Conclusion

This project has demonstrated that learning skew in the regularisation of VAEs can be done independently of hyper-parameter initialisation, and that optimal alpha corresponds to best reconstruction loss. Improving reconstruction loss, tends to come at the cost of generative performance however. These findings were observed for benchmark datasets, and for a real-world case study. Although in the real-world example performance improvements on the task investigated were marginal the fact that significant reconstruction performance improvements were observed on this dataset, as well as all the benchmarks, suggests that GJS with trainable skew has potential use cases in tasks where reconstruction is important.

References

[1] Jacob Deasy, Nikola Simidjievski, and Pietro Liò. Constraining variational inference with geometric jensen-shannon divergence. 2021.

[2] Peter Godfrey-Smith. Theory and reality: An introduction to philosophy of science. *Bibliovault OAI Repository, the University of Chicago Press*, 01 2003.

[3] Dan Guest, Kyle Cranmer, and Daniel Whiteson. Deep learning and its application to lhc physics. *Annual Review of Nuclear and Particle Science*, 68(1):161–181, Oct 2018.

[4] Dalya Baron. Machine learning in astronomy: a practical overview, 2019.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[7] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[8] Charles Fox and Stephen Roberts. A tutorial on variational Bayesian inference. *Artificial Intelligence Review*, 38(2):85–95, 2012.

[9] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. 2014.

[10] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models, 2014.

[11] I. Higgins, Loïc Matthey, A. Pal, Christopher P. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[12] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Balancing learning and inference in variational autoencoders. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5885–5892, Jul. 2019.

[13] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder, 2021.

[14] Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images, 2021.

[15] D. P. Kingma and M. Welling. *An Introduction to Variational Autoencoders*. 2019.

[16] Radford M. Neal and Geoffrey E. Hinton. A new view of the em algorithm that justifies incremental and other variants. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1993.

[17] Matthew Beal. Variational algorithms for approximate bayesian inference /. *PhD thesis*, 01 2003.

[18] Rober Hecht-Nielson. Iii.3 - theory of the backpropagation neural network. based on "nonindent" by robert hecht-nielsen, which appeared in proceedings of the international joint conference on neural networks 1, 593–611, june 1989. © 1989 ieee. In Harry Wechsler, editor, *Neural Networks for Perception*, pages 65–93. Academic Press, 1992.

[19] Michael A. Nielsen. Neural networks and deep learning, 2018.

[20] Léon Bottou. *Stochastic Gradient Descent Tricks*, pages 421–436. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[22] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. 2018.

[23] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. 2017.

[24] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. 2016.

[25] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. 2016.

[26] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow. 2017.

[27] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. 2017.

[28] Rui Shu. Stochastic video prediction with conditional density estimation. 2016.

[29] Jakub Tomczak and Max Welling. Vae with a vampprior. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1214–1223. PMLR, 09–11 Apr 2018.

[30] Mingtian Zhang, Thomas Bird, Raza Habib, Tianlin Xu, and David Barber. Variational f-divergence minimization, 2019.

[31] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007.

[32] William Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, Chicago, IL, USA, 1939.

[33] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*, pages 481–492, Berkeley and Los Angeles, 1951. University of California Press.

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[35] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[36] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[37] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

[38] Christina Curtis, Sohrab Shah, Suet-Feung Chin, Gulisa Turashvili, Oscar Rueda, Mark Dunning, Doug Speed, Andy Lynch, Shamith Samarajiwa, Yinyin Yuan, Stefan Gräf, Gavin Ha, Gholamreza Haffari, Ali Bashashati, Roslin Russell, Steven McKinney, Carlos Caldas, Samuel Aparicio, James Brenton, and Anne-Lise Børresen-Dale. The genomic and transcriptomic architecture of 2,000 breast tumors reveals novel subgroups. *Nature*, 486:–, 04 2012.

[39] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[40] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Chem. Biol. Drug Des.*, 297:273–297, 01 2009.

[41] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 01 2002.

[42] Nikola Simidjievski, Cristian Bodnar, Ifrah Tariq, Paul Scherer, Helena Andres Terre, Zohreh Shams, Mateja Jamnik, and Pietro Lio. Variational autoencoders for cancer data integration: Design principles and computational practice. *Frontiers in Genetics*, 10:1205, 12 2019.

[43] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[45] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models, 2021.

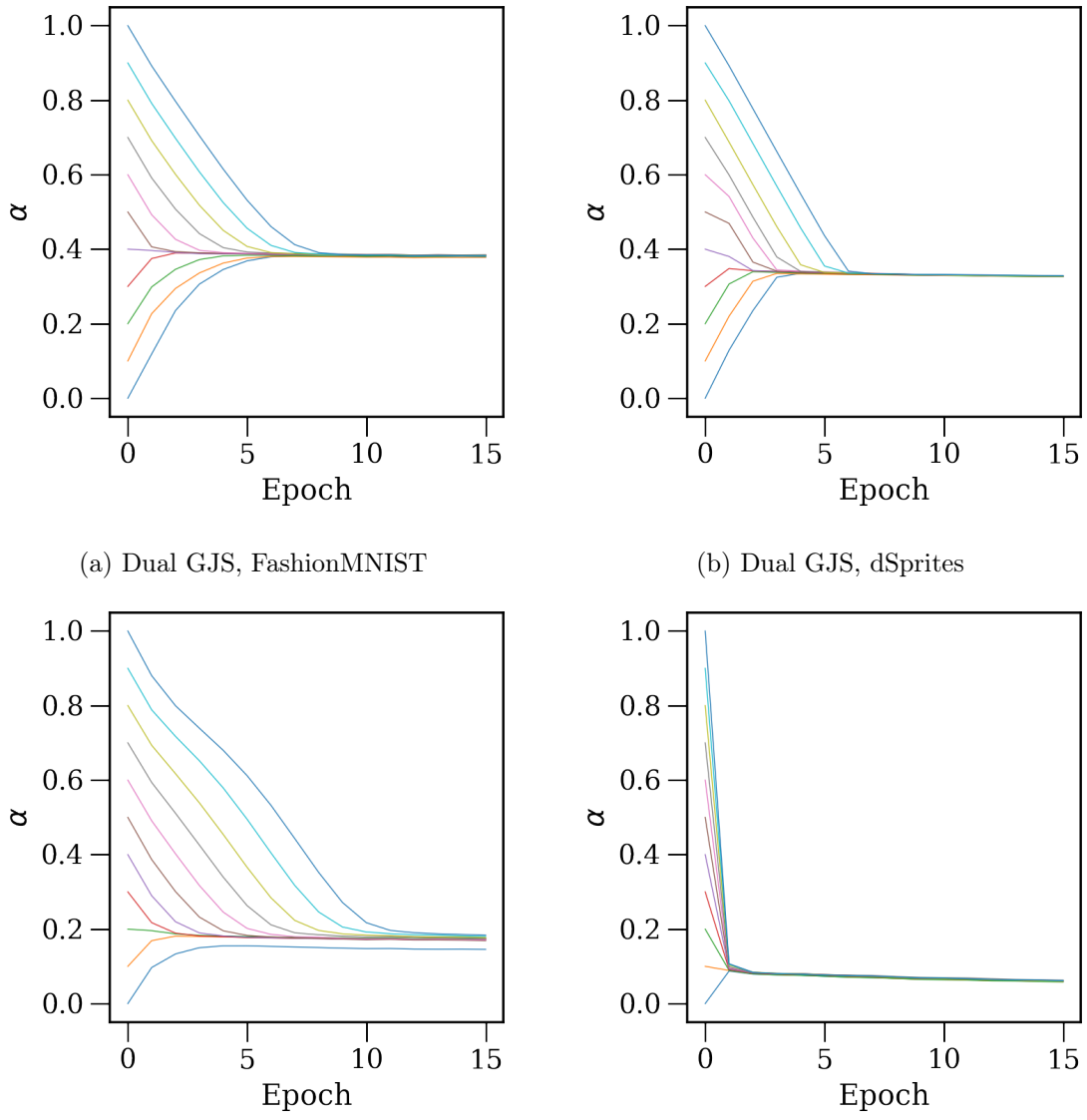
[46] Rob Brekelmans, Vaden Masrani, Frank Wood, Greg Ver Steeg, and Aram Galstyan. All in the exponential family: Bregman duality in thermodynamic variational inference, 2020.

[47] Danilo Jimenez Rezende and Fabio Viola. Taming vaes. 2018.

[48] Alexander A. Alemi, Ben Poole, Ian Fischer, Joshua V. Dillon, Rif A. Saurous, and Kevin Murphy. Fixing a broken elbo. 2018.

A Appendix

Figures 9 and 10 show the progression of alpha during training for the METABRIC integration tasks using a VAE with the X and CNC structures respectively.



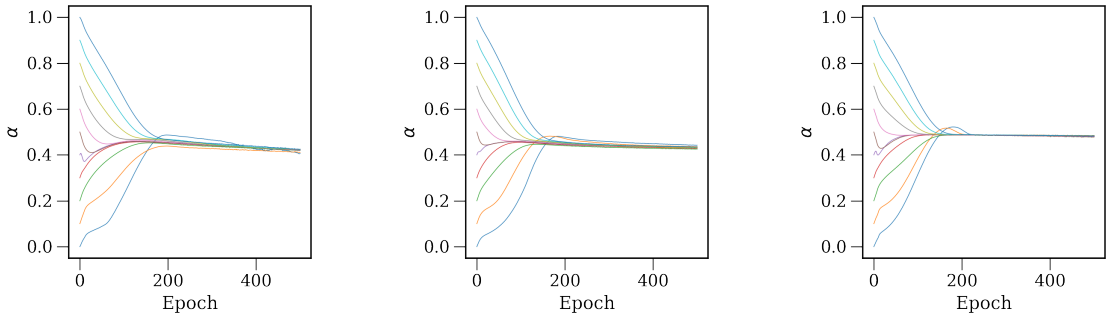
(a) Dual GJS, FashionMNIST

(b) Dual GJS, dSprites

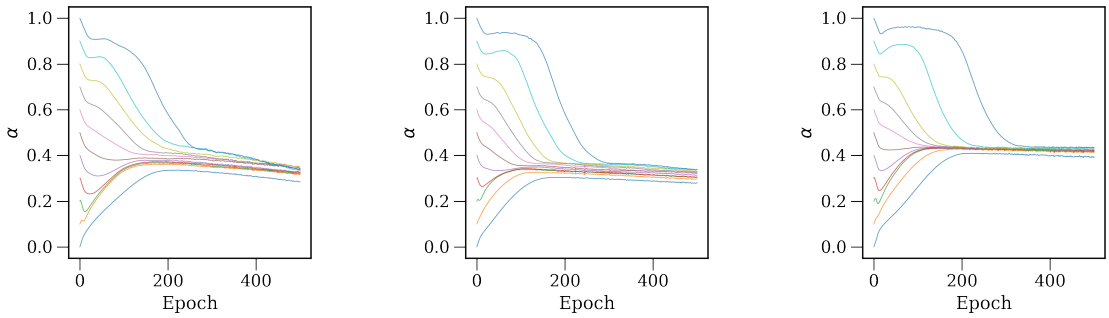
(c) GJS, FashionMNIST

(d) GJS, dSprites

Figure 8: Alpha convergence on FashionMNIST and dSprites.

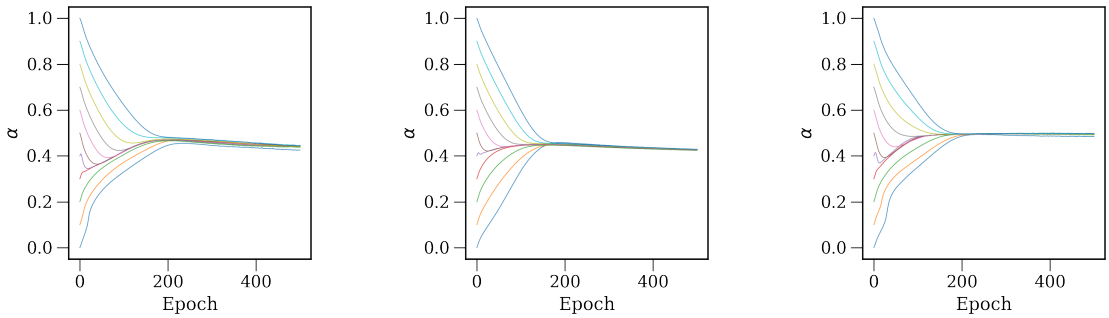


(a) Dual GJS, rna+cna (b) Dual GJS, clin+cna (c) Dual GJS, clin+rna

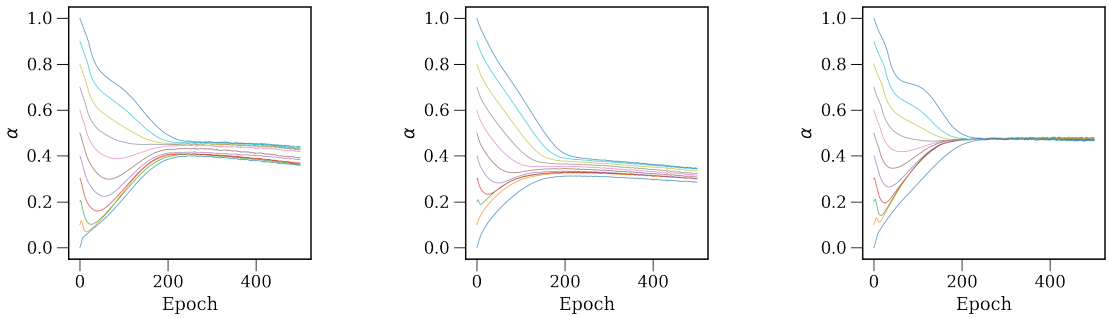


(d) GJS, rna+cna (e) GJS, clin+cna (f) GJS, clin+rna

Figure 9: Alpha convergence on METABRIC integration task using X VAE structure



(a) Dual GJS, rna+cna (b) Dual GJS, clin+cna (c) Dual GJS, clin+rna



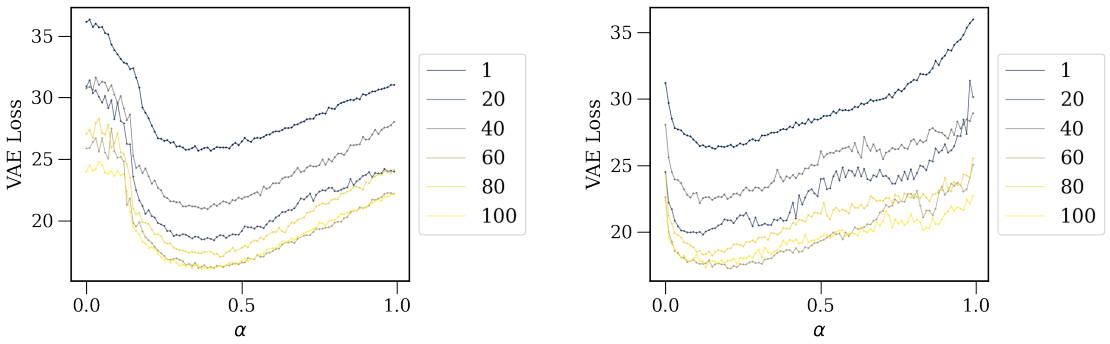
(d) GJS, rna+cna (e) GJS, clin+cna (f) GJS, clin+rna

Figure 10: Alpha convergence on METABRIC integration task using CNC VAE structure

B Appendix

Figure 11 shows the total VAE with respect to α, when training with constant alpha, for FashionMNIST. Also included here are plots of the components of the VAE loss when training with constant α - figures 12 and 13 show divergence with respect to constant alpha, and figures 12 and 13 show reconstruction. All plots are labelled with the epoch in training at which the data was recorded.

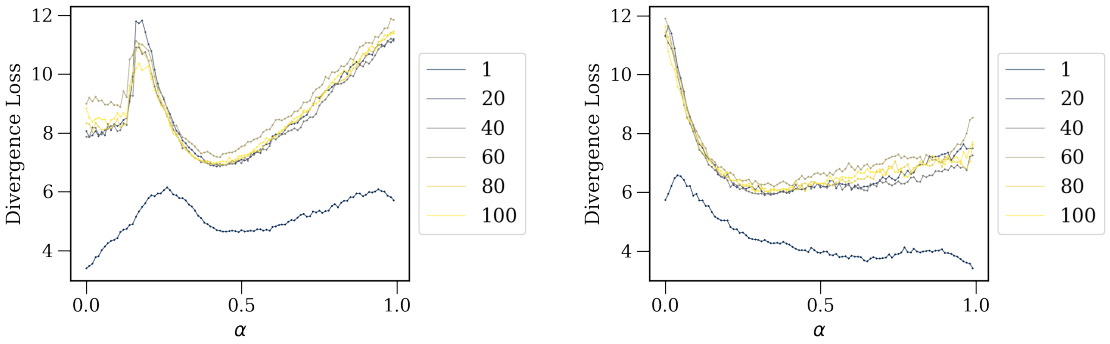
Notably, it is not clear why these plots have the exact form observed. Particularly interesting is the plots of dual GJS divergence and dual GJS reconstruction which show a peak and discontinuity, respectively, around $\alpha = 0.2$.



(a) Dual GJS, FashionMNIST

(b) GJS, FashionMNIST

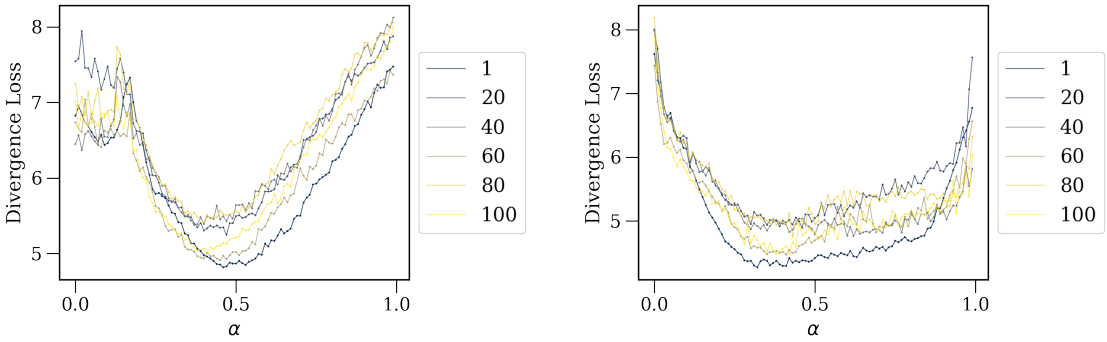
Figure 11: Total VAE loss versus skew during training of GJS VAE with constant alpha, for FashionMNIST.



(a) Dual GJS, MNIST

(b) GJS, MNIST

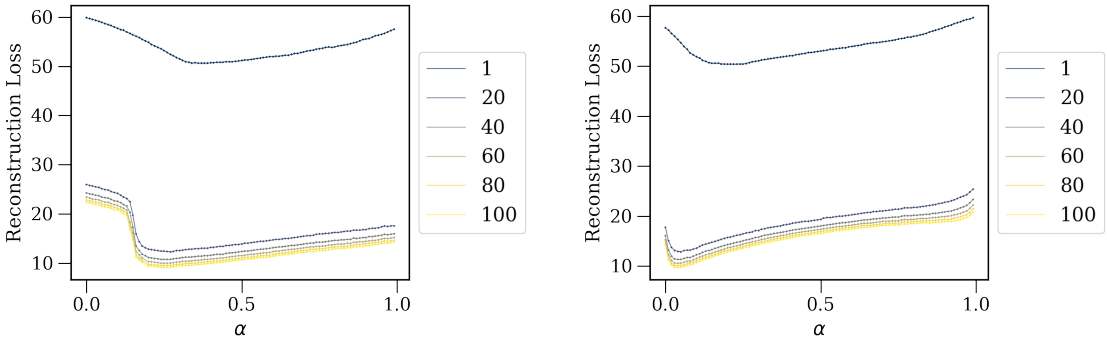
Figure 12: Divergence loss versus skew during training of GJS VAE with constant alpha, for MNIST.



(a) Dual GJS, FashionMNIST

(b) GJS, FashionMNIST

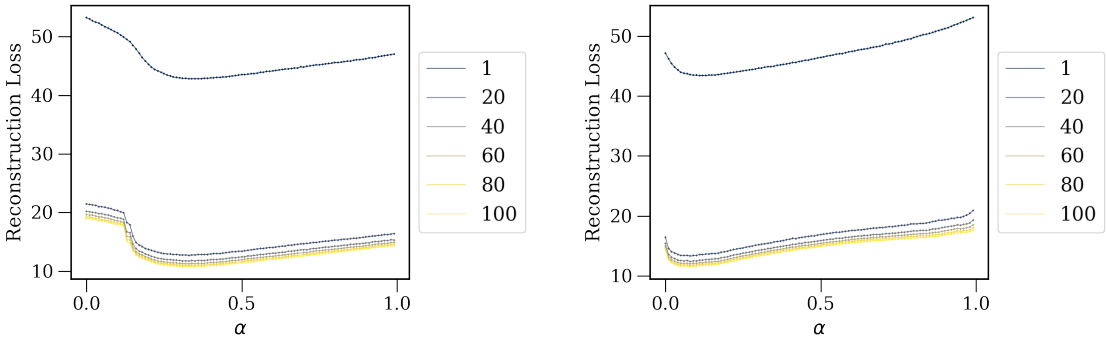
Figure 13: Divergence loss versus skew during training of GJS VAE with constant alpha, for FashionMNIST.



(a) Dual GJS, MNIST

(b) GJS, MNIST

Figure 14: Reconstruction loss versus skew during training of GJS VAE with constant alpha, for MNIST.



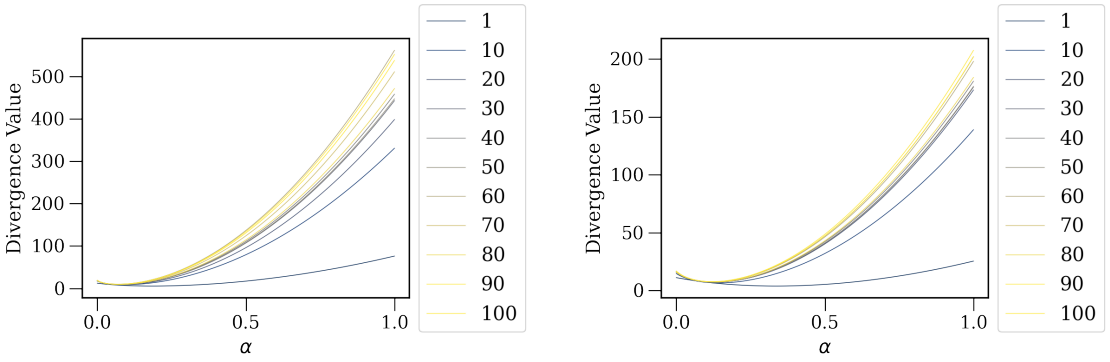
(a) Dual GJS, FashionMNIST

(b) GJS, FashionMNIST

Figure 15: Reconstruction loss versus skew during training of GJS VAE with constant alpha, for FashionMNIST.

C Appendix

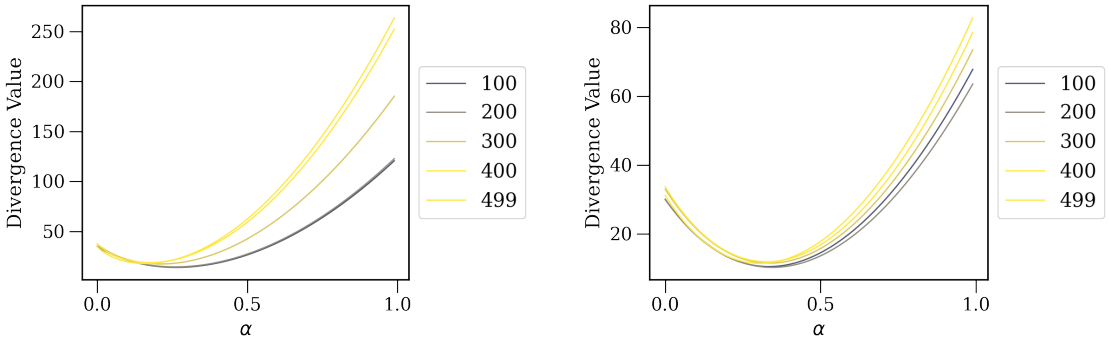
This appendix contains plots of GJS divergence value vs skew which were recorded during training with learning alpha. For each epoch, the value of the divergence was recorded by sweeping through alpha and simply measuring the GJS value.



(a) Dual GJS, MNIST

(b) GJS, MNIST

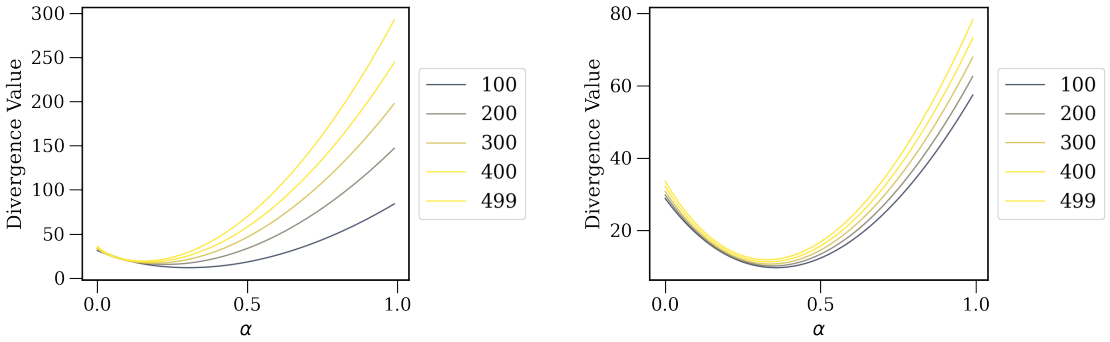
Figure 16: Divergence value vs skew during training, for MNIST.



(a) Dual GJS

(b) GJS

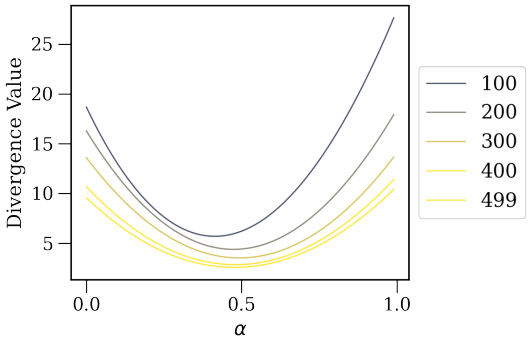
Figure 17: Divergence vs skew, clin + CNA, CNC VAE.



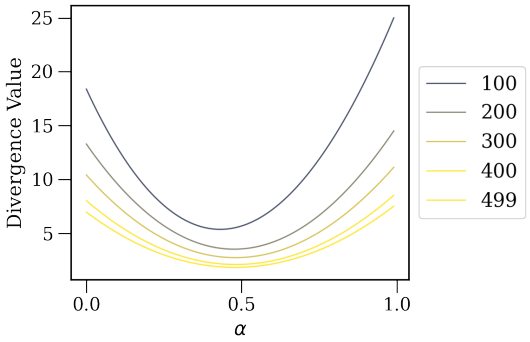
(a) Dual GJS

(b) GJS

Figure 18: Divergence vs skew, clin + CNA, X VAE.

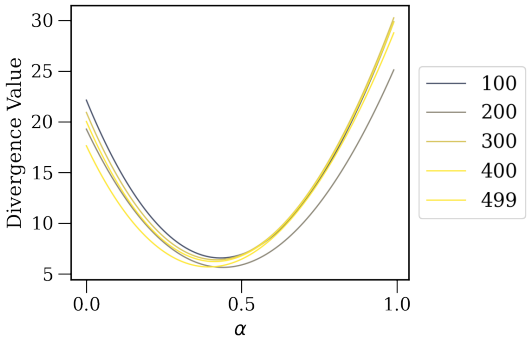


(a) Dual GJS

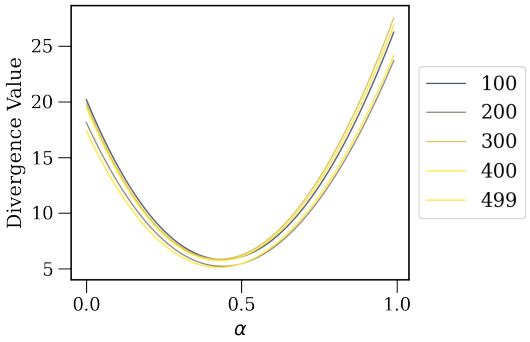


(b) GJS

Figure 19: Divergence vs skew, clin + RNA, CNC VAE.

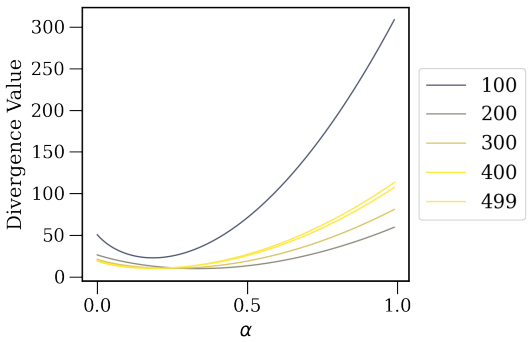


(a) Dual GJS

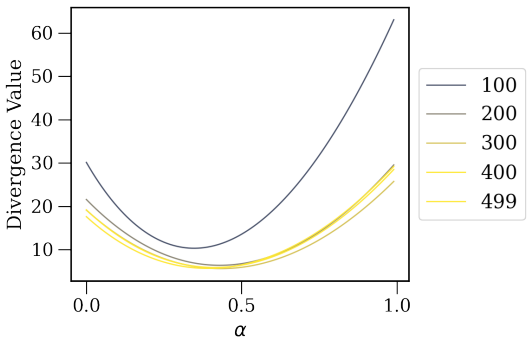


(b) GJS

Figure 20: Divergence vs skew, clin + RNA, X VAE.

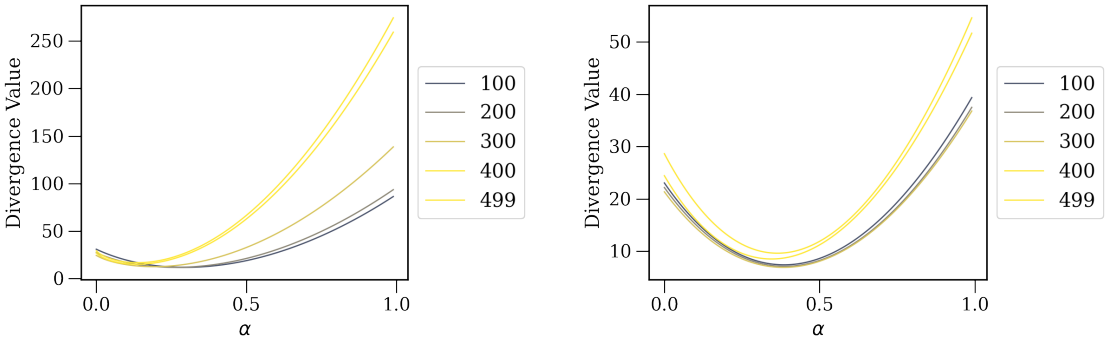


(a) Dual GJS



(b) GJS

Figure 21: Divergence vs skew, RNA + CNA, CNC VAE.

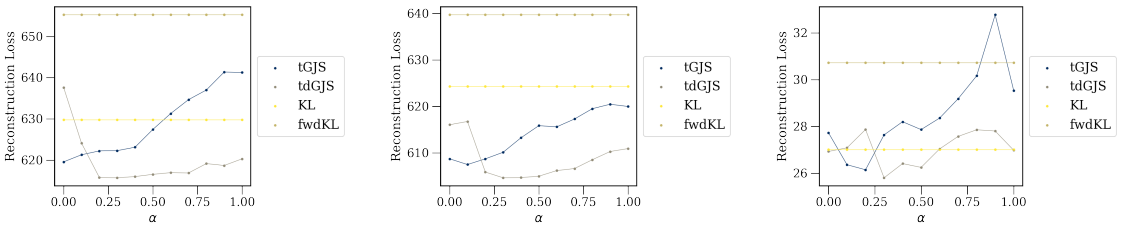


(a) Dual GJS

(b) GJS

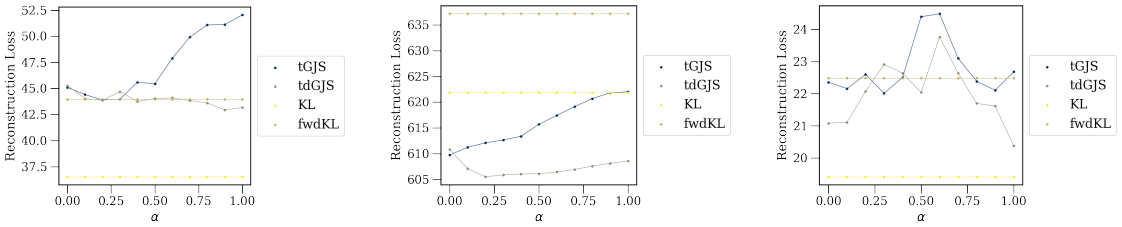
Figure 22: Divergence vs skew, RNA + CNA, X VAE.

D Appendix



(a) rna+cna (b) clin+cna (c) clin+rna

Figure 23: Reconstruction on METABRIC integration tasks with X VAE.



(a) rna+cna (b) clin+cna (c) clin+rna

Figure 24: Reconstruction on METABRIC integration tasks with CNC VAE.