Supplementary Materials

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

Louis J. Vincent
Implicity †
Université Paris Cité, INRIA, Inserm, SU
Centre de Recherche des Cordeliers *
louis.vincent@inria.fr

Stéphanie Allassonnière
Université Paris Cité, INRIA, Inserm, SU
Centre de Recherche des Cordeliers *
stephanie.allassonniere@inria.fr

A Usage of Pythae

In this section we illustrate through simple examples how to use **Pythae** pipelines. The library is documented[3] and also available on pypi[4] allowing a wider use and easier integration in other codes. All of the implementations proposed in the library are adaptations of the official code when available and allowed by the licence. If not, the method is re-implemented. Table. 1 lists all the implemented models as of June 2022.

1. **Training configuration** Before launching a model training, one must specify the training configuration that should be used. This can be done easily by instantiating a **BaseTrainerConfig** instance taking as input all the hyper-parameters related to the training (number of training epochs, learning rate to apply...). See the full documentation for additional arguments that can be passed to the **BaseTrainerConfig**.

```
from pythae.trainers import BaseTrainerConfig
# Set up the model configuration
my_training_config = BaseTrainerConfig(
    output_dir='my_model',
    num_epochs=50,
    learning_rate=1e-3,
    batch_size=200)
```

*15 Rue de l'École de Médecine, 75006 Paris
†https://www.implicity.com - Implicity Paris, France.
[3]https://pythae.readthedocs.io/en/latest/?badge=latest
[4]https://pypi.org/project/pythae/

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks.

2. **Model configuration** Similarly to the TrainerConfig, the model can then be instantiated with the model configuration specifying any hyper-parameters relevant to the model. Note that each model has its own configuration with specific hyper-parameters. See the online documentation for more details.

```
from pythae.models import BetaVAE, BetaVAEConfig
# Set up the model configuration
my_vae_config = BetaVAEConfig(
    input_dim=(1, 28, 28),
    latent_dim=16,
    beta=2)
# Build the model
my_vae_model = BetaVAE(model_config=my_vae_config)
```

3. **Training** A model training can then be launched by simply using the built-in training pipeline in which only the training/evaluation data need to be specified.

```
from pythae.pipelines import TrainingPipeline
pipeline = TrainingPipeline(
    training_config=my_training_config,
    model=my_vae_model)
# Launch the Pipeline
pipeline(
    train_data=your_train_data, # arrays or tensors
    eval_data=your_eval_data) # arrays or tensors

```

4. **Model reloading** The weights and configuration of the trained model can be reloaded using the **AutoModel** instance proposed in Pythae.

```
from pythae.models import AutoModel
my_trained_vae = AutoModel.load_from_folder('path/to/trained_model')
```

5. **Data generation** A data generation pipeline can be instantiated similarly to a model training. The pipeline can then be called with any relevant arguments such as the number of samples to generate or the training and evaluation data that may be needed to fit the sampler.

```
from pythae.samplers import GaussianMixtureSamplerConfig
from pythae.pipelines import GenerationPipeline
# Define your sampler configuration
gmm_sampler_config = GaussianMixtureSamplerConfig(
    n_components=10)
# Build the pipeline
pipeline = GenerationPipeline(
    model=my_trained_vae,
    sampler_config=gmm_sampler_config)
# Launch generation
generated_samples = pipeline(
    num_samples=100,
    return_gen=True,
    train_data=train_data,
    eval_data=None)
```

Table 1: List of implemented VAEs

Name	Reference
Variational Autoencoder (VAE)	Kingma and Welling [13]
Beta Variational Autoencoder (BetaVAE)	Higgins et al. [10]
VAE with Linear Normalizing Flows (VAE_LinNF)	Rezende and Mohamed [20]
VAE with Inverse Autoregressive Flows (VAE_IAF)	Kingma et al. [14]
Disentangled β-VAE (DisentangledBetaVAE)	Higgins et al. [10]
Disentangling by Factorising (FactorVAE)	Kim and Mnih [11]
Beta-TC-VAE (BetaTCVAE)	Chen et al. [4]
Importance Weighted Autoencoder (IWAE)	Burda et al. [1]
VAE with perceptual metric similarity (MSSSIM_VAE)	Snell et al. [21]
Wasserstein Autoencoder (WAE)	Tolstikhin et al. [22]
Info Variational Autoencoder (INFOVAE_MMD)	Zhao et al. [27]
VAMP Autoencoder (VAMP)	Tomczak and Welling [23]
Hyperspherical VAE (SVAE)	Davidson et al. [6]
Adversarial Autoencoder (Adversarial_AE)	Makhzani [19]
Variational Autoencoder GAN (VAEGAN)	Larsen et al. [16]
Vector Quantized VAE (VQVAE)	Van Den Oord et al. [24]
Hamiltonian VAE (HVAE)	Caterini et al. [2]
Regularized AE with L2 decoder param (RAE_L2)	Ghosh et al. [8]
Regularized AE with gradient penalty (RAE_GP)	Ghosh et al. [8]
Riemannian Hamiltonian VAE (RHVAE)	Chadebec et al. [3]

Maintenance plan: We intend for this library to be maintained in the long term. In that view, the main author's contact details will remain available and up-to-date on the github repository, which will remain the main discussion channel. Additionally, we are currently considering adding back-up contributors that will also support this effort in the long-term. Since this library has already started to be a community effort with external contributors, we further hope that the community will also continue to help reviewing and updating the current implementations.

Original papers reproducibility We validate the implementations by reproducing some results presented in the original publications when the official code has been released or when enough details about the experimental section of the papers were available (we indeed noted that in many papers key elements for reproducibility were missing such as the data split considered, which criteria is used to select the model on which the metrics are computed, the hyper-parameters are not fully disclosed or the network architectures is unclear making reproduction very hard if not impossible in certain cases). This insists on the fact that the framework is flexible enough to reproduce results from publications. Finally, we have open-sourced the scripts, configurations and results on the repository at `https://github.com/clementchadebec/benchmark_VAE/tree/main/examples/scripts/reproducibility` and made the trained models available on the HuggingFace Hub (e.g. `https://huggingface.co/clementchadebec/reproduced_iwae`).

B Interpolations

In this section, we show the interpolations obtained on the three considered datasets. For each model, we select both a starting image and an ending image from the test set and perform a linear interpolation between the corresponding embeddings in the learned latent space. We then show the decoded trajectory all along the interpolation line. For this task, we use the model configuration that obtained the lowest FID on the validation set with a GMM sampler from the generation task. We show the resulting interpolations for latent spaces of dimension 16 and 256 for MNIST, 32 and 256 for CIFAR10 and 64 for CELEBA. As mentioned in the paper, for this complex task, variational approaches tend to outperform the AE-based methods. This is well illustrated on MNIST with a latent space of dimension 256 since all the AE-based approaches eventually superpose the starting and ending image, making the interpolation visually irrelevant. Impressively, the regularisation imposed by the variational approaches prevents such undesirable behaviours from occurring. This adds to the observation made in Sec. 4.2.2 of the paper where we note some robustness to the latent dimension for the variational methods. Nonetheless, as stated in the paper this regularisation can also degrade image reconstruction, leading to very blurry interpolations, as illustrated on Fig. 3.

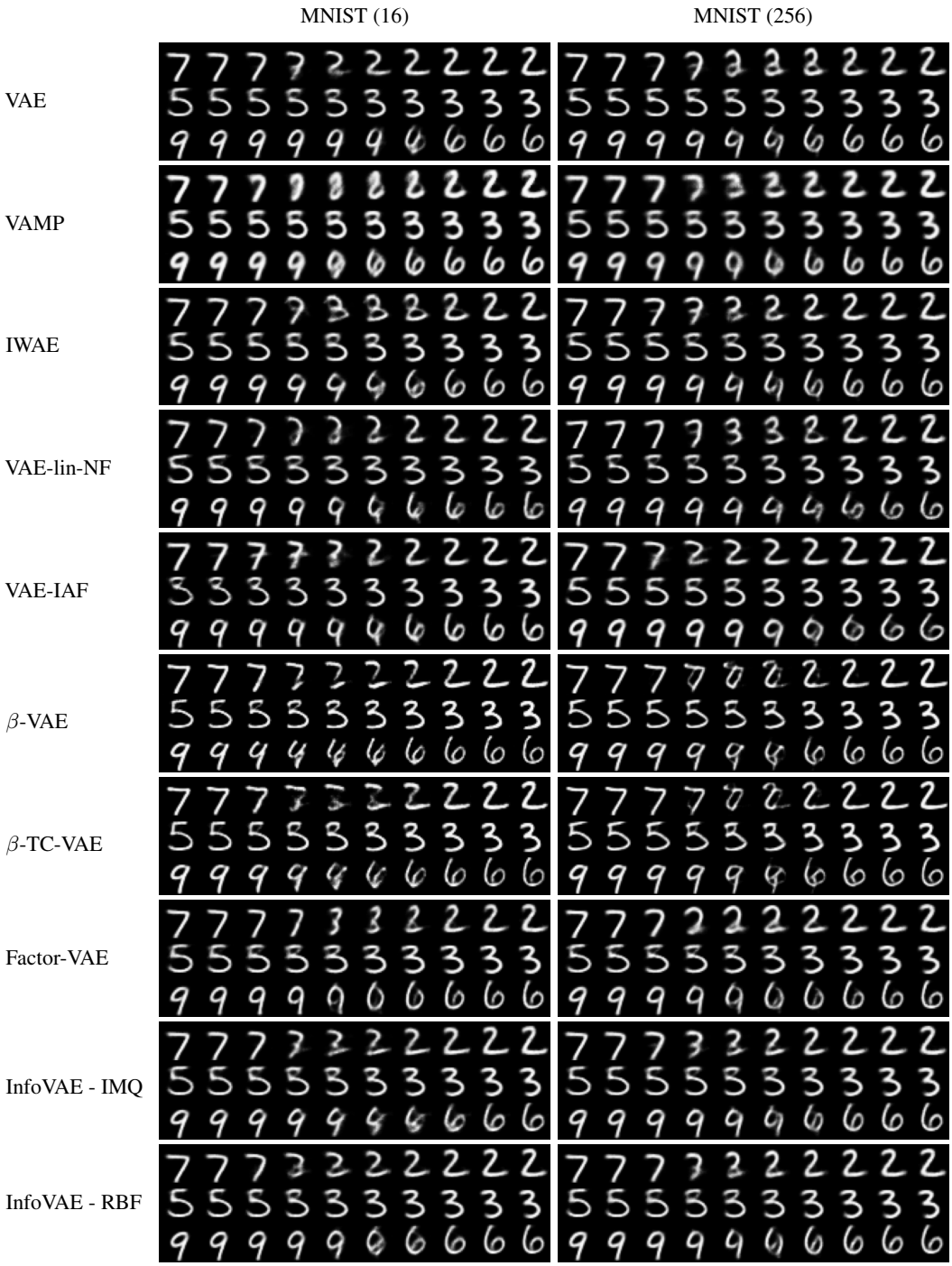

Figure 1: Interpolations on MNIST with the same starting and ending images for latent spaces of dimension 16 and 256. For each model we select the configuration achieving the lowest FID on the generation task on the validation set with a GMM sampler.

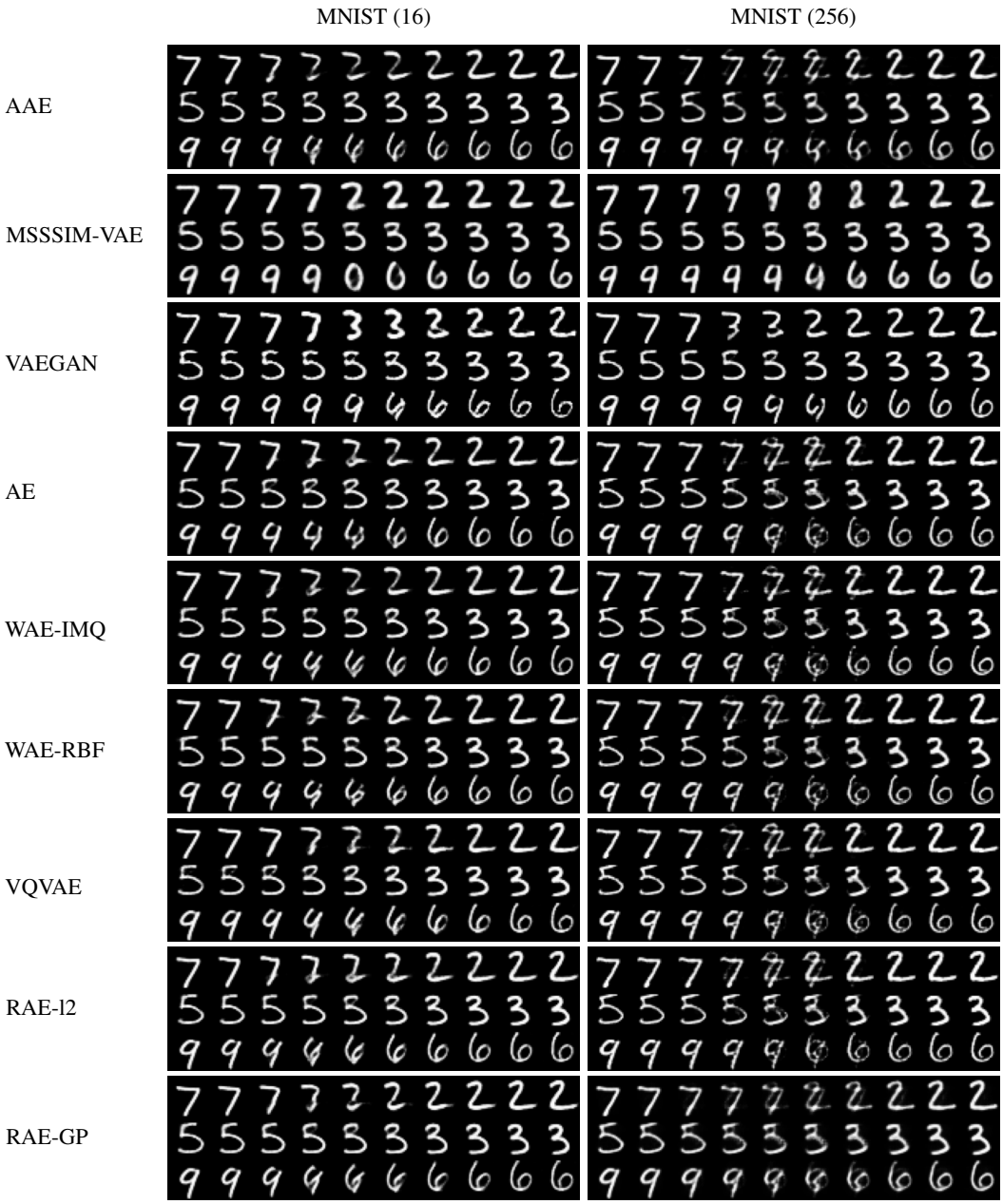

Figure 2: Interpolations on MNIST with the same starting and ending images for latent spaces of dimension 16 and 256. For each model we select the configuration achieving the lowest FID on the generation task on the validation set with a GMM sampler.

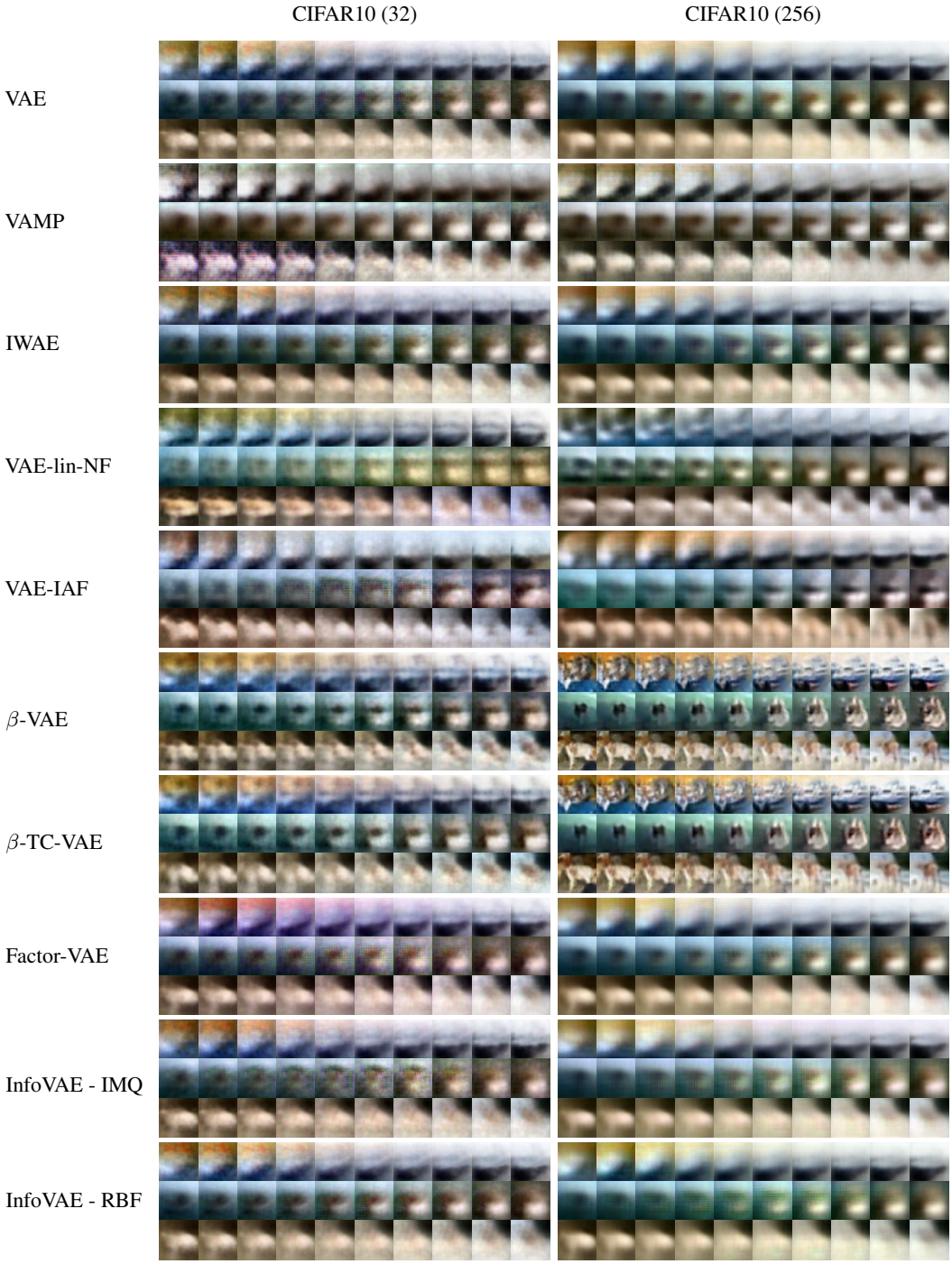

Figure 3: Interpolations on CIFAR10 with the same starting and ending images for latent spaces of dimension 32 and 256. For each model we select the configuration achieving the lowest FID on the generation task on the validation set with a GMM sampler.

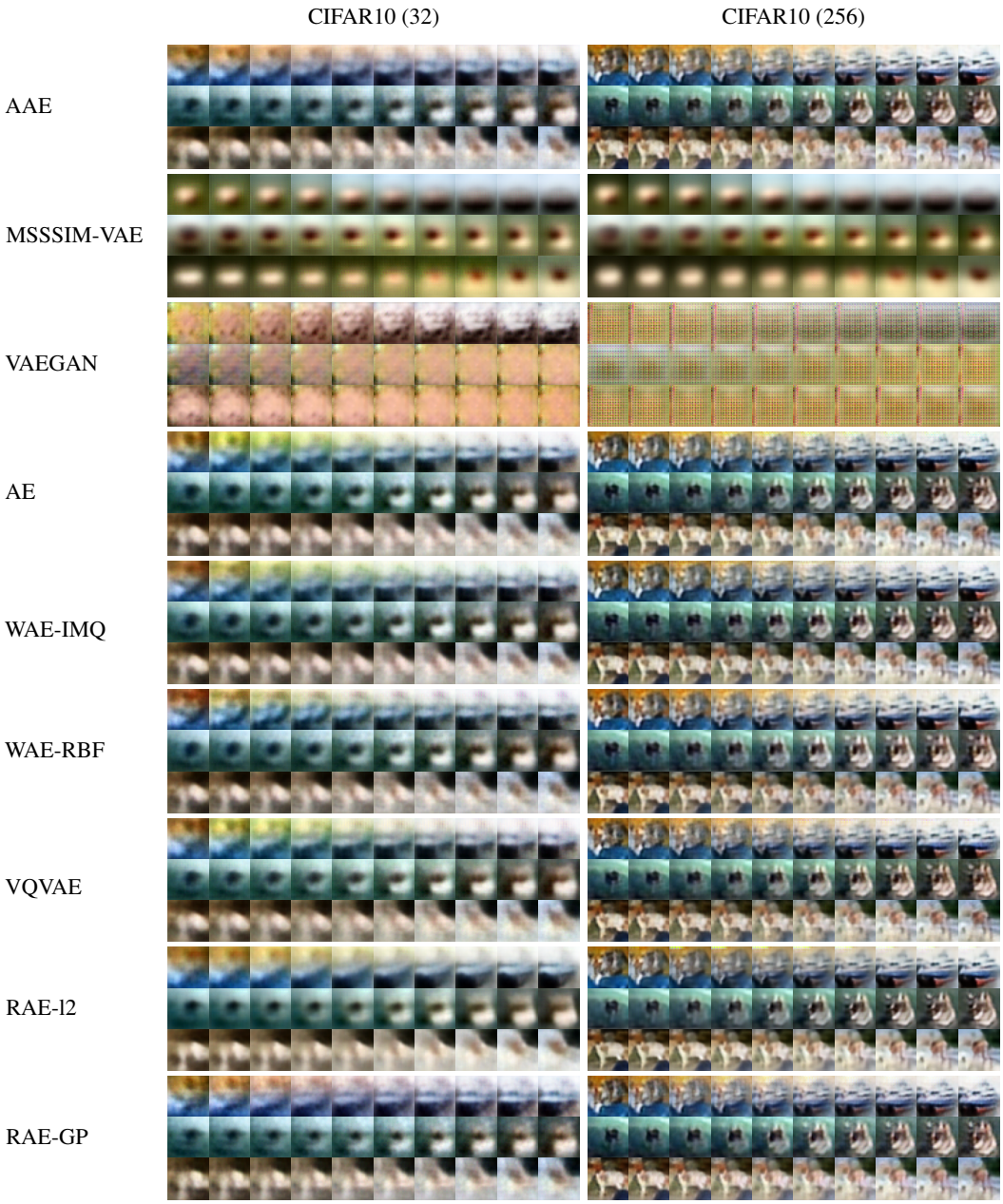

Figure 4: Interpolations on CIFAR10 with the same starting and ending images for latent spaces of dimension 32 and 256. For each model we select the configuration achieving the lowest FID on the generation task on the validation set with a GMM sampler.

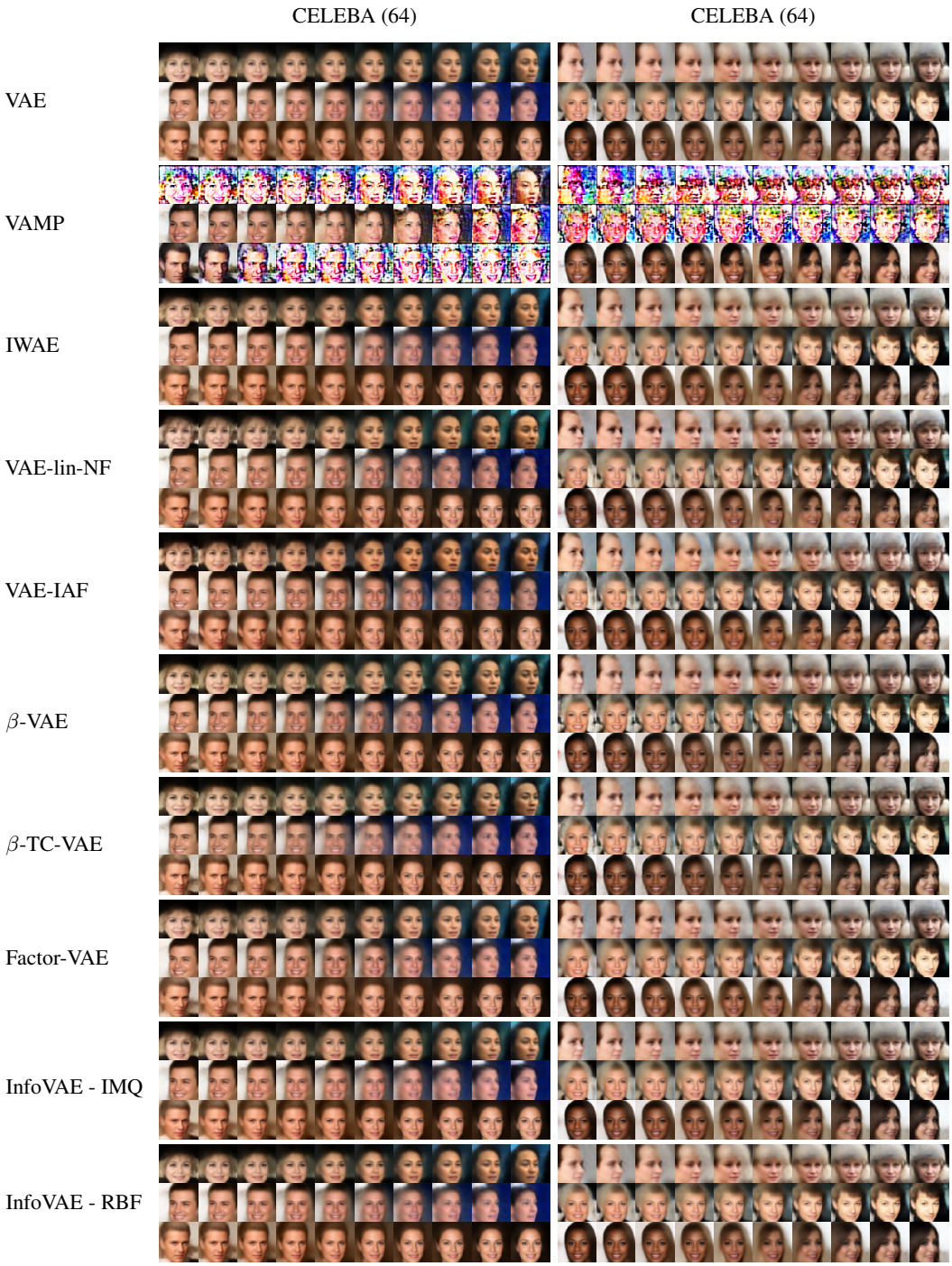

Figure 5: Interpolations on CELEBA with the same starting and ending images for a latent space of dimension 64. For each model we select the configuration achieving the lowest FID on the generation task on the validation set with a GMM sampler.

Figure 6: Interpolations on CELEBA with the same starting and ending images for a latent space of dimension 64. For each model we select the configuration achieving the lowest FID on the generation task on the validation set with a GMM sampler.

C Detailed experiments set-up

We detail here the main experimental set-up and implementation choices made in the benchmark. We let the reader refer to the code available online for specific implementation aspects

The data To perform the benchmaks presented in the paper, we select 3 classical *free-to-use* image datasets: MNIST [17], CIFAR10 [15] and CELEBA [18]. These datasets are publicly available, widely used for generative model related papers and have well known associated metrics in the literature. Each dataset is split into a train set, a validation set and a test set. For MNIST and CIFAR10 the validation set is composed of the last 10k images extracted from the official train set and the test set corresponds to the official one. For CELEBA, we use the official train/val/test split.

Training paradigm We equip each model used in the benchmark with the same neural network architecture for both the encoder and decoder, taken as a ConvNet and ResNet (architectures given in Tables. 2 and 3) leading to a comparable number of parameters [5]. For the 19 considered models, due to computational limitations, 10 different configurations are considered, allowing a simple exploration of the models' hyper-parameters. The sets of hyper-parameters explored are detailed in Appendix. D for each model. The models are then trained on MNIST and CIFAR10 for 100 epochs, a starting learning rate of $1e^{-4}$ and batch size of 100 with Adam optimizer [12]. A scheduler reducing the learning rate by half if the validation loss does not improve for 10 epochs is also used. For CELEBA, we use the same setting but we train the models for 50 epochs with a starting learning rate of $1e^{-3}$. Models with unstable training (NaN, huge training spikes...) are iteratively retrained with a starting learning rate divided by 10 until training stabilises. All 19 models are trained on a single 32GB V100 GPU. This leads to 10 trained models for each method, each dataset (MNIST, CIFAR10 or CELEBA) and each neural network (ConvNet or ResNet) leading to a total of 1140 models. The training setting (curves, configs ...) can be found at `https://wandb.ai/benchmark_team/trainings`.

Sampling paradigm for the MAF and VAE samplers For the Masked Autoregressive Flow sampler used for sampling we use a 3-layer MADE [7] with 128 hidden units and ReLU activation for each layer and stack 2 blocks of MAF to create the flow. For the masked layers, the mask is made sequentially and the ordering is reversed between each MADE. For this normalising flow we consider a starting distribution given by a standard Gaussian. For the auxiliary VAE sampling method proposed in [5], we consider a simple VAE with a Multi Layer Perceptron (MLP) encoder and decoder, with 2 hidden layers composed of 1024 units and ReLU activation. Both samplers are fitted with 200 epochs using the train and evaluation embeddings coming from the trained autoencoder models. A learning rate of $1e^{-4}$, a scheduler decreasing the learning rate by half if the validation loss does not improve for 10 epochs and a batch size of 100 are used for these samplers.

Table 2: Neural network architecture used for the convolutional networks.

	MNIST	CIFAR10	CELEBA
Encoder	(1, 28, 28)	(3, 32, 32)	(3, 64, 64)
Layer 1	Conv(128, 4, 2), BN, ReLU	Conv(128, 4, 2), BN, ReLU	Conv(128, 4, 2), BN, ReLU
Layer 2	Conv(256, 4, 2), BN, ReLU	Conv(256, 4, 2), BN, ReLU	Conv(256, 4, 2), BN, ReLU
Layer 3	Conv(512, 4, 2), BN, ReLU	Conv(512, 4, 2), BN, ReLU	Conv(512, 4, 2), BN, ReLU
Layer 4	Conv(1024, 4, 2), BN, ReLU	Conv(1024, 4, 2), BN, ReLU	Conv(1024, 4, 2), BN, ReLU
Layer 5	Linear(1024, latent_dim)*	Linear(4096, latent_dim)*	Linear(16384, latent_dim)*
Decoder			
Layer 1	Linear(latent_dim, 16384)	Linear(latent_dim, 65536)	Linear(latent_dim, 65536)
Layer 2	ConvT(512, 3, 2), BN, ReLU	ConvT(512, 4, 2), BN, ReLU	ConvT(512, 5, 2), BN, ReLU
Layer 3	ConvT(256, 3, 2), BN, ReLU	ConvT(256, 4, 2), BN, ReLU	ConvT(256, 5, 2), BN, ReLU
Layer 4	Conv(1, 3, 2), Sigmoid	Conv(3, 4, 1), Sigmoid	ConvT(128, 5, 2), BN, ReLU
Layer 5	-	-	ConvT(3, 5, 1), Sigmoid

*Doubled for VAE-based models

[5] Some models may actually have additional parameters in their intrinsic structure *e.g.* a VQVAE learns a dictionary of embeddings, a VAMP learns the pseudo-inputs, a VAE-IAF learns the auto-regressive flows. Nonetheless, since we work on images, the number of parameters remains in the same order of magnitude.

Table 3: Neural network architecture used for the residual networks.

	MNIST	CIFAR10	CELEBA
Encoder	(1, 28, 28)	(3, 32, 32)	(3, 64, 64)
Layer 1	Conv(64, 4, 2)	Conv(64, 4, 2)	Conv(64, 4, 2)
Layer 2	Conv(128, 4, 2)	Conv(128, 4, 2)	Conv(128, 4, 2)
Layer 3	Conv(128, 3, 2)	Conv(128, 3, 1)	Conv(128, 3, 2)
Layer 4	ResBlock**	ResBlock**	Conv(128, 3, 2)
Layer 5	ResBlock**	ResBlock**	ResBlock**
Layer 6	Linear(2048, latent_dim)*	Linear(8192, latent_dim)*	ResBlock**
Layer 7	-	-	Linear(2048, latent_dim)*
Decoder			
Layer 1	Linear(latent_dim, 2048)	Linear(latent_dim, 8192)	Linear(latent_dim, 2048)
Layer 2	ConvT(128, 3, 2)	ResBlock**	ConvT(128, 3, 2)
Layer 3	ResBlock**	ResBlock**	ResBlock**
Layer 4	ResBlock**, ReLU	ConvT(64, 4, 2)	ResBlock**
Layer 5	ConvT(64, 3, 2), ReLU	ConvT(3, 4, 2), Sigmoid	ConvT(128, 5, 2), Sigmoid
Layer 6	ConvT(1, 3, 2), Sigmoid	-	ConvT(64, 5, 2), Sigmoid
Layer 6	-	-	ConvT(3, 4, 2), Sigmoid

*Doubled for VAE-based models

**The ResBlocks are composed of one Conv(32, 3, 1) followed by Conv(128, 1, 1) with ReLU.

D Additional results

D.1 Effect of the latent dimension on the 4 tasks with the CIFAR10 database

Analogously to the results shown in the paper on the MNIST dataset for the 4 chosen tasks (reconstruction, generation, classification and clustering), Fig. 7 shows the impact the choice of the latent space dimension has on the performances of the models on the CIFAR10 dataset, whose image arguably have a greater intrinsic latent dimension than images of the MNIST dataset. Similarly to MNIST, two distinct groups appear: the AE-based methods and variational methods. Again, for all tasks but clustering, variational based methods demonstrate good robustness properties with respect to the dimension of the latent space when compared to AE approaches.

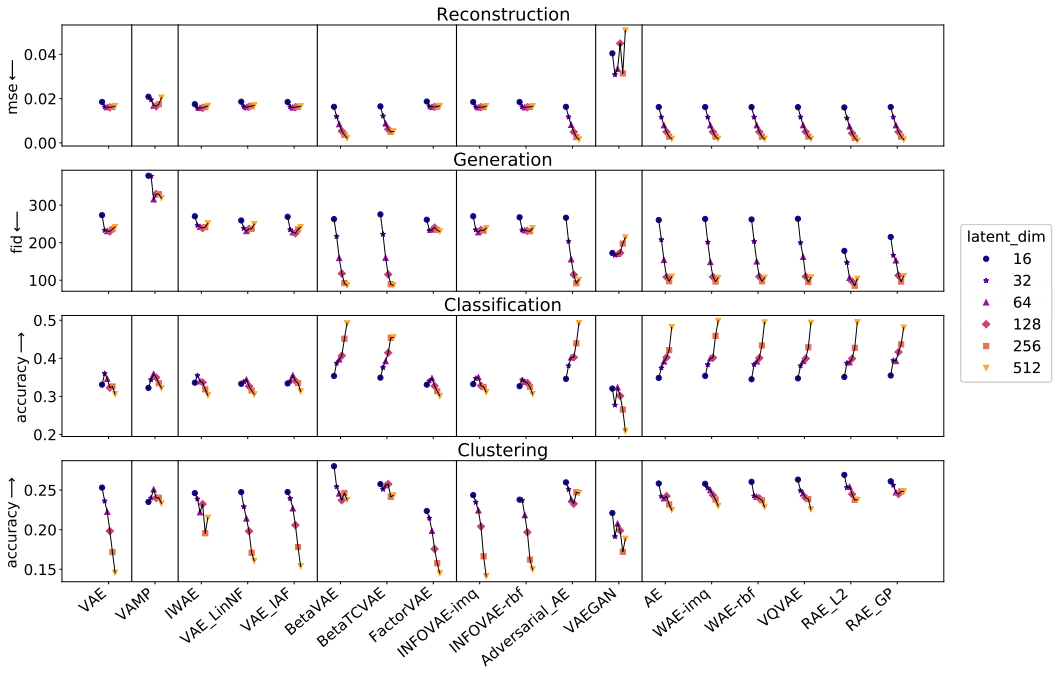

Figure 7: *From top to bottom:* Evolution of the reconstruction MSE, generation FID, classification accuracy and clustering accuracy with respect to the latent space dimension on the CIFAR dataset.

D.2 Complete generation table

In Table. 4 are presented the full results obtained for generation *i.e.* including the MAF and 2-stage VAE sampler [5]. As mentioned in the paper, it is interesting to note that fitting a GMM instead of using the prior for the variational-based approaches seems to often allow a better image generation since it allows a better prospecting of the learned latent space of each model. Interestingly, it seems that fitting more complex density estimators such as a normalising flow (MAF sampler) or another VAE (2-stage sampler) does not improve the generation results when compared to the GMM for those datasets.

Table 4: Inception Score (higher is better) and FID (lower is better) computed with 10k samples on the test set. For each model and sampler we report the results obtained by the model achieving the lowest FID score on the validation set.

Model	Sampler	ConvNet						ResNet					
		MNIST		CIFAR10		CELEBA		MNIST		CIFAR10		CELEBA	
		FID ↓	IS ↑	FID	IS	FID	IS ↑	FID ↓	IS ↑	FID	IS	FID	IS
VAE	$\mathcal{N}$	28.5	2.1	241.0	2.2	54.8	1.9	31.3	2.0	181.7	2.5	66.6	1.6
	GMM	26.9	2.1	235.9	2.3	52.4	1.9	32.3	2.1	179.7	2.5	63.0	1.7
	VAE	40.3	2.0	337.5	1.7	70.9	1.6	48.7	1.8	358.0	1.3	76.4	1.4
	MAF	26.8	2.1	239.5	2.2	52.5	2.0	31.0	2.1	181.5	2.5	62.9	1.7
VAMP	VAMP	64.2	2.0	329.0	1.5	56.0	1.9	34.5	2.1	181.9	2.5	67.2	1.6
IWAE	$\mathcal{N}$	29.0	2.1	245.3	2.1	55.7	1.9	32.4	2.0	191.2	2.4	67.6	1.6
	GMM	28.4	2.1	241.2	2.1	52.7	1.9	34.4	2.1	188.8	2.4	64.1	1.7
	VAE	42.4	2.0	346.6	1.5	74.3	1.5	50.1	1.9	364.8	1.2	76.4	1.4
	MAF	28.1	2.1	243.4	2.1	52.7	1.9	32.5	2.1	190.4	2.4	64.3	1.7
VAE-lin NF	$\mathcal{N}$	29.3	2.1	240.3	2.1	56.5	1.9	32.5	2.0	185.5	2.4	67.1	1.6
	GMM	28.4	2.1	237.0	2.2	53.3	1.9	33.1	2.1	183.1	2.5	62.8	1.7
	VAE	40.1	2.0	311.0	1.6	71.1	1.6	49.7	1.9	296.2	1.7	75.6	1.4
	MAF	27.7	2.1	239.1	2.1	53.4	2.0	32.4	2.0	184.2	2.5	62.7	1.7
VAE-IAF	$\mathcal{N}$	27.5	2.1	236.0	2.2	55.4	1.9	30.6	2.0	183.6	2.5	66.2	1.6
	GMM	27.0	2.1	235.4	2.2	53.6	1.9	32.2	2.1	180.8	2.5	62.7	1.7
	VAE	39.4	2.0	330.5	1.1	73.0	1.5	44.8	1.9	322.7	1.5	76.7	1.4
	MAF	26.9	2.1	236.8	2.2	53.6	1.9	30.6	2.1	182.5	2.5	63.0	1.7
β-VAE	$\mathcal{N}$	21.4	2.1	115.4	3.6	56.1	1.9	19.1	2.0	124.9	3.4	65.9	1.6
	GMM	9.2	2.2	92.2	3.9	51.7	1.9	11.4	2.1	112.6	3.6	59.3	1.7
	VAE	14.0	2.2	139.6	3.6	55.0	1.9	20.3	2.1	152.5	3.5	61.5	1.7
	MAF	9.5	2.2	100.9	3.5	51.5	2.0	12.0	2.1	120.0	3.6	59.7	1.8
β-TC VAE	$\mathcal{N}$	21.3	2.1	116.6	2.8	55.7	1.8	20.7	2.0	125.8	3.4	65.9	1.6
	GMM	11.6	2.2	89.3	4.1	51.8	1.9	13.3	2.1	**106.5**	**3.7**	59.3	1.7
	VAE	18.4	2.2	127.9	4.2	59.7	1.8	28.3	2.0	164.0	3.3	66.4	1.5
	MAF	12.0	2.2	95.6	3.6	52.2	1.9	13.7	2.1	116.6	3.4	60.1	1.7
FactorVAE	$\mathcal{N}$	27.0	2.1	236.5	2.2	53.8	1.9	31.0	2.0	185.4	2.5	66.4	1.7
	GMM	26.9	2.1	234.0	2.2	52.4	2.0	32.7	2.1	184.4	2.5	63.3	1.7
	VAE	41.2	1.9	338.3	1.5	75.0	1.5	54.7	1.8	316.2	1.3	77.7	1.4
	MAF	26.7	2.2	236.7	2.2	52.7	1.9	32.8	2.1	185.8	2.5	63.4	1.7
InfoVAE - RBF	$\mathcal{N}$	27.5	2.1	235.2	2.1	55.5	1.9	31.1	2.0	182.8	2.5	66.5	1.6
	GMM	26.7	2.1	230.4	2.2	52.7	1.9	32.3	2.1	179.5	2.5	62.8	1.7
	VAE	39.7	2.0	327.2	1.5	73.7	1.5	50.6	1.9	363.4	1.2	75.8	1.4
	MAF	25.9	2.1	233.5	2.2	52.2	2.0	30.5	2.1	181.3	2.5	62.7	1.7
InfoVAE - IMQ	$\mathcal{N}$	28.3	2.1	233.8	2.2	56.7	1.9	31.0	2.0	182.4	2.5	66.4	1.6
	GMM	27.7	2.1	231.9	2.2	53.7	1.9	32.8	2.1	180.7	2.6	62.3	1.7
	VAE	40.4	1.9	323.8	1.6	73.7	1.5	49.9	1.9	341.8	1.8	75.7	1.4
	MAF	27.2	2.1	232.3	2.1	53.8	2.0	30.6	2.1	182.5	2.5	62.6	1.7
AAE	$\mathcal{N}$	16.8	2.2	139.9	2.6	59.9	1.8	19.1	2.1	164.9	2.4	64.8	1.7
	GMM	9.3	2.2	92.1	3.8	53.9	2.0	11.1	2.1	118.5	3.5	58.7	1.8
	VAE	13.4	2.2	144.0	3.4	58.2	1.8	15.1	2.1	145.2	3.6	59.0	1.7
	MAF	9.3	2.2	101.1	3.2	53.8	2.0	11.9	2.1	133.6	3.1	59.2	1.8
MSSSIM-VAE	$\mathcal{N}$	26.7	2.2	279.9	1.7	124.3	1.3	28.0	2.1	254.2	1.7	119.0	1.3
	GMM	27.2	2.2	279.7	1.7	124.3	1.3	28.8	2.1	253.1	1.7	119.2	1.3
	VAE	51.2	1.9	355.5	1.1	137.9	1.2	51.6	1.9	372.1	1.1	136.5	1.2
	MAF	26.9	2.2	279.8	1.7	124.0	1.3	27.5	2.1	254.1	1.7	119.5	1.3
VAEGAN	$\mathcal{N}$	8.7	2.2	199.5	2.2	39.7	1.9	12.8	2.2	198.7	2.2	122.8	2.0
	GMM	**6.3**	**2.2**	197.5	2.1	**35.6**	1.8	**6.5**	2.2	188.2	2.6	84.3	1.7
	VAE	11.2	2.1	310.9	2.0	54.5	1.6	9.2	2.1	272.7	2.0	88.8	1.6
	MAF	6.9	2.3	199.0	2.1	36.7	1.8	6.6	**2.2**	191.9	2.5	84.8	1.7
AE	$\mathcal{N}$	26.7	2.1	201.3	2.1	327.7	1.0	221.8	1.3	210.1	2.1	275.0	2.9
	GMM	9.3	2.2	97.3	3.6	55.4	2.0	11.0	2.1	120.7	3.4	**57.4**	1.8
	MAF	9.9	2.2	108.3	3.1	55.7	2.0	12.0	2.1	136.5	3.0	58.3	1.8
WAE - RBF	$\mathcal{N}$	21.2	2.2	175.1	2.0	332.6	1.0	21.2	2.1	170.2	2.3	69.4	1.6
	GMM	9.2	2.2	97.1	3.6	55.0	2.0	11.2	2.1	120.3	3.4	58.3	1.7
	MAF	9.8	2.2	108.2	3.1	56.0	2.0	11.8	2.2	135.3	3.0	58.3	1.8
WAE - IMQ	$\mathcal{N}$	18.9	2.2	164.4	2.2	64.6	1.7	20.3	2.1	150.7	2.5	67.1	1.6
	GMM	8.6	2.2	96.5	3.6	51.7	2.0	11.2	2.1	119.0	3.5	57.7	1.8
	MAF	9.5	2.2	107.8	3.1	51.6	2.0	11.8	2.1	130.2	3.0	58.7	1.7
VQVAE	$\mathcal{N}(0,1)$	28.2	2.0	152.2	2.0	306.9	1.0	170.7	1.6	195.7	1.9	140.3	**2.2**
	GMM	9.1	2.2	95.2	3.7	51.6	2.0	10.7	2.1	120.1	3.4	57.9	1.8
	MAF	9.6	2.2	104.7	3.2	52.3	1.9	11.7	2.2	136.8	3.0	57.9	1.8
RAE-L2	$\mathcal{N}$	25.0	2.0	156.1	2.6	86.1	**2.8**	63.3	2.2	170.9	2.2	168.7	3.1
	GMM	9.1	2.2	**85.3**	**3.9**	55.2	1.9	11.5	2.1	122.5	3.4	58.3	1.8
	MAF	9.5	2.2	93.4	3.5	55.2	2.0	12.3	2.2	136.6	3.0	59.1	1.7
RAE - GP	$\mathcal{N}$	27.1	2.1	196.8	2.1	86.1	2.4	61.5	2.2	229.1	2.0	201.9	3.1
	GMM	9.7	2.2	96.3	3.7	52.5	1.9	11.4	2.1	123.3	3.4	59.0	1.8
	MAF	9.7	2.2	106.3	3.2	52.5	1.9	12.2	2.2	139.4	3.0	59.5	1.8

D.3 Further interesting results

Generated samples In addition to quantitative metrics, we also provide in Fig. 8 and Fig. 9 some samples coming from the different models using either a $\mathcal{N}(0, I_d)$ or fitting a GMM with 10 components on MNIST and CELEBA. This allows to visually differentiate the quality of the different sampling methods.

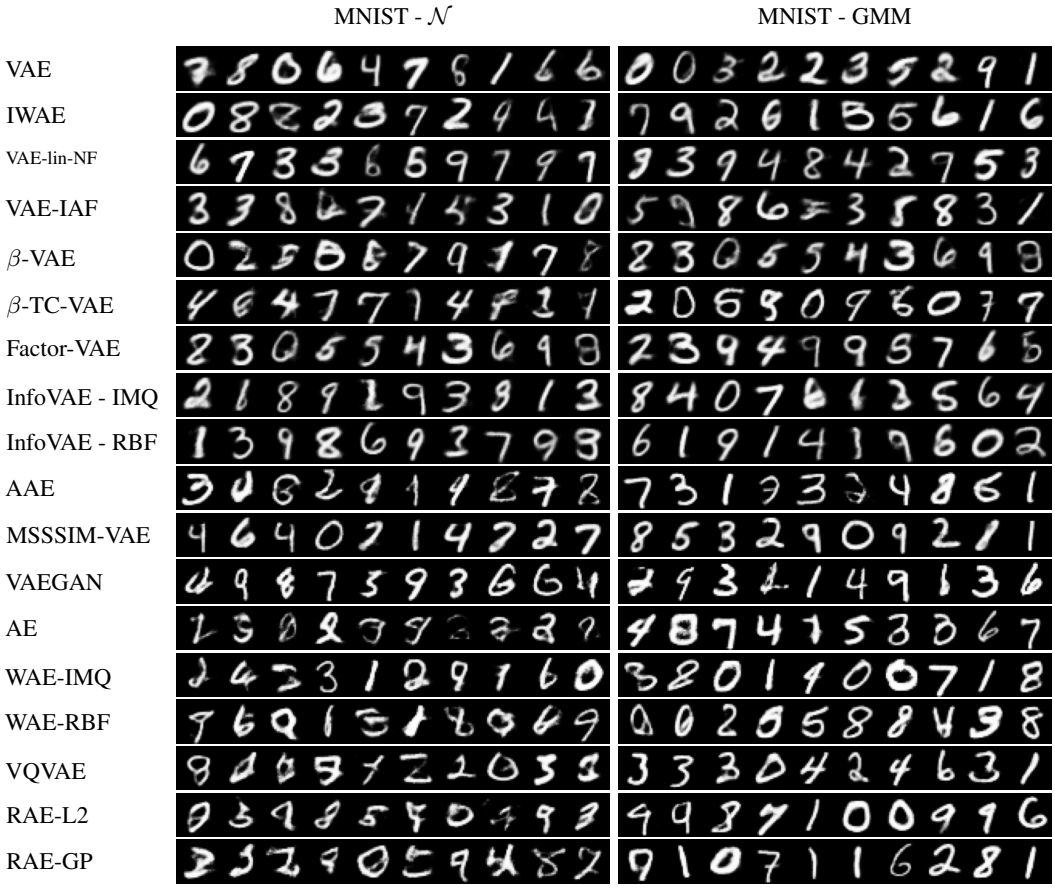

Figure 8: Generated samples on MNIST for a latent space of dimension 16 and ConvNet architecture. For each model, we select the configuration achieving the lowest FID on the validation set.

Figure 9: Generated samples on CELEBA for a latent space of dimension 64 and ConvNet architecture. For each model, we select the configuration achieving the lowest FID on the validation set.

Sampler ablation study Fig. 10 shows the same results as Table. 4 but under a different prism. In this plot, we show the influence each sampler has on the generation quality for all the models considered in this study. Note that sampling using a $\mathcal{N}(0, I_d)$ for an AE, RAE or VQVAE is far from being optimal since those models do not enforce explicitly the latent variables to follow this distribution. As mentioned in the paper, this experiment shows that using more complex density estimators such as a GMM or a normalising flow almost always improves the generation metric.

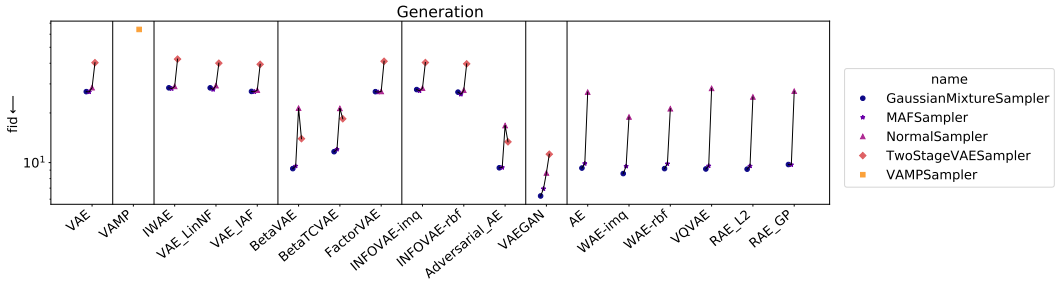

Figure 10: Evolution of the FID for the generation task depending on the sampler, for a ConvNet, the MNIST dataset and a latent dimension of 16. For each sampler and model, we select the configuration achieving the lowest FID on the validation set.

Neural network architecture ablation study As explained in the paper and in Appendix. C, we consider two different neural architectures for the encoder and decoder of each model: a ConvNet (convolutional neural network) and a ResNet (residual neural network). Fig. 11 shows the influence the choice of the neural architecture has on the ability of the model to perform the 4 tasks presented in the paper. The results are computed for each model on MNIST and a latent dimension of 16. The ConvNet architecture has approximately 20 times more parameters than the ResNet in such conditions. We select the best configuration for each model and each task on the validation set and report the results on the test set. Unsuprisingly, we see in Fig. 11 that the ConvNet architecture, more adapted to capture features intrinsic to images, leads to the best performances for reconstruction and generation. Interestingly, the ResNet outperforms the ConvNet for the classification and clustering tasks, meaning that in addition to the network complexity, its structure can play a major role in the representation learned by the models.

Training time Fig. 12 shows the training times required for each model for both network architectures on the MNIST dataset. For each model, we show the results obtained with the configuration giving the best performances on the generation task with fixed latent dimension 16. It is interesting to note that although VAEGAN outperforms other models on the generation task, it is at the price of a higher computational time. This is due to the discriminator network (a convolutional neural net) that is called several times during training and takes images as inputs. It should be noted that methods applying normalising flows to the posterior (VAE-lin-NF and VAE-IAF) maintain a reasonable training time, as the flows were chosen for their scalability.

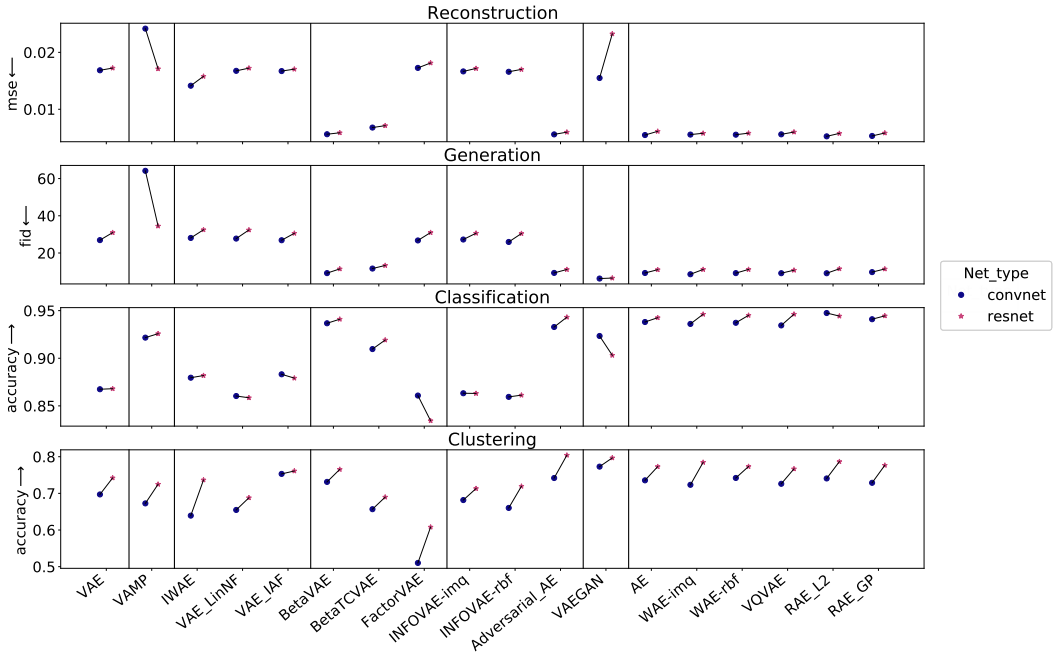

Figure 11: Evolution of the metrics for the 4 tasks depending on the network type on the MNIST dataset and a latent dimension of 16.

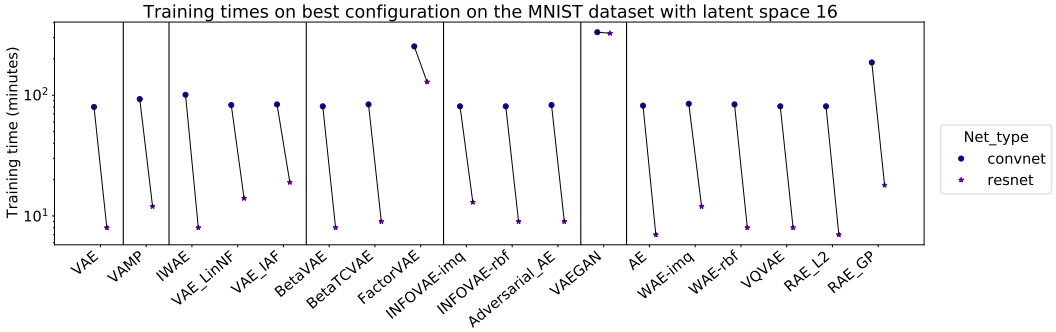

Figure 12: Total training time for the models trained on the MNIST dataset with latent dimension 16 with the best performance on the generation task.

D.4 Configurations and results by models

In this section we briefly explain each model considered in the benchmark, and show the evolution of performances on the 4 tasks and the training speed with respect to the choice of the hyper-parameters. For all 4 tasks we consider the MNIST dataset and a fixed latent space of dimension 16, as well as the Normal Gaussian sampler (if applicable) and the convolutional network architecture. For each model, 10 configuration runs with different hyper-parameters were tested. It should be noted that this configuration search was done empirically and is not exhaustive, therefore models with multiple hyper-parameters or that are sensitive to the choice of hyper-parameters will tend to have sub-optimal configuration choices. Although hyper-parameter choices are dependant on both the auto-encoder architecture and the dataset, it is interesting to note the relative evolution of the performances on the different tasks and the training time induced by different hyper-parameter choices.

Notations In order to better underline the differences between different models and for clarity purposes, we set the following unified notations:

- $X = \{x_1, \dots, x_N\} \in \mathcal{X}^N$ the input dataset
- $x \in \mathcal{X}$ an observation from the dataset, and $z \in \mathcal{Z} = \mathbb{R}^d$ its corresponding latent vector
- $\hat{x}$ the reconstruction of x by the auto-encoder model
- $p_z(z)$ the prior distribution, with $p_z \equiv \mathcal{N}(0, I_d)$ under standard VAE assumption
- $q_\phi(z|x)$ the approximate posterior distribution, modelled by the encoder. Kingma and Welling [13] set
$$q_\phi(z|x) \equiv \mathcal{N}\big(\mu_\phi(x), \Sigma_\phi(x)\big)\big)$$
where $\Sigma_\phi(x) = diag[\sigma_\phi(x)]$ and $\big(\mu_\phi(x), \sigma_\phi(x)\big) \in \mathbb{R}^{2\times d}$ are outputs of the encoder network. The sampling process $z \sim q_\phi(z|x)$ is therefore performed by sampling $\varepsilon \sim \mathcal{N}(0, I_d)$ and setting $z = \mu_\phi(x) + \Sigma_\phi(x)^{1/2} \cdot \varepsilon$ (re-parametrization trick).
- $p_\theta(x|z)$ the distribution of x given z
- $p_\theta(x) = \int_{\mathcal{Z}} p_\theta(x|z)p_z(z)dz$ the marginal distribution of x
- $q_\phi(z) = \dfrac{1}{N}\sum_{i=1}^N q_\phi(z|x_i)$ the aggregated posterior integrated over the training set.
- $\mathcal{D}_{KL}$ the Kullback-Leibler divergence

We further recall that Kingma and Welling [13] use the unbiased estimate $\widehat{p}_\theta(x)$ of $p_\theta(x)$ defined as

$$\widehat{p}_\theta(x) = \frac{p_\theta(x|z)p_z(z)}{q_\phi(z|x)}$$

to derive the standard Evidence Lower Bound (ELBO) of the log probability $\log p_\theta(x)$ which we wish to maximize:

$$\mathcal{L}_{ELBO} = \mathbb{E}_{x\sim p_\theta(x)}\big[\mathcal{L}_{ELBO}(x)\big]$$

with

$$\mathcal{L}_{ELBO}(x) = \underbrace{\mathbb{E}_{z\sim q_\phi}[\log p_\theta(x|z)]}_{\text{reconstruction}} - \underbrace{\mathcal{D}_{KL}\big[q_\phi(z|x)||p_z(z)\big]}_{\text{regularisation}}$$

The *reconstruction* loss is maximized when $\hat{x}$ is close to x, thus encouraging a good reconstruction of the input x, while the *regularisation* term is maximized when $q_\phi(z|x)$ is close to $p_z(z)$, encouraging the posterior distribution to follow the chosen prior distribution.

The integration over $p_\theta(x)$ is approximated by the empirical distribution of the training dataset, and the negative ELBO function acts as a loss function to minimize for the encoder and decoder networks.

VAE with a VampPrior (VAMP) Starting from the observation that a standard Gaussian prior may be too simplistic, Tomczak and Welling [23] proposes a less restrictive prior: the Variational Mixture of Posteriors (VAMP). A VAE with a VAMP prior aims at relaxing the posterior constraint by replacing the conventional normal prior with a multimodal aggregated posterior given by:

$$p_z(z) = \frac{1}{K} \sum_{k=1}^{K} q_\phi(z|u_k) \,,$$

where u_k are pseudo-inputs living in the data space $\mathcal{X}$ learned through back-propagation and acting as anchor points for the prior distribution. For the VAMP VAE implementation, we use the same architecture as the authors' implementation for the network generating the pseudo-inputs: a MLP with a single layer and Tanh activation.

Results by configuration

Table 5: VAMP configurations

Config	1	2	3	4	5	6	7	8	9	10
Number of pseudo-inputs (K)	10	20	30	500	100	150	200	250	300	500

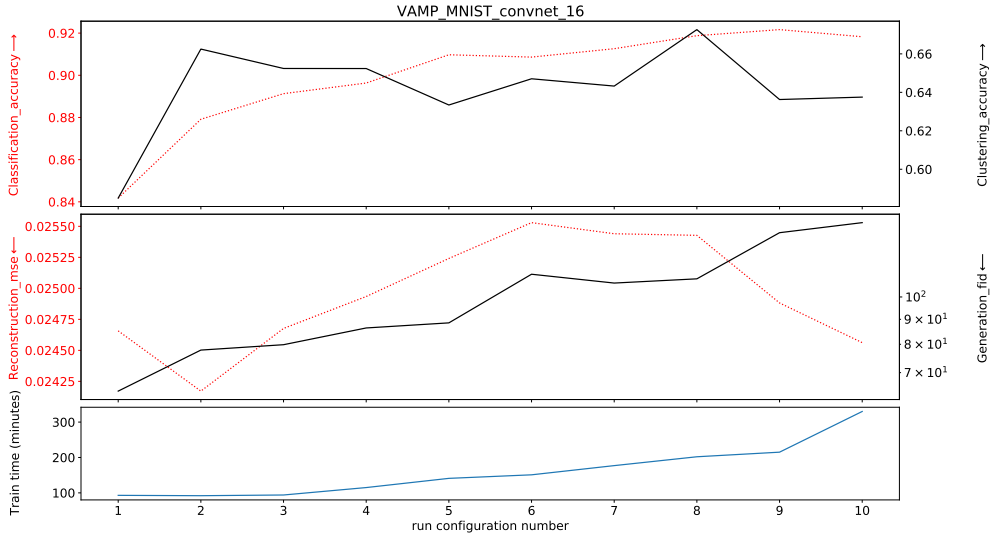

Figure 13: Results on VAMP

Importance Weighted Autoencoder (IWAE) Burda et al. [1] introduce an alternative lower bound to maximize, derived from importance weighting where the new unbiased estimate $\widehat{p}_\theta(x)$ of the marginal distribution $p_\theta(x)$ is computed with L samples $z_1, \ldots, z_L \sim q_\phi(z|x)$:

$$\widehat{p}_\theta(x) = \frac{1}{L} \sum_{i=1}^{L} \frac{p_\theta(x|z_i) p_z(z_i)}{q_\phi(z_i|x)} .$$

This estimate induces a new lower bound of the true marginal distribution $p_\theta(x)$ using Jensen's inequality:

$$\mathcal{L}_{\text{IWAE}}(x) := \mathbb{E}_{z_1,\ldots,z_L \sim q(z|x)} \left[\log \frac{1}{L} \sum_{i=1}^{L} \frac{p_\theta(x|z_i) p_z(z_i)}{q_\phi(z_i|x)} \right] \leq \log \underbrace{\mathbb{E}_{z_1,\ldots,z_L \sim q(z|x)} \left[\widehat{p}_\theta(x) \right]}_{p_\theta(x)} .$$

As the number of samples L increases, $\mathcal{L}_{\text{IWAE}}(x)$ becomes closer to $\log p_\theta(x)$, therefore providing a tighter bound on the true objective. Note that when $L = 1$ we recover the original VAE framework.

As expected the reconstruction quality increases with the number of samples. Nonetheless, we note that increasing the number of samples has a significant impact on the computation of a single training step, therefore leading to a much slower training process.

Results by configuration

Table 6: IWAE configurations

Config	1	2	3	4	5	6	7	8	9	10
Number of samples (L)	2	3	4	5	6	7	8	9	10	12

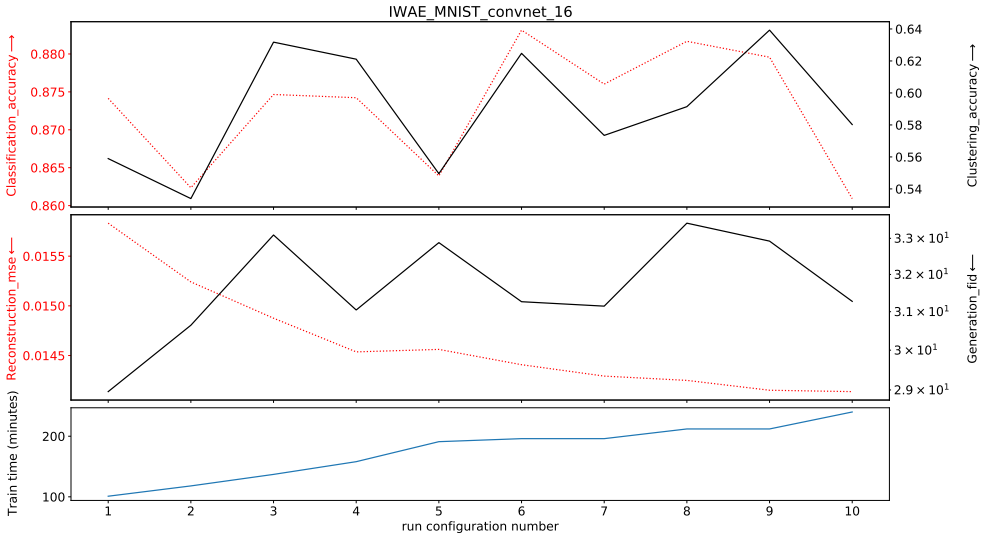

Figure 14: Results on IWAE

Variational Inference with Normalizing Flows (VAE-lin-NF) In order to model a more complex family of approximate posterior distributions, Rezende and Mohamed [20] propose to use a succession of normalising flows to transform the simple distribution $q_\phi(z|x)$, allowing it to model more complex behaviours. In practice, after having sampled $z_0 \sim q_\phi(z|x)$, z_0 is passed through a chain of K invertible smooth mappings from the latent space $\mathbb{R}^d$ to itself:

$$z_K = f_K \circ \cdots \circ f_2 \circ f_1(z_0)\,.$$

The modified latent vector z_K is then used as input z for the decoder network. In their paper, the authors propose to use two types of transformations: planar and radial flows.

$$f_{\text{planar}}(z) = z + uh(w^\top z + b) \quad ; \quad f_{\text{radial}}(z) = z + \beta g(\alpha, r)(z - z_0)\,,$$

where h is a smooth non-linearity with tractable derivatives and $g(\alpha, r) = \frac{1}{\alpha + r}$. The parameters are such that $r = \|z - z_0\|$, $u, w, z_0 \in \mathbb{R}^d$, $\alpha \in \mathbb{R}^+$ and $b, \beta \in \mathbb{R}$. We can easily compute the resulting density q given by

$$\log q(z_K) = \log q_\phi(z_0|x) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial z} \right|.$$

This makes the ELBO tractable and optimisation possible.

Results by configuration

Table 7: VAE-lin-NF configurations

Config	1	2	3	4	5	6	7	8	9	10
Flow sequence	PPPPP	RRRRR	PRPRP	10P	15P	20P	30P	PRPRPRPRPR	PPP	PRPPRPPPPR

'P' stands for planar flow - 'R' stands for radial flow

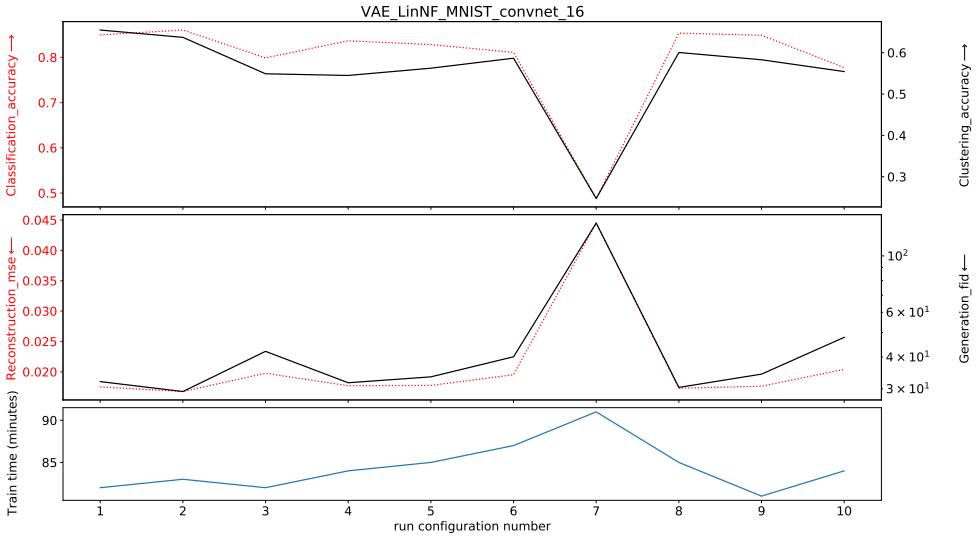

Figure 15: Results on VAE-lin-NF

Variational Inference with Inverse Autoregressive Flow (VAE-IAF) Kingma et al. [14] improve upon the works of [20] with a new type of normalising flow that better scales to high-dimensional latent spaces. The main idea is again to apply several transformations to a sample from a simple distribution in order to model richer distributions. Starting from $z_0 \sim q_\phi(z|x)$, the proposed IAF flow consists in applying consecutively the following transformation

$$z_k = \mu_k + \sigma_k \cdot z_{k-1} \,,$$

where μ_k and σ_k are the outputs of an autoregressive neural network taking z_{k-1} as input. Inspired from the original paper, to implement one Inverse Autoregressive Flow we use MADE [7] and stack multiple IAF together to create a richer flow. The MADE mask is made sequentially for the masked autoencoders and the ordering is reversed after each MADE.

Results by configuration

Table 8: VAE-IAF configurations

Config	1	2	3	4	5	6	7	8	9	10
hidden size in MADE	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	64.0	128.0
number hidden units in MADE	2.0	2.0	2.0	2.0	2.0	2.0	4.0	6.0	2.0	2.0
number of IAF blocks	1.0	2.0	5.0	10.0	20.0	4.0	4.0	4.0	4.0	4.0

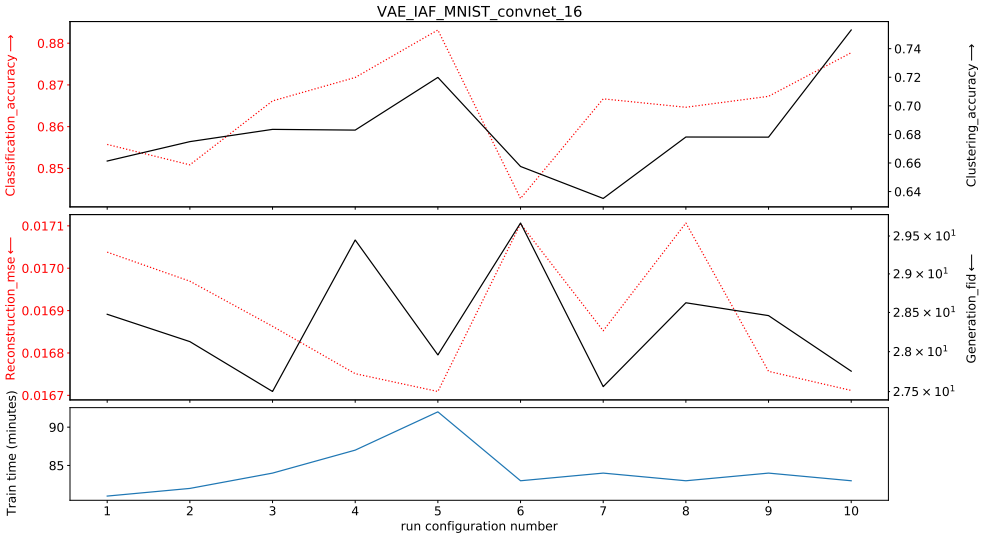

Figure 16: Results on VAE-IAF

β-VAE Higgins et al. [10] argue that increasing the weight of the KL divergence term in the ELBO loss enforces a stronger disentanglement of the latent features as the posterior probability is forced to match a multivariate standard Gaussian. They propose to add a hyper-parameter β in the ELBO leading to the following objective to maximise:

$$\mathcal{L}_{\beta\text{-VAE}}(x) = \mathbb{E}_{z\sim q_\phi}[\log p_\theta(x|z)] - \beta\mathcal{D}_{KL}\left[q_\phi(z|x)||p_z(z)\right].$$

Although the original publication specifies $\beta > 1$ to encourage a better disentanglement, a smaller value of β can be used to relax the regularisation constraint of the VAE. Therefore, for this model we consider a range of values for β from $1e^{-3}$ to $1e^{3}$.

As expected, we see in Fig. 17 a trade-off appearing between reconstruction and generation. Indeed, a very small β will tend to less regularise the model since the latent variables will no longer be driven to follow the prior, favouring a better reconstruction. On the other hand, a higher value for β will constrain the model, leading to a better generation quality. Moreover, as can be seen in Fig. 17, too high a value of β will lead to over-regularisation, resulting in poor performances on all evaluated tasks.

Results by configuration

Table 9: β-VAE configurations

Config	1	2	3	4	5	6	7	8	9	10
β	$1e^{-3}$	$1e^{-2}$	$1e^{-1}$	0.5	2	5	10	20	$1e^{2}$	$1e^{3}$

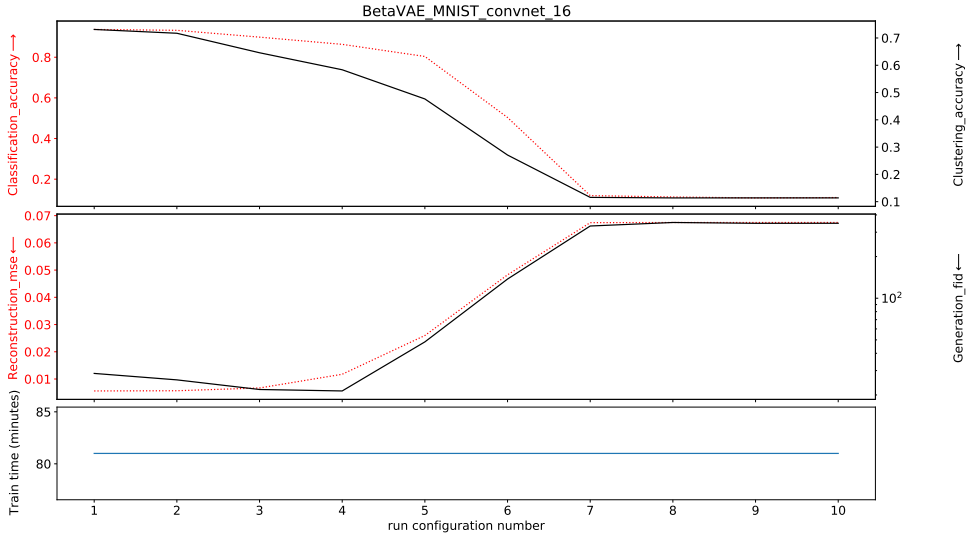

Figure 17: Results on β-VAE

β-TC-VAE Chen et al. [4] extend on the ideas of Higgins et al. [10] by rewriting and re-weighting specific terms in the ELBO loss with multiple hyperparameters. The authors note that the KL-divergence term of the ELBO loss can be rewritten as

$$\mathbb{E}_{x\sim p_\theta}\Big[\mathcal{D}_{KL}\big[q_\phi(z|x)||p_z(z)\big]\Big] = \underbrace{I(x,z)}_{\text{Mutual information}} + \underbrace{\mathcal{D}_{KL}\big[q_\phi(z)||\prod_{j=1}^{d} q_\phi(z_j)\big]}_{\text{TC-loss}} + \underbrace{\sum_{j=1}^{d}\mathcal{D}_{KL}\big[q_\phi(z_j)||p_z(z_j)\big]}_{\text{Dimension-wise KL}}$$

- The mutual information term corresponds to the amount of information shared by x and its latent representation z. It is claimed that maximising the mutual information encourages better disentanglement and a more compact representation of the data.
- The TC-loss corresponds to the total correlation between the latent distribution and its fully disentangled version, maximising it enforces the dimensions of the latent vector to be uncorrelated.
- Maximising the dimension-wise KL prevents the marginal distribution of each latent dimension from diverging too far from the prior Gaussian distribution

The authors therefore propose to replace the classical regularisation term with the more general term

$$\mathcal{L}_{\text{reg}} := \alpha I(x,n) + \beta\mathcal{D}_{KL}\big[q_\phi(z)||\prod_j q_\phi(z_j)\big] + \gamma\sum_j \mathcal{D}_{KL}\big[q_\phi(z_j)||p_z(z_j)\big]\,.$$

Similarly to the authors, we set $\alpha = \gamma = 1$ and only perform a search on the parameter β. Fig. 18 shows a reconstruction-generation trade-off similar to the β-VAE model

Results by configuration

Table 10: β-TC-VAE configurations

Config	1	2	3	4	5	6	7	8	9	10
β	$1e^{-3}$	$1e^{-2}$	$1e^{-1}$	0.5	1	2	5	10	50	$1e^{2}$

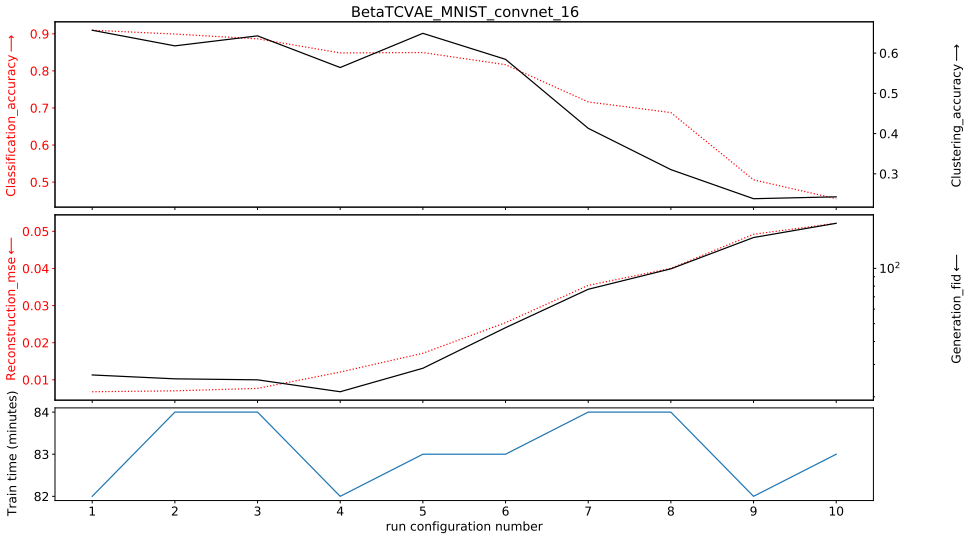

Figure 18: Results on β-TC-VAE

Factor VAE Kim and Mnih [11] augment the VAE objective with a penalty that encourages factorial representation of the marginal distributions, enforcing a stronger disentangling of the latent space. Noting that a high β value in β-VAE ELBO loss encourages disentanglement at the expense of reconstruction quality, FactorVAE proposes a new lower bound of the log likelihood with an added disentanglement term:

$$\mathcal{L}_{\text{FactorVAE}}(x) := \mathcal{L}_{ELBO}(x) - \gamma \mathcal{D}_{KL}\big(q_\phi(z)||\bar{q}_\phi(z)\big) \text{ , with } \bar{q}_\phi(z) := \prod_{j=1}^{d} q_\phi(z_j)$$

The distribution of representations $q_\phi(z) = \frac{1}{N}\sum_{i=1}^{N} q_\phi(z|x_i)$ of the entire dataset is therefore forced to be close to its fully-disentangled equivalent $\bar{q}_\phi(z)$ while leaving the ELBO loss as it is. They further propose to approximate the KL divergence with a discriminator network D that is trained jointly to the VAE:

$$\mathcal{D}_{KL}\big(q(z)||\bar{q}(z)\big) \approx \mathbb{E}_{q_z(z)}\left[\log \frac{D(z)}{1-D(z)}\right]$$

As suggested in the authors's paper, the discriminator is set as a MLP composed of 6 layers each with 1000 hidden units and LeakyReLU activation.

Results by configuration

Table 11: FactorVAE configurations

Config	1	2	3	4	5	6	7	8	9	10
γ	1	2	5	10	15	20	30	40	50	100

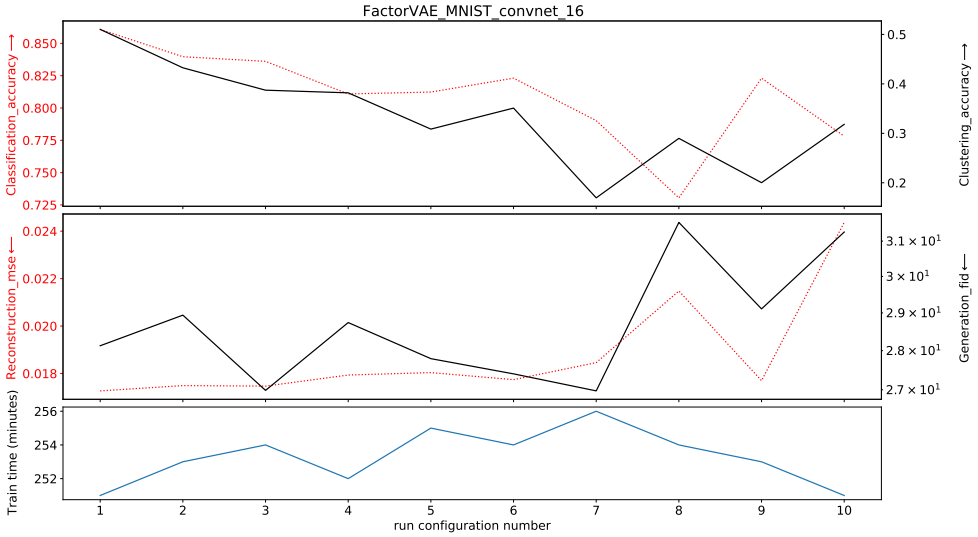

Figure 19: Results on FactorVAE

InfoVAE Zhao et al. [27] note that the traditional VAE ELBO objective can lead to both inaccurate amortized inference and VAE models that tend to ignore most of the latent variables, therefore not fully taking advantage of the modelling capacities of the VAE scheme and learning less meaningful latent representations. In order to counteract these two issues, they propose to rewrite and re-weight the ELBO objective in order to counterbalance the imbalance between the distribution in the data space and the latent space, and add a mutual information term between x and z to encourage a stronger dependency between the two variables, preventing the model from ignoring the latent encoding. One can re-write the ELBO loss in order to explicit the KL divergence between the marginalised posterior and the prior

$$\mathcal{L}_{\text{ELBO}}(x) := -\mathcal{D}_{KL}[q_\phi(z)||p_z(z)] - \mathbb{E}_{z\sim p_z}\left[\mathcal{D}_{KL}[q_\phi(x|z)||p_\theta(x|z)]\right].$$

Introducing an additional mutual information term $I_q(x;z)$ and extending the objective function to use any given divergence D between probability measures instead of the KL objective, the authors propose a new objective defined as

$$\mathcal{L}_{\text{InfoVAE}}(x) := -\lambda D[q_\phi(z)||p_z(z)] - \mathbb{E}_{z\sim p_z}\left[\mathcal{D}_{KL}[q_\phi(x|z)||p_\theta(x|z)]\right] + \alpha I_q(x;z)$$

where λ and α are hyperparameters. In our experiments, α is set to 0 as recommended in the paper in the case where $p_\theta(x|z)$ is a simple distribution. D is chosen as the Maximum Mean Discrepancy (MMD) [9], defined as

$$\text{MMD}_k(p_\lambda(z), q_\phi(z)) = ||\int_{\mathcal{Z}} k(z,.)dp_\lambda(z) - \int_{\mathcal{Z}} k(z,.)dq_\phi(z)||_{\mathcal{H}_k}$$

with $k : \mathcal{Z}\times\mathcal{Z} \to \mathbb{R}$ a positive-definite kernel and its associated RKHS $\mathcal{H}_k$. We choose to differentiate 2 cases in the benchmark: one with a Radial Basis Function (RBF) kernel, the other with the Inverse MultiQuadratic (IMQ) kernel as proposed in [22] where the kernel is given by $k(x,y) = \sum_{s\in\mathcal{S}} \frac{s\cdot C}{s\cdot C + \|x-y\|_2^2}$ with $s \in [0.1, 0.2, 0.5, 1, 2, 5, 10]$ and $C = 2\cdot d\cdot\sigma^2$, d being the dimension of the latent space and σ a parameter part of the hyper-parameter search.

The authors underline that choosing $\lambda > 0$, $\alpha = 1-\lambda$ and $D = \mathcal{D}_{KL}$, we recover the β-VAE model [10], while choosing $\alpha = \lambda = 1$ and setting D as the Jensen Shannon divergence we recover the Adversarial AE model [19].

Results by configuration

Table 12: InfoVAE configurations

Config	1	2	3	4	5	6	7	8	9	10
kernel bandwidth - σ	$1e^{-2}$	$1e^{-1}$	0.5	1	1	1	1	1	2	5
λ	10	10	10	$1e^{-2}$	$1e^{-1}$	10	100	100	10	10

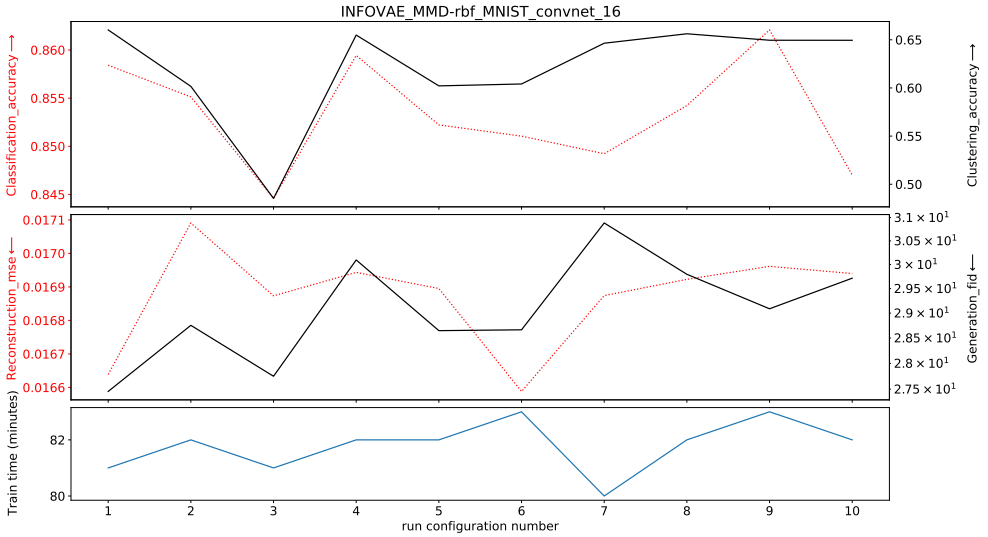

Figure 20: Results on InfoVAE-RBF

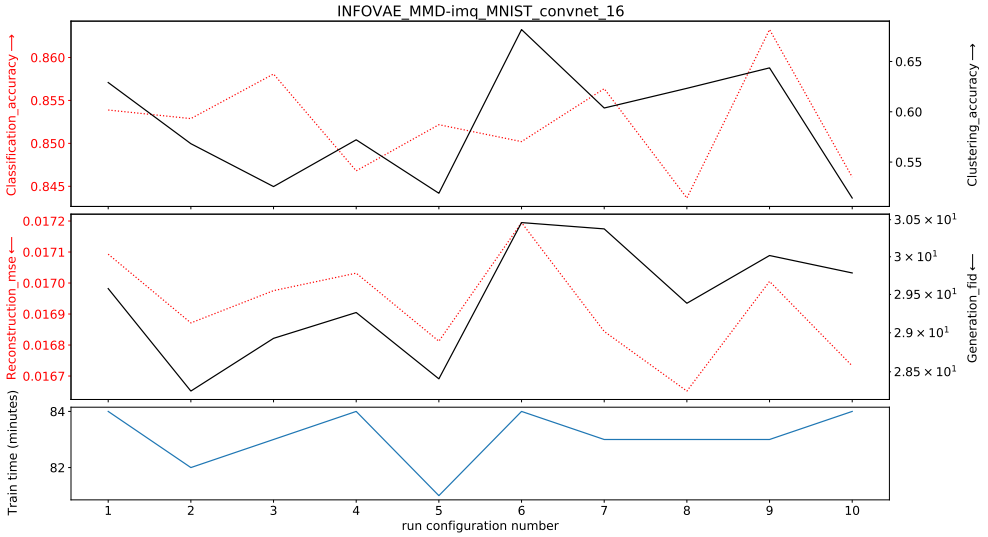

Figure 21: Results on InfoVAE-IMQ

Adversarial AE (AAE) Makhzani [19] propose to use a GAN-like approach by replacing the regularisation induced by the KL divergence with a discriminator network D trained to differentiate between samples from the prior and samples from the posterior distribution. The encoder network therefore acts as a generator network, leading to the following objective

$$\mathcal{L}_{\text{AAE}}(x) = \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] + \alpha\mathcal{L}_{\text{GAN}}\,,$$

with $\mathcal{L}_{\text{GAN}}$ the standard GAN loss defined by

$$\mathcal{L}_{\text{GAN}} = \mathbb{E}_{\tilde{z}\sim p_z(z)}\Big[\log(1 - D(\tilde{z}))\Big] + \mathbb{E}_{x\sim p_\theta}\Big[\mathbb{E}_{z\sim q_\phi(z|x)}[\log D(z)]\Big]\,.$$

For the Adversarial Autoencoder implementation, we use a MLP neural network for the discriminator composed of a single hidden layer with 256 units and ReLU activation.

We observe a similar trade-off between reconstruction and generation quality as observed with β-VAE type models, as the α term acts like the β term, balancing between regularisation and reconstruction.

Results by configuration

Table 13: AAE configurations

Config	1	2	3	4	5	6	7	8	9	10
α	$1e^{-3}$	$1e^{-2}$	$1e^{-1}$	0.25	0.5	0.75	0.9	0.95	0.99	0.999

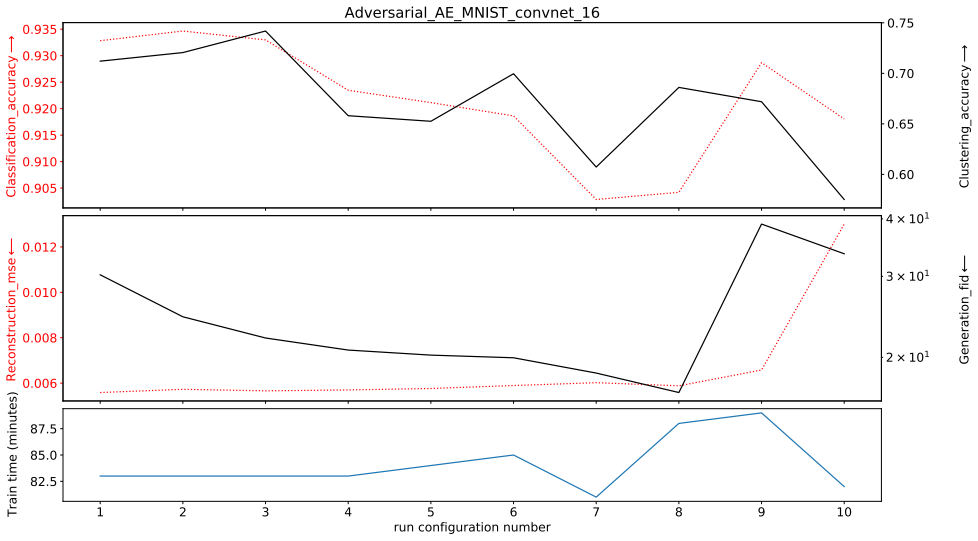

Figure 22: Results on Adversarial AE

EL-VAE (MSSSIM-VAE) Snell et al. [21] propose an extension of the ELBO loss to a more general case where any deterministic reconstruction loss $\Delta(x, \hat{x})$ can be used by replacing the probabilistic decoder p_θ with a deterministic equivalent f_θ such that the reconstruction $\hat{x}$ of x given $z \sim q_\phi(z|x)$ is defined as $\hat{x} = f_\theta(z)$. The modified ELBO objective is thus defined as

$$\mathcal{L}_{\mathrm{EL-VAE}}(x) = \Delta(x, \hat{x}) - \beta \mathcal{D}_{KL}\big(q_\phi(z|x)||p(z)\big)\,,$$

with $\beta \leq 1$. As suggested in the original paper we use a multi scale variant of the single scale SSIM [26]: the Multi Scale Structural Similarity Metric (MS-SSIM) [25].

Results by configuration

Table 14: MSSSIM-VAE configurations

Config	1	2	3	4	5	6	7	8	9	10
β	$1e^{-2}$	$1e^{-2}$	$1e^{-2}$	$1e^{-1}$	$1e^{-1}$	$1e^{-1}$	1	1	1	1
window size in MSSSIM	3	5	11	5	3	11	11	5	3	15

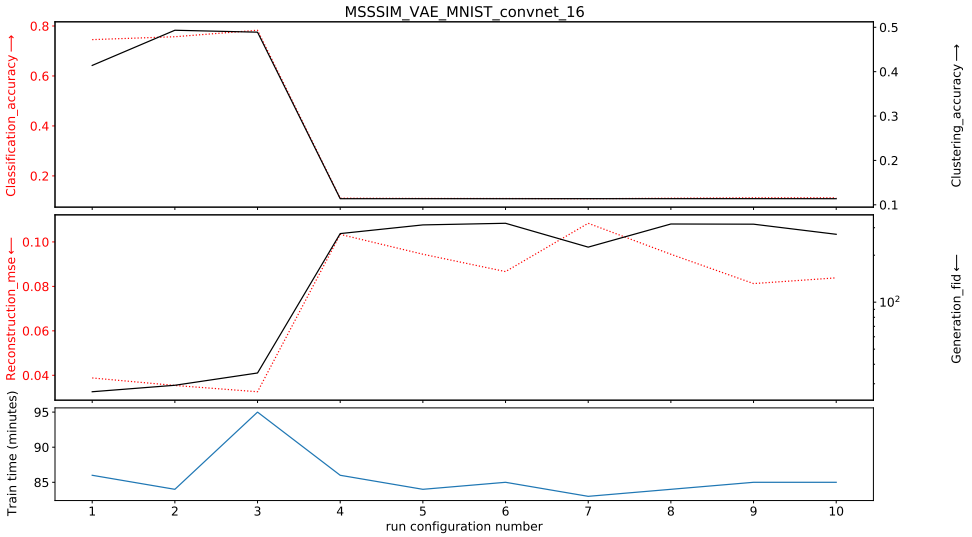

Figure 23: Results on MSSSIM-VAE

VAE-GAN Larsen et al. [16] use a GAN like approach by training a discriminator to distinguish real data from reconstructed data. In addition, the discriminator learns to distinguish between real data and data generated by sampling from the prior distribution in the latent space.

Noting that intermediate layers of a discriminative network trained to differentiate real from generated data can act as data-specific features, the authors propose to replace the reconstruction loss of the ELBO with a Gaussian log-likelihood between outputs of intermediate layers of a discriminative network D:

$$\mathcal{L}_{\text{VAE-GAN}} = \underbrace{\mathbb{E}_{z\sim q_\theta(z|x)}\Big[\log \mathcal{N}(D_l(x)|D_l(\hat{x}), I)\Big]}_{\text{reconstruction}} - \underbrace{\mathcal{D}_{KL}\big[q_\phi(z|x)||p_z(z)\big]}_{\text{regularisation}} - \mathcal{L}_{GAN}\,,$$

where D_l is the output of the l^{th} layer of the discriminator D, chosen to be representative of abstract intermediate features learned by the discriminator, and $\mathcal{L}_{\text{GAN}}$ is the standard GAN objective defined as

$$\mathcal{L}_{\text{GAN}} = \log\left(\frac{D(x)}{1 - D(x_{\text{gen}})}\right),$$

where x_{gen} is generated using $z \sim p_z(z)$. As encouraged by the authors, we add a hyper-parameter α to the reconstruction loss for the decoder only, such that a higher value of α will encourage better reconstruction abilities with respect to the features extracted at the l^{th} layer of the discriminator network, whereas a smaller value will encourage fooling the discriminator, therefore favouring regularisation toward the prior distribution. For the VAEGAN implementation, we use a discriminator whose architecture is similar to the model's encoder given in Table. 2. For MNIST and CIFAR we remove the BatchNorm layer and change the activation of layer 2 to Tanh instead of ReLU. For CELEBA, the BatchNorm layer is kept and the activation of layer 2 is also changed to Tanh. For all datasets, the output size of the last linear layer is set to 1 instead of d and followed by a Sigmoid activation.

Results by configuration

Table 15: VAEGAN configurations

Config	1	2	3	4	5	6	7	8	9	10
α	0.3	0.5	0.7	0.8	0.8	0.8	0.9	0.9	0.99	0.999
reconstruction layer (l)	3	3	3	3	2	4	3	3	3	3

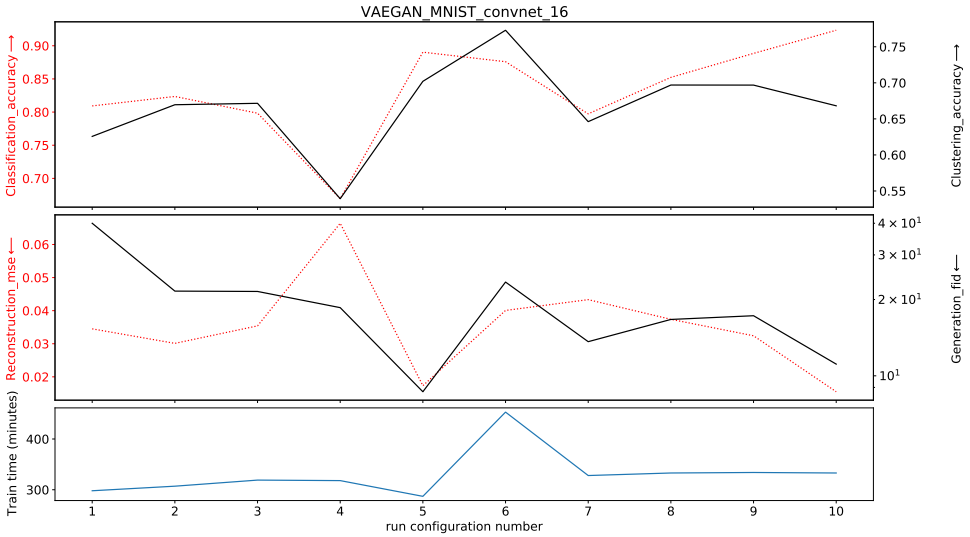

Figure 24: Results on VAEGAN

Wasserstein Autoencoder (WAE) Tolstikhin et al. [22] generalise the VAE objective by replacing both terms in the ELBO: similarly to [21] (EL-VAE), the reconstruction loss is replaced by any measurable cost function Δ, and the standard KL divergence is substituted with any arbitrary divergence $\mathcal{D}$ between two distributions, leading to the following objective function

$$E_{q_\phi(z|x)}[\Delta(x, \hat{x})] + \lambda \mathcal{D}_z(p_z(z), q_\phi(z)) \,,$$

with λ a hyper-parameter. The authors propose two different penalties for $\mathcal{D}_z$:

1. **GAN-based: WAE-GAN**
   An adversarial discriminatory network $D(z, z')$ is trained jointly to separate the "true" points sampled from the prior $p_z(z)$ from the "fake" ones sampled from $q_\phi(z|x)$, similarly to [19] (Adversarial AE).
2. **MMD-based**
   The Maximum Mean Discrepancy is used as a distance between the prior and the posterior distribution. This is the case considered in the benchmark. We choose to differentiate 2 cases in the benchmark: one with a Radial Basis Function (RBF) kernel, the other with the Inverse MultiQuadratic (IMQ) kernel as proposed in [22] where the kernel is given by $k(x, y) = \sum_{s \in \mathcal{S}} \frac{s \cdot C}{s \cdot C + \|x - y\|_2^2}$ with $s \in [0.1, 0.2, 0.5, 1, 2, 5, 10]$ and $C = 2 \cdot d \cdot \sigma^2$, d being the dimension of the latent space and σ a parameter part of the hyper-parameter search. As proposed by the authors, for this model we choose to use a deterministic encoder meaning that $q_\phi(z|x) = \delta_{\mu_\phi(x)}$.

Results by configuration

Table 16: WAE configurations

Config	1	2	3	4	5	6	7	8	9	10
kernel bandwidth - σ	$1e^{-2}$	$1e^{-1}$	0.5	1	1	1	1	1	2	5
λ	1	1	1	$1e^{-2}$	$1e^{-1}$	1	10	100	1	1

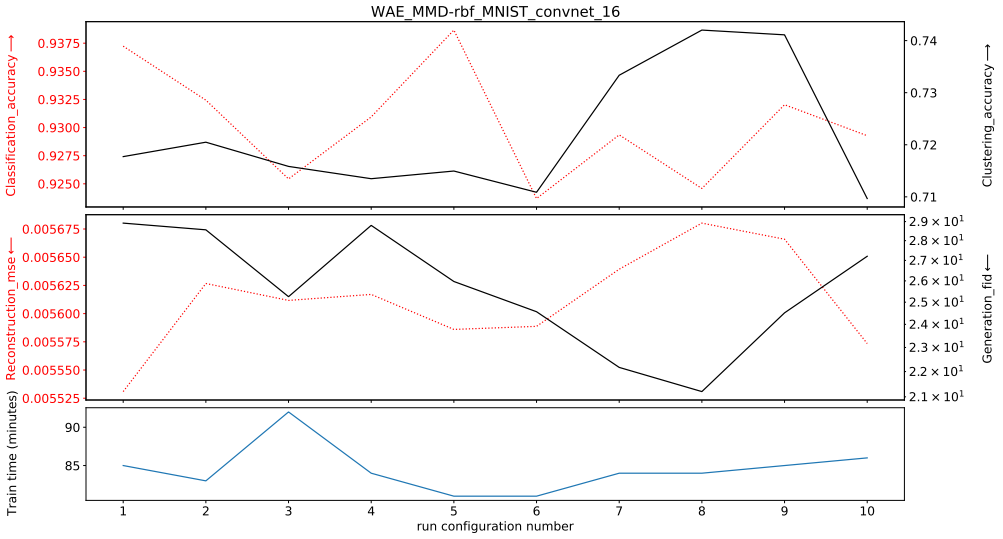

Figure 25: Results on WAE-RBF

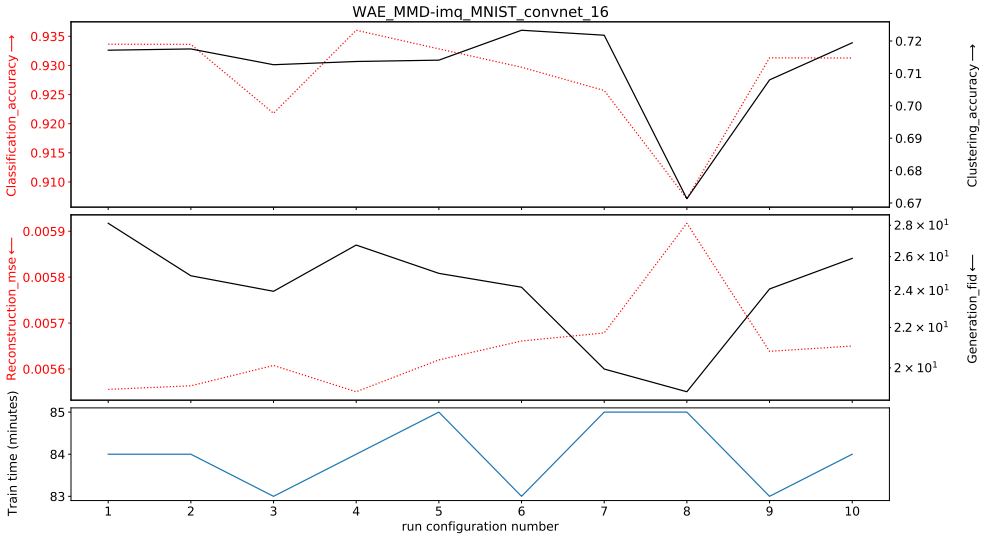

Figure 26: Results on WAE-IMQ

Vector Quantized VAE (VQ-VAE) Van Den Oord et al. [24] propose to use a discrete space. Therefore, the latent embedding space is defined as a $\mathbb{R}^{K\times D}$ vector space of K different D dimensional embedding vectors $\mathcal{E} = \{e_1, \ldots, e_K\}$ which are learned and updated at each iteration.

Given an embedding size d and an input x, the output of the encoder $z_e(x)$ is of size $\mathbb{R}^{d\times D}$. Each of its d elements is then assigned to the closest embedding vector resulting in an embedded encoding $z_q(x) \in \mathcal{E}^d$ such that $\big(z_q(x)\big)_j = e_l$ where $l = \text{argmin}_{1\leq l\leq d}||(z_e(x))_j - e_l||_2$ for $j \in [1, d]$. Since the argmin operation is not differentiable, learning of the embeddings and regularisation of the latent space is done by introducing the stopgradient operator sg in the training objective:

$$\mathcal{L}_{\text{VQ-VAE}}(x) := \log p(x|z_q(x)) + \alpha||sg[z_e(x)] - e||_2^2 + \beta||z_e(x) - sg[e]||_2^2 \,.$$

For the VQVAE implementation we use the Exponential Moving Average update as proposed in [24] to replace the term $||sg[z_e(x)] - e||_2^2$ in the loss. Thus, we consider only two hyper-parameters in the search: the size of the dictionary of embeddings K and the regularisation factor β.

Results by configuration

Table 17: VQVAE configurations

Config	1	2	3	4	5	6	7	8	9	10
K	128	256	512	512	512	512	512	512	1024	2948
β	0.25	0.25	0.9	0.1	0.5	0.25	0.75	0.25	0.25	0.25

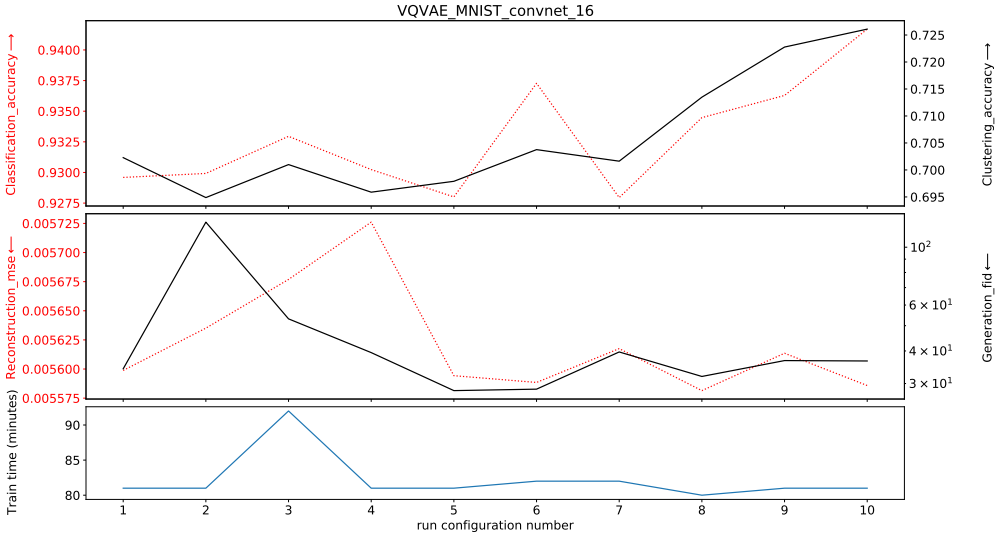

Figure 27: Results on VQVAE

RAE L2 and RAE GP Ghosh et al. [8] propose to replace the stochastic VAE with a deterministic autoencoder by adapting the ELBO objective to a deterministic case. Under standard VAE assumption with Gaussian decoder, both the reconstruction and the regularisation terms in the ELBO loss can be written in closed form as

$$\begin{cases} \mathcal{L}_{\text{reconstruction}}(x) = ||x - \hat{x}||_2^2 \,, \\ \mathcal{L}_{\text{regularisation}}(x) = \frac{1}{2}\Big[||z||_2^2 - d + \sum_{i=1}^{d}(\sigma_\phi(x)_i - \log \sigma_\phi(x)_i)\Big] \,. \end{cases}$$

Arguing that the regularisation of the VAE model is done through a noise injection mechanism by sampling from the approximate posterior distribution $z \sim \mathcal{N}(\mu_\phi, diag(\sigma_\phi))$, the authors propose to replace this stochastic regularisation with an explicit regularisation term, leading to the following deterministic objective:

$$\mathcal{L}_{\text{RAE}} = ||x - \hat{x}||_2^2 + \frac{\beta}{2}||z||_2^2 + \lambda \mathcal{L}_{\text{REG}} \,, \tag{1}$$

where $\mathcal{L}_{\text{REG}}$ is an explicit regularisation. They propose to use either

- a L2 loss on the weights of the decoder (RAE-L2), which amounts to applying weight decay on the parameters of the decoder.
- a gradient penalty on the output of the decoder (RAE-GP), which amounts to applying a L2 norm on the gradient of the output of the decoder.

Results by configuration

Table 18: RAE configurations

Config	1	2	3	4	5	6	7	8	9	10
β	$1e^{-6}$	$1e^{-4}$	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$1e^{-2}$	$1e^{-1}$	1
λ	$1e^{-3}$	$1e^{-3}$	$1e^{-6}$	$1e^{-4}$	$1e^{-2}$	$1e^{-1}$	1	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$

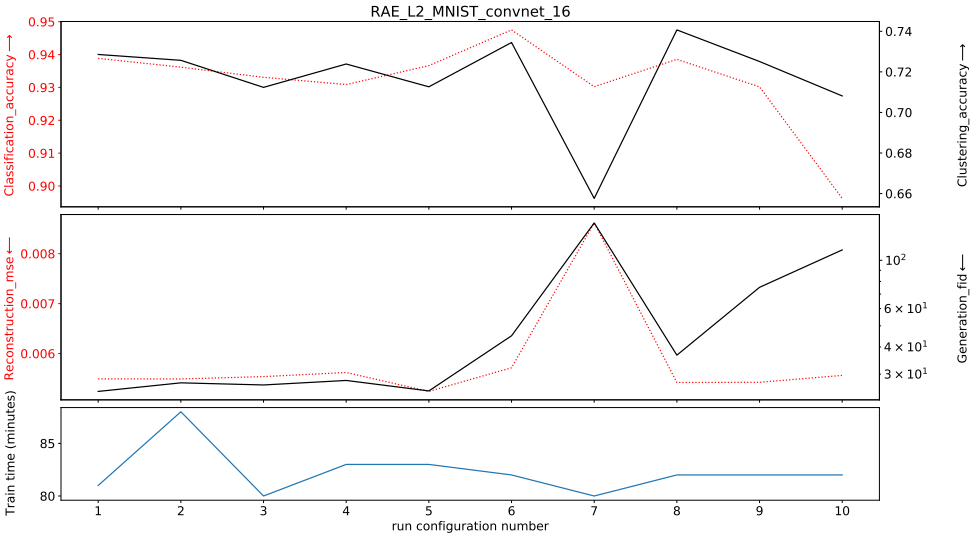

Figure 28: Results on RAE-L2

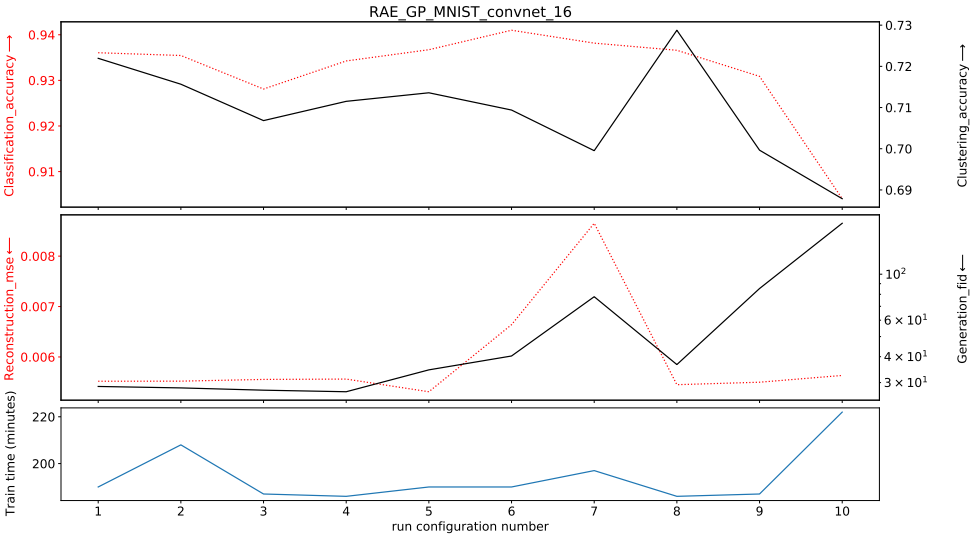

Figure 29: Results on RAE-GP

References

[1] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv:1509.00519 [cs, stat]*, 2016.

[2] Anthony L Caterini, Arnaud Doucet, and Dino Sejdinovic. Hamiltonian variational auto-encoder. In *Advances in Neural Information Processing Systems*, pages 8167–8177, 2018.

[3] Clément Chadebec, Elina Thibeau-Sutre, Ninon Burgos, and Stéphanie Allassonnière. Data Augmentation in High Dimensional Low Sample Size Setting Using a Geometry-Based Variational Autoencoder. *arXiv preprint arXiv:2105.00026*, 2021.

[4] Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

[5] Bin Dai and David Wipf. Diagnosing and Enhancing VAE Models. In *International Conference on Learning Representations*, 2018.

[6] Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. Hyperspherical variational auto-encoders. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 856–865. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

[7] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR, 2015.

[8] Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[9] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[10] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

[11] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR, 2018.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114 [cs, stat]*, 2014.

[14] Durk P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

[15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[16] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

[17] Yann LeCun. The MNIST database of handwritten digits. 1998.

[18] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[19] Alireza et al. Makhzani. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2016.

[20] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[21] Jake Snell, Karl Ridgeway, Renjie Liao, Brett D Roads, Michael C Mozer, and Richard S Zemel. Learning to generate images with perceptual similarity metrics. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 4277–4281. IEEE, 2017.

[22] I Tolstikhin, O Bousquet, S Gelly, and B Schölkopf. Wasserstein auto-encoders. In *6th International Conference on Learning Representations (ICLR 2018)*, 2018.

[23] Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR, 2018.

[24] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[25] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[26] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[27] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2016.