Extreme deconvolution reimagined: conditional densities via neural networks and an application in quasar classification

Kang, Y.; Hennawi, J.F.; Schindler, J.-T.; Tamanas, J.; Nanni, R.

Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **541,** 2815–2829 (2025) https://doi.org/10.1093/mnras/staf1068
Advance Access publication 2025 July 26

Extreme deconvolution reimagined: conditional densities via neural networks and an application in quasar classification

Yi Kang,[1,2⋆] Joseph F. Hennawi ,[1,2] Jan-Torge Schindler,[3] John Tamanas[4] and Riccardo Nanni [2]

[1] *Leiden Observatory, Niels Bohrweg 2, Leiden, NL-2333 CA, the Netherlands*
[2] *Department of Physics, University of California, Santa Barbara, CA 93106, USA*
[3] *Hamburg Observatory, Gojenbergsweg 112, D-21029 Hamburg, Germany*
[4] *The Department of Astronomy and Astrophysics, University of California Santa Cruz, 1156 High Street, Santa Cruz, CA 95064, USA*

Accepted 2025 June 27. Received 2025 June 27; in original form 2024 December 4

ABSTRACT

Density estimation is a fundamental problem that arises in many areas of astronomy, with applications such as selecting quasars via colour distributions and characterizing stellar abundances. Astronomical observations are inevitably noisy, while the density of a noise-free feature is often the desired outcome. The extreme-deconvolution (XD) method can be used to deconvolve the noise and estimate the underlying density distribution by fitting a mixture of Gaussians to data with heteroscedastic Gaussian noise. However, XD does not generalize to cases where some feature dimensions have distributions far away from Gaussian, and no established method exists to overcome this limitation. Requiring negligible noise in these non-Gaussian features, we introduce a possible solution that separates out the non-Gaussian features and models the Gaussian-like dimensions conditioned on the non-Gaussian features using a neural network and Gaussian mixture model. The result is the CondXD algorithm, a generalization of XD that takes in the non-Gaussian features and outputs the deconvolved conditional distribution of the Gaussian-like features on the input features. We apply CondXD to a toy model, and compare it with an existing method that divides the samples into bins of conditioning variables and applies XD separately to each bin. We find that CondXD is more accurate than the classical approach. We further test CondXD on a real-world high-redshift quasar versus contaminant classification problem. It achieves comparable results to the binning method but is roughly 10 times faster. Overall, our method has the potential to significantly improve the deconvolution of non-Gaussian distributions and enable new discoveries in astronomy.

Key words: methods: statistical – methods: data analysis – quasars: general.

1 INTRODUCTION

Density distribution estimation is an active area of research in astronomy, with key attention paid to uncovering the underlying distributions of various astronomical properties. For example, Buder et al. (2022) used deconvolution techniques to estimate the distribution of the abundances of accreted stars, while Mortlock et al. (2011a), Bovy et al. (2011b, 2012), and Nanni et al. (2022) et al. applied similar methods to measure the flux distribution of quasars. Other researchers, such as Bhave et al. (2022), Reddy Ch. & Desai (2022), and Arumugam & Desai (2023), have used deconvolution techniques to model the distributions of transients like pulsars and gamma-ray bursts. Moreover, Bird et al. (2021) and Ivezić & Ivezić (2021) have used density distribution estimation to infer the galactic structure and predict galaxy sizes measured by the Rubin Observatory (Ivezić et al. 2019), respectively.

In practice, the physical attributes of astronomical targets are rarely measured without substantial and heteroscedastic uncertainties, and the estimation of the underlying distribution is never an easy task. The observations can be regarded as samples drawn from a (noiseless) underlying distribution convolved with the distribution of noise; thus the estimation of the underlying distribution is also referred to as deconvolution. While density deconvolution of noisy distributions has been extensively studied in the literature, early works such as Devroye (1989), Stefanski & Carroll (1990), Zhang (1990), and Fan (1991a,b) often assumed that the distributions are univariate and the noise distribution is identical for every measurement, neglecting the heteroscedasticity of the measurements. Moreover, most of the early studies applied non-parametric approaches that cannot be implemented when samples with missing measurements (missing data) are encountered.

To address these complications, Bovy, Hogg & Roweis (2011a) developed the extreme-deconvolution (XD) algorithm that works for data with heterogeneous noise—even accommodating missing data.[1] They used a Gaussian mixture model (GMM) to fit the underlying distribution of a set of noisy samples. With the assumption of Gaussian underlying distribution and Gaussian noise distribution with zero mean, the noisy density distribution (i.e. convolution of the underlying and noise distribution) can be obtained efficiently [see equation (6) in Section 2.2]. Iteratively applying the expectation

⋆ E-mail: yi_kang@physics.ucsb.edu

[1] https://github.com/jobovy/extreme-deconvolution

and maximization process to increase the likelihood of the noisy samples to the noisy distribution, the underlying distribution is estimated. As demonstrated in Bovy et al. (2011a), XD is capable of inferring the 3D velocity distribution of stars around the Sun given the noisy 2D transverse velocity measurements from the *Hipparcos* satellite. It is also resistant to poor initialization to obtain the optimal fit. Later, the scalable XD algorithm developed by Ritchie & Murray (2019) improved the XD code with modern machine-learning algorithms (e.g. stochastic gradient descent and mini-batches) to seek for the GMM best-fitting parameters, instead of using an iterative expectation–maximization approach on the full data set. Similar studies were also conducted by Hosseini & Sra (2015, 2020) and Gepperth & Pfülb (2019).

Subsequent to its publication, XD has gained wide applications, particularly in the field of classifying quasars and contaminants based on their photometric multiband fluxes (e.g. Bovy et al. 2011b, 2012; White et al. 2012; DiPompeo et al. 2015; Myers et al. 2015; Nanni et al. 2022). In some cases, however, the joint distributions of the band fluxes cannot be easily deconvolved by XD. For example, Nanni et al. (2022) found that the probability density of quasars observed by *JWST* and *Euclid* will have a dominant power-law shape corresponding to the number counts as a function of the J-band magnitude, which is hard to approximate by combining a small number of Gaussian distributions. As an alternative, they tried to separate the J-band flux, and deconvolved the joint distribution of the other band fluxes (relative to the J band) conditioning on the J-band flux with XD. This idea is based on the definition of conditional distribution:

$$p(\boldsymbol{x}_1,\ \boldsymbol{x}_2) = p(\boldsymbol{x}_1|\boldsymbol{x}_2)p(\boldsymbol{x}_2), \tag{1}$$

where $p(\boldsymbol{x}_1,\ \boldsymbol{x}_2)$ is the joint distribution of two sets of features $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, $p(\boldsymbol{x}_1|\boldsymbol{x}_2)$ is the distribution of $\boldsymbol{x}_1$ conditioning on $\boldsymbol{x}_2$, and $p(\boldsymbol{x}_2)$ is the marginal distribution for $\boldsymbol{x}_2$. Nanni et al. (2022) found that their conditional joint distribution has no power-law-shaped marginal distribution and can be well described by GMM across their J-band flux range. Moreover, their J-band fluxes have significant signal-to-noise ratio, such that the noise is negligible and binning data based on the observed J-band flux is reliable. Consequently, the joint distribution for all bands can be obtained by 1) grouping the data into bins of the observed J-band fluxes, and deconvolving the joint distribution of the other band fluxes with XD in each bin; 2) deriving the marginal distribution of the J-band flux by estimating the J-band magnitude distribution with a power-law function; and 3) multiplying them according to equation (1). Similar separation and binning approaches are also employed in Bovy et al. (2011b, 2012), Bird et al. (2021), all noticing that the samples can only be well modelled by GMM when conditioning on the non-Gaussian features.

However, there are several major drawbacks to this binning approach. First, the bin size must be small to reduce the variation of the conditional distribution, but also large enough to include sufficient samples in each bin for accurate estimation. Unfortunately, there is no objective method to reconcile this trade-off and decide on the bin size. Secondly, the continuity of the estimated distribution is not guaranteed among the bins since the XD is applied individually. Nanni et al. (2022) implemented complicated strategies to mitigate these problems, resulting in inefficient algorithms. More details are discussed in Section 4.

To address these issues, we propose using a mixture density network (section 5.6 in Bishop 2006) to deconvolve the conditional distributions [the first factor on the right-hand side of equation (1)] under the assumption that the noise is negligible for the observed conditioning variables. A mixture density network can fit a conditional distribution with a density mixture, i.e. a combined set of weighted basic density distributions such as GMM. It allows the parameters of the density mixture (e.g. for GMM, mixing coefficients,[2] means, and covariance matrices) to be generated by a neural network (NN) that takes the conditioning variables as the input. In other words, once trained, the mixture density network can take in the conditioning variable and output the joint distribution of the other features conditioning on the input value of the conditioning variable. This motivates us to combine the XD and mixture density networks in order to deconvolve the conditional distributions of astronomical sources with heteroscedastic noise.

In this paper, following Ritchie & Murray (2019), we use modern machine-learning methods to deconvolve noisy conditional distributions for both a simple toy model and an astronomical real case classification problem, demonstrating the capabilities of our conditional XD algorithm, CondXD. In Section 2 we provide a general description of the CondXD method. In Section 3 we conduct an experiment to test the performance of CondXD on a simple toy model. Using the same toy model, in Section 4 we compare the deconvolving capability of both CondXD and a binning approach similar to the one from Nanni et al. (2022). In Section 5 we apply CondXD to a realistic astronomy case and compare the results with the binning method from Nanni et al. (2022). In Section 6 we provide our conclusions and discussions.

2 METHOD

In its standard form, XD estimates the underlying probability distribution $p(\mathbf{X})$ of a noisy multidimensional sample $\mathbf{X}$ using a GMM $\hat{p}(\mathbf{X} \mid \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\mu}}, \hat{\mathbf{V}})$, defined by a set of K mixing coefficients $\hat{\boldsymbol{\alpha}}$, means $\hat{\boldsymbol{\mu}}$, and covariance matrices $\hat{\mathbf{V}}$. We use hat notation to indicate estimated statistics and leave out the notation to represent the true values of these quantities when they are generated from an underlying Gaussian mixture. However, if the probability density, p, is conditioned on a variable $\boldsymbol{c}$ such that $p(\mathbf{X}|\boldsymbol{c})$, the correct form for the estimator is $\hat{p}(\mathbf{X} \mid \hat{\boldsymbol{\alpha}}(\boldsymbol{c}), \hat{\boldsymbol{\mu}}(\boldsymbol{c})), \hat{\mathbf{V}}(\boldsymbol{c}))$. Here the conditioning variable $\boldsymbol{c}$ is bold, since it can also be multidimensional, i.e. the model can depend on multiple variables. Furthermore, we only consider noiseless $\boldsymbol{c}$ to simplify our theory, which requires the noise of the conditioning features of observations to be negligible.

In this work, we build our conditional GMM estimator using an NN with weights $\boldsymbol{\phi}$. In this case, the GMM parameters become functions of $\boldsymbol{\phi}$ and the conditioning variable $\boldsymbol{c}$, written as: $\hat{\boldsymbol{\alpha}}(\boldsymbol{\phi}, \boldsymbol{c})$, $\hat{\boldsymbol{\mu}}(\boldsymbol{\phi}, \boldsymbol{c})$, $\hat{\mathbf{V}}(\boldsymbol{\phi}, \boldsymbol{c})$. Consequently, for notational simplicity we henceforth express the estimator as $\hat{p}(\mathbf{X} \mid \boldsymbol{\phi}, \boldsymbol{c})$.

In Section 2.1 we describe the architecture of our NN, while in Section 2.2 we describe how we define the loss function of our method; Section 3.3 introduces the technical details implemented to improve training. Hereafter, we call our technique CondXD, the code for which is available on GitHub.[3]

2.1 Architecture of the neural network

Our NN has a stem–branch structure, shown in Fig. 1. The stem constitutes a sequence of three linear layers, which branches off into three output layers for the mixing coefficients, means, and

[2] They are more often referred as 'weights', but we use the term 'mixing coefficients' to avoid confusion with the 'weights' of neural networks.

[3] https://github.com/enigma-igm/CondXD

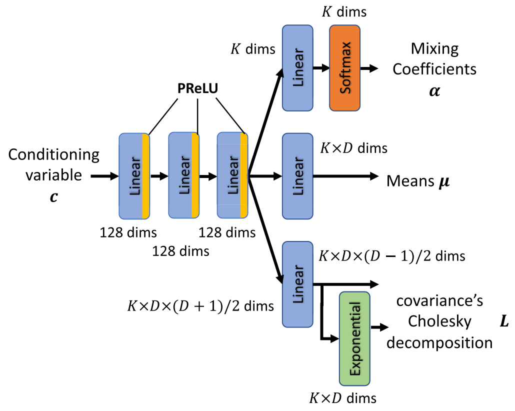

Figure 1. Schematic of the CondXD neural network. It takes in conditioning variable $\boldsymbol{c}$ and outputs the parameters of a GMM, i.e. the mixing coefficients, means, and Cholesky factors of the covariance matrices. Blocks are layers in the architecture, the types of which are indicated by annotations. The yellow ends refer to the `PReLU` activation functions after the current layer. K is the number of Gaussians in the model, and D is the dimension of the Gaussians; both are hyperparameters. In practice, D is automatically determined by the dimension of the samples. The dimensions of the outputs computed by every layer are also labelled.

covariances, respectively. It takes in the conditioning variable $\boldsymbol{c}$ and outputs the parameters of the GMM. In general, the density distribution of a noisy sample can depend on several variables, so that the conditioning variable $\boldsymbol{c}$ is actually multidimensional. The number of Gaussians of the GMM (K) is a hyperparameter, and the dimension of the GMM (D) is determined by the dimension of the data from the observations, whose density is to be estimated.

In the structure of the NN we simply use `Linear` layers everywhere. The `Linear` layers multiply the input with matrices, the elements of which are the weights $\boldsymbol{\phi}$ of our neural network. In the stem part, all the `Linear` layers are followed by the `PReLU` (parametric rectified linear unit; He et al. 2015) activation functions. At the output layer for the mixing coefficients branch (see orange block in Fig. 1) we follow standard practice, using a softmax activation to ensure that all mixing coefficients are positive and sum up to unity:

$$\hat{\alpha}_j = \frac{\exp(\beta_j)}{\sum_j \exp(\beta_j)}, \tag{2}$$

where $\hat{\alpha}_j$ is the mixing coefficient for the jth Gaussian, and β_j is the jth output of the last `Linear` layer of the mixing coefficients branch. For the means branch we do not implement any further processing than a `Linear` layer, since there is no rigorous requirement on it. Finally, instead of directly generating the covariances that have to satisfy symmetry and positive semidefiniteness, we again follow standard practice, generating the Cholesky decomposition factors $\hat{\mathbf{L}}(\boldsymbol{\phi}, \boldsymbol{c})$. The Cholesky decomposition is defined as:

$$\hat{\mathbf{V}}_j = \hat{\mathbf{L}}_j \hat{\mathbf{L}}_j^{\mathrm{T}} + 10^{-4} I, \tag{3}$$

where $\hat{\mathbf{L}}_j$ is the jth Cholesky factor, which is a lower triangular matrix with shape (D, D), $\hat{\mathbf{V}}_j$ is the jth covariance matrix of the K Gaussians, and I is the identity matrix. Each of the K Cholesky factors has all zero values in the upper right-hand triangle, and all diagonal elements are positive for every matrix. Moreover, a small positive value of 10^{-4} is added to the diagonal. In this way $\hat{\mathbf{V}}_j$ is guaranteed to be symmetric and positive semidefinite. In the covariance branch (see the green block in Fig. 1), the diagonal elements of every Cholesky factor have been processed by an exponential activation function to ensure positivity.

With this architecture we generate all the GMM parameters. In practice, these parameters define the model that describes the deconvolved density distribution of the noisy samples.

2.2 Loss function

To train the NN, or, in other words, to find the best-fitting weights $\boldsymbol{\phi}$ in the NN, we need to quantify how well the GMM represents the data samples by utilizing a loss function. In our case, the Kullback–Leibler divergence (KL divergence or D_{KL}; Kullback & Leibler 1951) is chosen as a standard practice to measure how different the GMM is from the underlying distribution. The KL divergence is defined as:

$$\begin{aligned} D_{\mathrm{KL}}\left(p \| \hat{p}\right) &= \int p(\boldsymbol{x} \mid \boldsymbol{c}; \mathbf{S}(\boldsymbol{x})) \ln \left(\frac{p(\boldsymbol{x} \mid \boldsymbol{c}; \mathbf{S}(\boldsymbol{x}))}{\hat{p}(\boldsymbol{x} \mid \boldsymbol{\phi}; \boldsymbol{c}; \mathbf{S}(\boldsymbol{x}))} \right) \mathrm{d}\boldsymbol{x} \\ &= \int p(\boldsymbol{x} \mid \boldsymbol{c}; \mathbf{S}(\boldsymbol{x})) \ln p(\boldsymbol{x} \mid \boldsymbol{c}; \mathbf{S}(\boldsymbol{x}))\, \mathrm{d}\boldsymbol{x} \\ &\quad - \int p(\boldsymbol{x} \mid \boldsymbol{c}; \mathbf{S}(\boldsymbol{x})) \ln \hat{p}(\boldsymbol{x} \mid \boldsymbol{\phi}; \boldsymbol{c}; \mathbf{S}(\boldsymbol{x}))\, \mathrm{d}\boldsymbol{x}, \end{aligned} \tag{4}$$

where $\mathbf{S}(\boldsymbol{x})$ is the noise covariance of the random sample $\boldsymbol{x}$, shaped (D, D). $\mathbf{S}$ is always a diagonal matrix in our case assuming independent noise. With the heteroscedastic assumption, each $\boldsymbol{x}_i$ can have its own noise; thus $\mathbf{S}$ can also be regarded as a function of $\boldsymbol{x}$, i.e. $\mathbf{S}(\boldsymbol{x})$. In practice, observed data are always noisy; thus both distributions p and $\hat{p}$ in equation (4) have been convolved with noise. In the above equation, $p(\boldsymbol{x} \mid \boldsymbol{c}; \mathbf{S}(\boldsymbol{x}))$ is short for $p(\mathbf{X} = \boldsymbol{x} \mid \boldsymbol{c}; \mathbf{S}(\boldsymbol{x}))$, and $p(\mathbf{X} \mid \boldsymbol{c}; \mathbf{S}(\boldsymbol{x}))$ is the noise-convolved underlying density $p(\mathbf{X} \mid \boldsymbol{c})$. Similarly, $\hat{p}(\boldsymbol{x} \mid \boldsymbol{\phi}; \boldsymbol{c}; \mathbf{S}(\boldsymbol{x}))$ is the probability of $\boldsymbol{x}$ under the noise-convolved GMM estimator for $p(\mathbf{X} \mid \boldsymbol{c}; \mathbf{S}(\boldsymbol{x}))$. As the samples $\boldsymbol{x}$ are from the noise-convolved distribution $p(\mathbf{X} \mid \boldsymbol{c}; \mathbf{S}(\boldsymbol{x}))$, the integration in equation (4) is averaged over the sample space.

Our goal is to find the model $\hat{p}(\mathbf{X}|\boldsymbol{\phi}; \boldsymbol{c}; \mathbf{S}(\boldsymbol{x}))$ that minimizes the KL divergence validated on the observation data $\{\boldsymbol{x}, \boldsymbol{c}, \mathbf{S}\}$. The first term in the second line of equation (4) is a constant that does not depend on the NN weights $\boldsymbol{\phi}$; therefore only the second term needs to be minimized. Therefore, we can define the loss function as the second term, i.e. the negative of the log-probability of the model averaged over the underlying distribution. Since we do not have access to the noise-convolved underlying distribution $p(\mathbf{X} \mid \boldsymbol{c}; \mathbf{S})$ (this is what we are trying to estimate), but we do have access to noisy data $\{\boldsymbol{x}, \mathbf{S}\}$, we rewrite the second term in equation (4) as a Monte Carlo integral:

$$\mathrm{loss}_{\mathrm{NN}}(\boldsymbol{\phi} \mid \{\boldsymbol{x}, \boldsymbol{c}, \mathbf{S}\}) = -\frac{1}{N} \sum_{i=1}^{N} \ln \hat{p}(\boldsymbol{x}_i \mid \boldsymbol{\phi}; \boldsymbol{c}_i; \mathbf{S}_i), \tag{5}$$

where N is the data sample size. Minimizing the loss in equation (5) is equivalent to finding the parameters $\boldsymbol{\phi}$ that maximize the probability of the data $\{\boldsymbol{x}\}$ given the corresponding conditionals $\{\boldsymbol{c}\}$ and the noise covariances $\{\mathbf{S}\}$.

To evaluate the probability of noisy data $\boldsymbol{x}_i$, we need to convolve the GMM with the noise probability distribution. Assuming that the noise $\boldsymbol{\epsilon}$ has a Gaussian distribution $\mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{S}_i)$, the convolution is trivial, and is simply the sum of $\mathbf{S}_i$ and every covariance $\hat{\mathbf{V}}_j$, due to the closure of the Gaussian distribution under convolutions. The model probability can thus be evaluated via

$$\begin{aligned} \hat{p}(\boldsymbol{x}_i \mid \boldsymbol{\phi}; \boldsymbol{c}_i; \mathbf{S}_i) = &\sum_{j=1}^{K} \hat{\boldsymbol{\alpha}}_j(\boldsymbol{\phi}; \boldsymbol{c}_i) \mathcal{N} \\ &\times \left(\boldsymbol{x}_i \mid \hat{\boldsymbol{\mu}}_j(\boldsymbol{\phi}; \boldsymbol{c}_i), \hat{\mathbf{V}}_j(\boldsymbol{\phi}; \boldsymbol{c}_i) + \mathbf{S}_i\right), \end{aligned} \tag{6}$$

at any noisy data given its location $\boldsymbol{x}_i$, conditioning variable $\boldsymbol{c}_i$, and noise covariance $\mathbf{S}_i$, where $\hat{\boldsymbol{\alpha}}_j$, $\hat{\boldsymbol{\mu}}_j$, and $\hat{\mathbf{V}}_j$ are the mixing coefficients, mean, and covariance of the jth Gaussian.

An issue that can arise during the optimization of the loss in equation (5) is that a Gaussian in the mixture can collapse on to a single sample. This will reduce the estimated covariance to nearly zero. Although a finite positive value [equation (3)] and a finite noise covariance are added to the diagonal of the covariance, a large number of Gaussian (like $K = 20$) components makes it possible that a certain component will focus on some outliers. This is clearly undesirable and does not represent a viable optimum. We regularize the loss by adding an additional term that amounts to a penalty when the covariance diagonal elements approach zero:

$$\mathrm{loss}_{\mathrm{reg}} = w \sum_j \sum_i \frac{1}{\mathrm{diag}(\mathbf{V}_j)_i}, \tag{7}$$

where w is a tunable parameter that is determined at 10^{-6} via trial and error, and $\mathrm{diag}(\mathbf{V}_j)_i$ is the ith diagonal element of the jth covariance. As we have forced all covariance diagonals to be positive, this regularization loss is also always positive but dominates only if the diagonal elements approach zero. The total loss is then the sum of the regularization loss and model loss:

$$\mathrm{loss} = \mathrm{loss}_{\mathrm{NN}} + \mathrm{loss}_{\mathrm{reg}}, \tag{8}$$

which is what we try to reduce for the samples.

2.3 Training strategies

After defining the NN architecture and the loss function, we now turn to the specific strategies implemented to train our NN.

Before training, we normalize the data in every dimension. Normalization ensures that each feature contributes equally during training and prevents vanishing or exploding gradients to stabilize convergence. Additionally, normalization generalizes our method to different applications with contrasting orders of magnitude. Specifically, we compute the mean and standard deviation of each feature of all data samples (including training, validation, and test sets; see later in this section), subtract the mean from each piece of data $\boldsymbol{x}_i$, and divide each piece of data by the standard deviation:

$$(x_{i,\mathrm{n}})_j = [(x_i)_j - m_j]/\sigma_j. \tag{9}$$

In the above equation, $(x_i)_j$ and $(x_{i,\mathrm{n}})_j$ are the jth feature of the ith piece of data before and after normalization respectively, m_j is the mean value of the jth feature over all the N samples (before separating into training/validation/test sets; see the following paragraphs), and σ_j is the standard deviation of the jth feature of all the N samples. The means and standard deviations are stored to later rescale the CondXD predicted samples after training. We proceed by dividing the noise covariance of each by the outer product of the standard deviation vector $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_D)$:

$$(\mathbf{S}_{i,\mathrm{n}})_{jk} = (\mathbf{S}_i)_{jk}/\sigma_j\sigma_k, \tag{10}$$

where $(\mathbf{S}_i)_{jk}$ and $(\mathbf{S}_{i,\mathrm{n}})_{jk}$ are the jth row and kth column of the noise covariance matrix of the ith piece of data. In this way, we are in fact deconvolving the normalized distribution; thus all the x_i and $\mathbf{S}_i$ in equations (4)–(6) should be replaced by $x_{i,\mathrm{n}}$ and $\mathbf{S}_{i,\mathrm{n}}$ respectively. After training the NN, we transform the samples from the output joint distribution back to the original scale: the output samples are multiplied by the standard deviation vector $\boldsymbol{\sigma}$ and then added to the mean vector $\boldsymbol{m} = (m_1, ..., m_D)$. All subsequent plots have already been rescaled.

We use stochastic gradient descent (Robbins & Monro 1951; Kiefer & Wolfowitz 1952) with the `torch.optim.Adam` optimizer (Kingma & Ba 2014) in the PYTORCH PYTHON package (Paszke et al. 2019) to train our NN. This updates the weights of the NN in stochastic directions around the loss gradient during training. The stochasticity prevents it from having local optima provided with only a limited number of samples. Moreover, we utilize the weight decay method that introduces an additional loss term accounting for the sum of squares of the NN weights multiplying with a coefficient, equipped in the `Adam` optimizer. It penalizes large NN weight values and encourages some weights to be close to 0, i.e. to prefer a simple model and avoid overfitting. With trial and error on our toy model in Section 3.1, we achieve a coefficient value of 10^{-6}.

To further avoid overfitting, we randomly divide the $\{\boldsymbol{x}_\mathrm{n}, \boldsymbol{c}, \mathbf{S}_\mathrm{n}\}$ data triplets (the subscript n denotes 'normalized') into two sets: a training set and a validation set with ratio 90 per cent : 10 per cent. As long as the loss [equation (5)] of the validation set remains close to that of the training set, the model does not overfit the training set. In each set, the samples are further divided into mini-batches with size equal to 250 samples. This number is determined rather randomly from a typical value in the literature. Compared with the sample size of 90 000 in our toy model in Section 3 and 1902 071 in the quasar contaminants in Section 5, the mini-batch has a small size. It can be increased as long as one mini-batch still fits in the computer memory. The NN weights are updated by the `torch.optim.Adam` optimizer based on the loss and gradient computed on each mini-batch. After looping over all of the mini-batches, we compute the average training loss of the whole training set. We then compute the validation loss, which is the loss averaged over the entire validation set. An epoch is defined as the execution of stochastic gradient descent on all the training mini-batches plus the computation of the validation loss. The best model is defined as the one that achieves the lowest value of the validation loss after 100 epochs.

The learning rate is the step size by which $\boldsymbol{\phi}$ are updated when trained on each mini-batch. In the early stages, the learning rate should be large to speed up convergence, while later it should be small to allow $\boldsymbol{\phi}$ to converge on precise values. The `Adam` optimizer automatically decreases the learning rate, and we find that implementing an additional decrease results in faster convergence. We set the initial learning rate to 0.001 in the `Adam` optimizer, and decrease it further by multiplying by 0.4 every time when there is no decrease of the validation loss for two subsequent epochs. The latter is achieved with the `torch.optim.lr_scheduler.ReduceLROnPlateau` scheduler. These values are obtained after experiments on the toy model in Section 3.

Although these hyperparameters, i.e. weight decay coefficient, training and validation set relative size, mini-batch size, learning rate and its manual decay rate, are all determined on our models, they generally exhibit reliable deconvolution performance (for example, in the application to high-z quasar contaminants in Section 5). For specific tasks, the user can decide on customized values via methods such as grid search.

3 EXPERIMENTS ON A SIMULATED NOISY GMM

3.1 Constructing the GMM toy model

To test the performance of our CondXD method when estimating the underlying density given observations with heteroscedastic noise, we constructed a simple toy model using a GMM with $K = 10$ Gaussian components and $D = 7$ dimensions. To construct the model, we first generate the mixing coefficients $\boldsymbol{\alpha}$, means $\boldsymbol{\mu}$, and Cholesky factors $\mathbf{L}$ of the covariances as a function of the conditioning variable. However, for simplicity we only consider the case of a 1D conditioning variable c, although our method can be generalized to an N-dimensional conditioning variable.

The mixing coefficients vector $\boldsymbol{\alpha}$ is calculated using power-law functions and is integral to the generation of the Gaussian mixture. Specifically, each component of the Gaussian mixture's mixing coefficients, α_i, is computed as:

$$\begin{aligned}\alpha_{i,0} &= A^{1-i/10}c^{1+i/10}\\ \alpha_i &= \frac{\alpha_{i,0}}{\sum_i \alpha_{i,0}},\end{aligned} \tag{11}$$

where A is a number drawn from the uniform distribution in the range $[0,\ 2]$.[4] This sequence introduces sufficient randomness into the mixing coefficients calculation process while constraining the range of values. The formulation of the mixing coefficients ensures that each α_i varies distinctively with the conditioning variable while collectively summing to unity.

The means for our Gaussian components are generated similarly. We randomly draw $K \times D$ numbers from the uniform distribution in the range $[0,\ 10]$.[5] The $K \times D$ numbers are reshaped into a matrix $\mathbf{B}$ with shape $(K,\ D)$, and the means are computed as:

$$\boldsymbol{\mu} = (\mathbf{B} - \overline{\mathbf{B}}) \cdot c^{1.2}, \tag{12}$$

where $\overline{\mathbf{B}}$ denotes the average of all the elements in $\mathbf{B}$ over both dimensions. For simplicity we keep using a power-law behaviour on the conditional, and the exponent 1.2 is randomly chosen and is different from that of the mixing coefficients. By subtracting $\overline{\mathbf{B}}$ from $\mathbf{B}$ we effectively centre the elements of the means such that the Gaussian clusters will be evenly distributed about the origin, which simplifies the training of the NN.

The generation of Cholesky factors follows a slightly different process. We opt to generate the diagonal and off-diagonal elements respectively. We first retrieve $K \times D$ random numbers from the uniform distribution in the range $[0,\ 0.2]$.[6] Then these numbers are reshaped into an array $\mathbf{C}_1$ of dimensions $(K,\ D)$. Simultaneously, we randomly select $K \times D \times (D-1)//2$ numbers from the uniform distribution in the range $[0,\ 0.2]$.[7] These numbers are then reshaped into an array $\mathbf{C}_2$ of dimensions $(K, D \times (D-1)//2)$. Finally, we compute the Cholesky factor $\mathbf{L}$ as follows:

$$\begin{aligned}\mathbf{L}_\mathrm{d} &= \mathbf{C}_1 \cdot c^{0.5} + 0.1^{0.5},\\ \mathbf{L}_\mathrm{l} &= \mathbf{C}_2 \cdot c^{0.5},\end{aligned} \tag{13}$$

where $\mathbf{L}_\mathrm{d}$ represents the diagonal part, and $\mathbf{L}_\mathrm{l}$ represents the unique off-diagonal elements of the lower diagonal Cholesky factor $\mathbf{L}$. To ensure positive definiteness of the covariances $\mathbf{V}$, a small constant factor of $0.1^{0.5}$ is added to $\mathbf{L}_\mathrm{d}$, which guarantees that the diagonal elements of $\mathbf{V}$ are always greater than 0.1. The exponent 0.5 on the conditioning variables allows $\mathbf{C}_1$ and $\mathbf{C}_2$ to intuitively indicate the level of covariance, instead of having to intuit them from the Cholesky factor. With the Cholesky factors $\mathbf{L}$ we can compute the underlying noiseless covariance of the toy model as:

$$\mathbf{V} = \mathbf{L}\mathbf{L}^\mathrm{T}. \tag{14}$$

In practice, real-world samples are always subject to noise. To construct a noisy toy model, we introduce the noise covariance matrices $\mathbf{S}$ using:

$$\mathbf{S} = \mathbf{L}_\mathbf{S}{\mathbf{L}_\mathbf{S}}^\mathrm{T}. \tag{15}$$

In this equation, the Cholesky factor $\mathbf{L}_\mathbf{S}$ is responsible for modelling the noise characteristics. The diagonal part of $\mathbf{L}_\mathbf{S}$ is sampled from a uniform distribution $U(0, 1)$, while the lower left-hand part is sampled from another uniform distribution $U(-0.5, 0.5)$. This choice of distribution introduces both positive and negative elements in the noise covariance Cholesky factors, simulating non-trivial covariant noise in a real astronomical application. Following equation (6), the noise covariance $\mathbf{S}$ can be added to the underlying covariance $\mathbf{V}$ in equation (14) to obtain the noisy distribution.

The choice of power-law behaviour as a function of the conditioning variable and the exponents used in our equations allows a broad range of behaviours for our toy model. In particular, the exponent on the conditioning variable used in the means [equation (12)] is larger than the one in the underlying covariance [equation (13)]. When c takes on smaller values, the larger exponent in equation (12) causes the Gaussian clusters to overlap. The orange points and contours in Fig. 2 illustrate the samples and their densities from the noise-convolved underlying distribution, in contrast to the noise-free underlying distribution shown in black. The influence of our noise dominates the dispersion within the clusters. Conversely, when c assumes larger values, as shown in Fig. 3, the Gaussian cluster centres separate more distinctly. Under such conditions, the influence of the noise diminishes, allowing the underlying covariances of the GMM to become more evident.

3.2 Training CondXD

To illustrate the capabilities of CondXD, we generate 90 000 $\{c, \boldsymbol{x}, \mathbf{S}\}$ training samples and 10 000 validation samples from a noisy toy model defined by the simulated parameters in equations (11)–(14). To obtain a single noisy sample $\boldsymbol{x}$, the conditioning variable c is uniformly sampled in the range $[0,\ 1]$, and input in our toy model. Then, we compute the noise covariance $\mathbf{S}$ using equation (15), add it to the noiseless covariance $\mathbf{V}$, and finally draw samples $\boldsymbol{x}$ from this noisy distribution. Samples from the Gaussian mixture are drawn following the standard approach (Harris et al. 2020): the specific Gaussian cluster to be sampled is first decided via a random draw employing the mixing coefficients as weights, and then a sample is drawn from that Gaussian cluster.

We train CondXD on the 90 000 training samples after normalization with the strategies described in Section 3.3, implementing a mini-batch size of 250. After training for 100 epochs, the loss [see equation (5)] for the training and validation sets converges to a constant value. The training and validation loss as a function of training epoch is shown in Fig. 4. Both losses decrease with training epoch, indicating that the NN has learned to fit the parameters governing the conditioned noisy distribution. In fact, overfitting is not significant, as there is only minimal disparity between the validation loss and the training loss.

3.3 Results from the toy model

In this subsection we provide a visual comparison of the aforementioned distributions. After training, to test the deconvolution

[4] These were actually generated by permuting random integers and are hence constrained to be integer multiples of 0.02.

[5] Randomly sampling and permuting non-repeating integers in $[0,\ 10 \times K \times D]$ and then multiplying by $1/(K \times D)$.

[6] Randomly sampling and permuting non-repeating integers in $[0,\ 10 \times K \times D]$ and multiplying by $1/(50 \times K \times D)$.

[7] Randomly sampling and permuting non-repeating integers in $[0, 10 \times K \times D \times (D-1)//2]$ and multiplying by $1/(50 \times K \times D \times (D-1)//2)$.

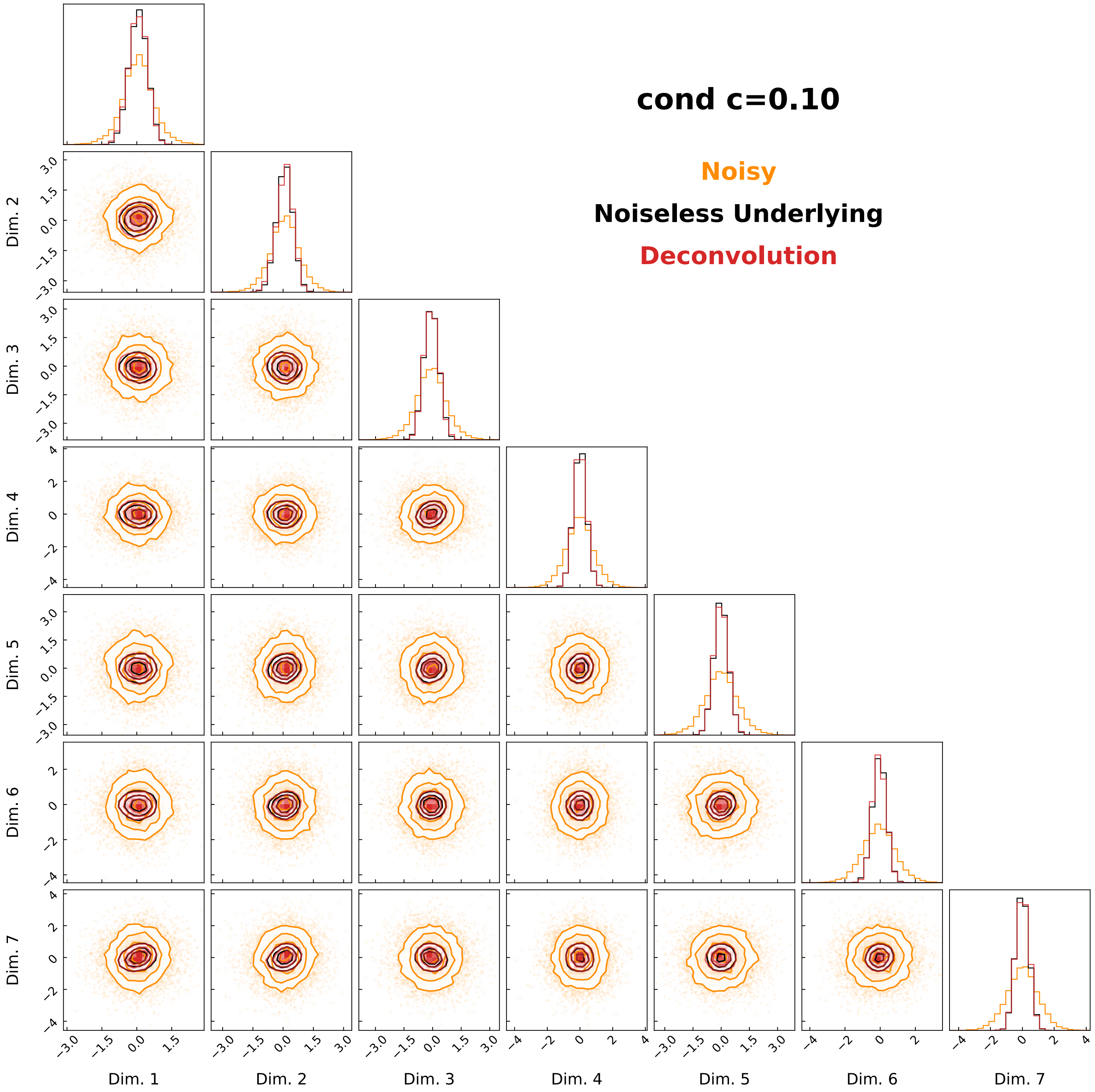

Figure 2. The distribution and density contours of 10 000 samples from the noisy toy model, the underlying toy model, and the deconvolution when $c = 0.10$. Orange scatters are samples from the noise-convolved underlying distribution with orange contours representing their density contours. Black scatters and contours are for the samples from the underlying distribution, while red is for the deconvolution result. The upper or right-hand panels show the 1D marginal distribution of the samples. Orange histograms represent the samples from the noise-convolved underlying distribution, black is for the underlying distribution, and red is for the deconvolution. All corner plots in this paper are created by the PYTHON package CORNER (Foreman-Mackey 2016).

capability of CondXD, we compare the estimated deconvolved distribution with the noiseless underlying distribution. Note that the underlying model is conditioned and we train on a continuous range of c over [0, 1], but in this section we evaluate the performance of our method only for two extreme values, $c = 0.1$ and $c = 0.9$, whereas the result for another intermediate case $c = 0.5$ is shown in the Appendix.

The best way to visualize how well we are deconvolving is to compare the distribution of samples drawn from our underlying noiseless model to the distribution of samples from the trained CondXD distribution, which is usually achieved by making density contour plots of these samples. We input the specific aforementioned values of c into our toy model, and generate 10 000 samples from the underlying noiseless GMM as the test set. For comparisons with the noisy distribution, we also generate 10 000 noisy samples by drawing 10 000 random noise covariances, adding each to the covariance matrice of the same noiseless GMM, and sampling the noisy GMM. For CondXD, the same c is input in the trained model, and 10 000 noiseless samples are drawn from it. Note that these samples have been transformed to the original scale in the way described in Section

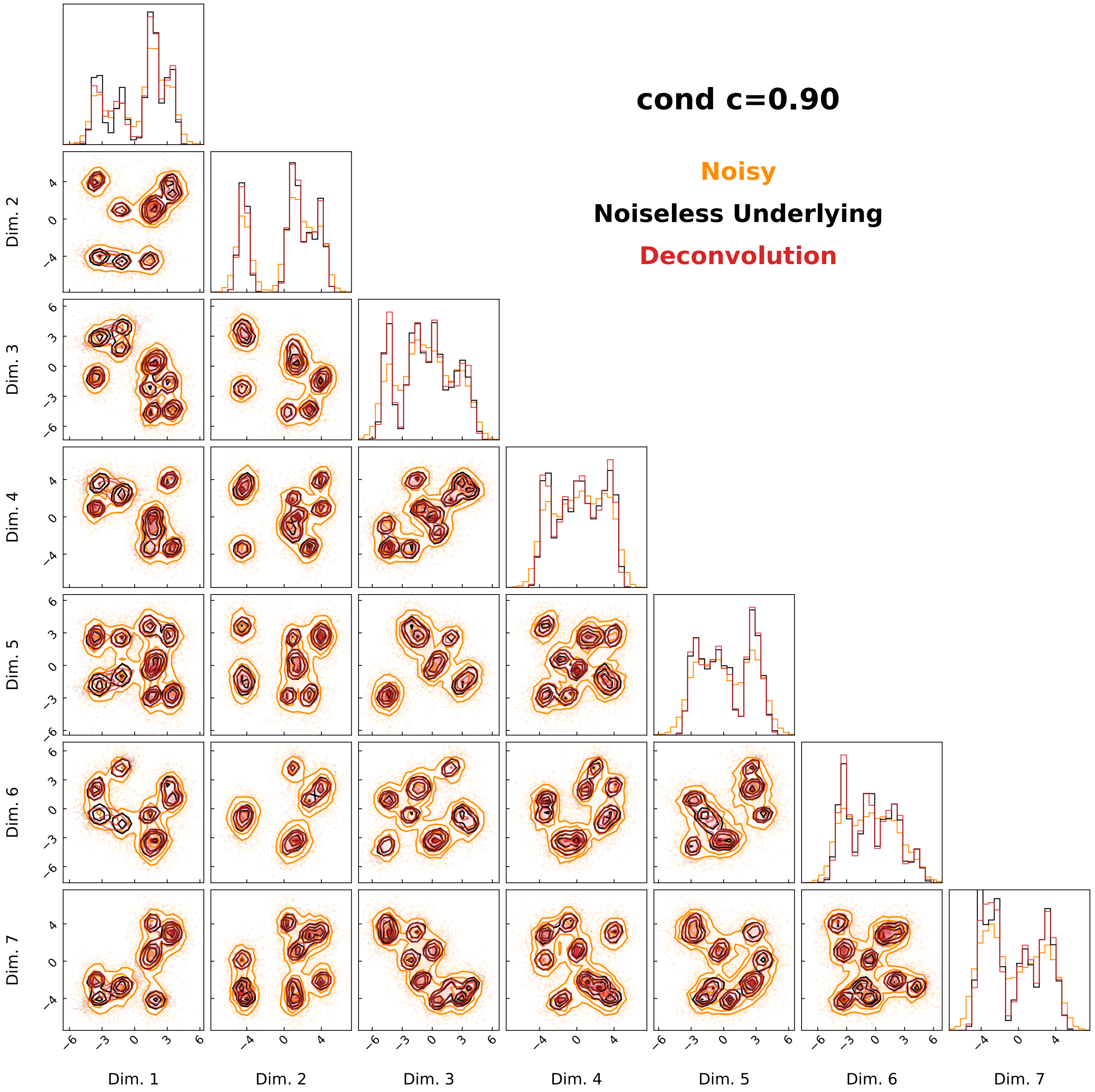

Figure 3. The distribution and density contours of 10 000 samples from the noisy toy model, the underlying toy model, and the deconvolution when $c = 0.90$. The upper or right-hand panels show the 1D marginal distribution of the samples. The colour scheme is the same as in Fig. 2.

2.3, i.e. multiplied by the standard deviation and added to the mean of the 90 000 training samples on each feature. The density contours and 1D marginal histograms for these three sets of samples are shown in Figs 2 ($c = 0.1$) and 3 ($c = 0.9$).

The black histograms and contours in Fig. 2 show that for $c = 0.1$ the underlying Gaussians in the Gaussian mixture strongly overlap. The orange lines show the density distribution of the noisy samples from the noise-convolved Gaussian mixture, which are significantly broader than the width of the underlying distribution, indicating that the noise level is larger than the underlying dispersion of the Gaussian mixture. Nevertheless, CondXD still successfully deconvolves and uncovers a robust estimate of the underlying distribution. Our estimate for the deconvolved distribution is shown by the red lines. One sees qualitatively that they differ negligibly from the underlying distribution in black.

On increasing the value of c to 0.9, the means of the Gaussians separate more, as shown in Fig. 3. The orange contours of noisy samples from the GMM toy model show that the noise level is comparable to the intrinsic dispersion of the Gaussians in the mixture, which blurs the distinction between the individual components of the mixture. CondXD is still capable of estimating the noiseless underlying distribution under such conditions. Most of the red contours are consistent with the black ones, indicating that most of the individual Gaussian clusters have been recovered correctly. This is also confirmed in the panels showing the 1D marginal distributions, as the estimated 1D histograms differ very little from the underlying

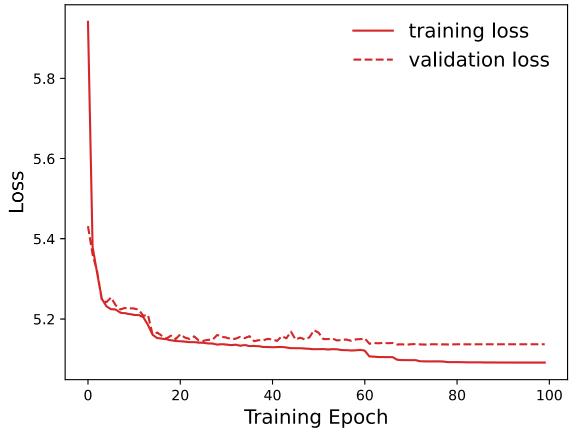

Figure 4. The loss reduction process using the 60 000 training samples from our GMM noisy toy model. The solid red line is the training loss and the dashed one is the validation loss.

distribution. Nevertheless, in rare cases, the deconvolution does not perform well. For example, in the subpanel showing dimensions 1 and 6, two nearby noiseless Gaussians (black contours) are fitted with a single deconvolved Gaussian (red contours). We repeat the whole training and testing process for 10 different toy models (each has a different random seed ξ), and our visual assessment shows that four of the 10 realizations fail to recover all the underlying Gaussians, while the other six succeed in recovering every Gaussian. In our toy model, the value of c controls the separation of Gaussians. At $c = 0.9$, we have almost reached the most extreme value for c. However, the Gaussian clusters remain insufficiently separated because the noise level amplitude is still relatively significant. As a result, CondXD struggles to perfectly differentiate every Gaussian. If we had allowed c to be beyond 1 and included more training samples, the Gaussian cluster could be more separated, and CondXD might be able to distinguish them.

4 COMPARISON WITH BINNING METHOD

One of the main advantages introduced by the method that we described in Section 2 is that it can deconvolve and fit distributions that depend on conditioning variables. This is usually a common situation in astrophysics, where physical properties of sources often depend on other properties (e.g. the variation of colour distributions with the magnitude of the sources). Capturing these dependences is not an easy task, and has no standard approach. Previous works usually divide the samples into bins of conditioning variables, and estimate the distribution of samples in every bin respectively (e.g. Bovy et al. 2011b, 2012; Bird et al. 2021; Nanni et al. 2022). The main drawback of the binning method is that the continuity of the distribution variation, which is dependent on the conditioning variable, among the different bins is not easily guaranteed. The distribution is supposed to vary smoothly among the bins, but the independent estimations within each bin might be trapped in some local optima, resulting in discontinuity. Furthermore, to limit the variance within each bin, the bin width should be narrow enough. However, the number of samples in each bin decreases as the bin width decreases, affecting the accuracy of the estimation. Therefore, a manual choice of a trade-off between the bin width and sample size is inevitable, and there is no objective way to define it. In contrast, the neural network of CondXD trained by all the samples naturally provides continuity, and this does not require any binning of the conditioning variable. To demonstrate the advantages of CondXD compared to the aforementioned binning approach, we apply a binning deconvolution algorithm (denoted as bin-XD hereafter) to the GMM toy model described in Section 3 and compare the results with CondXD.

Using the same training samples described in Section 3, the conditioning variables and corresponding data samples are split into 10 conditioning variable bins of equal size 0.1. Since the bins are narrow, we assume that the dependence of the sample properties with respect to the conditioning variable inside each bin is negligible. For every bin, we apply the XDGMM method (Holoien, Marshall & Wechsler 2017), which is an implementation of the extreme deconvolution, to the training sets. XDGMM is a PYTHON package that models mixed Gaussians with the SCIKIT-LEARN API.[8] It performs density estimation of noisy, heterogeneous, and incomplete data with the extreme-deconvolution algorithm (Bovy et al. 2011a) when an uncertainty covariance is provided, as in our case. The hyperparameters in XDGMM are the number of Gaussians, set to $K = 10$, and dimensions $D = 7$, which are consistent with those used in CondXD. The bin-XD code is progressively applied starting from the smallest conditioning variable values ($c \in [0,\ 0.1]$) to the largest ones ($c \in [0.9,\ 1]$). The fitting in individual bins does not necessarily guarantee the continuity of the model among different bins. Following Nanni et al. (2022), the bin-XD code fits for all the conditioning variable bins are initialized using the best-fitting parameters for the previous bin. The starting bin is the only one that is initialized without reference.

After training bin-XD, to derive the test set, we uniformly sample 25 000 conditioning variables in the range $[0,\ 1]$, and draw 25 000 corresponding samples from the noiseless toy model (underlying GMM). The test set is divided into conditioning variable bins as described in the previous paragraph. To quantify the performance of CondXD and bin-XD, we use the discrete KL divergence as a measure of the difference between the underlying density and the estimated. Similar to equation (4), the discrete KL divergence is defined as:

$$D_{\mathrm{KL}}(p\|\hat{p}; c) = \frac{1}{N}\sum_{i}^{N}\log\left(\frac{p(\boldsymbol{x}_i \mid c_i)}{\hat{p}(\boldsymbol{x}_i \mid c_i)}\right), \tag{16}$$

where $\boldsymbol{x}_i$ are the test samples from the underlying density without normalization, N is the sample size, $p(\boldsymbol{x}_i \mid c_i)$ is the probability density of sample $\boldsymbol{x}_i$ under the underlying GMM, and $\hat{p}(\boldsymbol{x}_i \mid c_i)$ is the probability under the GMM estimated by either CondXD or bin-XD without normalization. Specifically, since both CondXD or bin-XD are deconvolving the normalized samples, we scale the output mean and covariance of CondXD or bin-XD to obtain the unnormalized estimated GMM:

$$\begin{aligned} \boldsymbol{\mu}_j &= \boldsymbol{\mu}_{\mathrm{n},j} + m_j, \\ \boldsymbol{V}_{jk} &= (\boldsymbol{V}_{\mathrm{n}})_{jk}\sigma_j\sigma_k, \end{aligned} \tag{17}$$

where $\boldsymbol{m}$ is the mean and $\boldsymbol{\sigma}$ is the standard deviation of all the 90 000 training samples on each feature. In fact, equation (16) is calculated for every conditioning variable bin. When p and $\hat{p}$ are close, D_{KL} should be close to zero. In general, the probability for the underlying distribution, $p(\boldsymbol{x}_i \mid c_i)$, should be higher than the probability for the estimated distribution $\hat{p}(\boldsymbol{x}_i \mid c_i)$, since they are being evaluated at sample $\boldsymbol{x}_i$ from the underlying distribution. Thus the KL divergence is generically expected to be positive. Besides, if we instead consider

[8] https://github.com/tholoien/XDGMM

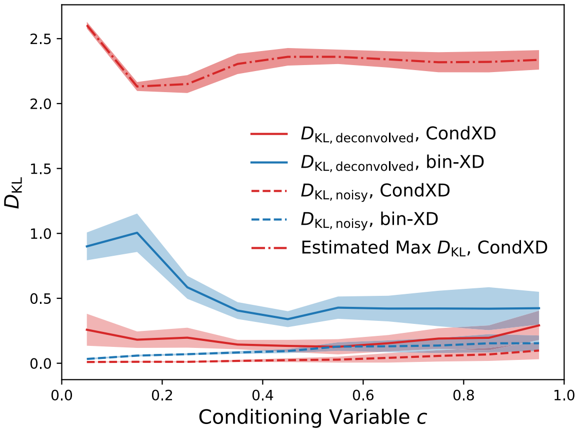

Figure 5. KL divergence of different methods as a function of the conditioning variable $\boldsymbol{c}$ in our experiments. Solid lines are KL divergence measured on the underlying distributions and CondXD estimated distributions. The dashed lines are computed on the noise-reconvolved underlying distributions and noise-reconvolved estimated distributions. The dash–dotted line is an estimation of the possible maximum D_{KL}, assuming that CondXD is only fitting the noisy underlying GMM while not deconvolving at all (for details see Section 4). The red curves show the KL divergence of CondXD, while the blue curves are for bin-XD (see Section 4).

p in equation (16) to be the underlying noiseless GMM and $\hat{p}$ as the noise-reconvolved estimated probability, equation (16) is just the KL divergence of an algorithm that simply fits a Gaussian mixture to the noisy distribution without deconvolving. This situation represents the worst case (no deconvolution performed) and it yields a maximum value for D_{KL}, which provides a useful reference. In fact, D_{KL} should lie within zero and the aforementioned maximum.

We compute the D_{KL} value of every bin, resulting in a relation between D_{KL} and c. For a more general examination we repeat our experiment 10 times with 10 different random seeds ξ that determine the toy model. In every experiment, CondXD and bin-XD are applied to the same training samples. We average the 10 D_{KL} versus c curves and compute the standard deviation. The results are shown as solid curves and shaded regions respectively in Fig. 5. Meanwhile, we also plot the estimated maximum of D_{KL} (defined in the previous paragraph) with CondXD (dash–dotted red line in Fig. 5) as a reference.

Fig. 5 shows that CondXD could deconvolve (solid red line) the noisy distribution for all values of conditioning variable c. The solid red line is flat and close to zero compared to the estimated maximum (dash–dotted red line). This indicates globally good performance. In contrast, bin-XD (solid blue line) shows less capability than CondXD at any value of the conditioning variable, as its D_{KL} is much higher. Especially at $c \leq 0.2$ values, the KL divergence of the bin-XD increases remarkably. This implies that bin-XD is not a promising method for cases of overlapping Gaussians and noise domination.

One may argue that the poor performance of our bin-XD method at small c values might be related to the fact that our fit to the first bin with the lowest conditioning variable value is not initialized with reference to a trained bin. To verify this, we perform more experiments by training bin-XD on the opposite direction: starting from the largest c value bin with random initialization and proceeding toward the smallest value bin. However, the results are consistent with those presented in Fig. 5 (solid blue line). Bin-XD's inability to effectively deconvolve the data in bins with low values of the conditioning variable is inherent. The poor performance may result from the fact that we did not implement any strategy to prevent overfitting in the bin-XD method. As c decreases and the Gaussian clusters merge, using $K = 20$ Gaussians for density estimation can lead to significant degeneracy.

By evaluating D_{KL} with c, we note that the value of D_{KL} of CondXD rises with increasing c. This increase in D_{KL} is likely due to the fact that CondXD is fitting two close underlying Gaussians with a single one, as described in Section 3. The noisy Gaussians in the noise-convolved toy model are not separated sufficiently so that CondXD may not be able to fit every single Gasussian correctly. If the c range is broadened to larger c, the Gaussians are more separated, and CondXD is more likely to estimate well.

The performance of the reconstruction of the noisy distributions can also be compared if we compute a set of noise covariances from equation (15) and convolve them with p and $\hat{p}$ in equation (16). The test samples $\boldsymbol{x}_i$ should also be resampled after reconvolution. We compute the same number, i.e. 25 000, of noise covariances, draw test samples after adding the noise covariances to the underlying GMM, and calculate the KL divergence of the two noise-reconvolved density distributions. The results are shown as dashed lines in Fig. 5. Both D_{KL} are very close to zero for all c values, which implies that the reconstruction is very precise. CondXD also outperforms bin-XD in the reconstruction globally.

5 DECONVOLVING THE DISTRIBUTION OF QUASAR CONTAMINANTS

Luminous high-redshift (high-z) quasars are a key tool for studying the primordial universe during the epoch of reionization (for some recent works, see Becker et al. 2021; Bosman 2021; Davies et al. 2021; Wolfson et al. 2024). However, finding the most distant quasars is challenging. Currently, only eight quasars are known at $z \geq 7$ (Mortlock et al. 2011b; Bañados et al. 2018; Wang et al. 2018, 2021; Matsuoka et al. 2019; Yang et al. 2019, 2020), primarily due to the limited photometric depth of current near-infrared surveys and the decreasing number density of quasars with increasing redshift ($\approx 10^{-3}$ deg^{-2} at $J = 21$, where J is a flux band in the VIKING survey; Wang et al. 2019). Moreover, the number of contaminants, which mostly consist of cool galactic dwarfs and early-type galaxies, is much higher (≈ 20 deg^{-2} at $J = 21$), making efficient classification methods critical. Bayesian probabilistic methods offer a principled way to classify quasar candidates (e.g. Mortlock et al. 2011a; Euclid Collaboration et al. 2019). One can estimate the density distribution of quasars and contaminants and compute the probability that a source belongs to quasars or contaminants (see Section 5.2). In this section, we apply our CondXD method to a real astrophysical example: to deconvolve the flux distribution of quasar contaminants. We train our model using the same contaminant data set described in Nanni et al. (2022). We present the results of our deconvolution and reconstruction, as well as a brief comparison with the previous method of Nanni et al. (2022).

5.1 Training data: quasar contaminants

The training data that we use to apply the CondXD method to the problem of high-z quasar classification is identical to the data set described in Nanni et al. (2022), which contains 1902 071 sources of quasar contaminants. In summary, our model is trained on 1076 deg^2 of overlapping area from the DELS (Dey et al. 2019), VIKING (Edge et al. 2013), and unWISE (Meisner et al. 2019; Schlafly, Meisner & Green 2019) imaging surveys. The multiband fluxes are obtained from the DELS z optical band, the VIKING $YJHK_s$ near-infrared

(NIR) bands, and the unWISE $W1W2$ mid-infrared (MIR) bands with forced photometry. The construction algorithms are described in detail in section 3.1 of Nanni et al. (2022). The aim of Nanni et al. (2022) is to find high-redshift quasars ($6 \leq z \leq 8$), whose Lyα lines shift to the Y band, while the VIKING J band could reach a depth of 22.1 at the 5σ level. Therefore, all sources in the sample are selected with high signal-to-noise ratio in the J band: SNR(J) ≥ 5.

5.2 Density in the Bayesian theorem

To classify sources based on observed fluxes $\{\hat{F}\}$, we need to calculate the conditioned probability that a source belongs to a certain class according to the Bayesian theorem:

$$P\left(O \in B \mid \{\hat{F}_i\}\right) = \frac{p\left(\{\hat{F}_i\} \mid O \in B\right) P(O \in B)}{p\left(\{\hat{F}_i\}\right)}, \tag{18}$$

where O is the object and B is the class, i.e. quasars or contaminants. If we denote quasars as A and contaminants as B, the denominator of the right-hand side in equation (18) is defined as

$$p\left(\{\hat{F}_i\}\right) = p\left(\{\hat{F}_i\} \mid O \in A\right) P(O \in A) \\ + p\left(\{\hat{F}_i\} \mid O \in B\right) P(O \in B), \tag{19}$$

as a source can only be a quasar or contaminant. The factor $P(O \in B)$ in the numerator on the right-hand side of equation (18) is the prior, which could be approximated as the fraction of quasars in the data set. The other factor, $p\left(\{\hat{F}_i\} \mid O \in B\right)$, is the density of quasars in flux space that is to be estimated.

As stated in Section 1, the number density of quasars observed by *JWST* and *Euclid* will have a dominant power-law shape as a function of the J-band magnitude. This means that the J-band magnitude feature has a distribution that is far from Gaussian. In the context of a Gaussian mixture model, a large number of Gaussian components would be required to model this distribution accurately. In contrast, their colour (logarithm of relative flux) distribution is flat enough to be modelled by a small number of Gaussians. Furthermore, crucial information for distinguishing quasars and contaminants lies mostly in colour. This motivates people to search for a colour-based distribution model. Additionally, relative fluxes are easier to derive and more straightforward to model than colours. In the case of faint sources that drop out in certain bands (e.g. high-z quasars), the measured fluxes could be non-positive, and it is infeasible to compute the colour, i.e. logarithm of zero or a negative value. Furthermore, the observational uncertainties of the relative fluxes are closer to Gaussian than colours, especially when the uncertainty in the reference J band is small. In fact, as both the numerators and denominators are noisy, the Gaussian approximation of the flux ratio density can only be validated when the noise of the denominators is small. If the noise of the denominators is large, the distribution of the ratio of two Gaussian random variables is not Gaussian. In our case, since the observed J-band flux, $\hat{F}_J$, is always significantly detected at great than 5σ significance, this condition is well satisfied. Hence, instead of fitting the distribution of the measured fluxes, we choose to model the fluxes relative to the J-band flux.

We separate the flux relative to the J band from the absolute flux in the likelihood as follows:

$$p\left(\{\hat{F}_i\} \mid O \in \mathrm{B}\right) \propto p\left(\{\hat{F}_i/\hat{F}_J\} \mid \hat{F}_J, O \in \mathrm{B}\right) \\ \times p\left(\hat{F}_J \mid O \in \mathrm{B}\right), \tag{20}$$

where $\hat{F}_i$ are the fluxes of the z, Y, H, K, $W1$, $W2$ bands. In this equation, the probability density of the absolute fluxes is separated into the distribution of the relative fluxes conditioned on the J-band flux and the distribution of the J-band fluxes. In this paper, we mainly discuss the first factor, and the derivation of the second factor can be found in section 4.3 in Nanni et al. (2022).

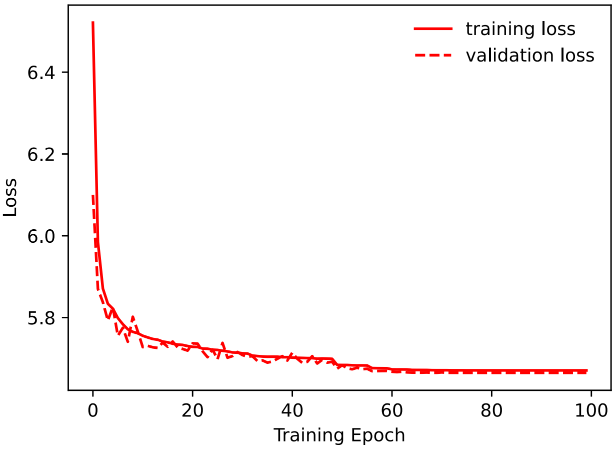

Figure 6. The loss decrease process of CondXD in the experiment of application to the quasar contaminants of Nanni et al. (2022). The solid red line is the training loss and the dashed line is the validation loss.

5.3 Density estimation

Nanni et al. (2022) employed their XD-based XDHZQSO algorithm to fit the distribution of the contaminants and simulated high-redshift quasars with a GMM. The algorithm has demonstrated high efficiency, accuracy, and stability. XDHZQSO, however, has to divide the contaminants into a discrete number of J-band magnitude bins, because the colour distribution of the contaminants is a strong function of magnitude, and XDHZQSO cannot be used to estimate in the continuous limit. They implemented complicated strategies to capture the variation of the relative flux distribution with magnitude and guarantee continuity. The authors used 50 overlapping bins, with the width of each bin determined by a broken sigmoid function of the J-band bin right edge. As the right edges are uniformly distributed, the bins overlap with their neighbors. The overlap between the bins improves continuity among adjacent bins as well as a sufficient number of sources at the faint and bright ends of the J-band magnitude. Within each bin, they used the XD algorithm to estimate the density, with the same initialization strategy as that described in Section 4, to further improve the model's continuity. These strategies slow the training process, as some samples belong to multiple bins and will be input to the training process multiple times. Moreover, this binning strategy results in additional problems. Since the bin width is very large at the two ends, e.g. a resulting magnitude range of 5 mag compared with the right edge step of 0.05 magnitude at the faintest end, it is hard to correctly capture the variation of the model.

Instead, with our CondXD method, the J-band magnitude is a conditioning variable c and we can build a continuous and general model by entering it into the NN and obtaining the Gaussian parameters. We model the six-dimensional density distribution of relative fluxes $\{f_z/f_J,\ f_Y/f_J,\ f_H/f_J,\ f_{K_s}/f_J,\ f_{W1}/f_J,\ f_{W2}/f_J\}$ using $K = 20$ Gaussian components. The number of Gaussians adopted is consistent with the number chosen by Bovy et al. (2011b) and Nanni et al. (2022). Empirically, models with fewer than 20 components overly smooth the observed distribution, while those with more than 20 components are likely to suffer from

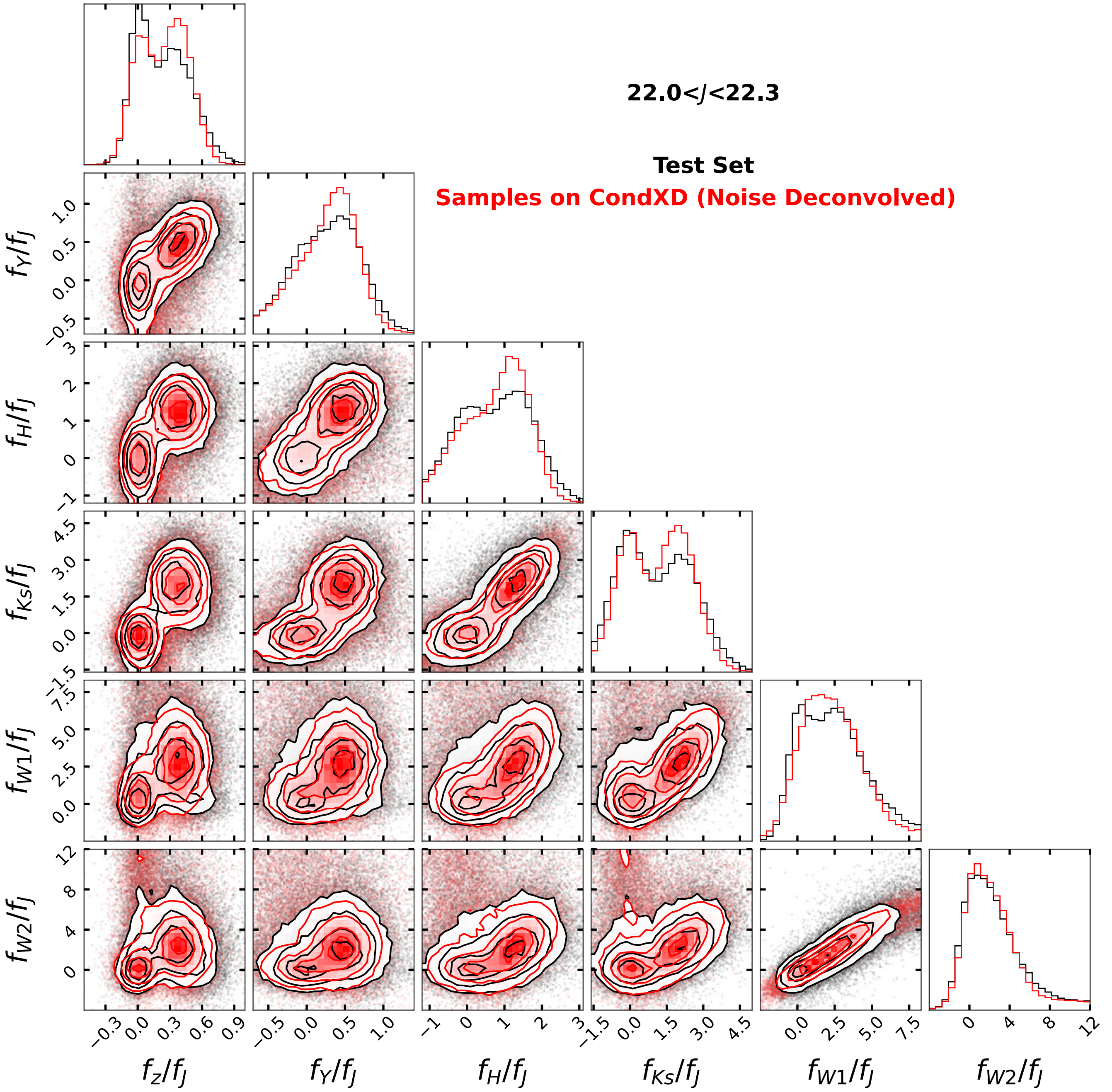

Figure 7. The relative fluxes of all quasar contaminant samples in the $22.0 < J < 22.3$ bin and their density contours are plotted in black. After deconvolution with CondXD, the samples from the deconvolved distribution and their density contours are shown in red. The red contours are narrower because the noise has been deconvolved.

overfitting. As we are deconvolving the relative noisy fluxes instead of the measured fluxes, the uncertainty covariance matrix should be computed. The validity and derivation of the uncertainty covariance matrix of relative fluxes have been discussed in appendix A of Nanni et al. (2022). Specifically, one needs to remove the off-diagonal elements (i.e. set them to 0) in the relative flux noise covariances when $J > 21$. This is because, in the limit of the faint J-band regime, the noise becomes significant compared with the flux, and the distribution of the relative fluxes violates the Gaussian assumption as discussed earlier. As we are estimating with a GMM assuming Gaussian noise, the non-Gaussian noise should be approximated by a Gaussian. We convolve the GMM output by CondXD with the uncertainties of the relative fluxes by adding the uncertainty covariance to the GMM covariance. The samples are divided into training and validation sets with ratio 9 : 1. After training and validating the NN with the strategies described in Section 3.3 for 100 epochs, our model converges. The loss decrease is shown in Fig. 6.

We compare the distribution of the entire contaminant set with the corresponding predictions by our trained model in Figs 7 and 8. As we do not have access to the underlying noiseless distribution of the relative fluxes, we can only compare our predictions, either noiseless or convolved with noise, with the noisy data set. We select the same J-band range as in the appendix of Nanni et al. (2022), i.e. $22.0 <$

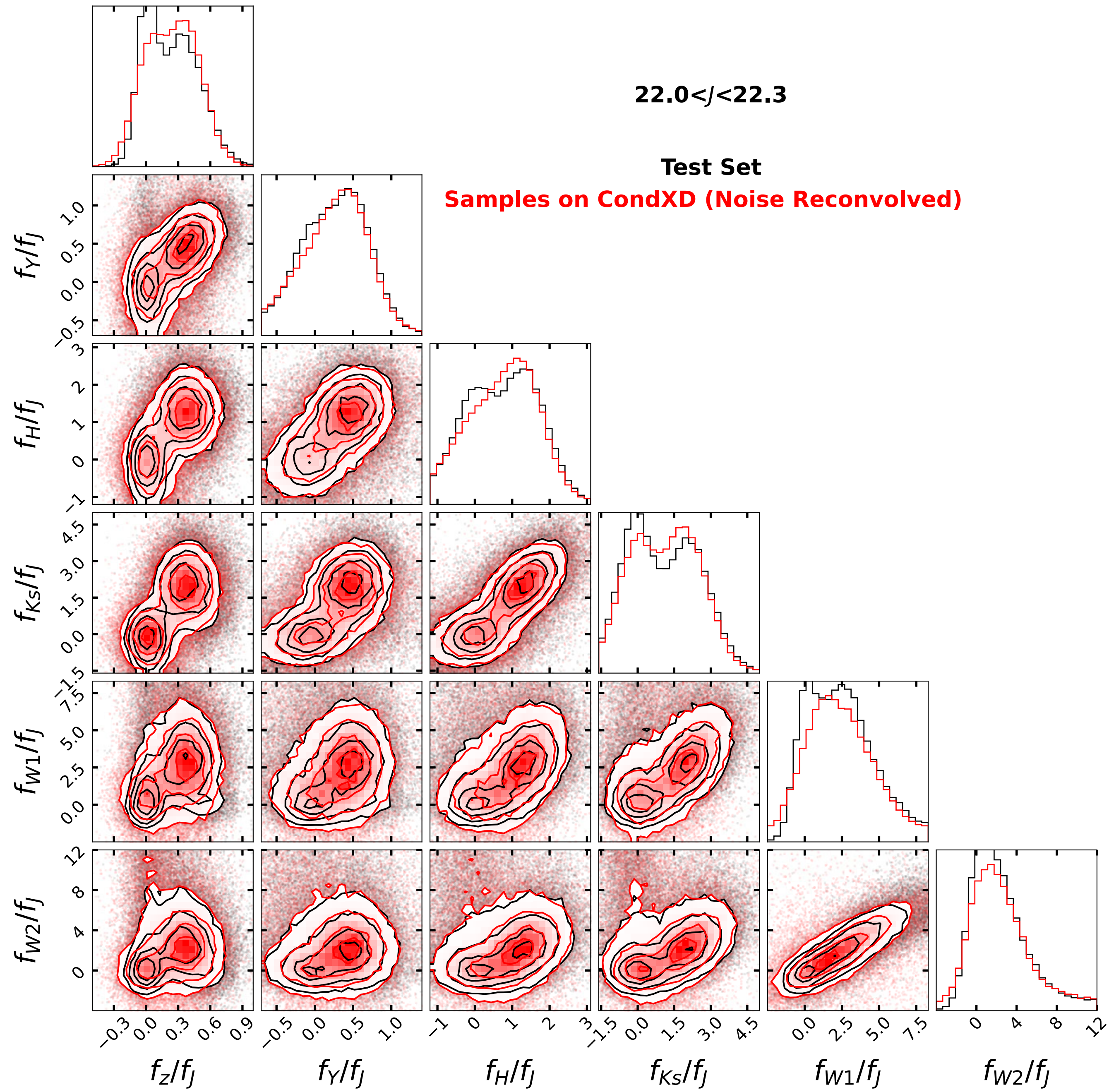

Figure 8. The relative fluxes of all quasar contaminant samples in the 22.0 < J < 22.3 bin and their density contours are plotted in black. In order to illustrate that, after deconvolution by CondXD, we are still capable of reconstructing the noisy data, we reconvolve the deconvolved distribution in Fig. 8 with the noise of the quasar contaminants. The samples from the noise-reconvolved model and their density contours are shown in red. The red contours show only negligible differences from the black, proving that the reconstruction is successful.

J < 22.3, for display and comparison purposes. For each object in this J-band bin, its J-band magnitude is input to the CondXD model and a GMM is output. Then, for each object one noiseless predicted data point is sampled from the GMM. After convolving the GMM with the object's uncertainty distribution, we also sample a noisy prediction. The distribution of the noisy predictions is shown in Fig. 8. When comparing Fig. 7 with fig. A1 in Nanni et al. (2022), the two deconvolutions yield similar results. In Fig. 8, the noisy prediction distribution (red) matches the original samples (black) promisingly. CondXD has reconstructed the noisy distribution precisely. When compared with fig. A2 in Nanni et al. (2022), we see that our model performs similarly to theirs. The distributions of the noisy predictions (red) in all the other bins produced with our model are also consistent with those in Nanni et al. (2022). Besides performance, our model finishes training within three hours on 1902 071 samples with a 2.8 GHz Quad-Core Intel Core i7 for MacBook, compared with ≈ 30 h with their model. This is partly because many pieces of data in the overlap of different bins are used in training multiple times, which substantially increases the time required to construct a model for all the bins. Note that no GPU is implemented in any of our experiments. With a GPU the time cost can be greatly reduced.

6 CONCLUSIONS AND DISCUSSION

In this paper we built a conditional density deconvolution algorithm, CondXD, with a neural network. This is an extension of the existing XD method and is combined with mixture density networks. It features the ability to estimate the underlying density of noisy properties, which depends on some conditioning variables, given a set of data with large and heteroscedastic uncertainties. CondXD works in the background of deconvolving data with features far from Gaussian with negligible noise: the distribution of the non-Gaussian features can be separated from the joint distribution and modelled independently, and CondXD can be applied to deconvolve the distribution of the other features conditioning on the non-Gaussian features.

We test CondXD on a toy model, a GMM whose parameters (i.e. mixing coefficients, means, and covariances) are dependent on a conditional. The samples are drawn from the GMM convolved with non-identical noise covariances. The result shows that CondXD is able to deconvolve the heteroscedastic uncertainties and estimate the underlying conditional GMM. It can also reconstruct the noisy distribution given the noise. Further experiments applying a classical binning XD to the same toy model show that CondXD consistently achieves higher continuity and better accuracy (Fig. 5). It shows a flat KL divergence D_{KL} curve throughout the conditional range, which is globally smaller than the binning method, indicating comprehensively more solid estimation. In particular, in the low signal-to-noise ratio region ($c < 0.05$ in Fig. 5), the D_{KL} of CondXD is close to 0, while the binning method approaches the estimated worst value. We further apply our method to a real astronomical case, i.e. inferring the underlying distribution of a set of noisy high-z quasar contaminant fluxes. Compared with the method used by Nanni et al. (2022), which used a binning approach, our method outputs a comparable result, but $\approx$ 10 times faster.

Although we only apply CondXD to 1D conditioning variables, it can be easily generalized to multidimensional conditioning variable cases. For example, Bovy et al. (2012) included not only the reference band flux but also the redshift as new features in addition to the original band fluxes, in order to obtain the flux density in different redshift ranges. There are no appreciable uncertainties on redshift, as quasar colours do not vary significantly within typical redshift uncertainties. Therefore, the redshift perfectly matches our requirement that the noise of conditioning variables should be negligible. In conclusion, the redshift is certainly another reasonable conditioning variable that is worth including.

However, restrictions still exist in our algorithm. This method only deconvolves the Gaussian features, and it cannot deconvolve the conditioning variables. Our conditioning variables need to be noiseless, while this is rarely satisfied in practice, like in Section 5. Therefore, the conditioning variables should all have a high signal-to-noise ratio to approximate the noise-free assumption. Another commonly used approach in density estimation that could possibly help solve such an issue is normalizing flow (Tabak & Vanden-Eijnden 2010; Tabak & Turner 2013). Normalizing flows transform a density that is easy to describe into a complicated density by a set of invertible functions, and have shown good scalability and flexibility in density estimation (e.g. Jimenez Rezende & Mohamed 2015; Cranmer et al. 2019). This class of methods does not require any feature (dimension) of data to be noise free, nor any marginal distribution to be Gaussian. Although these works did not consider conditional densities, normalizing flows can take the conditioning variables as new features (dimensions), deconvolve the general distribution, and further compute the conditional density like in Bovy et al. (2012). However, to our knowledge, only homoscedastic noise (identical noise distribution for all samples) has been considered (Dockhorn et al. 2020) when data have non-Gaussian features. Our CondXD might still be the best method for deconvolving conditional densities with heteroscedastic noise.

ACKNOWLEDGEMENTS

We would like to express our sincere appreciation for the constructive comments made by the ENIGMA group at Leiden Observatory and UCSB on this project. We also extend our gratitude to James A. Ritchie and Iain Murray for making their code SCALABLE_XD available, which served as a foundation for parts of our own code. Additionally, we are grateful to Tom Holoien, Phil Marshall, and Risa Wechsler for providing their XDGMM code, which made it possible to conduct the experiment comparing our model with the classical binning method.

We are also grateful to the developers and contributors of the following PYTHON packages, whose invaluable tools and libraries greatly facilitated our research: NUMPY for providing efficient array operations and numerical computing capabilities that formed the backbone of our data analysis and modelling; ASTROPY for its comprehensive set of astronomical tools and utilities, enabling us to manipulate astronomical data with ease; PYTORCH for its cutting-edge deep learning framework, which empowered us to implement and train our complex neural network model; and MATPLOTLIB and CORNER for their powerful visualization capabilities, which allowed us to create informative plots to present our results.

DATA AVAILABILITY

We present the code for our work, including the basics of CondXD and the toy model, in the CondXD repository at https://github.com/enigma-igm/CondXD. The pipeline for conducting the experiment comparing CondXD and the classical binning method is also made publicly available, as well as the example estimating the density of quasar contaminants.

REFERENCES

Arumugam S., Desai S., 2023, J. High Energy Astrophys., 37, 46
Bañados E. et al., 2018, Nature, 553, 473
Becker G. D., D'Aloisio A., Christenson H. M., Zhu Y., Worseck G., Bolton J. S., 2021, MNRAS, 508, 1853
Bhave A., Kulkarni S., Desai S., Srijith P. K., 2022, Ap&SS, 367, 39
Bird S. A., Xue X.-X., Liu C., Shen J., Flynn C., Yang C., Zhao G., Tian H.-J., 2021, ApJ, 919, 66
Bishop C. M., 2006, Pattern Recognition and Machine Learning. Springer, New York
Bosman S. E. I., 2021, preprint (arXiv:2108.12446)
Bovy J., Hogg D. W., Roweis S. T., 2011a, Ann. Applied Statistics, 5, 1657
Bovy J. et al., 2011b, ApJ, 729, 141
Bovy J. et al., 2012, ApJ, 749, 41
Buder S. et al., 2022, MNRAS, 510, 2407
Cranmer M. D., Galvez R., Anderson L., Spergel D. N., Ho S., 2019, preprint (arXiv:1908.08045)
Davies F. B., Bosman S. E. I., Furlanetto S. R., Becker G. D., D'Aloisio A., 2021, ApJ, 918, L35
Devroye L., 1989, Canadian J. Statistics, 17, 235
Dey A. et al., 2019, AJ, 157, 168
DiPompeo M. A., Bovy J., Myers A. D., Lang D., 2015, MNRAS, 452, 3124
Dockhorn T., Ritchie J. A., Yu Y., Murray I., 2020, preprint (arXiv:2006.09396)

Edge A., Sutherland W., Kuijken K., Driver S., McMahon R., Eales S., Emerson J. P., 2013, The Messenger, 154, 32
Euclid Collaboration, 2019, A&A, 631, A85
Fan J., 1991a, Statistica Sinica, 1, 541
Fan J., 1991b, Ann. Statistics, 19, 1257
Foreman-Mackey D., 2016, J. Open Source Software, 1, 24
Gepperth A., Pfülb B., 2019, preprint (arXiv:1912.09379)
Harris C. R. et al., 2020, Nature, 585, 357
He K., Zhang X., Ren S., Sun J., 2015, in Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, 2015 IEEE International Conference on Computer Vision (ICCV). IEEE, Piscataway, NJ, USA, p. 1026
Holoien T. W. S., Marshall P. J., Wechsler R. H., 2017, AJ, 153, 249
Hosseini R., Sra S., 2015, Advances Neural Inf. Processing Syst., 28, 910
Hosseini R., Sra S., 2020, Math. Programming, 181, 187
Ivezić V., Ivezić Ž., 2021, Icarus, 357, 114262
Ivezić Ž. et al., 2019, ApJ, 873, 111
Jimenez Rezende D., Mohamed S., 2015, preprint (arXiv:1505.05770)
Kiefer J., Wolfowitz J., 1952, Ann. Math. Statistics, 23, 462
Kingma D. P., Ba J., 2014, preprint (arXiv:1412.6980)
Kullback S., Leibler R. A., 1951, Ann. Math. Statistics, 22, 79
Matsuoka Y. et al., 2019, ApJ, 883, 183
Meisner A. M., Lang D., Schlafly E. F., Schlegel D. J., 2019, PASP, 131, 124504
Mortlock D. J., Patel M., Warren S. J., Hewett P. C., Venemans B. P., McMahon R. G., Simpson C., 2011a, MNRAS, 419, 390
Mortlock D. J. et al., 2011b, Nature, 474, 616
Myers A. D. et al., 2015, ApJS, 221, 27
Nanni R., Hennawi J. F., Wang F., Yang J., Schindler J.-T., Fan X., 2022, MNRAS, 515, 3224
Paszke A. et al., 2019, in Advances in Neural Information Processing Systems, Vol. 32. Curran Associates, Inc., New York, p. 8024
Reddy Ch. T. T., Desai S., 2022, New Astron., 91, 101673
Ritchie J. A., Murray I., 2019, preprint (arXiv:1911.11663)
Robbins H., Monro S., 1951, Ann. Math. Statistics, 22, 400
Schlafly E. F., Meisner A. M., Green G. M., 2019, ApJS, 240, 30
Stefanski L. A., Carroll R. J., 1990, Statistics, 21, 169
Tabak E., Turner C., 2013, Communications Pure Applied Math., 66, 145
Tabak E., Vanden-Eijnden E., 2010, Communications Math. Sci., 8, 217
Wang F. et al., 2018, ApJ, 869, L9
Wang F. et al., 2019, ApJ, 884, 30
Wang F. et al., 2021, ApJ, 907, L1
White M. et al., 2012, MNRAS, 424, 933
Wolfson M., Hennawi J. F., Davies F. B., Oñorbe J., 2024, 531, 3069
Yang J. et al., 2019, AJ, 157, 236
Yang J. et al., 2020, ApJ, 904, 26
Zhang C.-H., 1990, Ann. Statistics, 18, 806

APPENDIX A: DENSITY DISTRIBUTION AND CONTOURS

The deconvolution result of our toy model at $c = 0.5$ is plotted in Fig. A1, as an intermediate case in the range of$c \in [0, 1]$.

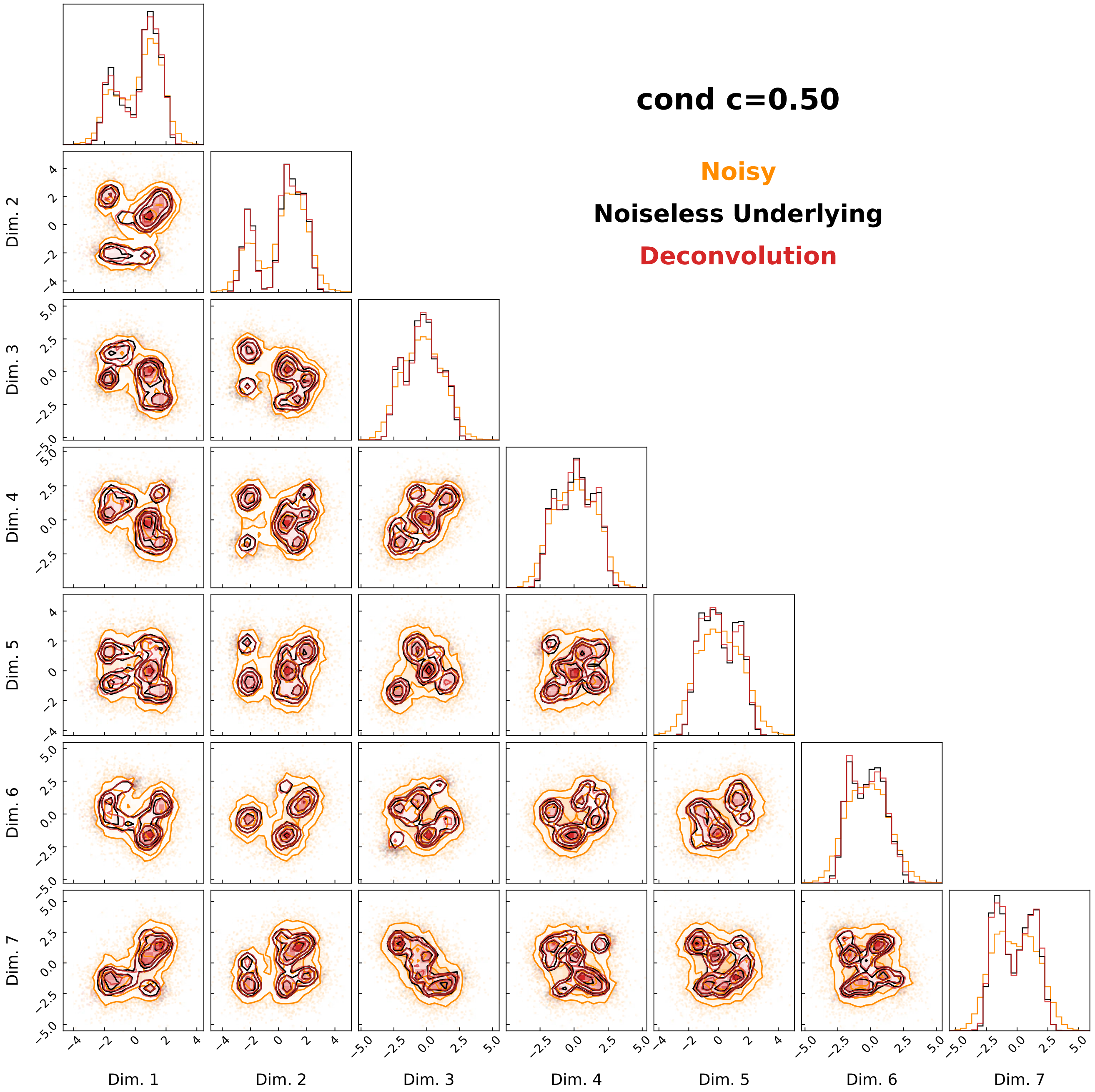

Figure A1. The distribution and density contours of 10 000 samples from the noisy toy model, the underlying toy model, and the deconvolution when $c = 0.90$. The upper or right-hand panels show the 1D marginal distribution of the samples. The colour scheme is the same as in Figs 2 and 3.

This paper has been typeset from a TEX/LATEX file prepared by the author.