Boosting heterogeneous VAEs via multi-objective optimization

Adrián Javaloy*
Department of Computer Science
Saarland University
Saarbrücken, Germany

Maryam Meghdadi
Department of Computer Science
Saarland University
Saarbrücken, Germany

Isabel Valera
Department of Computer Science, Saarland University
MPI Software Systems
Saarbrücken, Germany

Abstract

Variational autoencoders (VAEs) have been successfully applied to complex input data such as images and videos. Counterintuitively, their application to simpler, heterogeneous data—where features are of different types, often leads to underwhelming results. While the goal in the heterogeneous case is to *accurately approximate all observed features*, VAEs often perform poorly in a subset of them. In this work, we study this *feature overlooking* problem through the lens of multitask learning (MTL), relating it to the problem of *negative transfer* and the interaction between gradients from different features. With these new insights, we propose to train VAEs by leveraging off-the-shelf solutions from the MTL literature based on multi-objective optimization. Furthermore, we empirically demonstrate how these solutions significantly *boost* the performance of different VAE models and training objectives on a large variety of heterogeneous datasets.

1 Introduction

Deep generative models have enjoyed great success in the recent years. In the case of variational autoencoders (VAEs) [15], they have been successfully applied to different domains such as images, text, and temporal data [27, 38, 43, 22]. It thus comes as a surprise that their application to heterogeneous data—where each feature is of a different type—remains a challenge, as models tend to focus on modelling a subset of features while overlooking the rest, a phenomenon we refer to as *feature overlooking*.

In the recent years, a number of works have put attention on heterogeneous VAEs, and thus ways of addressing feature overlooking. As a result, new pre-processing mechanisms [11] and models [26, 2] to palliate feature overlooking have been proposed, as well as models that attempt to completely side-step the problem [20]. Interestingly, Nazabal et al. [26] hypothesized that feature overlooking was a result of disparities between gradients with respect to each feature, and Javaloy and Valera [11] drew some connections between heterogeneous VAEs and multitask learning (MTL).

In this work we formalize these connections, showing that heterogeneous VAEs are intimately related with hard parameter sharing architectures in MTL, and that feature overlooking can be thought of as a particular instance of negative transfer in MTL [29]. As a result, we propose finding the proper

*Correspondence to ajavaloy@cs.uni-saarland.de.

NeurIPS 2021 Workshop Your Model Is Wrong: Robustness and Misspecification in Probabilistic Modeling.

algorithm for training, thus decoupling modelling from optimization, and leveraging off-the-shelf MTL algorithms based on multi-objective optimization (MOO). We empirically validate our findings, showing that this multi-objective "cocktail" of algorithms significantly boosts the performance on a great variety of models, training losses, and heterogeneous datasets.

2 Preliminaries and background

2.1 Variational autoencoders for heterogeneous data

Variational autoencoders (VAEs) [15] are a popular class of deep generative models to optimize the marginal likelihood (*evidence*) of our dataset—i.i.d. samples from a random variable $\mathbf{X}$—by assuming the existence of a latent random variable $\mathbf{Z}$ that allows capturing the statistical dependencies across observed features, while leading to a tractable objective to optimize.

A VAE is composed of two main components. First, a generative model $p_\theta(\mathbf{X}, \mathbf{Z}) = p_\theta(\mathbf{X}|\mathbf{Z})p(\mathbf{Z})$, where $p(\mathbf{Z})$ is the prior distribution of $\mathbf{Z}$, and $p_\theta(\mathbf{X}|\mathbf{Z})$ is the likelihood of $\mathbf{X}$ given $\mathbf{Z}$, whose parameters are parametrized by a neural network η (*decoder*) with input $\mathbf{Z}$ and learnable parameters θ, that is, $p_\theta(\mathbf{X}|\mathbf{Z}) = p(\mathbf{X}; \eta(\mathbf{Z}; \theta))$. Second, an approximation to the posterior, $p(\mathbf{Z}|\mathbf{X})$, known as the variational distribution, $q_\phi(\mathbf{Z}|\mathbf{X})$, similarly parametrized by a neural network (*encoder*) that takes $\mathbf{X}$ as input, and which is governed by the set of parameters ϕ.

As mentioned above, introducing $\mathbf{Z}$ allows maximizing the log-evidence, $\log p_\theta(\mathbf{X})$, in a tractable way. More in detail, VAEs optimize the log-evidence by instead maximizing a tractable lower-bound, $\log p_\theta(\mathbf{X}) \geq \mathcal{L}(\mathbf{X}, p_\theta, q_\phi)$, and by leveraging first-order optimization techniques from deep learning. While there are a number of suitable losses (e.g., [15, 7, 17]), optimizing any of them translates to maximizing the log-evidence, and they all solely depend on θ and ϕ through p_θ and q_ϕ, respectively.

In this work, we assume $\mathbf{X}$ to be a heterogeneous random variable of size D. In other words, $\mathbf{X}$ is composed of D different univariate random variables that describe each feature of the dataset, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D]$, and each feature is potentially different from the rest (discrete vs. continuous, positive vs. real, etc). This is a key property, as it determines the choice of the parametric family for p_θ. Because there are no restrictions on each $\mathbf{x}_d$, the usual practice (see, e.g., [2, 20, 26, 24]) is to assume that the likelihood fully factorizes across features,

$$p_\theta(\mathbf{X}|\mathbf{Z}) = \prod_{d=1}^{D} p_d(\mathbf{x}_d; \eta_d(\mathbf{Z}; \theta)), \tag{1}$$

where each p_d is a potentially different parametric family (normal, log-normal, categorical, etc), and their parameters are jointly produced by the decoder given $\mathbf{Z}$ and θ.

2.2 Multitask learning and negative transfer

In multitask learning (MTL), our goal is to jointly solve a set of tasks. Let us assume that we have a set of K tasks that share input $\boldsymbol{X}$, and each task defined by its own loss function $\mathcal{L}_k$. Our goal then is to solve the following MOO problem [30]: $\min_\theta \ (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_K)$. Instead of solving this problem, in practice one instead optimizes a weighted sum of losses, $\min_\theta \sum_k \omega_k \mathcal{L}_K$, obtaining a scalar optimization problem for which first-order methods can be easily applied.

Unfortunately, this simplified objective come at the cost of *negative transfer* [29]. This phenomenon describes the negative effect that jointly optimizing different tasks can have, potentially hampering training progress and, as a result, ending up with a suboptimal model that poorly solves some tasks.

Extensive literature have studied ways to palliate negative transfer [42, 10, 35, 23, 39, 16, 19, 36, 45, 21, 33, 14]. Here, we focus on multi-objective gradient-based solutions, which study the effect of summing losses in the gradient updates of shared parameters. The premise is that the updates of θ during optimization follow the gradient $\sum_k \omega_k \nabla_\theta \mathcal{L}_k$, and thus differences in magnitude and direction across task gradients can lead to overlooking or cancelling out, respectively, the gradient updates of a subset of tasks. In the recent years, a number of works have appeared to address magnitude [5, 18, 30, 9] and direction [6, 44, 12, 40] conflicts between task gradients during training.

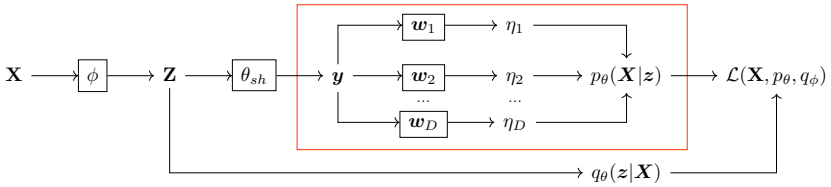

Figure 1: Simplified computational graph of a heterogeneous VAE (see § 3.1). To produce feature-specific parameters, the model splits its parameters into *shared* (ϕ, θ_{sh}) and *exclusive* ($\boldsymbol{w}_d$) parameters, resembling a hard parameter sharing architecture commonly used in MTL.

2.3 Feature overlooking in heterogeneous VAEs

Similar to the negative effect that naïvely adding up task losses has in MTL, assuming that the likelihood factorizes as in Eq. (1) in heterogeneous VAEs raises some difficulties during training. Specifically, a VAE trained on highly heterogeneous data—for example, combinations of continuous and discrete features, tends to incur some overlooking for a subset of the features, poorly performing on this subset despite obtaining a good model evidence on average, as illustrated in [26, 11, 20].

Exactly tracing back the first evidences of feature overlooking is tough, as modelling failures are rarely documented. The usual practice is to avoid heterogeneity by means of continuous extensions of discrete features (e.g., adding uniform noise [34]). In the recent years, however, feature overlooking has gained some attention. Nazabal et al. [26] first reported this problem, proposing a hierarchical VAE (HI-VAE) with the idea that more structured latent spaces should alleviate likelihood differences. More recently, Ma et al. [20] proposed VAEM, a model which attempts to side-step feature overlooking by first learning the marginals. Unfortunately, this model cannot perform better than learning the marginals independently under the presence of fully-observable data.

3 Boosting heterogeneous VAEs

3.1 Feature overlooking as negative transfer

We are now ready to draw connections between feature overlooking (§ 2.3) and negative transfer (§ 2.2), suggesting that both problems are intimately related. There are several aspects that bond feature overlooking and negative transfer together:

Goal and symptomatology. Task-impartiality in MTL [18] shares a similar spirit as heterogeneous data modelling. The goal is not to simply minimize the error, but doing so while properly learning all tasks/features, without overlooking any of them. Moreover, both settings manifest similar training difficulties, leading to models that overlook subsets of tasks/features.

Architectural similarity. In MTL, hard parameter sharing architectures [4] are characterized by having a shared network (*backbone*), followed by a set of task-specific networks (*heads*) that learn to solve their assigned task, given the (shared) output of the backbone.

One key insight is that the factorization assumed in Eq. (1) forces VAEs to follow a similar structure. As an example, suppose we use a multi-layer perceptron (MLP) as a decoder. Denote by $\boldsymbol{W}$ the last weight matrix of the decoder, and by $\boldsymbol{y}$ the output of the decoder up to that point, i.e., the input of the last layer. From Eq. (1), we know that the decoder produces all likelihood parameters, and by focusing only in the last layer we get that $\boldsymbol{\eta}(\mathbf{Z};\theta) \coloneqq [\eta_1, \eta_2, \ldots, \eta_D]\,(\mathbf{Z};\theta) = \boldsymbol{y}\boldsymbol{W}$. This last equation lets us easily identify $\boldsymbol{y}$ as the *last-shared feature of the network*, and the columns of $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_D]$ as the *exclusive parameters* for each likelihood function p_d, thus having a clear parallelism with MTL architectures. Figure 1 depicts the described decomposition.

Gradient conflicts. Similar to negative transfer, gradient conflicts across features can explain the appearance of feature overlooking. Moreover, we show that it occurs independently of the form of $\mathcal{L}$.

Feature overlooking happens when the model fails to explain some data features, and the goodness-of-fit of a model is measured by $p_d(\mathbf{x}_d|\eta_d(\mathbf{Z};\theta))$ and $p_\theta(\mathbf{X}|\mathbf{Z})$. Therefore, only the derivative of $\mathcal{L}$ *through* p_θ should lead to this unfair prioritization of features, as $\nabla_{q_\phi}\mathcal{L}$ is feature agnostic. Using

Table 1: Test reconstruction error for different VAE models and heterogeneous datasets. Numbers represent the median over five different seeds. Bold numbers are statistically different (p-value < 0.1) from their counterpart, according to a corrected paired t-test [25].

		Adult	*Credit*	*Wine*	*Diam.*	*Bank*	*IMDB*	*HI*	*rwm5yr*	*labour*
VAE	Vanilla	0.21	0.12	0.08	0.18	0.19	0.08	0.17	0.10	0.10
	MOO	**0.09**	**0.04**	**0.07**	**0.13**	**0.05**	**0.04**	**0.03**	**0.02**	**0.06**
IWAE	Vanilla	0.22	0.13	0.07	0.18	0.21	0.09	0.15	0.09	0.09
	MOO	**0.12**	**0.05**	**0.06**	**0.12**	**0.17**	**0.03**	**0.04**	**0.02**	**0.06**
DReG	Vanilla	0.23	0.13	0.07	0.17	0.20	0.08	0.15	0.09	0.09
	MOO	0.16	**0.07**	**0.06**	**0.13**	0.17	**0.04**	**0.04**	**0.02**	**0.07**
HI-VAE	Vanilla	0.12	0.10	0.12	0.11	0.11	0.07	0.10	0.04	0.10
	MOO	**0.08**	0.06	**0.11**	**0.01**	0.10	0.05	0.12	**0.02**	**0.06**

Fig. 1 as reference, we split the gradient with respect to the shared parameters ($\phi, \theta_{sh} \to p_\theta \to \mathcal{L}$) into two more parts ($\phi, \theta_{sh} \to \boldsymbol{y} \to \boldsymbol{\eta} \to p_\theta \to \mathcal{L}$) using the chain rule:

$$\nabla_{\phi,\theta_{sh}} p_\theta \cdot \nabla_{p_\theta} \mathcal{L}(\boldsymbol{X}, p_\theta, q_\phi) = \nabla_{\phi,\theta_{sh}} \boldsymbol{y} \cdot \left(\sum_d \nabla_{\boldsymbol{y}} \eta_d \cdot \nabla_{\eta_d} p_\theta \right) \cdot \nabla_{p_\theta} \mathcal{L}(\mathbf{X}, p_\theta, q_\phi). \tag{2}$$

Equation (2) shows that, in order to update any shared parameters to increase the model goodness-of-fit, the gradient computation involves a sum over the feature gradients, which can *lead to the same problems as gradient conflicts in MTL* (§ 2.2). Equation (2) also shows that this problem is agnostic to the form of $\mathcal{L}$, as the split-and-merge computations are encapsulated in a small part of the computational graph (red box in Fig. 1, and parenthesis in Eq. (2)).

3.2 The multi-objective cocktail

We have shown in § 3.1 that feature overlooking can be explained via differences in feature gradients, just as negative transfer, and that heterogeneous VAEs exhibit a similar structure as hard parameter sharing architectures in MTL. These similarities allow us to leverage existing solutions in MTL based on MOO to palliate conflicting gradients. More in detail, these solutions attempt to homogenize the gradient across tasks before adding them up, so that the contribution of each task to the parameters update is similar, and thus no task/feature is overlooked.

Limited overhead. Instead of trying to homogenize the gradients with respect to the shared parameters, $\nabla_{\phi,\theta_{sh}} p_\theta \cdot \nabla_{p_\theta} \mathcal{L}$ in Eq. (2) and Fig. 1 show that it is enough to homogenize the feature gradients with respect to $\boldsymbol{y}$, that is, to homogenize $\nabla_{\boldsymbol{y}} \eta_d \cdot \nabla_{\eta_d} p_\theta \cdot \nabla_{p_\theta} \mathcal{L}$ for all $d = 1, 2, \ldots, D$. This is a common choice in MTL to reduce computational overhead when possible.

Choosing algorithms. Similar to the regularization cocktail proposed by Kadra et al. [13] to train neural networks for tabular data, we propose a *multi-objective cocktail* to find the proper combination of MOO algorithms per dataset and model, running a hyperparameter sweep where we consider the use of a MOO algorithm as a binary hyperparameter.

4 Results and conclusions

To assess MOO algorithms on heterogeneous VAEs, we compare model performance in terms of reconstruction error. We consider 9 different heterogeneous datasets—with (log-)normal, categorical, and Poisson likelihoods—and 4 different models: VAE [15], IWAE [3], DReG [37], and HI-VAE [26]. A detailed description of the experimental setup can be found in Appendix A.

For the hyperparameter sweep, we consider all possible combinations of algorithms that deal with disparities in gradient magnitude [5, 30, 18], and gradient direction [44, 6]. Table 1 summarizes the final results. It shows that finding the proper combination of MOO algorithms significantly boosts the results across all models and datasets. While there is one case where the results do not improve (HI-VAE in HI), for most of the settings the improvements are outstanding. For example, all models

perform the best in *rwm5yr* after applying MOO. Remarkably, the original VAE heavily benefits from MOO, suggesting that simpler models can obtain outstanding results if we properly optimize them.

In this paper, we have bridged the gap between VAEs and MTL, and demonstrated that we can leverage existing knowledge from other fields to improve probabilistic models. As a promising research direction, we hope to explore other scenarios where MOO can further benefit probabilistic models, such as Gaussian Processes [41] or probabilistic multi-modal data modelling [31].

References

[1] R package datasets. URL `https://vincentarelbundock.github.io/Rdatasets/datasets.html`.

[2] Daniel Barrejón, Pablo M Olmos, and Antonio Artés-Rodríguez. Medical data wrangling with sequential variational autoencoders. arXiv preprint arXiv:2103.07206, 2021.

[3] Yuri Burda, Roger B Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In ICLR (Poster), 2016.

[4] Richard Caruana. Multitask learning: A knowledge-based source of inductive bias. In Proceedings of the Tenth International Conference on Machine Learning, pages 41–48. Morgan Kaufmann, 1993.

[5] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In International Conference on Machine Learning, pages 794–803. PMLR, 2018.

[6] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/16002f7a455a94aa4e91cc34ebdb9f2d-Abstract.html`.

[7] Adji B Dieng, Dustin Tran, Rajesh Ranganath, John Paisley, and David M Blei. Variational inference via x upper bound minimization. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pages 2729–2738, 2017.

[8] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[9] Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. Comptes Rendus Mathematique, 350(5):313–318, 2012. ISSN 1631-073X. doi: https://doi.org/10.1016/j.crma.2012.03.014. URL `https://www.sciencedirect.com/science/article/pii/S1631073X12000738`.

[10] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pages 3854–3863. PMLR, 2020. URL `http://proceedings.mlr.press/v119/guo20e.html`.

[11] Adrián Javaloy and Isabel Valera. Lipschitz standardization for multivariate learning. arXiv preprint arXiv:2002.11369, 2020.

[12] Adrián Javaloy and Isabel Valera. Rotograd: Dynamic gradient homogenization for multi-task learning. arXiv preprint arXiv:2103.02631, 2021.

[13] Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Regularization is all you need: Simple neural nets can excel on tabular data. arXiv preprint arXiv:2106.11189, 2021.

[14] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7482–7491, 2018.

[15] Diederik Kingma, Max Welling, et al. Auto-encoding variational bayes. In Proceedings of the International Conference on Learning Representations (ICLR), volume 1, 2014.

[16] Hila Levi and Shimon Ullman. Multi-task learning by a top-down control network. arXiv preprint arXiv:2002.03335, 2020.

[17] Yingzhen Li and Richard E Turner. Rényi divergence variational inference. In Proceedings of the 30th International Conference on Neural Information Processing Systems, pages 1081–1089, 2016.

[18] Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=IMPnRXEWpvr.

[19] Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-end multi-task learning with attention. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 1871–1880. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00197. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Liu_End-To-End_Multi-Task_Learning_With_Attention_CVPR_2019_paper.html.

[20] Chao Ma, Sebastian Tschiatschek, Richard Turner, José Miguel Hernández-Lobato, and Cheng Zhang. Vaem: a deep generative model for heterogeneous mixed type data. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 11237–11247. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/8171ac2c5544a5cb54ac0f38bf477af4-Paper.pdf.

[21] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 1851–1860. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00195. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Maninis_Attentive_Single-Tasking_of_Multiple_Tasks_CVPR_2019_paper.html.

[22] Nazanin Mehrasa, Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. A variational auto-encoder model for stochastic point processes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[23] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 3994–4003. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.433. URL https://doi.org/10.1109/CVPR.2016.433.

[24] Pablo Moreno-Muñoz, Antonio Artés-Rodríguez, and Mauricio A Álvarez. Heterogeneous multi-output gaussian process prediction. arXiv preprint arXiv:1805.07633, 2018.

[25] Claude Nadeau and Yoshua Bengio. Inference for the generalization error. Machine learning, 52(3):239–281, 2003.

[26] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. Pattern Recognition, 107:107501, 2020.

[27] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. Advances in neural information processing systems, 29:2352–2360, 2016.

[28] Tom Rainforth, Adam Kosiorek, Tuan Anh Le, Chris Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter variational bounds are not necessarily better. In International Conference on Machine Learning, pages 4277–4285. PMLR, 2018.

[29] Sebastian Ruder. An overview of multi-task learning in deep neural networks. CoRR, abs/1706.05098, 2017. URL http://arxiv.org/abs/1706.05098.

[30] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems 31, pages 525–536. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7334-multi-task-learning-as-multi-objective-optimization.pdf.

[31] Yuge Shi, Narayanaswamy Siddharth, Brooks Paige, and Philip HS Torr. Variational mixture-of-experts autoencoders for multi-modal deep generative models. arXiv preprint arXiv:1911.03393, 2019.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[33] Trevor Standley, Amir Roshan Zamir, Dawn Chen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pages 9120–9132. PMLR, 2020. URL http://proceedings.mlr.press/v119/standley20a.html.

[34] Suwon Suh and Seungjin Choi. Gaussian copula variational autoencoders for mixed data. arXiv preprint arXiv:1604.04960, 2016.

[35] Ximeng Sun, Rameswar Panda, Rogério Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/634841a6831464b64c072c8510c7f35c-Abstract.html.

[36] Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: The TC algorithm. In Lorenza Saitta, editor, Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996, pages 489–497. Morgan Kaufmann, 1996.

[37] George Tucker, Dieterich Lawson, Shixiang Gu, and Chris J Maddison. Doubly reparameterized gradient estimators for monte carlo objectives. arXiv preprint arXiv:1810.04152, 2018.

[38] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 19667–19679. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/e3b21256183cf7c2c7a66be163579d37-Paper.pdf.

[39] Simon Vandenhende, Stamatios Georgoulis, Luc Van Gool, and Bert De Brabandere. Branched multi-task networks: Deciding what layers to share. In 31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020. BMVA Press, 2020. URL https://www.bmvc2020-conference.com/assets/papers/0213.pdf.

[40] Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=F1vEjWK-lH_.

[41] Christopher K Williams and Carl Edward Rasmussen. Gaussian processes for machine learning, volume 2. MIT press Cambridge, MA, 2006.

[42] Sen Wu, Hongyang R Zhang, and Christopher Ré. Understanding and improving information transfer in multi-task learning. arXiv preprint arXiv:2005.00944, 2020.

[43] Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. Variational autoencoder for semi-supervised text classification. In Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[44] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 5824–5836. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/3fe78a8acf5fda99de95303940a2420c-Paper.pdf.

[45] Amir Roshan Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 3712–3722. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00391. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Zamir_Taskonomy_Disentangling_Task_CVPR_2018_paper.html.

A Experimental details

A.1 Dataset descriptions

Likelihood selection. Choosing the proper likelihood is a hard task which requires expert-domain knowledge for each specific setting. We attempt to simplify this process, and instead automatize likelihood selection based on basic properties of the data that can be programmatically verified. Specifically, we use the following criteria:

Real-valued:	$x_d \sim \mathcal{N}(\mu, \sigma)$
Positive real-valued:	$x_d \sim \log\mathcal{N}(\mu, \sigma)$
Count:	$x_d \sim \text{Poiss}(\lambda)$
Binary:	$x_d \sim \text{Bern}(p)$
Categorical:	$x_d \sim \text{Cat}(\pi_1, \pi_2, \ldots, \pi_K)$

Datasets. For the experiments shown in § 4, we use nine different heterogeneous datasets. First, we took *Adult*, *defaultCredit*, *Wine*, and *Bank marketing* datasets from the UCI repository [8]. Then, we included from the R package datasets [1] the following datasets: *Diamonds*, *Movies (IMDB)*, *Health Insurance (HI)*, *German health registry (rwm5yr)*, and *labour*. Table 2 provides the statistics per dataset in terms of sizes and number of likelihoods. It is important to remark that the *IMDB* and *Adult* datasets contain NaNs values (each only in two of the features). We replace them by non-NaNs values and ignore them during training and evaluation using binary masks.

Table 2: Datasets description. The first two columns describe number of instances, N, and number of features, D. The next columns describe the number of specific likelihoods per dataset.

Dataset	N	D	*Real*	*Positive*	*Count*	*Categorical*
Adult	32561	12	0	3	1	7
Credit	30000	24	6	7	1	10
Wine	6497	13	0	11	1	1
Diamonds	53940	10	7	0	0	3
Bank	41188	21	10	0	0	11
IMDB	28819	23	4	1	10	8
HI	22272	12	5	1	0	6
rwm5yr	19609	16	0	2	3	11
labour	15992	9	3	0	2	4

Preprocessing. When parsing the dataset, we centre all real-valued features by removing their mean. We further standardize real-valued features, computing their (training) standard deviation and dividing the data by this quantity. We perform the same standardization by in the log-space for positive real-valued features. These last two steps are omitted for HI-VAE, since it uses its own normalization layer as described by Nazabal et al. [26]. We also treat non-negative as positive real-valued features by adding a negligible value of 1×10^{-20}. Finally, we make sure that the support of count, binary, and categorical features are in accordance to that of the library used during implementation by removing their minimum value in the case of binary and categorical features, and 1 in the case of count features.

Additionally, we performed some extra preprocessing to the *IMDB* dataset. In said dataset, there are ten features that contain rating percentages of users to the movies, ranging from 0 to 100, at intervals of 0.5. We convert each of them into discrete features starting from one by performing $\mathbf{x}'_d = 2\mathbf{x}_d + 1$ to each of these features, treating them afterwards as count data.

A.2 Multi-objective cocktail

As explained in § 2.2 and § 4, for the multi-objective cocktail we consider multi-objective solutions that attempt to homogenize the gradient across features. This type of solutions can be classified in two big groups. First, solutions that scale the gradients according to different criteria. That is, they replace each gradient $\nabla_d \mathcal{L}$ with $\omega_d \nabla_d \mathcal{L}$, where each MOO algorithm sets ω_d differently. Second, direction-aware algorithms. These algorithms attempt to solve issues from gradients pointing towards different places of the parameter space, thus cancelling out each other.

Therefore, for the multi-objective cocktail we consider all the possible combinations between the following two blocks:

- Magnitude-aware: i) nothing; ii) GradNorm [5]; iii) MGDA-UB [30]; iv) IMTL-G [18].
- Direction-aware: i) nothing; ii) GradDrop [6]; iii) PCGrad [44].

This amounts to a total of 12 combinations, plus the hyperparameter of specific algorithms. In this case, we only tune the α parameter from GradNorm between the values zero and one.

A.3 Experimental settings

We train all experiments using Adam as optimizer, with a learning rate of 0.001 for all models. For all models (except HI-VAE) we set the batch size to 128, and train for 400 epochs for the all datasets (except for *Wine* with 2000 epochs). For HI-VAE, we set the batch size to 1000 and the number of epochs to 2000 as in the original paper. We randomly split the data into training ($70\,\%$), validation ($10\,\%$), and testing ($20\,\%$).

We set the latent size of $\mathbf{Z}$, d, to $50\,\%$ of the number of features of the dataset, D, and the hidden size of each layer to 50 for all the experiments, except for those of the *Bank* dataset which are set to 100.

Metric. Since we deal with heterogeneous data, where each feature has different type and range, we compute the reconstruction error using metrics that account for these differences. For numerical features (real, positive, and count data) we compute the normalized root mean squared error:

$$\text{err}(d) = \frac{1}{N} \frac{||x_d - \hat{x}_d||_2}{\max(x_d) - \min(x_d)}, \tag{3}$$

where $\hat{x}$ is the model prediction. For the case of nominal features (categorical and binary data) we use the error rate as reconstruction error:

$$\text{err}(d) = \frac{1}{N} \sum_{n=1}^{N} I(x_{n,d} \neq \hat{x}_{n,d}). \tag{4}$$

The final metric shown in Table 1 is the average across dimensions, $\text{err} = \frac{1}{D} \sum_d \text{err}(d)$.

Model selection. In order to make fair comparisons, for each model and dataset we first tuned the hyperparameters (for example, hidden/latent/batch size, number of epochs, etc.) for the vanilla implementations (i.e., without MOO). To this end, we ran grid searches and averaged the validation metric over five random seeds, just as in Table 1, choosing the set of hyperparameters that performed the best in terms of reconstruction error during validation. Note that all these hyperparameters (including optimization hyperparameters such as learning rate) are shared across all methods of the same setting. Additionally, we verified that the vanilla models were performing well by visual inspection of the marginal reconstructions.

Multi-objective cocktail selection. Similar to model selection, we chose the best combinations of MOO by averaging over five random seeds and taking the combination of methods that performed the best in terms of reconstruction error in validation. In general, it was enough to focus on the median to select the best combination. However, some combinations happened to overfit, and therefore we needed to choose those having a good balance between median, mean, and standard deviation.

Statistical test. In order to compare the performance of the proposed method with vanilla, we employ the corrected paired t-test [25]. The usual paired t-test assumes that the data used to perform the test is independently sampled, which usually does not hold in the machine learning as we sample the training and test data from the same distribution. As a consequence, paired t-test might suggest statistical significance between the compared models, whereas there is no such significance (type I error). Corrected paired t-test considers the dependency of the sampled data, correcting the variance of the differences of the paired samples in the two testing models.

A.4 Models

In this section we explain all the architectural details for each model, please refer to the original papers for a detailed explanation of each model. We use the following notation to describe the models:

- D - number of features.
- D' - total number of likelihood parameters.
- d - latent size.
- h - hidden size.
- `[Linear-`h`]` - Linear layer with output of size h.
- `[Dropout-10 %]` - Dropout Srivastava et al. [32] with $10\,\%$ of dropping probability.
- `[ReLU]`- Rectified linear unit activation function.
- `[Tanh]`- Hyperbolic tangent activation function.

A.4.1 Variational autoencoder (VAE)

We implement the original VAE [15] assuming the following probabilistic model:

$$\begin{aligned} &\text{Prior:} && p(\mathbf{Z}) = \mathcal{N}(0, I) \\ &\text{Likelihood:} && p_\theta(\mathbf{X}|\mathbf{Z}) = \textstyle\prod_d p_d(\mathbf{x}_d|\eta_d(\mathbf{Z};\theta)) \\ &\text{Variational approx.:} && q_\phi(\mathbf{Z}|\mathbf{X}) = \mathcal{N}(\mu(\mathbf{X};\phi), \sigma(\mathbf{X};\phi)). \end{aligned}$$

Here μ and σ are modelled by the encoder, and all η_d are jointly modelled by the decoder. These two neural networks are of the following form:

- **Encoder:** `[Dropout-10 %][BN][Linear-`h`][Tanh][Linear-`h`][Tanh][Linear-`h`][Tanh][Linear-`$2d$`]`.
- **Decoder:** `[Linear-`h`][ReLU][Linear-`h`][ReLU][Linear-`h`][ReLU][Linear-`D'`]`.

Additionally, we make sure that each parameter fulfils its distributional constraints (e.g., the variance has to be positive) by passing it through a softplus functions when necessary. It is also important to note that, while we parametrize the latent space using the mean and standard deviation, we parametrize the parameters of the likelihoods using their natural parameters.

Loss. We use the negative ELBO as training loss:

$$\text{ELBO}(\mathbf{X}, \theta, \phi) \coloneqq \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{X}|\mathbf{Z})\right] - \text{KL}(q_\phi(\mathbf{Z}) \| \, p(\mathbf{Z})). \tag{5}$$

Imputation. We impute data by taking the modes of $q_\phi(\boldsymbol{z}|\mathbf{X})$ and $p_d(\mathbf{x}_d; \eta_d(\mathbf{Z};\theta))$.

MOO. As explained in the main text, for this model we simply apply the MOO algorithms on the last-shared feature $\boldsymbol{y}$ (see Fig. 1).

A.4.2 Importance weighted autoencoder (IWAE)

Importance weighted autoencoder (IWAE) [3] shares the same settings as VAE, being the only difference the training loss.

Loss. Instead of maximizing the ELBO, IWAE maximizes a tighter loss that makes use of K i.i.d samples from $\mathbf{Z}$:

$$\text{IWAE}(\mathbf{X}, \theta, \phi) \coloneqq \mathbb{E}_{\mathbf{Z}_1,\ldots,\mathbf{Z}_K \sim q_\phi}\left[\log \frac{1}{K}\sum_k \frac{p_\theta(\mathbf{X}|\mathbf{Z}_k)p(\mathbf{Z}_k)}{q_\phi(\mathbf{Z}_k|\mathbf{X})}\right]. \tag{6}$$

For all the results shown in Table 1 we set the number of importance samples to $K = 20$.

A.4.3 Doubly reparametrized gradient estimator (DReG)

Rainforth et al. [28] showed long time ago that the gradient estimators produced by IWAE had some undesired properties that could hamper properly learning the inference parameters (encoder). A strict improvement of this negative result was provided by Tucker et al. [37], as they provided a simple way of addressing these issues by applying the reparametrization trick a second time. As a result, we obtain again a model structurally identical to VAE, but which is optimized with two different losses: one for the encoder, and one for the decoder. We use $K = 20$ importance samples as for IWAE.

Encoder loss. For one importance sample $\mathbf{Z}_k$, let us define

$$\omega_k \coloneqq \frac{p_\theta(\mathbf{X}|\mathbf{Z}_k)p(\mathbf{Z}_k)}{q_\phi(\mathbf{Z}_k)}, \text{ and } \quad \widetilde{\omega}_k \coloneqq \frac{\omega_k}{\sum_i \omega_i} \text{ such that } \quad \sum_k \widetilde{\omega}_k = 1. \tag{7}$$

Then, we optimize the parameters of the encoder by maximizing

$$\mathrm{DReG}_{enc}(\mathbf{X}, \theta, \phi) \coloneqq \mathbb{E}_{\mathbf{Z}_1,\ldots,\mathbf{Z}_K \sim q_\phi}\left[\sum_k \widetilde{\omega}_k^2 \log \omega_k\right], \tag{8}$$

where we consider $\widetilde{\omega}_k$ to be a constant value (i.e., we do not backpropagate through it), and we compute the derivative w.r.t. ϕ only through $\mathbf{Z}$ (i.e., we do not compute the partial derivative w.r.t. ϕ).

Decoder loss. Similarly, we optimize the parameters of the decoder by maximizing the following loss (same assumptions on $\widetilde{\omega}_k$ and ϕ):

$$\mathrm{DReG}_{dec}(\mathbf{X}, \theta, \phi) \coloneqq \mathbb{E}_{\mathbf{Z}_1,\ldots,\mathbf{Z}_K \sim q_\phi}\left[\sum_k \widetilde{\omega}_k \log \omega_k\right]. \tag{9}$$

MOO. As we now have two different losses, we employ two different set of parameters for the MOO algorithms (one for the encoder, and one for the decoder).

A.4.4 HI-VAE

We have faithfully re-implemented the original version of HI-VAE [26], this includes implementing their architecture with the same number of parameters, as well as implementing their methods (such as the proposed normalization and denormalization layers). Regarding the architecture, we have maintained the same one as the original authors used in their experiments. Therefore, results between HI-VAE and the rest of the models in Table 1 are not completely comparable.

HI-VAE assumes a hierarchical latent space. Thus, we assume the following probabilistic model:

$$\begin{aligned}
&\text{Prior:} && p(\mathbf{Z}, \mathbf{S}) && = p(\mathbf{S})p(\mathbf{Z}|\mathbf{S}) \\
& && && = Cat(\tfrac{1}{d_s}, \tfrac{1}{d_s}, \ldots, \tfrac{1}{d_s})\,\mathcal{N}(\mu_0(\mathbf{S}), I) \\
&\text{Likelihood:} && p_\theta(\mathbf{X}|\mathbf{Z}) && = \textstyle\prod_d p_d(\mathbf{x}_d|\eta_d(\mathbf{Z};\theta)) \\
&\text{Variational approx.:} && q_\phi(\mathbf{Z}, \mathbf{S}|\mathbf{X}) && = q_\phi(\mathbf{S}|\mathbf{X})q_\phi(\mathbf{Z}|\mathbf{X}, \mathbf{S}) \\
& && && = Cat(\pi(\mathbf{X}))\,\mathcal{N}(\mu(\mathbf{X}, \mathbf{S}; \phi), \sigma(\mathbf{X}, \mathbf{S}; \phi)).
\end{aligned}$$

Similar to VAE, μ_0, μ, and σ are all neural networks, and all likelihood parameters η_d are jointly modelled by the decoder. Note also the introduction of new variables to describe the size of each latent variable, d_z and d_s. We set in our experiments $d_z = d_s = 10$, and the hidden size to $h = 5D$, just as in the original paper.

Loss. We maximize the ELBO as originally proposed by Nazabal et al. [26]:

$$\mathrm{ELBO}(\mathbf{X}, p_\theta, q_\phi) \coloneqq \mathbb{E}_{\mathbf{Z},\mathbf{S}\sim q_\phi}\left[\log p_\theta(\mathbf{X}|\mathbf{Z}, \mathbf{S})\right] - \mathrm{KL}(q_\phi(\mathbf{Z}, \mathbf{S}) \| \, p(\mathbf{Z}, \mathbf{S})). \tag{10}$$

MOO. Like in the VAE case, we apply MOO at the last shared feature produced by the model. However, in this case there are *two* last shared features, $\boldsymbol{y}$ (as before), and $\mathbf{S}$. Therefore, we use two different set of parameters for each of the last shared features.