Robust Variational Autoencoders for Outlier Detection and Repair of Mixed-Type Data

Simão Eduardo[1*] **Alfredo Nazábal**[2*] **Christopher K. I. Williams**[12] **Charles Sutton**[123]
[1]School of Informatics, University of Edinburgh, UK
[2]The Alan Turing Institute, UK; [3]Google Research

1 Dataset details

Table 1: Properties of the tabular datasets employed in the experiments.

Dataset	Rows	Real features	Categorical features
Wine	6497	12	1
Adult	32561	5	10
Credit Default	30000	14	10
Letter	20000	0	17

2 Derivation of Coordinate Step for Weights

From (6), we can write the bound $\mathcal{L}$ on $\log p(X)$ with respect to $\pi_{nd}(\mathbf{x}_n)$ as

$$\begin{aligned}\mathcal{L} \;\propto\; & \sum_{n=1}^{N}\sum_{d=1}^{D}\pi_{nd}(\mathbf{x}_n)\mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)}[\log p_\theta(x_{nd}|\mathbf{z}_n)] \\ & + \sum_{n=1}^{N}\sum_{d=1}^{D}(1-\pi_{nd}(\mathbf{x}_n))\,\mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)}[\log p_0(x_{nd})] \\ & - \pi_{nd}(\mathbf{x}_n)\log\frac{\pi_{nd}(\mathbf{x}_n)}{\alpha} \\ & - (1-\pi_{nd}(\mathbf{x}_n))\log\frac{1-\pi_{nd}(\mathbf{x}_n)}{1-\alpha}\end{aligned}$$

The derivative of this bound w.r.t. $\pi_{nd}(\mathbf{x}_n)$ can be easily computed:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\pi_{nd}(\mathbf{x}_n)} &= \mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)}[\log p_\theta(x_{nd}|\mathbf{z}_n)] \\ &- \mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)}[\log p_0(x_{nd})] \\ &- \log\frac{\pi_{nd}(\mathbf{x}_n)}{\alpha} + \log\frac{1-\pi_{nd}(\mathbf{x}_n)}{1-\alpha}\end{aligned}$$

Evaluating $\frac{\partial\mathcal{L}}{\partial\pi_{nd}(\mathbf{x}_n)}=0$ and solving for $\pi_{nd}(\mathbf{x}_n)$, we obtain the coordinate update for the weights:

$$\hat{\pi}_{nd}(\mathbf{x}_n)=\frac{1}{1+\exp\left(-\left(\mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)}[\log\frac{p_\theta(x_{nd}|\mathbf{z}_n)}{p_0(x_{nd})}]+\log\frac{\alpha}{1-\alpha}\right)\right)},$$

which is the sigmoid function applied to the expected log density ratio between the clean model and the outlier model plus the logit of the prior probability.

3 Additional details for RVAE and Competing Methods

- **Data Pre-Processing:** For all models and competitor methods the real features were standardized, i.e. subtracting by the empirical mean and dividing by standard deviation. One-hot encoding for categorical features was used depending on the method, as defined below.
- **Validation Set**: 10% of each dataset was separated from the rest of the data to be employed as a validation set, with known ground truth of the corrupted cells, for hyper-parameter selection on all baselines. Our RVAE model does not use this validation set in any of the experiments.
- **Hyper-parameter Selection:** The criterion used for hyper-parameter selection on all baselines was the AVPR in the outlier detection task registered in the validation set. The exception is the Marginals Distribution baseline, where the number of components is chosen via BIC score.

3.1 RVAE, VAE, DeepRPCA and Conditional Predictor methods

- **Architecture:** For VAE, RVAE and DeepRPCA, we used an intermediate hidden layer in both encoder and decoder, size 400. The latent space dimension was chosen to be size 20. In the Cond-Pred baseline, we found that a deep version of the base conditional predictor was superior than

[*] Joint first authorship.

a linear one in both outlier and repair metrics. Two inner layers of dimension 200 and 50 for each predictor were employed, which made this model substantially slower than all autoencoder baselines. The non-linear activation used throughout was ReLU (Rectified Linear Unit).

- **Optimization:** We used the Adam optimizer as provided in Pytorch to train the encoder and decoder parameters, for all VAE-based models. In the case of RVAE, VAE and CondPred models we minimized their respective negative losses. In CondPred, each conditional predictor had its own Adam optimizer, we found this to work better. The initial learning rate used in experiments was 0.001. All models ran for 100 epochs on all datasets, noise levels and noise processes. Since access to a validation set is impossible in a unsupervised learning setting, no standard early stopping can be defined.

  In the case of DeepRPCA, we use Adam to train the encoder and decoder parameters, as in the original paper. The optimization process used to obtain data matrix R, and noise matrix S, was carried out using ADMM (Alternating Method of Multipliers). We use row structured $\ell_{2,1}$ version of DeepRPCA for outlier detection as it performed better. In order for the ADMM optimization procedure to work, in terms of categorical reconstruction loss we follow the work in (Udell et al., 2016) (Section 6, Categorical PCA), using cross-entropy loss to aggregate the different one-hot dimensions. This yielded better experimental results than one-vs-all type aggregation. All models ran for 20 ADMM iterations, each using 10 intermediate epochs of Adam to train the autoencoder component R. All the above are in accordance to DeepRPCA paper (Zhou and Paffenroth, 2017). It should be noted that, in our experiments, running more ADMM iterations eventually led to performance degradation, even after an extensive hyper-parameter search and optimizer tuning.

- ℓ_2 **Regularization (Weight Decay):** We used the weight decay option of the Adam optimizer in Pytorch. We performed a grid search over the values $\lambda_{\ell_2} = [0, 0.1, 1, 5, 10, 100]$, each run for 100 epochs, and chose the best on the validation set The search was performed for each dataset in Table 1. For VAE, the best performance was obtained with we $\lambda_{\ell_2} = 0.1$ in the Letter dataset, $\lambda_{\ell_2} = 1$ in the Adult dataset and $\lambda_{\ell_2} = 10$ in the Wine and Credit Default datasets. For the conditional predictors, the best performance was obtained for $\lambda_{\ell_2} = 1$ in Adult, Credit default and Letter datasets, and $\lambda_{\ell_2} = 5$ in the Wine dataset. For RVAE-CVI and RVAE-AVI no regularization was needed.

- **Categorical Encoding:** VAE, RVAE and CondPred models we used categorical embedding matrices to codify the categorical features at the input level of the encoder. The dimensionality used in all experiments was size 50, as it provided generally good results. For CondPred, embeddings were not shared between individual feature predictors. In the case of DeepRPCA we had to use on-hot encoding, as this was the only way to make the ADMM procedure to work properly, given the projection step (using proximity operator). This relies on subtracting the noise matrix S from the data matrix X, which is non-trivial using embedding representations. One-hot encoding is standard in PCA-type models when dealing with categorical features.

- **DeepRPCA hyper-parameter:** The coefficient that regulates how many of the data-points (cells) will be represented by sparse matrix S was chosen from the range $\lambda = [0.001, 0.01, 0.1, 1]$. The best outlier detection performance was obtained for 0.01 in Wine and Adult datasets and 0.1 in Credit Default and Letter datasets.

- **RVAE (hyper-parameters):** The value for the prior probability α was set to 0.95 throughout (it is fair to assume in general that most of the data is clean). A full evaluation on its effect on the performance of the model was conducted in the main text. In the case of the hyper-parameter S of the outlier model for real features, we used 2 throughout, with good results. This was the setting used for all RVAE-based models in the experiment section, and the validation set was not employed at any time while selecting parameters.

- **Encoder of the weights for RVAE-AVI:** We used a feed-forward neural network with the same architecture as the one specified above for the encoder of , which parameterizing the variational distribution of the latent space. An intermediate hidden layer of size 400 was used. In this case, no coordinate optimization procedure was performed.

3.2 OC-SVM

We use a scikit-learn implementation, with RBF (radial basis function) kernel. We conducted an hyper-parameter search on both ν and γ, from 0 to 1 in intervals of 0.1. The best performance for all the datasets was obtained with $\nu = 0.2$ and $\gamma = 0.1$, on the validation set.

3.3 Marginal Method:

The Marginal method has no hyper-parameters to tune, apart from the maximum number of Gaussian Mixture Model components that can be selected by BIC score. We found a maximum of 40 components to be sufficient.

3.4 OCSVM + Marginals method

We employed a combination of both the OCSVM and Marginals implementations described above. The parameters were selected based on the previous details ($\nu = 0.2$,$\gamma = 0.1$ and maximum number of components of the GMM set to 40).

3.5 Isolation Forest:

We use scikit-learn implementation. A maximum number of samples of 50% of the size of the datasets, and a contamination parameter of 0.2 seemed to work best for all the scenarios. Again, these parameters were selected using the validation set.

4 Outlier detection additional details

In this section, we present the full disclosure of all the models in both row and cell outlier detection in each of the datasets of the experiments, in Figures 1-4 Notice

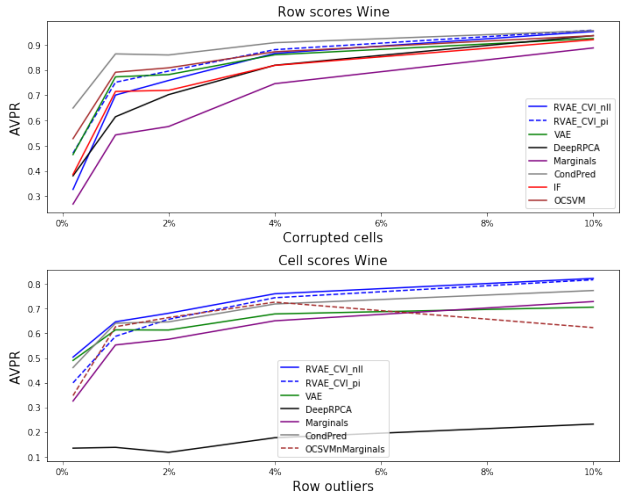

Figure 1: Row and cell outlier detection scores on Wine dataset in 5 different cells corruption levels. Upper figure shows the AVPR at row level. Lower figure shows the AVPR at cell level.

that RVAE-CVI is stable across datasets and noise corruption levels, while other models suffer in some specific datasets for either row or cell outlier detection.

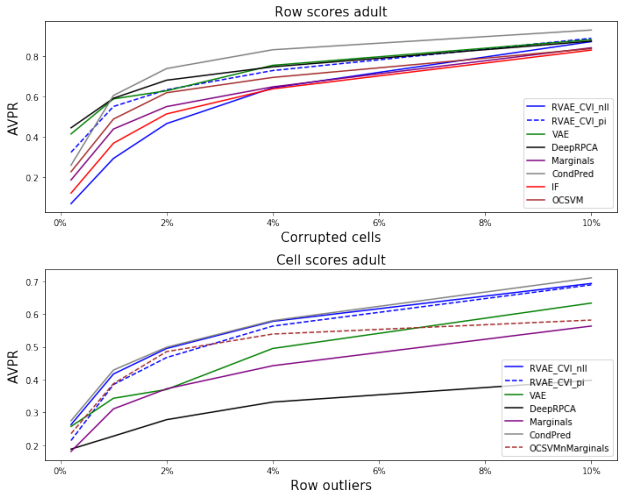

Figure 2: Row and cell outlier detection scores on Adult dataset in 5 different cells corruption levels. Upper figure shows the AVPR at row level. Lower figure shows the AVPR at cell level.

5 Repair additional details

In this section, we present the full disclosure of all the models in while repairing dirty cells in each of the datasets of the experiments, in Figure 5. RVAE-CVI performs better than the other methods for low level corruption, except for the adult dataset where RVAE-CVI and the conditional predictor are equivalent and the Letter dataset, where the conditional predictor does slightly better.

6 RVAE-CVI vs RVAE-AVI

We present here the AVPR evolution of RVAE-CVI and RVAE-AVI for each dataset and all noise corruption levels. RVAE-CVI outperforms RVAE-AVI in all datasets in both cell and row outlier detection, obtaining a similar performance only for the Letter dataset.

Additionally, in Figure 7 we show the difference in repair performance of the dirty cells for both models. We can observe that RVAE-CVI performs better than RVAE-AVI for all datasets and noise corruption levels.

7 Different noise processes additional details

In this section we present all the results in row and cell outlier detection and repair for all six combinations of noise processes, which are:

- Gaussian noise ($\mu = 0$, $\eta = 5\hat{\sigma}_d$), Tempered Categorical ($\beta = 0$)

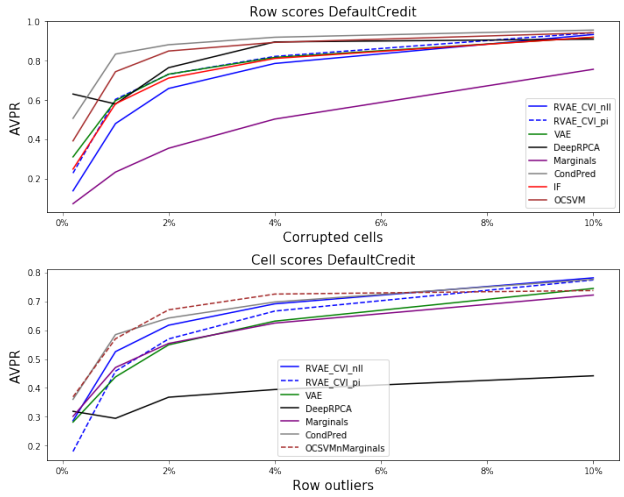

Figure 3: Row and cell outlier detection scores on Credit default dataset in 5 different cells corruption levels. Upper figure shows the AVPR at row level. Lower figure shows the AVPR at cell level.

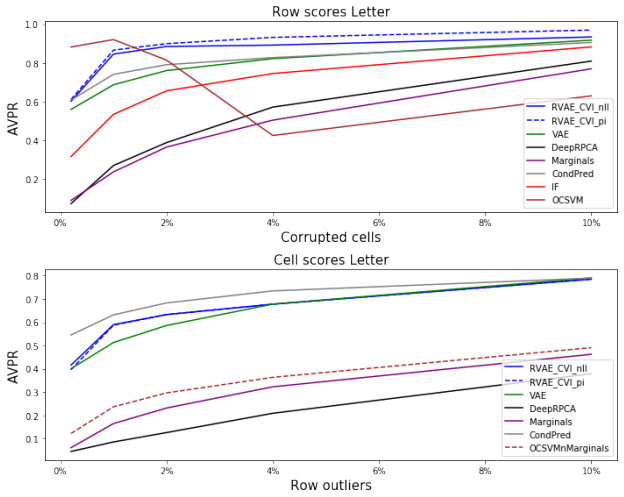

Figure 4: Row and cell outlier detection scores on Letter dataset in 5 different cells corruption levels. Upper figure shows the AVPR at row level. Lower figure shows the AVPR at cell level.

- Laplace noise ($\mu = 0$, $\eta = 4\hat{\sigma}_d$), Tempered Categorical ($\beta = 0.5$)
- Laplace noise ($\mu = 0$, $\eta = 4\hat{\sigma}_d$), Tempered Categorical ($\beta = 0.8$)
- Laplace noise ($\mu = 0$, $\eta = 8\hat{\sigma}_d$), Tempered Categorical ($\beta = 0.8$)
- Log normal noise ($\mu = 0$, $\eta = 0.75\hat{\sigma}_d$), Tempered Categorical ($\beta = 0$)
- Mixture of two Gaussian noise components ($\mu_1 = -0.5, \eta_1 = 3\hat{\sigma}_d$, with probability 0.6 and $\mu_2 = 0.5, \eta_2 = 3\hat{\sigma}_d$ with probability 0.4), Tempered Categorical ($\beta = 0$)

Figures 8-10 show a disclosure of the full results on all noise processes across the different models for both row and cell outlier detection and repair.

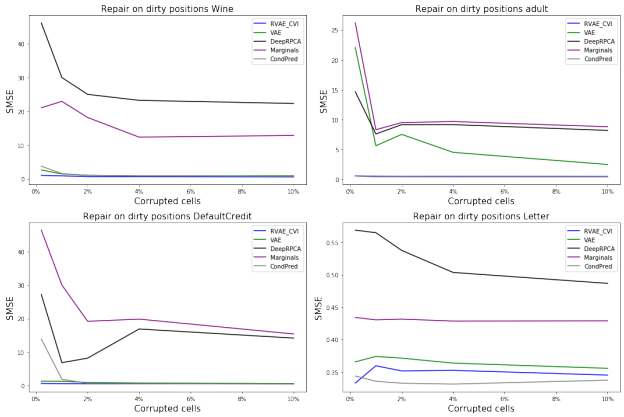

Figure 5: Repair performance on the dirty cells of all models for each datasets

8 Error Bars per Noise Level

Here, we show results for VAE, RVAE and CondPred with error bars provided for each noise level. The error bars were obtained by generating five independent instances of corruption – randomly corrupting different cells in the dataset each time. The corruption process is the same as in section 4.5. of the main paper. The inference mechanism for repair is MAP like in section 4.6. of main paper. We report the results both for OD and repair (Figure 11). In lower noise levels, the standard deviation tends to be higher, more significantly in repair (last row, Figure 11). Since fewer cells are affected at lower noise levels, this leads to more diverse behaviours in repair and OD, and thus to larger error bar.

We can see that the main conclusions about the "ranking" of our method against baselines still holds in either OD or repair. Further, in repair, in the two lowest noise levels RVAE (MAP) seems to less dependent on the corrupted cells (see Adult and Credit Default figures, in Figure 11).

To further complete this analysis, we provide in Table 2 the p-values computed from an independent t-test between RVAE, VAE and CondPred. These were averaged across datasets and noise levels.

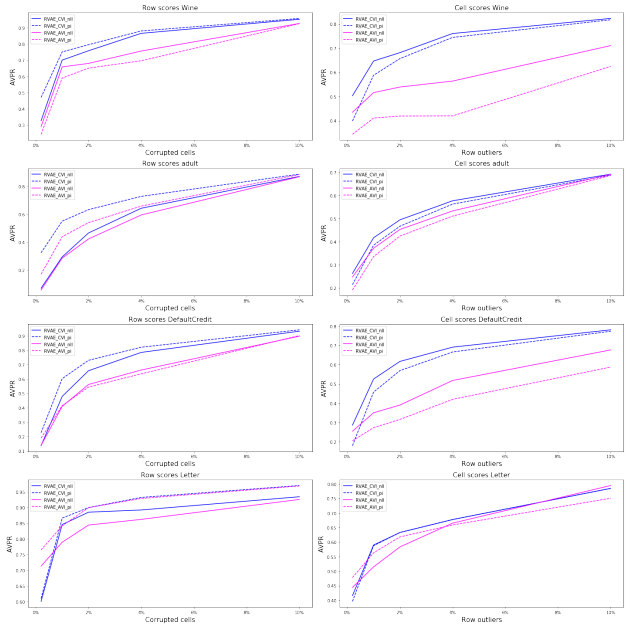

Figure 6: Comparison between RVAE-CVI and RVAE-AVI for each dataset in row outlier detection (left figures) and cell outlier detection (right figures)

Table 2: Independent t-test between RVAE, VAE and CondPred. If p-values in range 0.05-0.10 assume that models have different performance.

	avg. p-values RVAE vs CondPred	avg. p-values RVAE vs VAE
Cell AVPR	0.121	0.070
Row AVPR	0.040	0.227
Repair SMSE	0.025	0.013

9 Different OD Task: RVAE vs ABDA

In this section we compare RVAE to ABDA (Vergari et al., 2019), a recent algorithm employed both in OD and missing data imputation. We followed the details in the OD section of the ABDA paper and compare RVAE with ABDA in terms of row AUC ROC as used therein (we use the results reported by the ABDA authors). Table 3 shows that we perform better in average than ABDA, with 7 out 10 cases being better in OD. Notice that, the noising scenarios for these datasets (described in (Goldstein and Uchida, 2016)) are based of standard row outlier detection, where one or some classes are considered normal while another class or classes are considered outliers. This scenario is completely different to the scenarios described on our paper. In our work, we assume that some cells in the data corrupt several rows in a tabular dataset, and we need to detect and correct them. These experiments showcase the robustness of RVAE to a different outlier detection process.

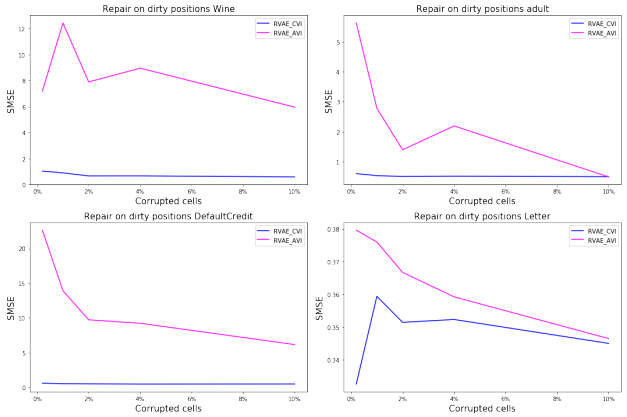

Figure 7: Comparison between RVAE-CVI and RVAE-AVI for each dataset in repair of dirty dells. The lower SMSE the better.

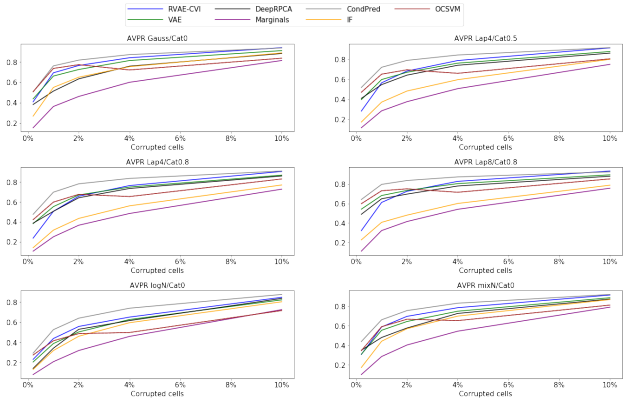

Figure 8: Row outlier detection across all models and noise processes, averaging all datasets

10 Different Inference Method

In this section, we compare the MAP inference (reconstruction, eq. (12)) for VAEs employed throughout the paper with more powerful inference methods (Figure 12). In particular, we provide results for pseudo-Gibbs sampling, (see Rezende et al., 2014, section F), applying it on a trained RVAE at evaluation time. The final repair estimate was provided after running the MCMC procedure for $T = 5$ iterations (samples), since larger values of T provided marginal improvements. We used the same scenarios of sections 4.5 and 4.6 of paper.

A mask removing anomalous entries needs to be either defined, or inferred. We provide two options to do this automatically:

- **OneStage** (Algorithm 1): Treat all cells in a row as anomalous and perform pseudo-Gibbs, for T iterations. Final repair value is line 6 of Algorithm 1, and OD is line 8.

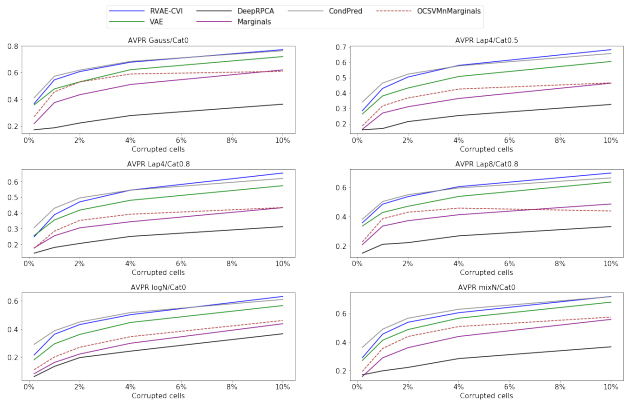

Figure 9: Cell outlier detection across all models and noise processes, averaging all datasets

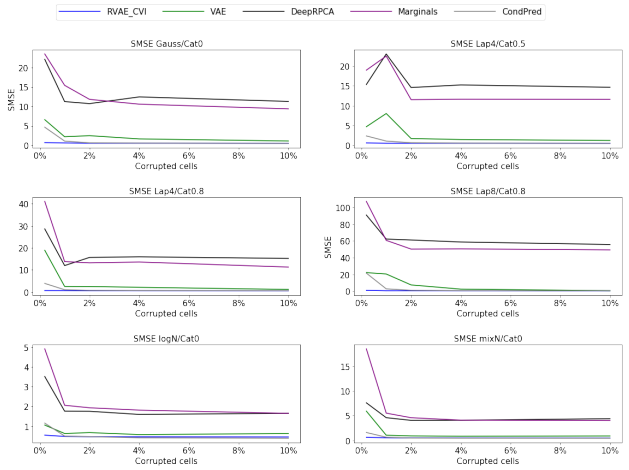

Figure 10: Repair of dirty cells across all models and noise processes, averaging all datasets

- **TwoStage** (Algorithm 2): Use OneStage, obtaining a more stable estimate of π_{nd}, then sample mask w_{nd} using it to perform pseudo-Gibbs (as described in (Rezende et al., 2014)). The assumed clean cells (i.e. $w_{nd} = 1$) have their value x^o_{nd} fixed throughout the MCMC chain (of T iterations). Meanwhile cells that are dirty are initialized with mean behaviour imputation, i.e. $\bar{x}_{nd}$ (Algorithm 2, line 4). For continuous features since our data is standardized ($\hat{\mu}_d = 0$), so we use 0. For categorical features, given our VAE models use normalized (word) embeddings, we use vectors of the same dimension with zeros – such strategy has been applied for imputation when using embeddings. Final repair value is line 8 of Algorithm 2, and OD is in line 2 (i.e. $\hat{\pi}_n$ from *OneStage*).

Note that in the *OneStage* method the mask $\mathbf{w}_n$ is not inferred, while in *TwoStage* it is. In addition, we remind the reader that $\mathbf{x}^o_n$ is the observed row, which can be clean or dirty.

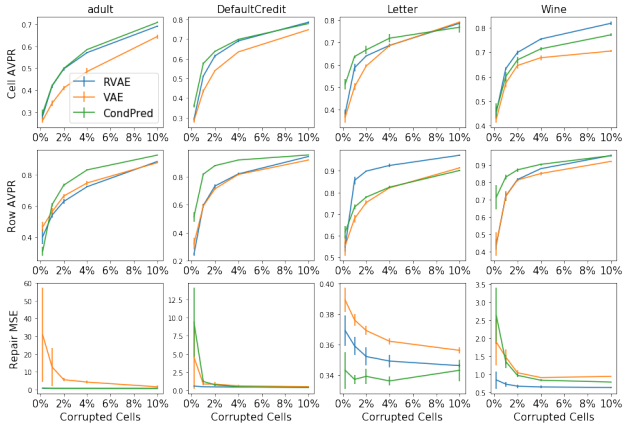

Figure 11: Plots with error-bars for each dataset (column), using 5 different instances of corruption, at each corruption level (x-axis). We show cell OD (upper row), row OD (middle row) and repair (lower row).

Table 3: Comparison between RVAE and ABDA in row AUC ROC for 10 different datasets.

Dataset	AUC RVAE	AUC ABDA
Letter	**0.8359**	0.7036
Breast	0.9815	**0.9836**
Pen Global	**0.9316**	0.8987
Pen Local	0.9053	**0.9086**
Satellite	**0.9460**	0.9455
Thyroid	0.8211	**0.8488**
Shuttle	**0.9985**	0.7861
Aloi	**0.5515**	0.4720
Speech	**0.5584**	0.4696
KDD	**0.9993**	0.9979
Average	**0.8529**	0.8014

Figure 12 shows that there were gains on average in OD and repair using *TwoStage*, particularly for repair at low noise levels. These are still close to MAP, specifically in the case of higher noising levels.

For completion, we disclose in Figure 13 the comparison across the inference methods per dataset. In general, we can see that TwoStage has better repair performance (last row of Figure 13), particularly in low level noise.

Lastly, other methods like (Mattei and Frellsen, 2019) could also have been used to improve repair. However, more powerful inference schemes can sometimes lead to overfitting to noise. On the other hand, inference schemes like MCMC (vs MAP) can provide more stable solutions (lower error bars), particularly in lower noise levels or in smaller datasets (number of rows).

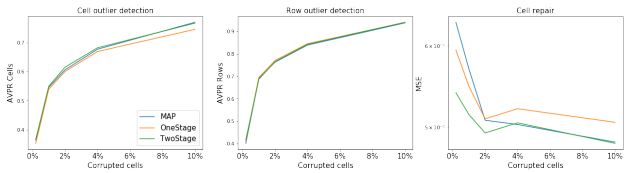

Figure 12: Comparison between MAP, OneStage, TwoStage inference methods in terms of both row / cell OD, and repair.

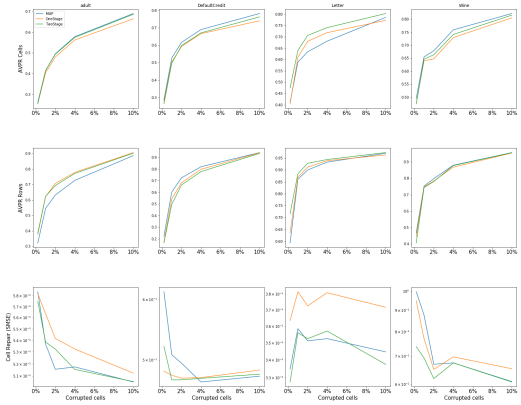

Figure 13: Comparison between MAP, OneStage, TwoStage, CondPred inference methods in terms of both row / cell OD, and repair. Results for each dataset.

Algorithm 1 OneStage: pseudo-Gibbs sampling

procedure OneStage(T, fixed $\{\phi, \theta\}$)
  $\mathbf{x}_n^{(1)} = \mathbf{x}_n^o$
  for $1, ..., T$ **do**
    $\mathbf{z}_n^{(t+1)} \sim q_\phi(\mathbf{z}|\mathbf{x}_n^{(t)})$
    $\tilde{\mathbf{x}}_n^{(t+1)} \sim p_\theta(\mathbf{x}|\mathbf{z}_n^{(t+1)})$
  $\hat{\mathbf{x}}_n^i = \tilde{\mathbf{x}}_n^{(T+1)}$ ▷ for repair
  $\hat{\mathbf{z}}_n^i = \mathbf{z}_n^{(T+1)}$
  $\hat{\pi}_{nd} = g\left(r(\mathbf{x}_{nd}^o, \hat{\mathbf{z}}_n^i) + \log \frac{\alpha}{1-\alpha}\right)$ ▷ eq.(9), OD
  return $(\hat{\mathbf{x}}_n^i, \hat{\mathbf{z}}_n^i, \hat{\pi}_n)$

Algorithm 2 TwoStage: pseudo-Gibbs sampling

procedure TwoStage(T, fixed $\{\phi, \theta\}$)
  $(\hat{\mathbf{x}}_n^i, \hat{\mathbf{z}}_n^i, \hat{\pi}_n) \leftarrow$ OneStage(T,$\{\phi, \theta\}$)
  $\hat{w}_{nd} \sim q_{\hat{\pi}_{nd}}(w_{nd})$
  $x_{nd}^{(1)} = \hat{w}_{nd} \times x_{nd}^o + (1 - \hat{w}_{nd}) \times \bar{x}_{nd}$
  for $1, ..., T$ **do**
    $\mathbf{z}_n^{(t+1)} \sim q_\phi(\mathbf{z}|\mathbf{x}_n^{(t)})$
    $\tilde{\mathbf{x}}_n^{(t+1)} \sim p_\theta(\mathbf{x}_n|\mathbf{z}_n^{(t+1)})$
  $\hat{x}_{nd}^i = \hat{w}_{nd} \times x_{nd}^o + (1 - \hat{w}_{nd}) \times \tilde{x}_{nd}^{(T+1)}$
  $\hat{\mathbf{z}}_n^i = \mathbf{z}_n^{(T+1)}$
  return $(\hat{\mathbf{x}}_n^i, \hat{\mathbf{z}}_n^i, \hat{\pi}_n)$

References

Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4): e0152173, 2016.

Pierre-Alexandre Mattei and Jes Frellsen. Miwae: Deep generative modelling and imputation of incomplete data sets. In *ICML*, 2019.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Madeleine Udell, Corinne Horn, Reza Zadeh, Stephen Boyd, et al. Generalized low rank models. *Foundations and Trends® in Machine Learning*, 9(1):1–118, 2016.

Antonio Vergari, Alejandro Molina, Robert Peharz, Zoubin Ghahramani, Kristian Kersting, and Isabel Valera. Automatic bayesian density analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5207–5215, 2019.

Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674. ACM, 2017.