MultiVae: A Python library for Multimodal Generative Autoencoders

Agathe Senellart, Clément Chadebec, Stéphanie Allassonnière

MultiVae: A Python library for Multimodal Generative Autoencoders

Agathe Senellart
Université de Paris-Cité,Inria, Inserm, SU
Centre de Recherche des Cordeliers
agathe.senellart@inria.fr

Clément Chadebec
Université de Paris-Cité,Inria, Inserm, SU
Centre de Recherche des Cordeliers
clement.chadebec@inria.fr

Stéphanie Allassonnière
Université de Paris-Cité,Inria, Inserm, SU
Centre de Recherche des Cordeliers
stephanie.allassonniere@u-paris.fr

Abstract

In recent years, there has been a major boom in the development of multimodal machine learning models. Among open topics, representation (fusion) and generation of multimodal data are very active fields of research. Recently, multimodal variational autoencoders have been attracting growing interest for both tasks, thanks to their versatility, scalability, and interpretability as probabilistic latent variable models. They are also particularly interesting models in the partially observed setting, as some models can learn even with missing data. In this article, we present *MultiVae*, an open-source Python library designed to bring together unified implementations of multimodal generative autoencoders models. It has been designed for easy, customizable use of these models on partially or fully observed data. This library facilitates the development and benchmarking of algorithms by integrating several popular datasets, variety of evaluation metrics and tools for monitoring and sharing models. For each model included, a key result is reproduced to validate our implementation. We also propose a case study in the less studied partially observed setting, evaluating the robustness of the models as a function of the missing ratio in a five-modalities dataset.

1 Introduction

Over the past few years, there has been growing interest for deep generative models and in particular multimodal models. In particular, Variational Autoencoders (VAEs) [29] have demonstrated promising results to model complex data distributions such as images or text. Their ability to learn a lower dimensional representation of potentially high dimensional data makes them attractive models to perform various tasks such as clustering [12], image synthesis [38], speech modelling [6] or multi-omics data representation and generation [33]. Initially used in a unimodal setting, VAEs have also rapidly drawn interest in multimodal contexts as well. Multimodal Variational Autoencoders aim to embed all the different modalities in a *common representation* and learn inference encoder distributions to estimate that embedding from any single modality. This inference of the latent code then allows to perform *cross-modal data generation* from any modality. Sampling from a prior distribution in the latent space would also enable joint generation of all modalities simultaneously, creating multimodal synthetic samples. So far, many variants of the unimodal VAE have been proposed to adapt it to the multimodal framework. Among them, we can distinguish the *joint encoder models* [46, 50, 40] using a dedicated network to model the joint posterior, *aggregated models* [41, 54, 43] modelling the joint posterior distribution as an aggregation of the unimodal encoders, and

Preprint. Under review.

coordinated models [51, 24, 47] where different unimodal VAEs are trained per modality while enforcing the learned latent spaces to share similarity. The flexibility of these models to various modality types through the choice of encoders and decoders architectures, their scalability to the number of modalities as well as their interpretability as latent variable models have contributed to the popularity of multimodal VAEs. For some of them, another advantage is that they can be applied in a *weakly-supervised* setting where not all modalities are observed for all samples which can reveal useful in medical applications for instance.

However, to the best of our knowledge, there exists no unifying framework for those models providing easy-to-use, flexible implementations while replicating the results given in the original papers. This absence slows down benchmarking as well as the development of new methods and their deployment in other application fields. Moreover, the original implementations are often no longer maintained or only available with hard-coded parameters, making them difficult to adapt and test for other use-cases. In particular, although many models are theoretically able to learn on *partially observed datasets* with missing modalities, most available implementations do not handle that setting. Hence, in this paper we propose the following contributions:

- We introduce the *MultiVae* (`https://github.com/AgatheSenellart/MultiVae`) library that implements and unifies implementation of multimodal generative autoencoders proposed in the literature. MultiVae has a modular structure that enables a flexible use of these models on any dataset with or without missing samples.
- We pay particular attention to the validation of the implementations available in *MultiVae* by reproducing one key result from the original publication.
- We propose several small case studies illustrating the flexibility of the proposed library and its usability for benchmarking and comparison purposes.

2 Multimodal Variational Autoencoders

In Multimodal Machine Learning, two goals are generally targeted: (1) Learn a shared representation from multiple modalities; (2) Perform cross-modal generation from one modality to another. In the literature, many models have been proposed to tackle either one or both of these tasks. For instance, on the one hand, Contrastive Learning approaches [16, 27, 15, 9] impose to learn representations of the multimodal data that are invariant to modalities and to given transformations of the data. On the other hand, many models like **Pix2Pix** and more generally **CycleGANs** tackle the issue of cross-modal generation by modelling the conditional distribution of one modality x_1 given another x_2, $p(x_2|x_1)$ [55, 21, 56].

In this paper, we decide to specifically focus on models referred to as Multimodal Generative Autoencoders [45] which aim at solving both issues at the same time. These models learn a latent representation z of all modalities in a lower dimensional common space and are able to infer z from any modality to generate the others. In other words, given a set of observations with M modalities $X = (x_1, x_2, ...x_M)$, it is assumed that a shared latent representation z exists, from which all modalities can be generated with decoding parametric distributions $(p_\theta(x_j|z))_{j\in\{1,...,M\}}$. In most cases, each modality is supposed to be conditionally independent of the others given z such that the joint model writes:

$$p_\theta(X, z) = p_\theta(X|z)p_\theta(z) = p_\theta(z)\prod_{j=1}^{M} p_\theta(x_j|z)\,, \tag{1}$$

where $p_\theta(z)$ is a *prior* distribution set over the latent variables. The choice of the decoding distributions $(p_\theta(x_j|z))_{1\leq j\leq M}$ depends on the modality type. For instance, for an image modality x_1, $p_\theta(z|x_1)$ can be a Gaussian distribution $\mathcal{N}(\mu_\theta(z), \Sigma_\theta(z))$ with the parameters $\mu_\theta(z), \Sigma_\theta(z)$ being the output of a decoder neural network taking z as input. In that framework, the two goals mentioned above translate as follows. First, we want to learn the best possible θ to model the observations. Second, we want to approximate the inference distributions $p_\theta(z|(x_j)_{j\in S})$ to infer the latent variable from any given subset of modalities $S \in \mathcal{P}(\{1, \dots, M\})$. For the sake of scalability to the number of modalities, most models focus on inferring the *unimodal posteriors* $p_\theta(z|(x_j))$ for all modalities $1 \leq j \leq M$, and then aggregate those distributions with a simple operation to model any subset posterior $p_\theta(z|(x_j)_{j\in S})$ [41, 54, 43, 20, 35].

Estimating the generative model's parameter θ Given N observations, a natural objective to estimate θ is to optimize the log-likelihood of the data:

$$\theta^* = \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(X^{(i)}) = \arg\max_\theta \sum_{i=1}^{N} \left(\log \int_z p_\theta(X^{(i)}, z) dz \right) \tag{2}$$

Since this objective is often intractable, one can recourse to Variational Inference [25] in a similar fashion as the Variational Autoencoder (VAE) [29] by introducing an auxiliary parametric distribution $q_\phi(z|X)$ allowing to derive an unbiased estimate of the likelihood of the data:

$$\widehat{p}_\theta(X, z) = \frac{p_\theta(X, z)}{q_\phi(z|X)} \quad \text{such that} \quad p_\theta(X) = \mathbb{E}_{q_\phi(z|X)} \left[\widehat{p}_\theta \right] . \tag{3}$$

Then, using Jensen's inequality allows to derive a lower bound on $p_\theta(X)$, referred to as the Evidence Lower Bound (ELBO).

$$\log p_\theta(X) = \log \mathbb{E}_{q_\phi(z|X)} \left[\widehat{p}_\theta \right] \geq \mathbb{E}_{q_\phi(z|X)} \left[\log p_\theta(X|z) \right] - KL(q_\phi(z|X)||p_\theta(z)) = \mathcal{L}(X) . \tag{4}$$

This bound is tractable and can be optimized through Stochastic Gradient Descent. Noteworthy, the first term can be seen as a reconstruction error and the second as a regularization term imposing the latent codes to follow the prior distribution [14]. The distribution $q_\phi(z|X)$ is generally called the *encoder* and one may notice that the optimization of the ELBO leads to minimizing the Kullback-Leibler divergence between the true posterior distribution $p_\theta(z|X)$ and $q_\phi(z|X)$ [29]. While most Multimodal VAEs use such an auxiliary distribution and optimize Eq. (4) to learn θ, some models also rely on variations of Eq. (4). For instance, the MMVAE [41] model uses a k-sampled importance weighted estimate of the log-likelihood (IWAE bound) [7] because it promotes a higher entropy of the learned posterior $q_\phi(z|X)$. Other models such as MoPoE [43], MVTCAE [20], MMVAE+ [35] use a β factor in the ELBO to weigh the regularization term. That hyperparameter can be tuned to promote disentanglement in the latent space [19].

Choice of the approximate inference distribution A natural choice is to model $q_\phi(z|X)$ as a Gaussian distribution $\mathcal{N}(\mu_\phi(X), \Sigma_\phi(X))$ where a dedicated joint encoder network takes all modalities as input and outputs the parameters $\mu_\phi(X), \Sigma_\phi(X)$. By doing so, we obtain an estimation of θ and an approximation of the joint posterior $p_\theta(z|X)$ by $q_\phi(z|X)$. However, we do not have access to the unimodal posteriors $(p_\theta(z|x_j))_{1 \leq j \leq M}$ which are needed for cross-modal generation. Indeed, we want to be able to infer z from any modality to then *decode* z to generate other modalities. To estimate these posterior distributions, two approaches have been proposed. First, several models, referred to as *aggregated models*, suggest to introduce a dedicated encoder $q_{\phi_j}(z|x_j)$ per modality $1 \leq j \leq M$, which models the unimodal posterior distribution, and to aggregate them to model $q_\phi(z|X)$. As an example, the MVAE [54] model uses a Product-of-Experts (PoE) operation $q_\phi(z|X) \propto p_\theta(z) \prod_j q_{\phi_j}(z|x_j)$ while the MMVAE [41] model uses a Mixture-of-Experts (MoE) and the MoPoE-VAE [43] model uses a Mixture of Product of Experts (MoPoE). Such a choice for $q_\phi(z|X)$ allows to obtain trained unimodal inference distributions when optimizing Eq. 4. In particular, [43] rewrites the ELBO to explicitly highlight how those aggregation methods encourage the estimated posteriors $q_{\phi_j}(z|x_j)$ to be close to the true joint posterior $p_\theta(z|X)$.

Moreover, additional terms might be added to the ELBO to further ensure certain properties of the unimodal encoders. For instance, the product of experts in the MVAE model may cause some of the unimodal encoders $q_{\phi_j}(z|x_j)$ to be uninformative if the others already contain all the necessary information. To solve that issue, the MVTCAE model derives a bound from a Total Correlation Analysis that adds Conditional Variational Information Bottleneck (CVIB) terms to the ELBO to avoid that issue [20].

However, *aggregated models* restrict the flexibility of the inference distribution $q_\phi(z|X)$. Hence, other approaches choose to still use a joint encoder network modeling $q_\phi(z|X)$ and introduce additional terms to Eq. (4) to learn the unimodal approximate posteriors $(q_\phi(z|x_j))_{1 \leq j \leq M}$. Some models introduce similar CVIB terms as used in the MVTCAE [46, 40], while the TELBO model uses additional unimodal ELBOs terms [50].

Coordinated models A slightly different approach than the one detailed above, referred to as *coordinated models* and proposed for instance in [51, 24, 47], does not directly optimize Eq. (4)

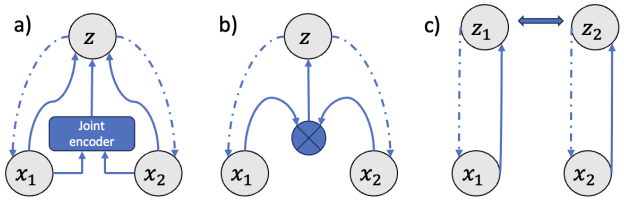

Figure 1: Graphical representation of the three presented types of multimodal VAEs: a) models using a joint encoder network, b) aggregated models, c) coordinated models. Plain lines represent encoders and dashed lines represent decoders.

but starts from learning individual representations z_j for each modality x_j (using unimodal ELBOs) while enforcing the representations from each modality to be *similar* $z_j \approx z_k$ for paired modalities x_j, x_k. This notion of similarity is defined differently depending on the model. Figure 1 summarizes the graphical models of families of models we have presented here. [45] presents a more in-depth survey of multimodal generative autoencoders.

Recent developments Recently, methods have been proposed with a more complex generative model including multiple, separated [31, 42] or hierarchical latent variables [49, 40]. For instance, having multiple latent spaces allows to improve the diversity in the generation of certain aggregated models [10]. An additional goal of these models is to separate into different latent spaces the information shared across modalities from modality specific factors. Models using those multiple latent variables are sometimes sensitive to the *shortcut* issue referring to shared information leaking into the modality specific latent spaces. Recently, the MMVAE+ model was proposed with an amended ELBO loss to limit that phenomenon [35].

Learning with incomplete datasets An important issue in multimodal machine learning is to design models that can learn from *partially observed data*, i.e. datasets where each sample contains some, but not necessarily all modalities. Models using a joint network encoder for $q_\phi(z|X)$ in the ELBO are not easily adaptable to this setting since the joint network requires all modalities as inputs to compute the latent representation. On the other hand, *aggregated models* have a natural extension to the partially observed setting: for each sample, the ELBO bound can be computed using only observed modalities:

$$\mathbb{E}_{q_\phi(z|(x_s)_{s\in S_{\text{obs}}})}\left[\log \frac{p_\theta((x_s)_{s\in S_{\text{obs}}}|z)p(z)}{q_\phi(z|(x_s)_{s\in S_{\text{obs}}})}\right] \tag{5}$$

with S_{obs} the subset of observed modalities, and $q_\phi(z|(x_s)_{s\in S_{\text{obs}}})$ being computed by aggregating the unimodal encoders for observed modalities: $q_{\phi_s}(z|(x_s))_{s\in S_{\text{obs}}}$. Even when the loss is a modified version of the ELBO, for instance with additional objective terms for the unimodal encoders or an IWAE bound, it can easily be computed with only the observed modalities. Appendix A summarizes the loss of the aggregated models we implemented and their adaptation to the partially observed setting. That setting has been explored by [54, 20] but not for all aggregated models.

3 The MultiVae Library

Why MultiVae? Since they are generic generative methods that are scalable to an arbitrary number of views, Multimodal VAEs have applications in many different fields: biomedicine [33], fake-news detection [28] and robotics [22] to cite a few. However, original implementations of the models are not always available, maintained or easy to adapt to a new use case. Easy to use, flexible and reproducible implementations in a unified framework is a key step to foster applied research. Furthermore, such unifying framework promotes the development of new algorithms by facilitating experiments and benchmark of existing models.

Project Vision Starting from these observations, we created MultiVae, a Python library that implements Multimodal Autoencoder models in a unified framework. MultiVae is designed with the following points in mind:

- **Modular and easy to use**. The library is designed to be easy to use, even for people that are beginners in the field of multimodal machine learning. Its modular structure means that it can be plugged in any Pytorch project with any user-defined dataset and custom architectures. A complete online documentation is provided along with tutorial notebooks. A model instance can be created and trained with only a few lines of code.
- **Reliable implementations**. Most of the implementations have been verified by reproducing results from original papers (cf. Table 3). This means that the models are implemented as closely as possible to the original code, including all modelling choices, recommended training scheme, and implementations details. This way, each model can be used following the papers' guidelines. Nonetheless, all options included can also be easily tuned by the user to experiment with other parameter choices.
- **Including data and metrics**. Our framework includes easy loading of several datasets including Mnist-SVHN [11, 34], PolyMNIST [43], CelebA [32] and CUB [17]. For easy benchmarking of models, we also include modules to visualize results and compute several metrics: coherences, likelihoods, Fréchet Inception Distance (FID), clustering and reconstruction metrics. This allows for a unified and straightforward evaluation of models. By taking work out of data management and metrics computing, our library helps focusing on model benchmarking and design.
- **Monitoring and sharing tools for transparent benchmarks**. MultiVae integrates easy logging of metrics and models configurations on the Wandb [5] platform and straightforward uploading/downloading of models on the HuggingFace Hub [13]. This allows to perform a completely open-source and reproducible benchmarking of models.
- **A framework compatible with the partially observed setting** As mentioned above, aggregated models can be applied to the setting where the training data is only partially observed (*e.g.* missing modalities). That setting was explored in [54, 20], however in those studies, the available implementations are not flexible enough to handle any arbitrary partially observed dataset. This is why, in our library, we propose a simple framework for working with incomplete datasets using any of the aggregated models.

Related work To the best of our knowledge, three available python frameworks exist attempting to regroup multimodal variational autoencoders implementations in a unified package. The **Multimodal VAE Comparison Toolkit** [39] only contains implementations of some of the *aggregated models* leaving behind models using a joint encoder network. The **Pixyz** library [44] regroups generative models including only two multimodal VAEs: the JMVAE/JMVAE-GAN. Very recently, the **multi-view-ae** library [1] was released implementing several Multimodal Autoencoders in a modular way. This library is the most similar to ours. However, it is not (yet) usable with completely customizable architectures and does not handle the partially observed setting. Importantly, none of the libraries mentioned above validated their implementations by reproducing results from the original papers nor support incomplete datasets.

Code Structure Our implementation is based on PyTorch [37] and is inspired by the architecture of [8] and [53]. The code has been designed with simplicity in mind, allowing quick and easy model setup and training, and with a strong emphasis on reproducibility. The implementations of the models are collected in the module `multivae.models`. For each of the models, the actual implementation of the model is accompanied by a configuration as a *dataclass* gathering the collection of any relevant hyperparameter which enables them to be saved and loaded straightforwardly. The models are implemented in a unified way, so that they can be easily integrated within the `multivae.trainers`. Like the models, the trainers are also accompanied by a training configuration *dataclass* used to specify any training-related hyperparameters (number of epochs, optimizers, schedulers, etc..). Models that have a multistage training [50, 40] benefit from their dedicated `trainer` that makes them as straightforward to use as other models. *MultiVae* also supports distributed training, allowing users to train their models on multiple GPUs straightforwardly. Partially observed datasets can be conveniently handled using the `IncompleteDataset` class that contains masks informing on missing or corrupted modalities in each sample. Finally, the *MultiVae* library also integrates an evaluation pipeline for all models where common metrics such as likelihoods, coherences, FID scores [18] and visualizations can be computed in a unified and reliable way.

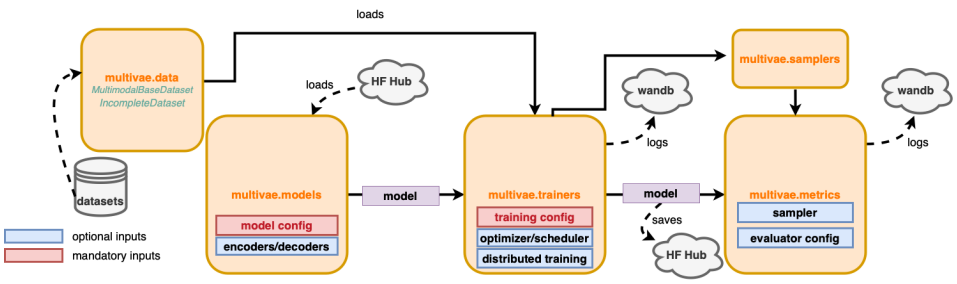

Figure 2: Schematic diagram of the MultiVae library.

Furthermore, a significant advantage of the *MultiVae* library is its capacity to enable users in defining their own encoder and decoder architectures, which can be supplied as arguments to any implemented model. This allows the training of any model on any type of modality without affecting the model's implementation. This is made possible by the library's design, which imposes a unified API for all models. In addition, the library integrates *wandb* [5], a callback-based experiment tracker that lets users monitor and compare their model trainings or evaluations using a graphical interface. *MultiVae* also supports the sharing of trained models via the Hugging Face Hub[13]. Code examples in Appendix F illustrates ease of use of these functionalities.

The library is unit-tested with a code coverage above 84 % to ensure code quality and continuous collaborative development. The main features are illustrated through tutorials made available either as notebooks or scripts allowing users to get started easily. An online documentation is also made available at `https://multivae.readthedocs.io/en/latest`.

Validating our implementation Most models implemented have been validated by reproducing a key result of the original paper. One experiment presented in the paper was reproduced using MultiVaE implementation with the same exact architectures and training parameters. Table 3 presents a summary of reproduced results for each model. More details and visualizations are given in the Appendix D. The scripts used for reproducing results are available in the library and saved models can be downloaded from Hugging Face Hub[13]. In addition to the models present in Table. 3, the TELBO model has been implemented but not yet validated. The library also contains the original implementations of the JNF and JNF-DCCA models used to produced the results in [40] of which we are the authors. A brief description of each implemented model is given in Appendix A.

Model	Dataset	Metrics	Original paper values	Our values
JMVAE	BinaryMNIST	Joint Likelihood	-86.86	-86.85 ± 0.03
MMVAE	Mnist-Svhn	Coherences	59 ± 11 / 87 ± 2 / 39 ± 3	60 ± 6 /87 ± 2 /38 ± 3
MVAE	BinaryMNIST	ELBO	188.8	188.3 ± 0.4
MoPoE	PolyMNIST	Coherences	66/77/81/83	67/79/84/85
MVTCAE	PolyMNIST	Coherences	59/77/83/86	64/82/88/91
MMVAE+	PolyMNIST	Coherence/FID	86.9/92.81	88.6 ± 0.8/93 ± 5

4 Case Study: Training Models on a Incomplete Dataset

Although a few papers have experimented with the partially observed setting [20, 54], most results on multimodal autoencoders have been obtained on complete and relatively large datasets. The MNIST-SVHN dataset used in [41, 26] contains (60,000$\times$5 or $\times$20) samples depending on how many pairings are done, the CelebA [46, 43] dataset contains 200,000 samples and the PolyMNIST dataset [46, 35, 20] contains 60,000 training samples. In this section, we propose to compare models trained on partially observed or small datasets. In sections 4.1 and 4.2 we use the PolyMNIST dataset that contains 5 modalities (or views). Each view is an image with a MNIST digit overlayed on top of a random crop from a background image specific to that view [43]. For each multimodal sample, the digit in each view has the same numerical value but not the same handwriting.

To create a partially observed dataset, we simulated a bernoulli distribution for each of the last four views of each sample. We kept the view available with probability η and discarded it otherwise. The first view is kept for all samples to ensure that each sample had at least one available modality. For $\eta = 1, 0.8, 0.5$ we compared all models with two paradigms: training with all samples even incomplete ones, or training only on complete samples. In that second case, the size of obtained datasets were $N = 24576$ when $\eta = 0.8$ and $N = 3750$ when $\eta = 0.5$.

In all study cases presented below, we test the models on the *complete* test set containing 10,000 samples.

Figure 3: PolyMNIST dataset. Each row presents a modality and the columns present samples.

4.1 Example: cross-modal generation using the MMVAE+ model

As an aggregated model using additional modality specific latent spaces, the MMVAE+ [35] benefits of a natural adapted objective for the partially observed setting, where for each sample, the objective is computed for the largest available subset of modalities (cf. Eq. (5)). Using the same architecture and training paradigm for PolyMNIST as in the original paper, we trained the model on the partially observed datasets. Presented results are obtained with $\beta = 2.5$ in the ELBO. Figure 4 shows that the model adapts well to this setting and still produces diverse and coherent generations when $\eta = 0.5$. The coherence of cross-modal generation shown on the right part of the figure is defined as the percentage of matching digits between the conditioning modality and the generated modality. It is assessed using pretrained classifiers. The advantage of keeping incomplete samples when training the model is evident as the coherence drops from 0.8 to 0.5 when they are set aside.

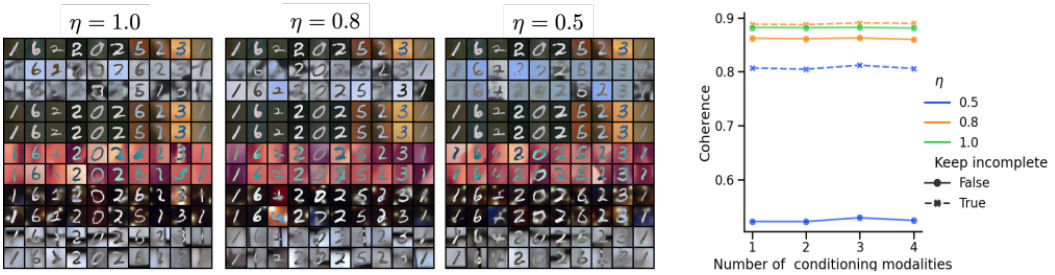

Figure 4: Left: Cross-modal generations for the MMVAE+ model. The generations are conditioned on the first line, then two rows of generations per modality are presented. Right: We plot the mean coherences of the cross-modal generations as a function of the number of conditioning modalities. The coherences are averaged over all subsets of a given length. The color of the line indicates the degree of missing modalities η, the style of the line indicates the training paradigm (keep incomplete samples or not).

4.2 Comparing metrics across models

In this paragraph, we present results for different models in the partially observed PolyMNIST setting presented above. The same simple convolutional architectures and latent dimension of 512 were used for all models as done in [43, 20]. Parameters for each model were chosen according to original papers and are specified in Appendix B.2. In particular we set $\beta = 2.5$ for the MVAE, MoPoE and MVTCAE models following [43, 20]. However we use the $K = 10$ sampled version of the MMVAE objective to follow the author's guidelines. Note that we do not tune the models' hyperparameters on each task but fix them following author's recommendations and then evaluate the trained models on different tasks. For all *aggregated models*, we use the natural adaptation of the objective functions to the partially observed setting recalled in Appendix A.

Cross-modal generation Figure 5 presents all coherences for cross-modal generations of seven models, for the three degrees of missing data when training with all samples or only complete samples. Figure 5 shows that for all *aggregated models*, learning with all observations gives better results that

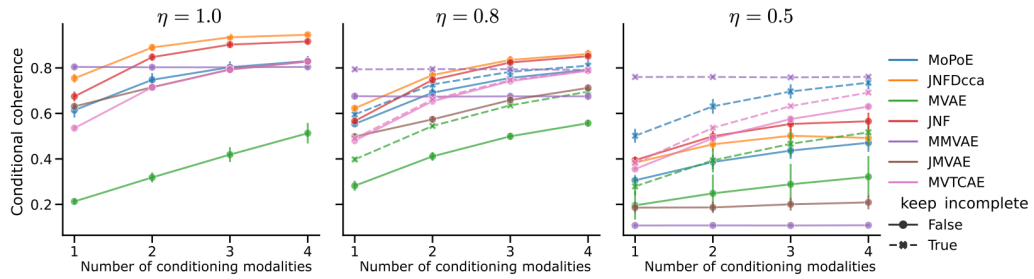

Figure 5: Coherences of cross-modal generation for different degrees of missing data. The accuracies of cross-modal generations are averaged over conditioning subsets of same length. The presented results are averaged on four different seeds. The color of the line indicates the model, and the style of the line indicates the training paradigm: keep all samples or only the complete ones.

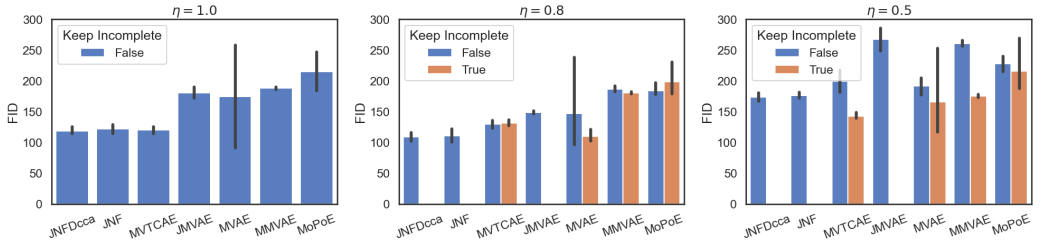

Figure 6: FID Scores (↓) for conditional generation of the first modality given all the others. For FID scores, we recall that lower is better. The results are averaged over four seeds. The color of the bar plot indicates the training paradigm. The error bars indicates the 0.95 confidence interval.

learning with only the complete observations. This is most remarkable in the case $\eta = 0.5$ where the MMVAE maintains a high accuracy in all cross-modal generations. Figure 5 also allows to evaluate the robustness of the models on smaller datasets: the plain lines on the right plot ($\eta = 0.5$) present the results of all models when trained on complete, smaller datasets (3750 training samples). The MoPoE, MVTCAE and JNFDcca models seem to perform best in that setting, whereas the performance of the JMVAE and MMVAE model are drastically reduced. Figure 6 presents the FID scores of generated samples for the first modality given all the others. The FID score gives an indication on the diversity and realism of generated images [18]. Examples of images are presented in Appendix C. Once again, learning with all samples results in an improved (lower) FID score compared to learning only with complete ones. When training on smaller datasets, we observe that the FID score tends to increase for all models while mainly keeping the same ordering; the JNF, JNFDcca, MVTCAE and MVAE model having the best values.

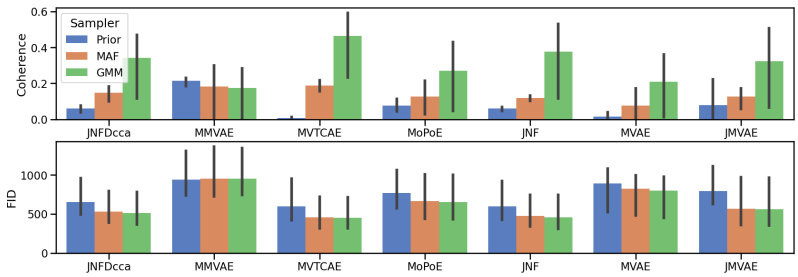

Figure 7: Joint coherence (↑) and FID (↓) for models trained on the complete dataset ($\eta = 1$) using different samplers in the latent space.

Joint generation with samplers The most natural way to sample from a generative model latent space is to use the prior distribution on the latent variables. However, using another, *a-posteriori*, estimated sampling distribution on the latent space can lead to improved quality of the generated samples [14]. In MultiVae, we propose a Gaussian-Mixture Model Sampler (GMM) and a Masked Autoregressive Flow (MAF) [36] Sampler plugins that can be used to sample from the latent space of any model. In both cases, the samplers' parameters are estimated on the training dataset, using latent codes sampled from $q_\phi(z|X)$. Samplers can also be used within the evaluation modules of MultiVae to compute metrics (coherences and FIDs) on samples generated with one of these samplers. In particular, to perform data augmentation of a multimodal dataset, joint coherence is a particularly important metric. On the PolyMNIST dataset, it is defined as the ratio of generated samples having the same digit for all modalities. Figure 7 presents how joint coherence as well as FID scores can be drastically improved when using a GMM sampler. Here the GMM performs best for most models, which also informs on the structure of the latent space: a GMM distribution can have multiple modes where MAF models a smoother, unimodal distribution. The fact that the GMM works best tends to indicate that the data is not distributed continuously, across the latent space which is coherent with the fact that the latent space is large (512 dimensions) in this experiment.

Clustering on the latent representation Finally we investigate, the relevance of the joint latent representation, taken as the mean of $q_\phi(z|X)$, for clustering. Here we perform a K-means [3] on the training latent embeddings with $K = 10$. To each cluster is assigned the majority label among the training samples in that cluster. That model is used to predict the labels for the test set in Figure 8. As a general comment, we observe that *aggregated models* tend to perform best on that task and are able to learn a meaningful representation even with incomplete datasets.

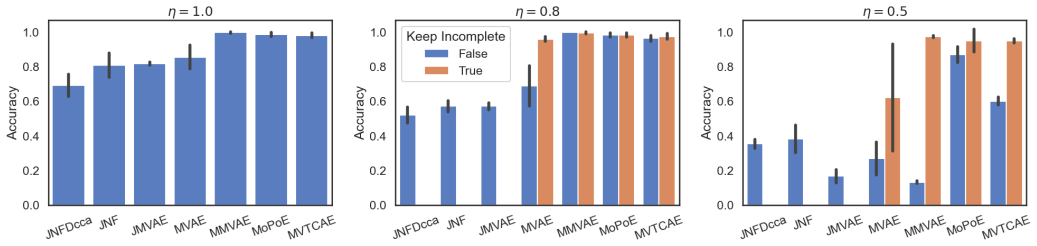

Figure 8: Accuracy of the K-means computing on latent representations for all the models. The color of the bar specifies the training paradigm. Standard deviations on 4 seeds are indicated.

4.3 Exploring another incomplete dataset

In section 4.2 we compared models on an incomplete dataset where gaps in the data where simulated at random, independently of the data. In real world incomplete datasets, missingness is often correlated with the data variables. To further explore the robustness of models to gaps in the data, we simulated another incomplete dataset where the probability of a modality sample to be missing depends on the label of the samples: samples with higher labels have a higher chance of being removed. Following such a principle, we create gaps in the multimodal 'Multimodal Handwritten Digits' (MHD) dataset introduced in [49] which contains digits in three modalities; image, sound and trajectory coordinates. Results for these experiments are given in Appendix E.

5 Conclusion

We have introduced *MultiVae*, a Python library unifying implementations of multimodal VAEs. The implementations were verified by reproducing a key result from the original papers and the library enables a flexible use of the models on new use-cases including incomplete datasets. This library intends to evolve through time with the addition of more models and tutorials. An interesting addition would be to include the impartial optimization training paradigm that [23] proposed to improve performance of aggregated models.

Impact statement and Ethics MultiVae was designed to foster research and development in the field of multimodal machine learning and its applications. Like other generative models, ill-intentionned or misuse of these models might facilitate misinformation. Developing safeguards around this issue is a crucial and active field of research [30] [2]. Another possible issue is representation biases which might occur when the models are trained on datasets that contains intrinsic biases [4]. We encourage users to check the distribution biases that may be contained in their dataset with regard to relevant sensitive attributes.

References

[1] Ana Lawry Aguila et al. "Multi-view-AE: A Python package for multi-view autoencoder models". In: *Journal of Open Source Software* 8.85 (2023), p. 5093. DOI: 10.21105/joss.05093. URL: https://doi.org/10.21105/joss.05093.

[2] Alim Al Ayub Ahmed et al. "Detecting fake news using machine learning: A systematic literature review". In: *arXiv preprint arXiv:2102.04458* (2021).

[3] Khaled Alsabti, Sanjay Ranka, and Vineet Singh. "An efficient k-means clustering algorithm". In: (1997).

[4] Federico Bianchi et al. "Easily Accessible Text-to-Image Generation Amplifies Demographic Stereotypes at Large Scale". In: *2023 ACM Conference on Fairness, Accountability, and Transparency*. ACM, June 2023. DOI: 10.1145/3593013.3594095. URL: https://doi.org/10.1145%2F3593013.3594095.

[5] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[6] Merlijn Blaauw and Jordi Bonada. "Modeling and transforming speech using variational autoencoders". In: *Morgan N, editor. Interspeech 2016; 2016 Sep 8-12; San Francisco, CA.[place unknown]: ISCA; 2016. p. 1770-4.* (2016).

[7] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. *Importance Weighted Autoencoders*. arXiv:1509.00519 [cs, stat]. Nov. 2016. DOI: 10.48550/arXiv.1509.00519. URL: http://arxiv.org/abs/1509.00519 (visited on 01/12/2023).

[8] Clément Chadebec, Louis J Vincent, and Stéphanie Allassonnière. "Pythae: Unifying Generative Autoencoders in Python–A Benchmarking Use Case". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks* (2022).

[9] Xinlei Chen et al. "Improved baselines with momentum contrastive learning". In: *arXiv preprint arXiv:2003.04297* (2020).

[10] Imant Daunhawer et al. "ON THE LIMITATIONS OF MULTIMODAL VAES". en. In: (2022), p. 27.

[11] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[12] Nat Dilokthanakul et al. "Deep unsupervised clustering with gaussian mixture variational autoencoders". In: *arXiv preprint arXiv:1611.02648* (2016).

[13] Hugging Face. *Hugging Face Hub*. 2023. URL: https://huggingface.co/docs/hub/index.

[14] Partha Ghosh et al. *From Variational to Deterministic Autoencoders*. en. arXiv:1903.12436 [cs, stat]. May 2020. URL: http://arxiv.org/abs/1903.12436 (visited on 08/18/2022).

[15] Jean-Bastien Grill et al. "Bootstrap your own latent-a new approach to self-supervised learning". In: *Advances in neural information processing systems* 33 (2020), pp. 21271–21284.

[16] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: 2016, pp. 770–778. URL: https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html (visited on 01/25/2023).

[17] Xiangteng He and Yuxin Peng. "Fine-Grained Visual-Textual Representation Learning". In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.2 (Feb. 2020), pp. 520–531. DOI: 10.1109/tcsvt.2019.2892802. URL: https://doi.org/10.1109%2Ftcsvt.2019.2892802.

[18] Martin Heusel et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in Neural Information Processing Systems*. 2017.

[19] Irina Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." In: *ICLR* 2.5 (2017), p. 6.

[20] HyeongJoo Hwang et al. "Multi-View Representation Learning via Total Correlation Objective". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 12194–12207. URL: https://proceedings.neurips.cc/paper/2021/hash/65a99bb7a3115fdede20da98b08a370f-Abstract.html (visited on 03/20/2023).

[21] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. arXiv:1611.07004 [cs]. Nov. 2018. DOI: 10.48550/arXiv.1611.07004. URL: http://arxiv.org/abs/1611.07004 (visited on 01/17/2023).

[22] Boris Ivanovic et al. “Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach”. In: *IEEE Robotics and Automation Letters* 6.2 (2020), pp. 295–302.

[23] Adrian Javaloy, Maryam Meghdadi, and Isabel Valera. “Mitigating Modality Collapse in Multimodal VAEs via Impartial Optimization”. In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 9938–9964. URL: https://proceedings.mlr.press/v162/javaloy22a.html.

[24] Dae Ung Jo et al. “Associative variational auto-encoder with distributed latent spaces and associators”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 11197–11204.

[25] Michael I. Jordan et al. “An Introduction to Variational Methods for Graphical Models”. en. In: *Learning in Graphical Models*. Ed. by Michael I. Jordan. Dordrecht: Springer Netherlands, 1998, pp. 105–161. DOI: 10.1007/978-94-011-5014-9_5. URL: http://link.springer.com/10.1007/978-94-011-5014-9_5 (visited on 01/17/2023).

[26] Tom Joy et al. “Learning multimodal VAEs through mutual supervision”. In: *arXiv preprint arXiv:2106.12570* (2021).

[27] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. “Contrastive Representation Learning: A Framework and Review”. In: *IEEE Access* 8 (2020), pp. 193907–193934. DOI: 10.1109/ACCESS.2020.3031549.

[28] Dhruv Khattar et al. “Mvae: Multimodal variational autoencoder for fake news detection”. In: *The world wide web conference*. 2019, pp. 2915–2921.

[29] Diederik P. Kingma and Max Welling. *Auto-Encoding Variational Bayes*. en. arXiv:1312.6114 [cs, stat]. May 2014. URL: http://arxiv.org/abs/1312.6114 (visited on 07/20/2022).

[30] Pavel Korshunov and Sébastien Marcel. “Deepfakes: a new threat to face recognition? assessment and detection”. In: *arXiv preprint arXiv:1812.08685* (2018).

[31] Mihee Lee and Vladimir Pavlovic. “Private-shared disentangled multimodal vae for learning of latent representations”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1692–1700.

[32] Ziwei Liu et al. “Deep Learning Face Attributes in the Wild”. In: 2015, pp. 3730–3738. URL: https://openaccess.thecvf.com/content_iccv_2015/html/Liu_Deep_Learning_Face_ICCV_2015_paper.html (visited on 01/24/2023).

[33] Kodai Minoura et al. “A mixture-of-experts deep generative model for integrated analysis of single-cell multiomics data”. In: *Cell reports methods* 1.5 (2021), p. 100071.

[34] Yuval Netzer et al. “Reading Digits in Natural Images with Unsupervised Feature Learning”. In: *NIPS* (Jan. 2011).

[35] Emanuele Palumbo, Imant Daunhawer, and Julia E Vogt. “MMVAE+: ENHANCING THE GENERATIVE QUALITY OF MULTIMODAL VAES WITHOUT COMPROMISES”. en. In: (2023).

[36] George Papamakarios, Theo Pavlakou, and Iain Murray. “Masked Autoregressive Flow for Density Estimation”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/hash/6c1da886822c67822bcf3679d04369fa-Abstract.html (visited on 01/17/2023).

[37] Adam Paszke et al. “Automatic differentiation in pytorch”. In: (2017).

[38] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. *Generating Diverse High-Fidelity Images with VQ-VAE-2*. arXiv:1906.00446 [cs, stat]. June 2019. DOI: 10.48550/arXiv.1906.00446. URL: http://arxiv.org/abs/1906.00446 (visited on 08/31/2022).

[39] Gabriela Sejnova, Michal Vavrecka, and Karla Stepanova. “Benchmarking Multimodal Variational Autoencoders: GeBiD Dataset and Toolkit”. In: (2022). arXiv: 2209.03048 [cs.LG].

[40] Agathe Senellart, Clément Chadebec, and Stéphanie Allassonnière. “Improving Multimodal Joint Variational Autoencoders through Normalizing Flows and Correlation Analysis”. In: *arXiv preprint arXiv:2305.11832* (2023).

[41] Yuge Shi et al. “Variational Mixture-of-Experts Autoencoders for Multi-Modal Deep Generative Models”. In: *arXiv:1911.03393 [cs, stat]* (Nov. 2019). arXiv: 1911.03393. URL: http://arxiv.org/abs/1911.03393 (visited on 04/20/2022).

[42] Thomas Sutter, Imant Daunhawer, and Julia Vogt. "Multimodal generative learning utilizing jensen-shannon-divergence". In: *Advances in neural information processing systems* 33 (2020), pp. 6100–6110.

[43] Thomas M. Sutter, Imant Daunhawer, and Julia E. Vogt. "Generalized Multimodal ELBO". In: *ICLR* (2021).

[44] Masahiro Suzuki, Takaaki Kaneko, and Yutaka Matsuo. *Pixyz: a library for developing deep generative models*. 2021. arXiv: 2107.13109 [cs.LG].

[45] Masahiro Suzuki and Yutaka Matsuo. "A survey of multimodal deep generative models". en. In: *Advanced Robotics* 36.5-6 (Mar. 2022). arXiv:2207.02127 [cs, stat], pp. 261–278. ISSN: 0169-1864, 1568-5535. DOI: 10.1080/01691864.2022.2035253. URL: http://arxiv.org/abs/2207.02127 (visited on 12/09/2022).

[46] Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. "Joint Multimodal Learning with Deep Generative Models". In: *arXiv:1611.01891 [cs, stat]* (Nov. 2016). arXiv: 1611.01891. URL: http://arxiv.org/abs/1611.01891 (visited on 04/15/2022).

[47] Yingtao Tian and Jesse Engel. "Latent Translation: Crossing Modalities by Bridging Generative Models". In: *ArXiv* (2019).

[48] George Tucker et al. "Doubly reparameterized gradient estimators for monte carlo objectives". In: *arXiv preprint arXiv:1810.04152* (2018).

[49] Miguel Vasco et al. "Leveraging hierarchy in multimodal generative models for effective cross-modality inference". en. In: *Neural Networks* 146 (Feb. 2022), pp. 238–255. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2021.11.019. URL: https://www.sciencedirect.com/science/article/pii/S0893608021004470 (visited on 03/20/2023).

[50] Ramakrishna Vedantam et al. "Generative Models of Visually Grounded Imagination". en. In: *arXiv:1705.10762 [cs, stat]* (Nov. 2018). arXiv: 1705.10762. URL: http://arxiv.org/abs/1705.10762 (visited on 05/12/2022).

[51] Weiran Wang et al. "Deep Variational Canonical Correlation Analysis". In: (2017). arXiv: 1610.03454 [cs.LG].

[52] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[53] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[54] Mike Wu and Noah Goodman. "Multimodal Generative Models for Scalable Weakly-Supervised Learning". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/1102a326d5f7c9e04fc3c89d0ede88c9-Abstract.html (visited on 04/15/2022).

[55] Jun-Yan Zhu et al. *Toward Multimodal Image-to-Image Translation*. arXiv:1711.11586 [cs, stat]. Oct. 2018. DOI: 10.48550/arXiv.1711.11586. URL: http://arxiv.org/abs/1711.11586 (visited on 01/17/2023).

[56] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This paper presents a Open-source Python library meant to facilitates the development and benchmark of models.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] This paper doesn't introduce any new theoretical results.
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We use well-known database.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We use well-known database.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Details on the models implemented in the library

In this appendix, we provide a brief explanation of each model and, when possible, how they have been adapted for incomplete datasets in the MultiVae library. Table A summarize key informations about the models. Note that when a model uses a multistage training, a specific trainer has been implemented in MultiVae to handle it without further complications. The trainer module suited to each model is indicated in Appendix F, along with snippets of code.

Model	Type	Incomplete data	Multiple Latent Spaces	Multistage training	Higher inference time
JMVAE	Joint encoder, Fig 1.a)	No	No	No	
TELBO	Joint encoder, Fig 1.a)	No	No	Yes	
MVAE	Aggregated, Fig 1.b)	Yes	No	No	
MMVAE	Aggregated, Fig 1.b)	Yes	No	No	
MVTCAE	Aggregated, Fig 1.b)	Yes	No	No	
MoPoE	Aggregated, Fig 1.b)	Yes	Possible	No	
JNF	Joint encoder, Fig 1.a)	No	No	Yes	x
JNFDcca	Joint encoder, Fig 1.a)	No	No	Yes	x
MMVAE+	Aggregated, Fig 1.b)	Yes	Yes	No	

Table 1: For each model, we indicate if it is a joint encoder model or an aggregated model. The relevant diagram depicting the model can be found in Figure 1. We also indicate the possibility to train with incomplete data, and if they consider multiple latent spaces. For the JNF, JNFDcca models, we indicate a higher inference time, because Monte-Carlo sampling is involved for infering the latent code from a subset of modalities when it is not a singleton.

A.1 JMVAE

The JMVAE model [46] is one of the first multimodal variational autoencoders models that were proposed by [46]. It has a dedicated joint encoder network $q_\phi(z|X)$ and surrogate unimodal encoders $q_{\phi_j}(z|x_j)$. The JMVAE-KL loss has additional terms to the ELBO to fit the unimodal encoders:

$$\mathcal{L}_{JMVAE}(X) = \mathbb{E}_{q_\phi(z|X)}\left[p_\theta(z|X)\right] - KL\left(q_\phi(z|X)||p_\theta(z)\right) - \alpha\sum_{j=1}^{M} KL\left(q_\phi(z|X)||q_{\phi_j}(z|x_j)\right) \tag{6}$$

Those additional terms are interpreted in [46] in terms of of minimizing the Variation of Information (VI) between modalities. Another insightful interpretation is provided in [50]. α is the parameter that controls a trade-off between the quality of reconstruction and the quality of cross-modal generation [46]. This model has been proposed for only two-modalities, but an extension has been proposed in [40] for dealing with more modalities. During inference, when $M \leq 2$, the subset posteriors $p_\theta(z|(x_j)_{j\in S})$ can be approximated by the product of experts (PoE) of the already trained unimodal encoders $q_\phi(z|x_j)_{1\leq j\leq M}$. Since the unimodal posteriors are normal distributions, the PoE has a closed-form and can easily be computed.

The original JMVAE model uses annealing during training: which means that a weighting factor that ponders the regularizations terms is linearly augmented from 0 to 1 during the N_t first epochs.

As it uses a joint encoder network, this model can not be trained with partially observed samples.

A.2 TELBO

The TELBO [50] model use a joint encoder as the JMVAE but uses the following Triple ELBO loss:

$$\mathcal{L}(X) = \mathbb{E}_{q_\phi(z|X)}\left[\frac{p_\theta(z,X)}{q_\phi(z|X)}\right] + \sum_{j=1}^{M} \mathbb{E}_{q_\phi(z|x_j)}\left[\frac{p_\theta(z,x_j)}{q_\phi(z|x_j)}\right] \tag{7}$$

It is trained with a two-steps training, first learning the joint encoder and decoders then, training the unimodal encoders with previous parameters fixed.

As it uses a joint encoder network, this model can not be trained with partially observed samples.

A.3 MVAE

The MVAE model was the first aggregated model proposed by [54]. To solve the scalability issue of the JMVAE model, the MVAE suggest to model the joint posterior as a PoE $q_\phi(z|X) \propto p(z)\prod_j q_{\phi_j}(z|x_j)$. The ELBO is then optimized:

$$\mathcal{L}_{MVAE}(X) = \mathbb{E}_{q_\phi(z|X)}\left[\log p_\theta(X|z)\right] - KL(q_\phi(z|X)||p_\theta(z)) \tag{8}$$

This ELBO can be computed on a subset S by taking only the modalities in the subset to compute the PoE:

$$q_\phi(z|X) \propto p(z)\prod_{j\in S} q_{\phi_j}(z|x_j) \tag{9}$$

To ensure all unimodal encoders are correctly trained, the MVAE uses a sub-sampling training paradigm, meaning that at iteration, the ELBO is computed for several subsets: the joint subset $\{1,..,M\}$, the unimodal subsets and for K random subsets. For each sample X, the objective then becomes

$$\mathcal{L}_{MVAE}(X) + \sum_j \mathcal{L}_{MVAE}(x_j) + \sum_{k=1}^{K} \mathcal{L}_{MVAE}((x_j)_{j\in s_k}) \tag{10}$$

where s_k are random subsets. As an aggregated model, this model can be used in the partially observed setting. In the partially observed setting, we don't use the sub-sampling paradigm since the dataset is naturally sub-sampled, and we compute the ELBO with only the observed modalities in $S_{obs}(X) : q_\phi(z|X) \propto p(z)\prod_{j\in S_{obs}(X)} q_{\phi_j}(z|x_j)$.

A.4 MMVAE

The MMVAE model [41] uses a mixture-of-experts (MoE) aggregation. It also uses a k-samples IWAE lower bound. The MMVAE loss writes as follows:

$$\frac{1}{M}\sum_{j=1}^{M} \mathbb{E}_{z^{(1)},\dots z^{(k)}\sim q_{\phi_j}(z|x_j)}\left[\log \frac{1}{K}\sum_k \frac{p_\theta(z^{(k)},X)}{q_\phi(z|X)}\right] \tag{11}$$

Note the sum over the modalities; for each sample X, K latent codes are sampled from $q_{\phi_k}(z|x_j)$ for each $1 \leq j \leq M$. In total $M \times K$ latent codes are sampled per data-point. The original MMVAE model uses Laplace posteriors while constraining their scaling in each direction to sum to D, the dimension of the latent space. This favours disentanglement in the latent space [41].

A DreG estimator [48] is used to train the IWAE objective.

To use this model in the partially observed setting, it suffices to sum over the available modalities and to take the mixture of experts $q_\phi(z|X)$ over the available modalities as well.

For instance, if $S_{obs}(X)$ is the subset of observed modalities for sample X the loss becomes:

$$\frac{1}{|S_{obs}(X)|}\sum_{j\in S_{obs}(X)} \mathbb{E}_{z^{(1)},\dots z^{(k)}\sim q_{\phi_j}(z|x_j)}\left[\log \frac{1}{K}\sum_k \frac{p_\theta(z^{(k)},X)}{q_\phi(z|X)}\right] \tag{12}$$

with the joint posterior $q_\phi(z|X)$ computed as the mixture of available experts.

A.5 MoPoE-VAE

The MoPoE-VAE [43] suggest to combines the advantages of the two models above by using a Mixture of Product of Experts. Formally, for each subset $S \in \mathcal{P}(\{1, \ldots, M\})$ a PoE distribution is defined $\tilde{q}_\phi(z|(x_j)_{j\in S}) = PoE((q_{\phi_j}(z|x_j))_{j\in S})$. Then the joint posterior is defined as:

$$q_\phi(z|X) = \frac{1}{2^M - 1} \sum_{S\in P(\{1,\ldots,M\}, S\neq\{\}} \tilde{q}_\phi(z|(x_j)_{j\in S})$$

. The ELBO is optimized:

$$\mathcal{L}(X) = \mathbb{E}_{q_\phi(z|X)} \left[\ln p_\theta(X|z)\right] - KL(q_\phi(z|X)|p_\theta(z)) \tag{13}$$

The MoPoE loss resemble the MVAE subsampled objective with several notable differences:

- The MoPoE objective includes all subsets while the MVAE objective only include the joint, unimodal subsets and K others random subsets.
- The Product of experts in MoPoE does not include the prior distribution except for the entire subset $S = \{1, \ldots, M\}$.
- In the MVAE subsampled objective, all the subsampled ELBOs are computed for each datapoint X. In the MoPoE objective, for each datapoint a subset is *sampled* at each iteration and only that subset ELBO is computed.

The MoPoE uses a β parameter to tune the regularization in the ELBO. To adapt this model to the partially observed setting, the loss is computed with all available subsets $S \in S_{obs}(X)$, where $S_{obs}(X)$ is the set of observed modalities for the sample X at hand, instead of all subsets $S \in P(\{1, \ldots, M\}$.

A.6 MVTCAE

The MVTCAE [20] model uses a PoE in a similar fashion as the MVAE but without the prior:

$$q_\phi(z|X) \sim \prod_j q_{\phi_j}(z|x_j) \tag{14}$$

The MVTCAE loss is derived from a Total Correlation Analysis and writes as follows:

$$\begin{aligned}\mathcal{L}(X) = &\frac{M-\alpha}{M}\mathbb{E}_{q_\phi(z|X)}\left[\log p_\theta(X|z)\right] \\ &- \beta\left[(1-\alpha)KL(q_\phi(z|X)||p_\theta(z)) + \frac{\alpha}{M}\sum_{j=1}^{M} KL(q_\phi(z|X)||q_{\phi_j}(z|x_j)\right]\end{aligned} \tag{15}$$

Although this loss derives from a different analysis, it uses same terms that in the JMVAE model. A β factor weighs the regularization, while the α parameters is used to weight out the different divergence terms: the Kullback-Leilbler divergence of the joint encoder to the prior refered to as Variational Information Bottleneck (VIB), and the Kullback-Leibler divergence between posteriors referred to as Conditional Variational Information Bottlenecks (CVIB).

For the partially observed setting, we follow the authors' indications setting the variance for the missing modalities' decoders to ∞ which amounts to setting the reconstruction loss to 0 for those modalities. The KL terms for missing modalities are also set to 0.

A.7 JNF, JNFDcca

The JNF, JNFDcca [40] models use a joint encoder to model $q_\phi(z|X)$ and surrogate unimodal encoders $q_{\phi_j}(z|x_j)$ for $1 \leq j \leq M$.

For the JNF model, the loss used is the same as the JMVAE (with $\alpha = 1$) but the unimodal encoders $q_{\phi_j}(z|x_j)$ are modelled with Masked Autoregressive Flows [36] to improve the flexibility and correctness of the inference.

$$\mathcal{L}_{JNF}(X) = \mathbb{E}_{q_\phi(z|X)}\left[\frac{p_\theta(z,X)}{q_\phi(z|X)}\right] - \sum_{j=1}^{M} KL(q_\phi(z|X)||q_{\phi_j}(z|x_j)) \tag{16}$$

For the JNFDcca model, the unimodal encoders distributions are conditioned not on the modality samples directly but on a DCCA embedding: $q_{\phi_j}(z|f_j(x_j))$ where $f_j(x_j)$ are pretrained DCCA embeddings for $j = 1, ..M$. The loss then writes:

$$\mathcal{L}_{JNFDcca}(X) = \mathbb{E}_{q_\phi(z|X)}\left[\frac{p_\theta(z,X)}{q_\phi(z|X)}\right] - \sum_{j=1}^{M} KL(q_\phi(z|X)|q_{\phi_j}(z|f_j(x_j)) \tag{17}$$

Conditioning on DCCA embeddings allows improve coherence by extracting the shared information relevant for cross-modal generation.

Those two models are trained with a two-steps training as the TELBO model; first the joint encoder and decoders are trained, and then those parameters are fixed while the unimodal encoders are trained. This two steps training replaces the α parameter in the original JMVAE loss.

A.8 MMVAE+

The MMVAE+ model [35] is an aggregated model that uses multiple latent spaces : z is the latent code *shared* accross modalities and w_j is private latent code of modality j for $1 \leq j \leq M$.

It has twice as many encoders as the MMVAE: $q_{\phi_{z_m}}(z|x_m)$ is the encoder for shared latent code given the modality m, $q_{\phi_{w_m}}(w_m|x_m)$ is the encoder for private latent code given the modality m.

It also uses specific prior distributions for each private latent spaces $r_n(w_n)$ with a scale parameter that is tuned.

As for the MMVAE, the joint posterior for the shared latent code is a mixture of experts of the unimodal posteriors:

$$q_{\phi_z}(z|X) = \frac{1}{M}\sum_{m=1}^{M} q_{\phi_{z_m}}(z|x_m) \tag{18}$$

The loss of the MMVAE+ model then writes as follows:

$$\frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{\substack{z_m^{1::K}\sim q_{\phi_{z_m}}(z|x_m) \\ w_m^{1::K}\sim q_{\phi_{w_m}}(z|w_m) \\ \tilde{w}_{n\neq m}^{1::K}\sim r_n(w_n)}} \log \frac{1}{K}\sum_{k=1}^{K} D^{\beta}_{\Phi,\Theta}(X, z^k, \tilde{w}_1^k, \tilde{w}_2^k, .., w_2^m, .., \tilde{w}_M^k) \tag{19}$$

with

$$D^{\beta}_{\Phi,\Theta}(X, z^k, \tilde{w}_1^k, \tilde{w}_2^k, .., w_2^m, .., \tilde{w}_M^k) = \frac{p_{\theta_m}(x_m|z^k, w_m^k)(p(z^k)p(w_m^k))^\beta}{(q_{\phi_z}(z^k|X)q_{\phi_{w-m}}(w_m^k|x_m))^\beta} \prod_{n\neq m} p_{\theta_n}(x_n|z^k, \tilde{w}_n^k) \tag{20}$$

It uses a k-importance sampled estimator of the likelihood and a β factor that can be tuned to promote disentanglement in the latent space. In this objective function, the modality private information w_m is only used for self reconstruction and not for cross-modal generation. For cross modal reconstruction, only the shared latent code can be used which forces it to contain the information shared across modalities.

For the partially observed case, that loss can be computed using only available sample instead of all modalities.

For instance if we only observe modalities in $S_{obs}(X)$ the computed loss for sample X becomes:

$$\frac{1}{|S_{obs}(X)|}\sum_{m\in S_{obs}(X)}\mathbb{E}_{\substack{z_m^{1::K}\sim q_{\phi_{z_m}}(z|x_m)\\ w_m^{1::K}\sim q_{\phi_{w_m}}(z|w_m)\\ \tilde{w}_{n\neq m}^{1::K}\sim r_n(w_n)}}\log\frac{1}{K}\sum_{k=1}^{K}D^{\beta}_{\Phi,\Theta}(X,z^k,\tilde{w}_1^k,\tilde{w}_2^k,..,w_2^m,..,\tilde{w}_M^k) \quad (21)$$

with

$$D^{\beta}_{\Phi,\Theta}(X,z^k,\tilde{w}_1^k,\tilde{w}_2^k,..,w_2^m,..,\tilde{w}_M^k)=\frac{p_{\theta_m}(x_m|z^k,w_m^k)(p(z^k)p(w_m^k))^{\beta}}{(q_{\phi_z}(z^k|X)q_{\phi_{w_m}}(w_m^k|x_m))^{\beta}}\prod_{n\neq m}p_{\theta_n}(x_n|z^k,\tilde{w}_n^k) \quad (22)$$

computed with the joint posterior computed with available modalities:

$$q_{\phi_z}(z|X)=\frac{1}{|S_{obs}(X)|}\sum_{m\in S_{obs}(X)}q_{\phi_{z_m}}(z|x_m) \quad (23)$$

B Technical details on the experiments

B.1 PolyMMNIST dataset

In Figure B.1, we plot example images of the incomplete PolyMNIST dataset used in the experiments.

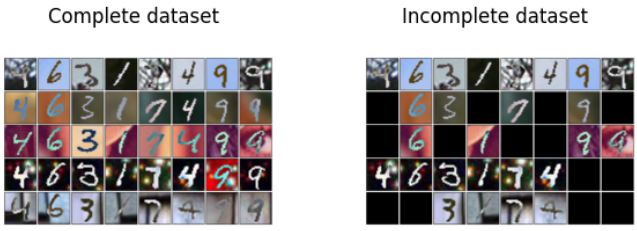

Figure 9: Examples from a complete PolyMNIST dataset and an incomplete one obtained with $\eta = 0.5$. Each row represents a modality and each column a sample.

B.2 Architectures

We use Laplace decoder distribution for all five modalities, with scaling parameter $\sigma = 0.75$ and a latent dimension of size 512, as done in [43, 20, 35].

The convolutional architectures used in section 4.2 are very similar to the ones used in [43, 20]. Figure 10 describes the encoders and decoders architectures used on the PolyMNIST images.

For more details on the training of each model, we refer to the scripts available at `https://github.com/AgatheSenellart/nips_experiments`.

B.3 Retrieving trained models

All models trained in this project are hosted on the HuggingFace Hub and can be downloaded. For instance see `https://huggingface.co/asenella/mmnist_MMVAEconfig2_seed_0_ratio_02_`

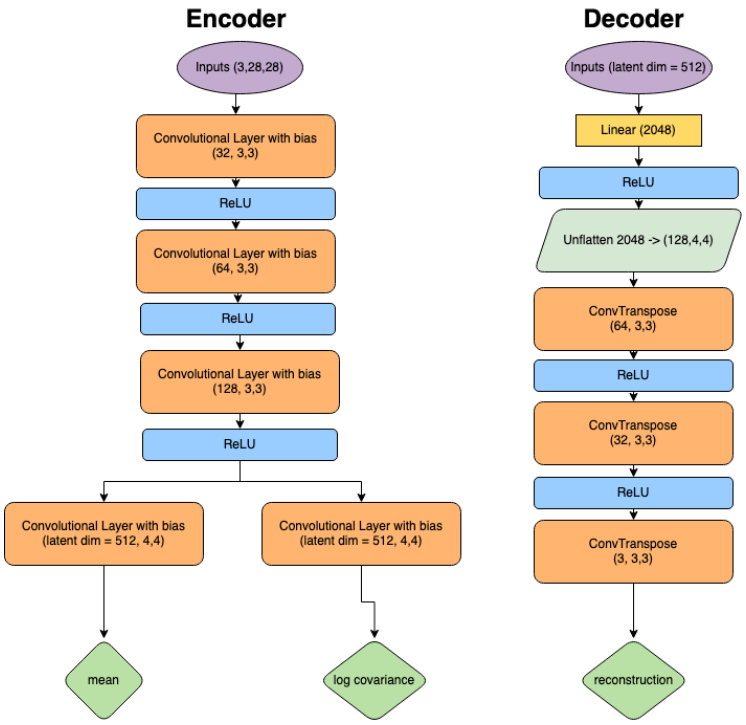

Figure 10: The encoder, decoder architectures used for the comparison between models in section 4.2.

`i`. Each run has a specific id corresponding to its configuration. To see how to retrieve them all, please look at the file 'eval_hf_models.py' as well the 'README.md' in the github repository of the project: `https://github.com/AgatheSenellart/nips_experiments`.

C Additional experimental results

C.1 Visualization of generated samples

C.1.1 Conditional generation and reconstruction

In this section, we present conditionally generated samples for all models, for the different degrees of missing data (best viewed with zoom). For the aggregated models, presented samples corresponds to models trained using all, even incomplete samples. For the joint encoder models that don't handle missing data, the presented samples correspond to models trained only on complete samples.

In Figure. 11, we condition on two modalities (presented as the first two rows) to infer a latent code and reconstruct all modalities from that latent code.

In Figure. 12, we condition on all five modalities (presented as the first five rows) to infer a latent code and reconstruct all modalities from that latent code.

C.1.2 Unconditional generations

Figure 13 presents unconditionally generated samples, when sampling the latent code from a), the prior and b) a Gaussian Mixture Model (GMM) with 10 components, fit on training latent codes (best viewed with zoom). For all models, the generated samples are much more realistic and coherent when using the GMM sampler.

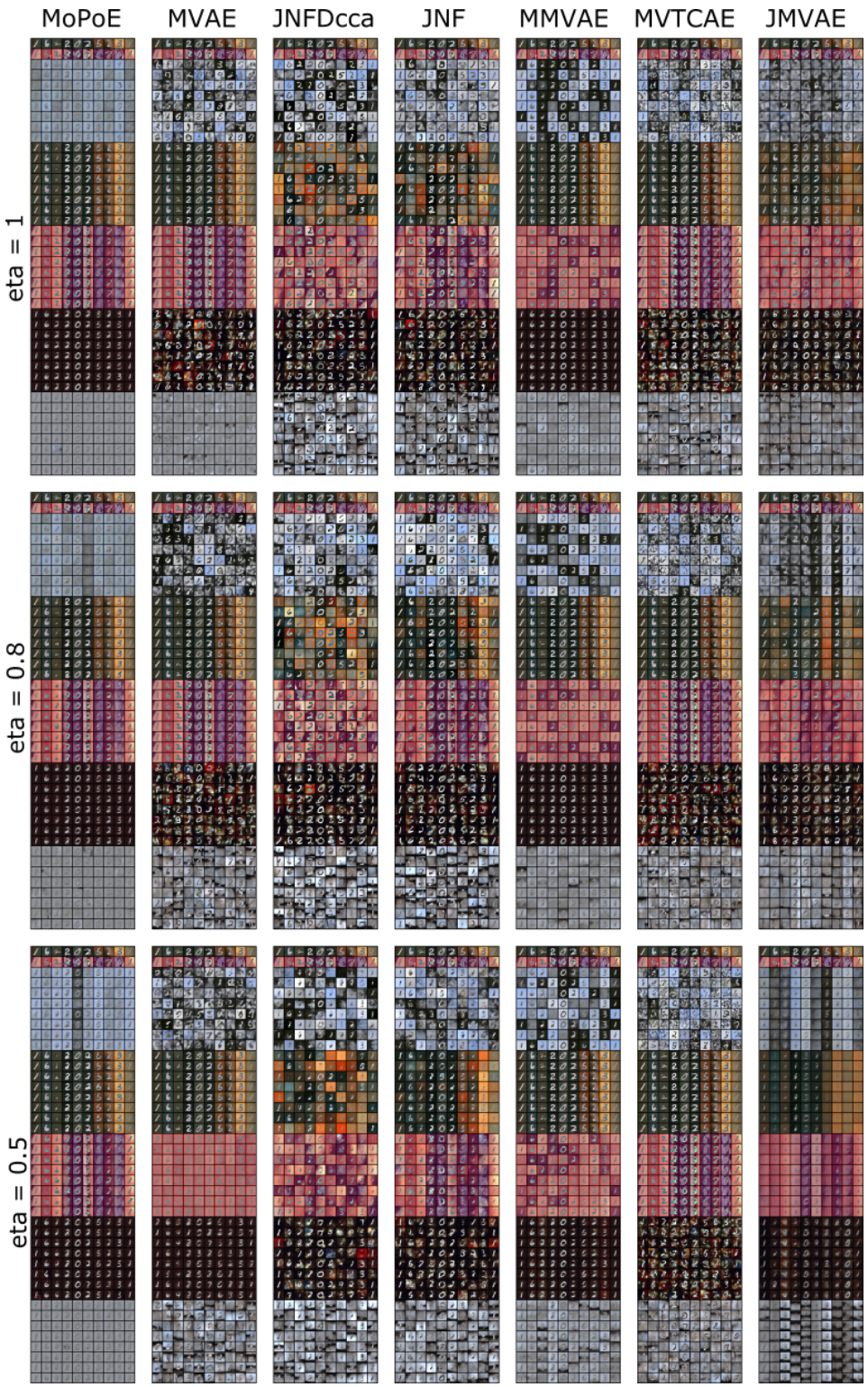

Figure 11: Conditional generation for all models for different η. For aggregated models, we present the results when training on all samples even incomplete ones. For joint encoder models, we only use complete samples. The first two lines are the conditioning images, while all the next rows are generated images.

Figure 12: Joint reconstruction of all modalities simultaneously. For aggregated models, we present the results when training on all samples even incomplete ones. For joint encoder models, we only use complete samples. The first five lines are the conditioning images, while all the next rows present the reconstructions.

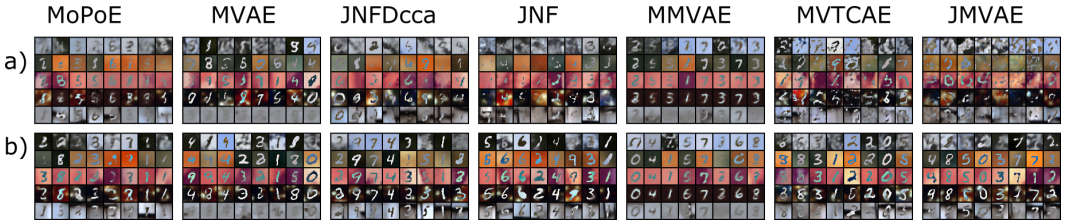

Figure 13: Unconditional generation sampling a) from the prior, b) from the Gaussian Mixture Model Prior. This generation are obtained in the case $\eta = 1$. Note that each column represent samples generated from the same shared latent code, and therefore are supposed to be *coherent* meaning sharing digit information.

C.2 Joint likelihoods for all models

In Figure 14 we present computed likelihoods for all models and all configurations. The likelihoods are computed by taking the joint posterior $q_\phi(z|X)$ as importance sampling distribution and 1000 importance samples for all datapoints.

We see that joint encoders models and the MVTCAE tend to obtain better joint likelihoods values.

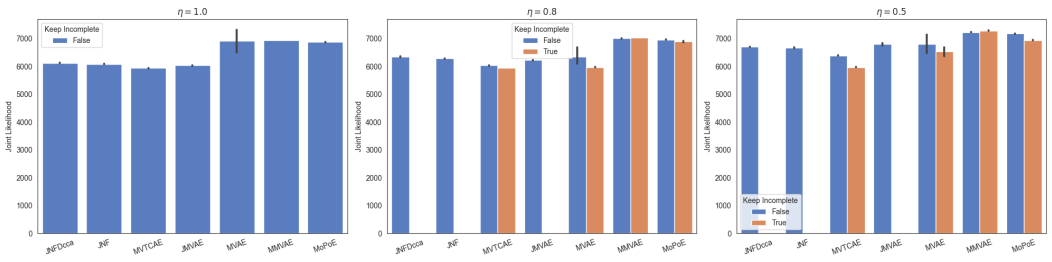

Figure 14: Negative log likelihoods for all models and configurations, averaged on 4 seeds.

C.3 Training times for all models

In Table. 2 we indicate approximate training times for the experiments in 4.2 with $\eta = 1$. To do so, we have used the training curves available in the Wandb workspace `https://wandb.ai/multimodal_vaes/compare_on_mmnist/` to get the number of epochs necessary for each model to reach convergence and the average computing time for each epoch. Note that the training time for each model may vary depending on the GPU used, the architectures, and the dataset.

MODEL	TIME FOR ONE EPOCH	NUMBER OF EPOCHS FOR CONVERGENCE	TIME FOR CONVERGENCE
JMVAE	19S	200	61 MIN
MMVAE (K=10)	85S	50	71 MIN
MVAE	24S	200	80 MIN
MOPOE	24S	150	59 MIN
MVTCAE	14S	400	92MIN
JNF	13S	500	108 MIN
JNFDCCA	10S	700	112MIN

Table 2: Approximate training times for each model to reach convergence on the validation set. The indicated values refer to the experiments in paragraph 4.2, obtained in the complete dataset scenario ($\eta = 1.0$). For the JNF and JNFDcca model that have a multistage training, the values refer to the entire training process. The GPUs used for those values are Quadro RTX 6000.

C.4 Inference times for all models

In Table. 3 we provide a comparison in inference times for all models trained in the case-study 4.2 with $\eta = 1.0$, and seed = 0. As expected the JNF, JNFDcca models have a larger inference time since it uses Monte-Carlo Sampling when infering the latent code from a subset of modalities that is not a singleton [40].

NUMBER OF INPUT MODALITIES	MVTCAE	MOPOE	MMVAE	JNFDCCA	JNF	JMVAE
1	0.05	0.08	0.04	0.95	1.16	0.06
2	0.03	0.07	0.03	12.54	20.75	0.04
3	0.04	0.09	0.03	19.17	31.75	0.04
4	0.05	0.08	0.03	26.01	42.26	0.05
5	0.06	0.07	0.03	0.03	0.06	0.06

Table 3: Inference time in seconds to generate all modalities from a subset of modalities for a minibatch of a hundred samples. These results are obtained on CPU.

C.5 Reconstruction metrics for all models

In this section, we provide metrics that quantify the quality of the reconstructions for each model, in each scenario of the case study. To compare original and reconstructed images, we use two different metrics:

- The Mean Squared Error (MSE) `https://en.wikipedia.org/wiki/Mean_squared_error` that is summed over the image's pixels.
- The Structural Similarity Index Measure (SSIM) (`https://fr.wikipedia.org/wiki/Structural_Similarity`) [52]

The SSIM metric was constructed to better reflect human perception of resemblance between images than MSE. [52]

C.5.1 Unimodal reconstructions

First we investigate the quality of reconstructed images when using the unimodal posteriors to reconstruct each modality independently of the others. More specifically, for each modality $1 \leq j \leq M$, we sample a latent code $z \sim q_{\phi_j}(z|x_j)$ from the unimodal posterior and reconstruct the modality sample by taking the mean of the decoder distribution $p_\theta(x_j|z)$. Figure. 15 present the MSE

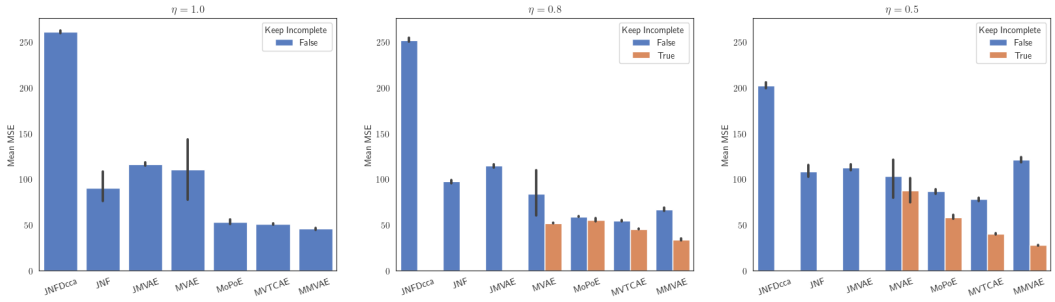

Figure 15: MSE (↓) averaged over the 5 modalities for unimodal reconstructions. We present the mean over four seeds with standard errors. For this metric, smaller is better.

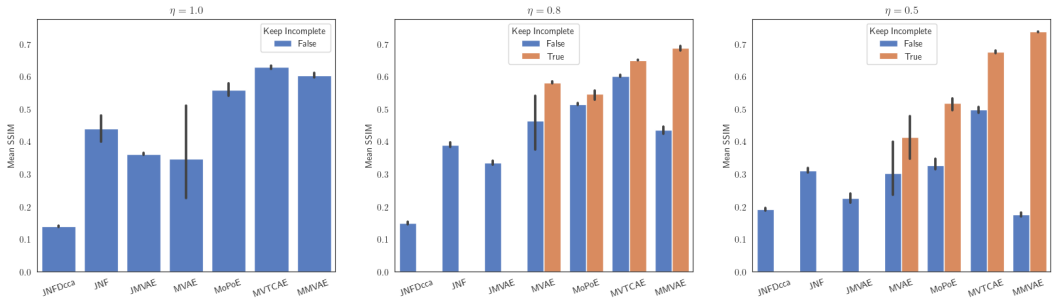

Figure 16: SSIM (↑) averaged over the 5 modalities for unimodal reconstructions. We present the mean over four seeds with standard errors. For this metric, higher is better.

values, averaged over the five modalities for the entire test set of PolyMNIST. Similarly, Figure. 16 presents the SSIM values, averaged over the five modalities for the entire test set of PolyMNIST. The JNFDcca model is the worst performing model for those metrics, which is coherent with the model design. In this method the unimodal posteriors learned are $q_{\phi_j}(z|f_j(x_j))$ are conditioned on the DCCA representation of the modality which extract the information shared accross modalities. The unimodal posteriors in that model are designed to perform well in cross-modal generation but not in the reconstruction task.

C.5.2 Joint reconstruction

Secondly, we present results for the joint reconstruction of all the modalities. To be precise, we sample a latent code $z \sim q_\phi(z|(x_j)_{1\leq j\leq M})$ from the joint posterior distribution and reconstruct all modalities from this latent code by taking the mean of each decoder distribution $p_\theta(x_j|z)$ for $1 \leq j \leq M$.

Figure. 17 presents the MSE for joint reconstruction, while Figure. 18 show the SSIM for joint reconstruction. Both metrics are computed over the entire test set and averaged over the four seeds.

For joint reconstruction, the MMVAE is the worst performing model because reconstructed images are somewhat averaged and do not preserve the background as can be seen in Figure. 12. On the other hand, we note that the MVTCAE obtain excellent values, even in with incomplete datasets.

D Details on the reproduction of previous results

D.1 JMVAE

To validate the JMVAE model, we reproduce the experiment on the binary MNIST dataset. For this dataset, the images are randomly binarized (with bernoulli laws that have for parameters the intensity of the grey-level MNIST images) at each epoch.

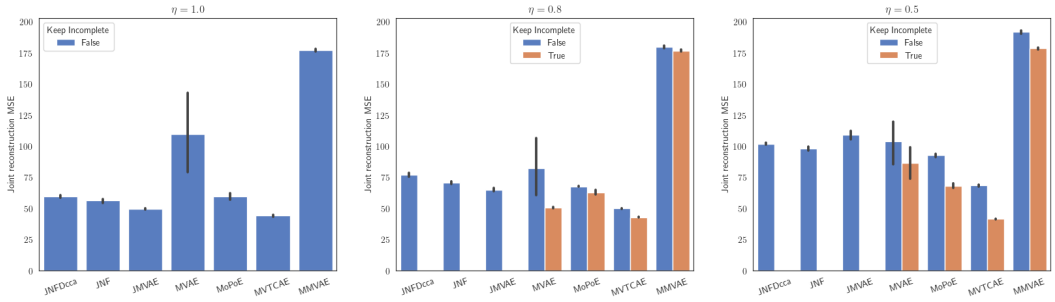

Figure 17: MSE (↓) for joint reconstruction. We present the mean over four seeds with standard errors. For this metric, smaller is better.

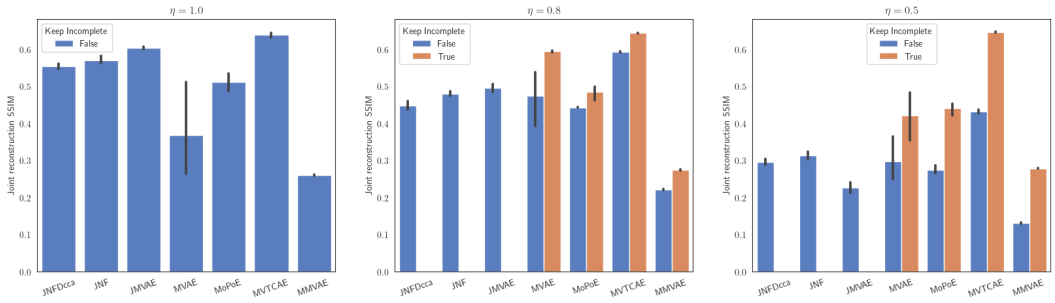

Figure 18: SSIM (↑) for joint reconstruction. We present the mean over four seeds with standard errors. For this metric, higher is better.

The script used for reproducing this model is under "/reproducing_experiments/jmvae" in `https://github.com/AgatheSenellart/nips_experiments`.

The wandb training runs can be visualized on wandb at `https://wandb.ai/multimodal_vaes/reproduce_jmvae`, with some sample images.

Three pretrained models can be downloaded from HuggingFace Hub at:

- `https://huggingface.co/asenella/reproduce_jmvae_seed_1`
- `https://huggingface.co/asenella/reproduce_jmvae_seed_2`
- `https://huggingface.co/asenella/reproduce_jmvae_seed_8`

Validation scripts in `https://github.com/AgatheSenellart/nips_experiments` detail how to load the models.

The metric we reproduced is the likelihood of the image modality $\log p(x)$ where x is the image using the joint posterior $q_\phi(z|x, y)$ as the importance sampling distribution. We obtain -86.85 ± 0.03 on three different runs while the value presented in the original paper is -86.86.

D.2 MMVAE

For the MMVAE model, we reproduce the MNIST-SVHN experiment. The metrics we focus on reproducing are the accuracies of cross-modal generation. The values presented in the original paper are 86.4% for (MNIST $\longrightarrow$ SVHN) and 69.1 for (SVHN $\longrightarrow$ MNIST). However those results are obtained on one specific seed. When running the original code on four different seeds, we obtain 87 ± 2 and 59 ± 11.

Using our implementation, we obtain similar values; 60 ± 6 and 87 ± 2 when averaging on four seeds as well. We also obtain similar joint generation coherence: 39 ± 3 with the original code, 38 ± 3 with our implementation.

The script used for reproducing the experiment are available under /reproducing_experiments/mmvae in https://github.com/AgatheSenellart/nips_experiments.

One trained model is available at https://huggingface.co/asenella/reproduce_mmvae_model.

The training runs of the four models used for reproduction can be visualized at https://wandb.ai/multimodal_vaes/reproducing_mmvae.

D.3 MVAE

For the MVAE model, we reproduce the Binary MNIST experiment using the same dataset as for the JMVAE experiment. Unfortunatly we weren't able to reproduce the likelihoods presented in the paper, as the code for computing the likelihoods have not been made available by the authors. Therefore we validated our implementation by running the original code and our code, and comparing the final ELBO values on the test dataset: we obtain 188.8 with the original code and 188.3 ± 0.4 with our implementation.

The script used for reproducing the experiment are available under /reproducing_experiments/mvae/mnist in https://github.com/AgatheSenellart/nips_experiments.

The runs can be visualized on Wandb at https://wandb.ai/multimodal_vaes/reproduce_mvae_mnist.

The trained models can be retrieved on the HuggingFace Hub under:

- https://huggingface.co/asenella/reproduce_mvae_mnist_0
- https://huggingface.co/asenella/reproduce_mvae_mnist_1
- https://huggingface.co/asenella/reproduce_mvae_mnist_2

D.4 MoPoE

For the MoPoE model, we reproduce the PolyMNIST experiment. We focus on reproducing the cross-modal generation coherence, averaged on three different runs.

The script used for reproducing the experiment are available under /reproducing_experiments/mopoe in https://github.com/AgatheSenellart/nips_experiments.

The runs can be visualized on Wandb at https://wandb.ai/multimodal_vaes/reproducing_mopoe.

The trained models can be retrieved at

- https://huggingface.co/asenella/reproducing_mopoe_seed_0
- https://huggingface.co/asenella/reproducing_mopoe_seed_1
- https://huggingface.co/asenella/reproducing_mopoe_seed_2

The coherences presented in 3 are averaged on the five runs and correspond to Figure 3. in the original paper. The results we obtain on our side are slightly better than the ones obtained in the original paper. After investigating the original code, we found that this small discrepancy comes from forgetting setting the classifiers in eval mode in the original code.

D.5 MVTCAE

For the MVTCAE model we also reproduce the PolyMNIST experiment using the same dataset as the one used in MoPoE.

The script used for reproducing the experiment are available under /reproducing_experiments/mvtcae in https://github.com/AgatheSenellart/nips_experiments.

The runs can be visualized on Wandb at https://wandb.ai/multimodal_vaes/reproducing_mvtcae.

The trained models can be retrieved at

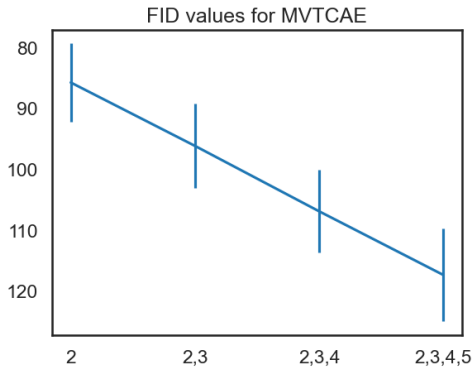

Figure 19: FID values computed on three seeds with the standard deviations plotted as error bars. This Figure is the reproduction of the rightmost graph of Figure 2. in the original paper.

- `https://huggingface.co/asenella/reproducing_mvtcae_seed_0`
- `https://huggingface.co/asenella/reproducing_mvtcae_seed_1`
- `https://huggingface.co/asenella/reproducing_mvtcae_seed_2`

For this model, in addition to the cross-modal coherences we also computed the FID values that are presented 19 and are similar to the ones presented in the rightmost plot of Figure 2 in the original paper [20].

D.6 MMVAE+

To validate the MMVAE+ model, we reproduce the PolyMNIST experiment using the same dataset as above with $K = 1$ and $\beta = 2.5$.

The script used for reproducing the experiment are available under /reproducing_experiments/mmvae_plus in `https://github.com/AgatheSenellart/nips_experiments`.

The runs can be visualized on Wandb at `https://wandb.ai/multimodal_vaes/reproducing_mmvae_plus`.

The trained models can be retrieved at

- `https://huggingface.co/asenella/reproduce_mmvaep_K__1__seed_0`
- `https://huggingface.co/asenella/reproduce_mmvaep_K__1__seed_1`
- `https://huggingface.co/asenella/reproduce_mmvaep_K__1__seed_2`

In table 3, we compared the mean coherence from one modality to another and the mean FID generating each modality from the prior. We found 88.6 ± 0.8 for the coherences, a value close to the one in the paper 86.9 ± 1.3, and 93 ± 5 for the unconditional FID reported at 96 ± 2 in the paper. In the original paper, those values can be found in Appendix G. Table 3.

E Additional experimental results on the MHD dataset

E.1 The Multimodal Handwritten Digits dataset

The 'Multimodal Handwritten Digits' (MHD) introduced in [49] contains 4 modalities:

- Image: gray digit images with one channel and size (28,28)
- Audio: spectrograms images with shape (1,32,128)
- Trajectory : flat arrays with 128 values

- Label : 10 categorical values

In our experiments, we don't use the label modality to make the task more challenging.

E.2 Incomplete version of the MHD dataset

To create an incomplete version of this dataset, we defined missing probabilities for each modality, depending on the label.

- For the image modality, we kept all samples.
- For the audio modality, the missing probabilities were evenly distributed between 0.1 for the class 0 and 0.8 for the class 9.
- For the trajectory modality, the missing probabilites were evenly distributed between 0.3 (for the class 0) and 0.4 (for the class 9).

This procedure creates an incomplete dataset of type MAR (Missing at Random) while the incomplete dataset in section 4.2, was of type MCAR (Missing Completely At Random). It is more realistic and more challenging as it creates imbalance between the classes.

E.3 Architectures and training parameters

On the complete dataset we train seven models while on the incomplete dataset we only train the 5 aggregated models available in the library. We use Normal decoder distributions for all three modalities and a latent dimension of size 64. For MMVAE+ that uses private and shared latent spaces, the shared space is of size 32 and each private latent space has size 32. We use the same encoders and decoders architectures for all models. Those architectures are very close to the ones used in [49], except that we don't pretrain the architectures for the audio modality. The MMVAE, and MMVAE+ models are trained with $K = 10$. For models that had a β parameter we tried three values $\beta \in [0.5, 1, 2.5]$. $\beta = 0.5$ was the best parameter for all models on the complete dataset and so we used that parameter on the incomplete dataset as well. The code used for the experiments is available at `https://github.com/AgatheSenellart/nips_experiments` and all training parameters can be retrieved on the Wandb workspaces:

- For the complete dataset: `https://wandb.ai/multimodal_vaes/MHD`
- For the incomplete one: `https://wandb.ai/multimodal_vaes/incomplete_MHD`

E.4 Experimental results

E.5 Metrics

E.5.1 Coherences

In Figure. 20, we present the mean coherences results for all trained models. The coherences are averaged over all classes and modalities. In Figure. 21 we present detailed coherences results for each label for models trained on the incomplete dataset. As expected, higher labels have lower coherences as they were more often missing in this dataset.

From the results in Figure. 20 it seems that the MMVAE and MMVAE+ model with (K=10) are less robust in this setting than the MVTCAE and MoPoE model.

E.5.2 Reconstruction metrics

Reconstruction metrics are presented for the models trained on the complete dataset in Figure. 22 and the incomplete one in Figure. 23. On both the complete and incomplete dataset, it seems that the MoPoE, MVTCAE and MVAE model reach the best reconstructions metrics.

E.5.3 Quantitative evaluation of the diversity of generations

Finally we present metrics to quantify the diversity of the cross-modality generation. In the case-study presented in 4.2 we used the Fréchet Inception Distance to quantify this diversity. However in the

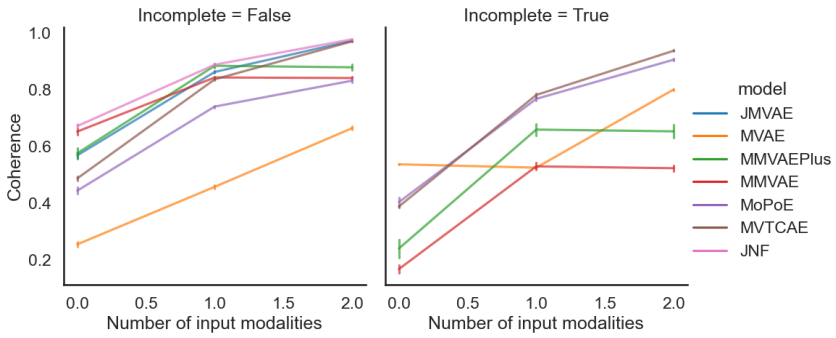

Figure 20: Coherence results for models trained on the complete dataset (on the left) and the incomplete dataset (on the right). The x-axis indicates the number of input modalities for computing the coherences. For zero input modalities, it is the joint coherence value when sampling from the prior that is plotted. Each value is averaged on four different seeds.

.

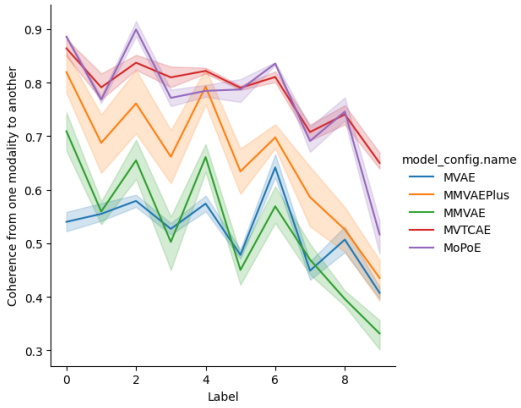

Figure 21: Detailed coherences results per classes, on the incomplete dataset, when generating from one modality to another. The results are averaged over all possible pairs of modalities. Standard errors on the four different seeds are shown. As expected the coherence is decreased for higher labels that were more often missing.

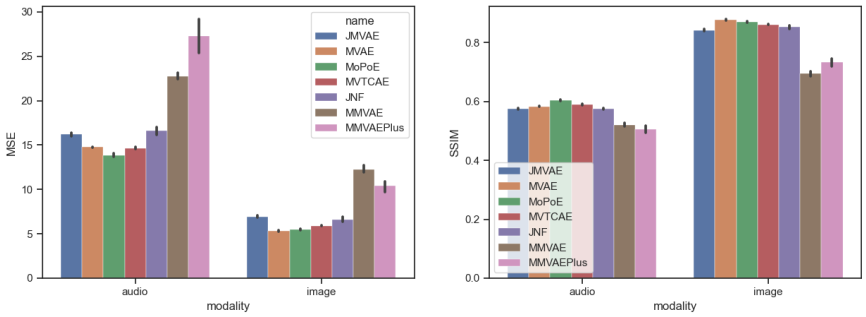

Figure 22: Reconstruction metrics for the models trained on the complete MHD dataset. On the left, we present the MSE (lower is better) and on the right the SSIM (higher is better). The models are averaged over four seeds. For both the audio and image modality, we sample a latent code from the considered modality and reconstruct it.

case of the MHD dataset, the Inception network is not appropriate to quantify the diversity of the different modalities. Indeed, the Inception network is trained on natural images and the MHD dataset

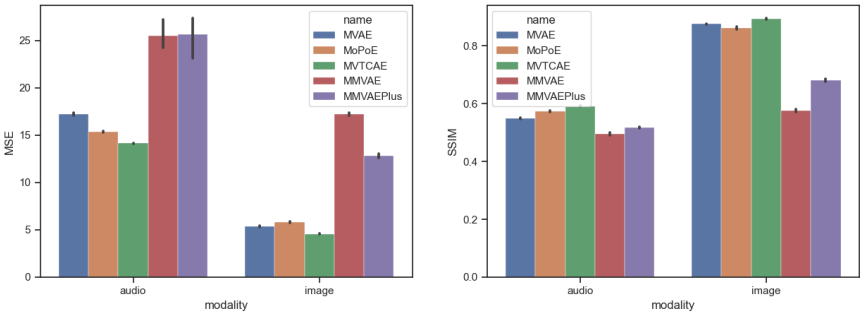

Figure 23: Reconstruction metrics for the models trained on the incomplete MHD dataset. On the left, we present the MSE (lower is better) and on the right the SSIM (higher is better). The models are averaged over four seeds. For both the audio and image modality, we sample a latent code from the considered modality and reconstruct it.

contains a black and white image modality along with audio and trajectory modalities. Therefore, we used the methodology used in [49] to quantify the diversity of the generations. For each class of each of modality, we trained a dedicated unimodal variational encoder. Then we used these encoders to compute latent representations of each original sample and cross-modal generated sample and compute a Mean Fréchet Distance between the true and generated distributions in the encoding space. More specifically, for every pair of modalities, and for each class of samples in the test set:

- We encoded the first modality with the dedicated pretrained encoder.
- We used the second modality to conditionally generate samples in the first modality.
- We encoded the generated samples with the same pretrained encoder.
- We computed the Fréchet Distance to compare the distribution of encodings of the true and generated samples.

In Figure. 24, we present Mean Fréchet Distance results for each model trained on the complete and incomplete dataset.

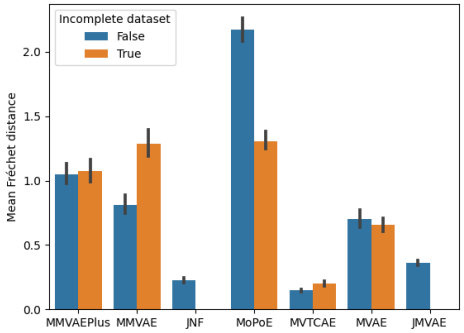

Figure 24: Mean Fréchet distance (lower is better) results for each model in the complete and incomplete setting. We average the Fréchet distance accross modalities and labels. The presented results are averaged on 4 seeds.

Interestingly we notice that for the MoPoE and MVAE model, the Mean Fréchet Distance are smaller when the models are trained on the incomplete dataset instead of complete one. These models seem to benefit from the gaps in the data which is already something that is known for the MVAE model that even uses a subsampling training paradigm to benefit from that effect [54]. As in the case study presented in 4.2, we see that joint models (JNF and JMVAE) as well as the MVTCAE reach the best performances in terms of the diversity of generated samples.

All codes used to compute these metrics are provided at https://github.com/AgatheSenellart/nips_experiments.

E.6 Visualization of generated samples and reconstructions

We present some examples of conditionally generated samples and reconstruction for each model trained on the complete dataset (in Figure. 25 and Figure. 27) and on the incomplete dataset (Figure. 26 and figure. 28).

E.7 Retrieving models on the HuggingFace Hub

All models trained in this project are made available on the HuggingFace Hub and can be downloaded. For instance see: https://huggingface.co/asenella/incomplete_mhd_MMVAE_beta_5_scale_True_seed_1.

Each run has a specific id corresponding to its experiment's configuration. To see how to retrieve them all, please look at the python file 'compute_mfd.py' in the github repository of the project: https://github.com/AgatheSenellart/nips_experiments in either the folder 'Comparison_on_MHD' or the folder 'Comparison_on_incomplete_MHD'.

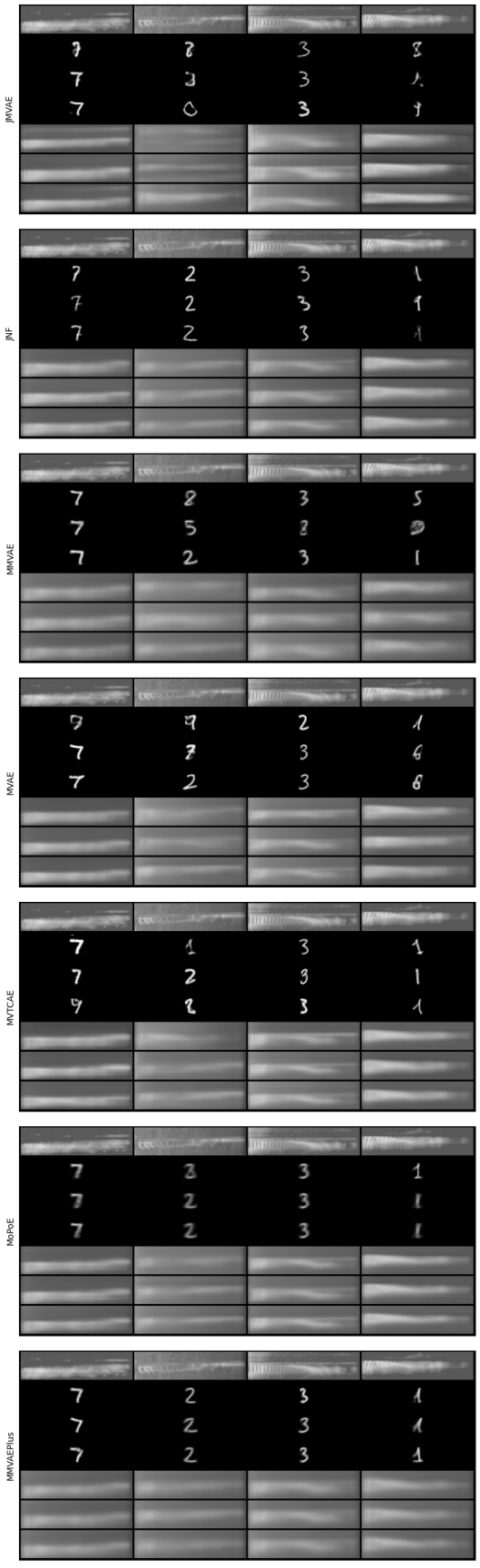

Figure 25: For each model, we sample a latent code from the audio modality to conditionally generate images and reconstruct the audio spectrogram.

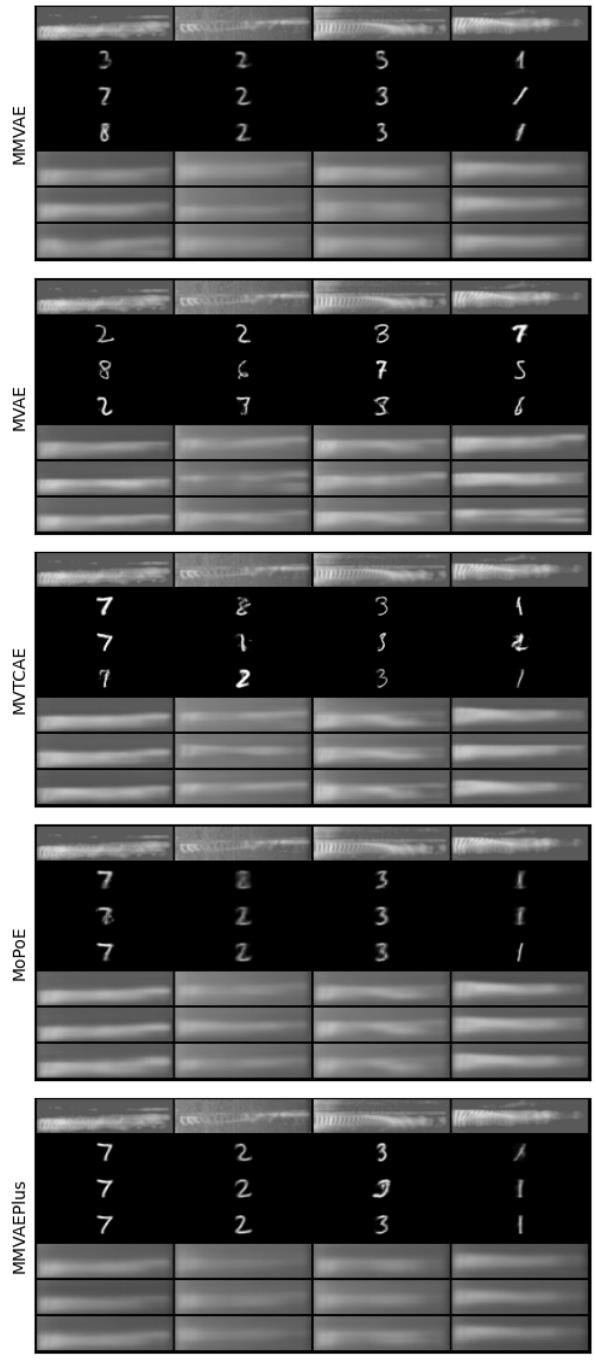

Figure 26: For each model trained on the incomplete dataset, we sample a latent code from the audio modality to conditionally generate images and reconstruct the audio spectrogram.

Figure 27: For each model trained on the complete dataset, we sample a latent code from the image modality and reconstruct the image from that latent code.

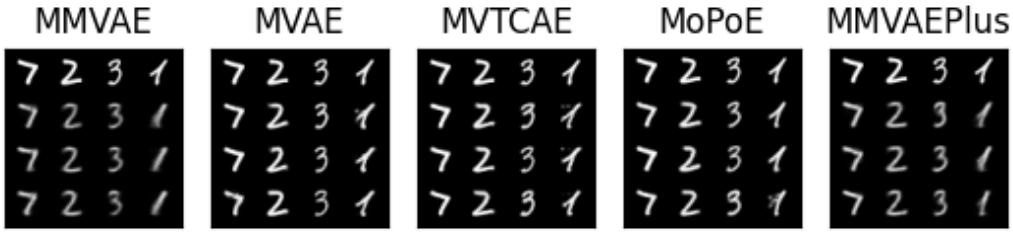

Figure 28: For each model trained on the incomplete dataset, we sample a latent code from the image modality and reconstruct the image from that latent code.

F How to use MultiVae ? Basic examples.

In Table. 4 we provide basic indications to use each model in MultiVae. You can find additional details in the MultiVae documentation.

Model	Suited MultiVae trainer	Architectural Components
JMVAE	BaseTrainer	Joint encoder
TELBO	TwoStepsTrainer	Joint encoder
MVAE	BaseTrainer	/
MMVAE	BaseTrainer	/
MVTCAE	BaseTrainer	/
MoPoE	BaseTrainer	/
JNF	TwoStepsTrainer	Joint encoder, flows
JNFDcca	AddDccaTrainer	Joint encoder, Dcca networks, flows
MMVAE+	BaseTrainer	Each encoder must output the shared and private latent variable

Table 4: For each model, we present the MultiVae trainer that it must be used with to respect original guidelines from the authors. The second column mentions the architectural components used in each model in addition to the encoders, decoders for each modality. For each architectural component a default version is provided but can be customized by the user. See the example codes in Figures **??**

In Figure 29, we present a simple snippet of code on how to create a model (in the example a JMVAE model) and train it. This examples also illustrate how to integrate *Wandb monitoring* and *HuggingFace model sharing* in just a few lines.

For a model using a more complex training (for instance multistage), you simply have to adapt the trainer according to 4. Figure 30 provides another code example using a two-steps trainer.

Finally, Figure 31 provides a last example of code that shows how to evaluate the models, with quick loading from HuggingFace Hub and metric computing with the MultiVae metrics modules.

```python
from multivae.models import JMVAE, JMVAEConfig
from multivae.trainers import BaseTrainer, BaseTrainerConfig
from multivae.trainers.base.callbacks import WandbCallback

train_dataset = ... # import your multimodal dataset class

# Set the model configuration
model_config = JMVAEConfig(
    n_modalities = 2, # set the number of modalities corresponding
                      # to your dataset
    latent_dim = 10,
    input_dims = {'modality 1' : (1,28,28), 'modality 2': (3,32,32)},
    alpha = 0.1 )

# Create the model
model = JMVAE(
    model_config=model_config,
    encoders = None, # Default MLPs will be used for each modality
    decoders=None )   # Otherwise, you can  provide your architectures.

# Set the training configuration
trainer_config = BaseTrainerConfig(
    output_dir = ..., # choose your output_directorys
    num_epochs=100,
    optimizer_cls='Adam') # There are many other parameters that you can set
                          # See the documentation for details

# Create a Wandb Callbacks to monitor
wandb_cb = WandbCallback()
wandb_cb.setup(trainer_config, model_config, project_name='your_wandb_project')

# Create the trainer
trainer = BaseTrainer(
    model=model,
    train_dataset=train_dataset,
    training_config=trainer_config,
    callbacks=[wandb_cb])

# Train
trainer.train()

# Save to Hugging Face Hub
trainer._best_model.push_to_hf_hub('your_user_name/your_repo_name')
```

Figure 29: Simple example code on how to create a model and train it in MultiVae.

```python
from multivae.models import JNF, JNFConfig
from multivae.trainers import TwoStepsTrainer, TwoStepsTrainerConfig

train_dataset = ... # import your multimodal dataset class

# Set the model configuration
model_config = JNFConfig(
    n_modalities = 2, # set the number of modalities corresponding
                      # to your dataset
    latent_dim = 10,
    input_dims = {'modality 1' : (1,28,28), 'modality 2': (3,32,32)},
    warmup=200 # number of epochs for the first step of training
)

# Create the model
model = JNF(
    model_config=model_config,
    encoders = None, # Default MLPs will be used for each modality
                     # Otherwise, provide your architectures.
    decoders = None, # Default MLPs will be used for each modality
    flows = None ) #  Default MAF flows will be used. You can also provide
                   # custom flows.


# Set the training configuration
trainer_config = TwoStepsTrainerConfig(
    output_dir = ..., # choose your output_directorys
    per_device_train_batch_size= 64,
    num_epochs=100,
    optimizer_cls='Adam') # There are many other parameters that you can set
                          # See the documentation for details

# Create the trainer
trainer = TwoStepsTrainer(
    model=model,
    train_dataset=train_dataset,
    training_config=trainer_config
    )
# Train
trainer.train()

# Save to HuggingFace Hub
trainer._best_model.push_to_hf_hub('your_user_name/your_repo_name')
```

Figure 30: Simple example code that shows how to train the JNF model using the dedicated TwoStepsTrainer class.

```python
from multivae.models import AutoModel
from multivae.metrics import LikelihoodsEvaluator, LikelihoodsEvaluatorConfig
from multivae.metrics import Visualization, VisualizationConfig

# Load model from HuggingFace Hub (or from a folder with `load_from_folder`)
model = AutoModel.load_from_hf_hub('your_user_name/your_repo_name')


# Compute joint likelihood
likelihood_config = LikelihoodsEvaluatorConfig(
    batch_size=128,
    wandb_path='your_wandb_run_path', # (you can use the same as for training
                                      # or create a new wandb path)
    num_samples=1000
)

likelihood = LikelihoodsEvaluator(
    model=model,
    test_dataset=..., # provide your dataset
    eval_config=likelihood_config
)

likelihood.eval()

# Visualize some conditionally generated/ jointly generated samples
vis_config = VisualizationConfig(
    wandb_path='your_wandb_path',
    n_samples=4
)
vis_module = Visualization(
    model=model,
    test_dataset=... # your test dataset for conditional generation
    output='your_path', # where to save the images
    sampler=..., # eventually provide a sampler for joint generation
)

vis_module.eval() # computes unconditional samples.

# you can also use more specific functions for conditional generation
# on a subset
vis_module.conditional_samples_subset(subset=['modality_1', 'modality_2'])

```

Figure 31: Simple example code that shows how to reload a model with AutoModel and compute metrics.