Supplementary Material of Learning Disentangled Representations via Mutual Information Estimation

Eduardo Hugo Sanchez[1,2], Mathieu Serrurier[1,2], and Mathias Ortner[3]

[1] IRT Saint Exupéry, Toulouse, France
{eduardo.sanchez, mathieu.serrurier}@irt-saintexupery.com
[2] IRIT, Université Toulouse III - Paul Sabatier, Toulouse, France
[3] Airbus, Toulouse, France
mathias.ortner@airbus.com

1 Implementation details

In this section, we provide the architecture details, the learning hyperparameters and the optimizer for each experiment.

1.1 Colored-MNIST experiment

Dataset creation The following 12 colors in RGB format are used to create the colored-MNIST dataset: (255.0, 0.0, 0.0), (255.0, 128.0, 0.0), (255.0, 255.0, 0.0), (128.0, 255.0, 0.0), (0.0, 255.0, 0.0), (0.0, 255.0, 128.0), (0.0, 255.0, 255.0), (0.0, 128.0, 255.0), (0.0, 0.0, 255.0), (128.0, 0.0, 255.0), (255.0, 0.0, 255.0), (255.0, 0.0, 128.0). These colors are randomly selected to change the background/digit color. The training dataset is used to train our model and the classifiers while the test dataset is used to compute the classification accuracy.

Shared representation learning In order to learn the shared representation, we use Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ during 50000 iterations. We use a batch size of 64 image pairs of size $28 \times 28 \times 3$ at each iteration. Concerning the constant coefficients to weight the terms of the objective function $\mathcal{L}^{shared}$, we use $\alpha^{sh} = 0.5$, $\beta^{sh} = 1.0$ and $\gamma = 0.1$. The size of the shared representation is 64.

During this stage, the shared representation encoders and the statistics networks are learned. The shared representation encoders $E^{sh}_{\psi_X}$ and $E^{sh}_{\psi_Y}$ are implemented as convolutional neural networks described in Table 1 with $z_{dim} = 64$. The architecture of the global statistics networks $T^{sh}_{\theta_X}$ and $T^{sh}_{\theta_Y}$ is shown in Table 2. The global statistics network shares some layers with the shared representation encoder. To compute the mutual information, the global statistics network takes as input two output layers from the shared representation encoder:

– ShGInput0: the output of the Conv2 layer from the shared representation encoder

– ShGInput1: the output of the Output0 layer from the shared representation encoder

The architecture of the local statistics networks $T^{sh}_{\phi_X}$ and $T^{sh}_{\phi_Y}$ is shown in Table 3. Similarly, the local statistics network shares some layers with the shared representation encoder by taking as input the output of the Conv2 layer and the output of the Output0 layer from the shared representation encoder.

Table 1: Representation encoder architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
Input0	Input	-	-	None	None	$28 \times 28 \times 3$
Conv0	Convolutional	4×4	1×1	LeakyReLU	None	$25 \times 25 \times 64$
Conv1	Convolutional	4×4	2×2	LeakyReLU	BatchNorm	$11 \times 11 \times 128$
Conv2	Convolutional	4×4	2×2	LeakyReLU	BatchNorm	$4 \times 4 \times 256$
Flat0	Flatten	-	-	None	None	4096
Output0	Dense	-	-	None	None	z_{dim}

Table 2: Global statistics network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
ShGInput0	Input	-	-	None	None	$4 \times 4 \times 256$
ShGFlat0	Flatten	-	-	None	None	4096
ShGInput1	Input	-	-	None	None	64
ShGConcat0	Concatenation: ShGFlat0 + ShGInput1	-	-	None	None	4160
ShGDense0	Dense	-	-	ReLU	None	512
ShGDense1	Dense	-	-	ReLU	None	512
ShGOutput0	Dense	-	-	None	None	1

Table 3: Local statistics network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
ShLInput0	Input	-	-	None	None	$4 \times 4 \times 256$
ShLInput1	Input	-	-	None	None	64
ShLConcat0	Tile+concatenation: ShLInput0 + ShLInput1	-	-	None	None	$4 \times 4 \times 320$
ShLConv0	Convolutional	1×1	1×1	ReLU	None	$4 \times 4 \times 512$
ShLConv1	Convolutional	1×1	1×1	ReLU	None	$4 \times 4 \times 512$
ShLOutput0	Convolutional	1×1	1×1	None	None	$4 \times 4 \times 1$

Exclusive representation learning To learn the exclusive representation, we use Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ during 10000 iterations. We use a batch size of 64 image pairs at each iteration. Concerning the constant coefficients to weight the terms of the objective function $\mathcal{L}^{exclusive}$, we use $\alpha^{ex} = 0.5$, $\beta^{ex} = 1.0$ and $\lambda_{adv} = 0.05$. The size of the exclusive representation is 8.

During this stage, the exclusive representation encoders, the statistics networks and the discriminator are learned. The exclusive representation encoders $E^{ex}_{\omega_X}$ and $E^{ex}_{\omega_Y}$ are implemented as convolutional neural networks using the architecture shown in Table 1 with $z_{dim} = 8$. The architecture of the global statistics networks $T^{ex}_{\theta_X}$ and $T^{ex}_{\theta_Y}$ is shown in Table 4. To compute the mutual information, the global statistics network takes two inputs:

– ExGInput0: the concatenation of the outputs of the Conv2 layer from the shared and exclusive representation encoders
– ExGInput1: the concatenation of the outputs of the Output0 layer from the shared and exclusive representation encoders

The architecture of the local statistics networks $T^{ex}_{\phi_X}$ and $T^{ex}_{\phi_Y}$ is shown in Table 5 and takes the same inputs as the global statistics network.

In order to perform representation disentanglement, we use a discriminator D_{ρ_X} in an adversarial setting. The discriminator architecture is implemented as a dense neural network which takes as input the shared and exclusive representations (the outputs of the Output0 layer from the shared and exclusive representation encoders) as can be seen in Table 6.

Classification To perform classification using the learned representations, we use Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ during 10000 iterations. We use a batch size of 64 images at each iteration. Regarding the classifier architecture, we use a very simple model as can be seen in Table 7 where $z_{dim} = 64$ when learning from the shared representations and $z_{dim} = 8$ when learning from the exclusive representations. The first layer is composed of

Table 4: Global statistics network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
ExGInput0	Input	-	-	None	None	$4 \times 4 \times 512$
ExGFlat0	Flatten	-	-	None	None	8192
ExGInput1	Input	-	-	None	None	72
ExGConcat0	Concatenation ExGFlat0 + ExGInput1	-	-	None	None	8264
ExGDense0	Dense	-	-	ReLU	None	512
ExGDense1	Dense	-	-	ReLU	None	512
ExGOutput0	Dense	-	-	None	None	1

Table 5: Local statistics network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
ExLInput0	Input	-	-	None	None	$4 \times 4 \times 512$
ExLInput1	Input	-	-	None	None	72
ExLConcat0	Tile + concatenation: ExLInput0 + ExLInput1	-	-	None	None	$4 \times 4 \times 584$
ExLConv0	Convolutional	1×1	1×1	ReLU	None	$4 \times 4 \times 512$
ExLConv1	Convolutional	1×1	1×1	ReLU	None	$4 \times 4 \times 512$
ExLOutput0	Convolutional	1×1	1×1	None	None	$4 \times 4 \times 1$

$C = 32$ neurons while the next layers are composed of N neurones corresponding to number of classification classes ($N = 10$ for digit number classification and $N = 12$ for digit/background color classification). We use the same architecture to train a classifier using the disentangled representations from the models proposed by Gonzalez-Garcia et al. [11] (their model uses different representation dimensions: $z_{dim} = 8 \times 8 \times 512$ when learning from the shared representations and $z_{dim} = 128$ when learning from the exclusive representations) and Jha et al. [18].

Table 6: Discriminator network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
DInput0	Input	-	-	None	None	64
DInput1	Input	-	-	None	None	8
DConcat0	Concatenation	-	-	None	None	72
	DInput0 + DInput1	-	-			
DDense0	Dense	-	-	ReLU	None	1000
DDense1	Dense	-	-	ReLU	None	200
DOutput0	Dense	-	-	None	None	1

Table 7: Classifier architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
CInput0	Input	-	-	None	None	z_{dim}
CDense0	Dense	-	-	ReLU	BatchNorm	C
CDense0	Dense	-	-	ReLU	BatchNorm	N
COutput0	Dense	-	-	Softmax	None	N

1.2 3D Shapes experiment

Shared representation learning To learn the shared representation, we use Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ during 50000 iterations. We use a batch size of 64 image pairs of size $64 \times 64 \times 3$ at each iteration. Concerning the constant coefficients to weight the objective function $\mathcal{L}^{shared}$, we use $\alpha^{sh} = 0.5$, $\beta^{sh} = 1.0$ and $\gamma = 0.1$. The size of the shared representation is 64.

As the image size is higher than in the previous case, the network architectures are slightly modified. Moreover, since data comes from a single domain we can share weights (i.e. ψ_X=ψ_Y, θ_X=θ_Y, etc) to reduce the number of networks. The shared representation encoder $E^{sh}_{\psi_X}$ is defined by a convolutional neural networks which is shown in Table 8 with $z_{dim} = 64$. The architecture of the global statistics network $T^{sh}_{\theta_X}$ is shown in Table 9. The global statistics network takes as input:

- ShGInput0: the output of the Conv3 layer from the shared representation encoder
- ShGInput1: the output of the Output0 layer from the shared representation encoder

The architecture of the local statistics network $T^{sh}_{\phi_X}$ is shown in Table 10. Similarly, the local statistics network shares the same layers as the global statistics network with respect to the shared representation encoder by taking as input the output of the Conv3 layer and the output of the Output0 layer from the shared representation encoder.

Table 8: Representation encoder architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
Input0	Input	-	-	None	None	$64 \times 64 \times 3$
Conv0	Convolutional	4×4	1×1	LeakyReLU	None	$61 \times 61 \times 64$
Conv1	Convolutional	4×4	2×2	LeakyReLU	BatchNorm	$29 \times 29 \times 128$
Conv2	Convolutional	4×4	2×2	LeakyReLU	BatchNorm	$13 \times 13 \times 256$
Conv3	Convolutional	4×4	2×2	LeakyReLU	BatchNorm	$5 \times 5 \times 512$
Flat0	Flatten	-	-	None	None	12800
Output0	Dense	-	-	None	None	z_{dim}

Table 9: Global statistics network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
ShGInput0	Input	-	-	None	None	$5 \times 5 \times 512$
ShGConv0	Convolutional	3×3	1×1	ReLU	None	$3 \times 3 \times 64$
ShGConv1	Convolutional	3×3	1×1	None	None	$1 \times 1 \times 32$
ShGFlat0	Flatten	-	-	None	None	32
ShGInput1	Input	-	-	None	None	64
ShGConcat0	Concatenation: ShGFlat0 + ShGInput1	-	-	None	None	96
ShGDense0	Dense	-	-	ReLU	None	512
ShGDense1	Dense	-	-	ReLU	None	512
ShGOutput0	Dense	-	-	None	None	1

Table 10: Local statistics network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
ShLInput0	Input	-	-	None	None	$5 \times 5 \times 512$
ShLInput1	Input	-	-	None	None	64
ShLConcat0	Tile + Concatenation: ShLInput0 + ShLInput1	-	-	None	None	$5 \times 5 \times 576$
ShLConv0	Convolutional	1×1	1×1	ReLU	None	$5 \times 5 \times 512$
ShLConv1	Convolutional	1×1	1×1	ReLU	None	$5 \times 5 \times 512$
ShLOutput0	Convolutional	1×1	1×1	None	None	$5 \times 5 \times 1$

Exclusive representation learning To learn the exclusive representation, we use Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ during 10000 iterations. We use a batch size of 32 image pairs at each iteration. Concerning the constant coefficients to weight the objective function $\mathcal{L}^{exclusive}$, we use $\alpha^{ex} = 0.5$, $\beta^{ex} = 1.0$ and $\lambda_{adv} = 0.01$. The size of the exclusive representation is 64.

During this stage, the exclusive representation encoder, the statistics networks and the discriminator are learned. The exclusive representation encoder $E^{ex}_{\omega_X}$ is defined by a convolutional neural networks using the architecture shown in Table 8 with $z_{dim} = 64$. The architecture of the global statistics network $T^{ex}_{\theta_X}$ is shown in Table 11. The global statistics network takes two inputs to compute the mutual information:

– ExGInput0: the concatenation of the outputs of the Conv3 layer from the shared and exclusive representation encoders

– ExGInput1: the concatenation of the outputs of the Output0 layer from the shared and exclusive representation encoders

The architecture of the local statistics networks $T^{ex}_{\phi_X}$ is shown in Table 12. The local statistics network takes the same inputs as the global statistics network to compute the mutual information.

Table 11: Global statistics network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
ExGInput0	Input	-	-	None	None	$5 \times 5 \times 1024$
ExGConv0	Convolutional	3×3	1×1	ReLU	None	$3 \times 3 \times 64$
ExGConv1	Convolutional	3×3	1×1	None	None	$1 \times 1 \times 32$
ExGFlat0	Flatten	-	-	None	None	32
ExGInput1	Input	-	-	None	None	128
ExGConcat0	Concatenation ExGFlat0 + ExGInput1	-	-	None	None	160
ExGDense0	Dense	-	-	ReLU	None	512
ExGDense1	Dense	-	-	ReLU	None	512
ExGOutput0	Dense	-	-	None	None	1

Table 12: Local statistics network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
ExLInput0	Input	-	-	None	None	$5 \times 5 \times 1024$
ExLInput1	Input	-	-	None	None	128
ExLConcat0	Tile + Concatenation: ExLInput0 + ExLInput1	-	-	None	None	$5 \times 5 \times 1152$
ExLConv0	Convolutional	1×1	1×1	ReLU	None	$5 \times 5 \times 512$
ExLConv1	Convolutional	1×1	1×1	ReLU	None	$5 \times 5 \times 512$
ExLOutput0	Convolutional	1×1	1×1	None	None	$5 \times 5 \times 1$

A discriminator D_{ρ_X} is used to minimize the mutual information between the shared and exclusive representations. The discriminator architecture is implemented as a dense neural network which takes as input these representations (the outputs of the Output0 layer from the shared and exclusive representation encoders) as can be seen in Table 13.

Table 13: Discriminator network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
DInput0	Input	-	-	None	None	64
DInput1	Input	-	-	None	None	64
DConcat0	Concatenation	-	-	None	None	128
	DInput0 + DInput1	-	-			
DDense0	Dense	-	-	ReLU	None	1000
DDense1	Dense	-	-	ReLU	None	200
DOutput0	Dense	-	-	None	None	1

Classification To perform classification using the learned representations, we use Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ during 10000 iterations. We use a batch size of 64 images at each iteration. As previously, we use the classifier architecture shown in Table 7 with $z_{dim} = 64$. The first layer is composed of $C = 32$ neurons while the next layers are composed of N neurones corresponding to number of classification classes ($N = 10$ to classify the floor color, wall color and the object color, $N = 8$ to classify the object scale, $N = 4$ to classify the object shape and $N = 15$ to classify the scene orientation). We use the same architecture to train a classifier using the disentangled representations from the models proposed by Gonzalez-Garcia et al. [11] (their model uses different representation dimensions: $z_{dim} = 8 \times 8 \times 512$ when learning from the shared representations and $z_{dim} = 128$ when learning from the exclusive representations) and Jha et al. [18].

1.3 IAM experiment

Dataset creation In order to perform our disentanglement experiments, we use a subset of the IAM dataset. We select 150 words for each writer from the top 50 writers resulting in a subset of 8750 images. This subset is split into two parts: a) 6711 images to learn the shared and exclusive representations of our model and b) 2039 images to train the classifiers. These 2039 images are in turn split into train/test subsets: a) 1427 images to train the classifiers and b) 612 to test and compute the classification accuracy. Each of these images corresponds to one of 100 possible word classes.

All the images are binarized using the provided threshold and padded with ones to have a shape of $64 \times 256 \times 1$.

Shared representation learning In order to learn the shared representation, we use Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ during 4150 iterations. We use a batch size of 64 image pairs of size $64 \times 256 \times 1$ at each iteration. Concerning the constant coefficients to weight the objective function $\mathcal{L}^{shared}$, we use $\alpha^{sh} = 0.5$, $\beta^{sh} = 1.0$ and $\gamma = 0.01$. The size of the shared representation is 64.

During this stage, the shared representation encoder and the statistics networks are learned. The shared representation encoders $E^{sh}_{\psi_X}$ is implemented as convolutional neural networks described in Table 14 with $z_{dim} = 64$. The architecture of the global statistics network $T^{sh}_{\theta_X}$ is shown in Table 15. The architecture of the local statistics network $T^{sh}_{\phi_X}$ is shown in Table 16. The global and local statistics network share some layers with the shared representation encoder. The statistics networks take as input:

– ShGInput0/ShLInput0: the output of the Conv3 layer from the shared representation encoder
– ShGInput1/ShLInput1: the output of the Output0 layer from the shared representation encoder

Table 14: Representation encoder architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
Input0	Input	-	-	None	None	$64 \times 256 \times 1$
Conv0	Convolutional	4×4	1×1	LeakyReLU	None	$61 \times 253 \times 64$
Conv1	Convolutional	4×4	2×2	LeakyReLU	BatchNorm	$29 \times 125 \times 128$
Conv2	Convolutional	4×4	2×2	LeakyReLU	BatchNorm	$13 \times 61 \times 256$
Conv3	Convolutional	4×4	2×2	LeakyReLU	BatchNorm	$5 \times 29 \times 512$
Flat0	Flatten	-	-	None	None	74240
Output0	Dense	-	-	None	None	z_{dim}

Table 15: Global statistics network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
ShGInput0	Input	-	-	None	None	$5 \times 29 \times 512$
ShGConv0	Convolutional	3×3	1×1	ReLU	None	$3 \times 27 \times 64$
ShGConv1	Convolutional	3×3	1×1	None	None	$1 \times 25 \times 32$
ShGFlat0	Flatten	-	-	None	None	800
ShGInput1	Input	-	-	None	None	64
ShGConcat0	Concatenation	-	-	None	None	864
ShGDense0	Dense	-	-	ReLU	None	512
ShGDense1	Dense	-	-	ReLU	None	512
ShGOutput0	Dense	-	-	None	None	1

Table 16: Local statistics network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
ShLInput0	Input	-	-	None	None	$5 \times 29 \times 512$
ShLInput1	Input	-	-	None	None	64
ShLConcat0	Concatenation	-	-	None	None	$5 \times 29 \times 576$
ShLConv0	Convolutional	1×1	1×1	ReLU	None	$5 \times 29 \times 512$
ShLConv1	Convolutional	1×1	1×1	ReLU	None	$5 \times 29 \times 512$
ShLOutput0	Convolutional	1×1	1×1	None	None	$5 \times 29 \times 1$

Exclusive representation learning To learn the exclusive representation, we use Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_1 = 0.999$ during 4150 iteration. We use a batch size of 64 image pairs at each iteration. Concerning the constant coefficients to weight the objective function $\mathcal{L}^{exclusive}$, we use $\alpha^{ex} = 0.5$, $\beta^{ex} = 1.0$ and $\lambda_{adv} = 0.01$. The size of the exclusive representation is 64.

The exclusive representation encoder $E^{ex}_{\omega_X}$ is defined by a convolutional neural networks using the architecture shown in Table 14 with $z_{dim} = 64$. The architecture of the global statistics network $T^{ex}_{\theta_X}$ is shown in Table 17. The architecture of the local statistics network $T^{ex}_{\phi_X}$ is shown in Table 18. The statistics networks take two inputs to compute the mutual information:

- ExGInput0/ExLInput0: the concatenation of the outputs of the Conv3 layer from the shared and exclusive representation encoder
- ExGInput1/ExLInput1: the concatenation of the representations Output0 from the shared and exclusive representation encoder

The discriminator architecture is implemented as a dense neural network which takes as input the shared and exclusive representations as can be seen in Table 19.

Classification To perform classification using the learned representations, we use Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ during 10000 iterations. We use a batch size of 64 images at each iteration. As previously, we use the classifier architecture shown in Table 7 with $z_{dim} = 64$. The first layer is composed of $C = 64$ neurons while the next layers are composed of N neurones corresponding to number of classification classes ($N = 50$ to perform writer classification and $N = 100$ to perform word classification).

As the models of Gonzalez-Garcia et al. [11] and Jha et al. [18] fails to converge we train a VAE model to provide a comparison. To learn the representations, we use Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ during 50000 iterations. For fair comparison, the VAE model is trained using the same encoder architecture of our model (see Table 14) and the decoder architecture shown in Table 20. Using the previous setting, we train

Table 17: Global statistics network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
ExGInput0	Input	-	-	None	None	$5 \times 29 \times 1024$
ExGConv0	Convolutional	3×3	1×1	ReLU	None	$3 \times 27 \times 64$
ExGConv1	Convolutional	3×3	1×1	None	None	$1 \times 25 \times 32$
ExGFlat0	Flatten	-	-	None	None	800
ExGInput1	Input	-	-	None	None	128
ExGConcat0	Concatenation: ExGFlat0 + ExGInput1	-	-	None	None	928
ExGDense0	Dense	-	-	ReLU	None	512
ExGDense1	Dense	-	-	ReLU	None	512
ExGOutput0	Dense	-	-	None	None	1

Table 18: Local statistics network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
ExLInput0	Input	-	-	None	None	$5 \times 29 \times 1024$
ExLInput1	Input	-	-	None	None	128
ExLConcat0	Tile + Concatenation: ExLInput0 + ExLInput1	-	-	None	None	$5 \times 29 \times 1152$
ExLConv0	Convolutional	1×1	1×1	ReLU	None	$5 \times 29 \times 512$
ExLConv1	Convolutional	1×1	1×1	ReLU	None	$5 \times 29 \times 512$
ExLOutput0	Convolutional	1×1	1×1	None	None	$5 \times 29 \times 1$

a classifier using the VAE representations but using a different representation dimension: $z_{dim} = 128$.

Table 19: Discriminator network architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
DInput0	Input	-	-	None	None	64
DInput1	Input	-	-	None	None	64
DConcat0	Concatenation	-	-	None	None	128
	DInput0 + DInput1	-	-			
DDense0	Dense	-	-	ReLU	None	1000
DDense1	Dense	-	-	ReLU	None	200
DOutput0	Dense	-	-	None	None	1

Table 20: VAE decoder architecture.

ID	Layer	Kernel	Stride	Activation	Normalization	Output
VInput0	Input	-	-	None	None	128
VDense0	Dense	-	-	None	None	131072
VReshape0	Reshape	-	-	LeakyReLU	BatchNorm	$8 \times 32 \times 512$
VDeconv0	Deconvolutional	4×4	2×2	LeakyReLU	BatchNorm	$16 \times 64 \times 256$
VDeconv1	Deconvolutional	4×4	2×2	LeakyReLU	BatchNorm	$32 \times 128 \times 128$
VDeconv2	Deconvolutional	4×4	2×2	LeakyReLU	BatchNorm	$64 \times 256 \times 64$
VDeconv3	Deconvolutional	4×4	1×1	None	None	$64 \times 256 \times 1$

1.4 Sentinel-2 experiment

Dataset creation The Sentinel-2 mission provides optical imagery of the Earth's surface at high spatial resolution. Optical images are acquired at 13 spectral bands using different spatial resolutions. L-1C processing is applied to optical images. To organize the data acquired by the mission, the Earth surface is divided into square tiles of approximately 100 km on each side.

In this paper, the following tiles are used to create our Sentinel-2 dataset: 11SLT, 19KDQ, 22LBP, 30SYJ, 30TXP, 36RYV, 37RDP, 37SCA, 39RUK, 41TLM, 31TCH, 20JPS, 31TCJ, 36RUU, 21HUB, 55HDU, 51JYN, 50TMK, 44QPE, 22KGU, 34HBH, 21HTU, 19HCT, 11SPS, 33SVD, 39RUM, 29TMF, 12SUC, 12SUF, 39QVE, 22LBR, 33TVF, 30SWC, 34SGH, 39SWV, 23KPQ, 31TFJ, 30STF, 51RUQ, 32SLB, 20GLT, 37MBU. For each tile, we extracted 12 optical images (cloud coverage tolerance of 2%) between 2016 and 2018 keeping a regular time-step between images.

For each tile, 100 non-overlapping patches of size 512×512 are extracted from these images. Images are composed of the RBGI bands (Red (band 4), Green (band 3), Blue (band 2) and Near infrared (band 8) bands) which correspond to the bands at 10m spatial resolution. The training dataset size is around 100GB.

Shared representation learning To learn the shared representation, we use Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ during 200000 iterations. We use a batch size of 64 image pairs of size $64 \times 64 \times 4$ at each iteration. Pairs of images of size $64 \times 64 \times 4$ are extracted from our dataset by selecting patches from the same region but acquired at different times. Concerning the constant coefficients to weight the objective function $\mathcal{L}^{shared}$, we use $\alpha^{sh} = 0.5$, $\beta^{sh} = 1.0$ and $\gamma = 0.1$.

We use the same model architecture employed to learn the representations of the 3D Shapes dataset. The shared representation encoder $E^{sh}_{\psi_X}$ is defined by a convolutional neural network described in Table 8. The architectures of the global statistics network $T^{sh}_{\theta_X}$ and the local statistics network $T^{sh}_{\phi_X}$ are shown in Tables 9 and 10, respectively.

Classification To perform classification using the learned representations, we use Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ during 10000 iterations. We use a batch size of 64 images of size $64 \times 64 \times 4$ from the EuroSAT dataset at each iteration. As previously, we use the classifier architecture shown in Table 7. The first layer is composed of $C = 32$ neurons while the next layers are composed of $N = 10$ neurones corresponding to number of classification classes (residential area, sea, river, highway, etc.). The confusion matrix can be observed in Table 21.

Table 21: Confusion matrix. Classes: Annual crop:0, Forest:1, Herbaceous:2, Highway:3, Industrial:4, Pasture:5, Permanent crop:6, Residential:7, River:8, Sea/Lake:9.

	0	1	2	3	4	5	6	7	8	9
0	0.94	0.00	0.01	0.01	0.00	0.02	0.02	0.00	0.00	0.00
1	0.00	0.99	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
2	0.00	0.01	0.89	0.00	0.00	0.01	0.07	0.00	0.00	0.00
3	0.02	0.00	0.02	0.88	0.01	0.01	0.03	0.01	0.03	0.00
4	0.00	0.00	0.01	0.01	0.97	0.00	0.00	0.01	0.00	0.00
5	0.01	0.01	0.04	0.01	0.00	0.91	0.02	0.00	0.00	0.00
6	0.04	0.00	0.11	0.02	0.01	0.02	0.80	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.98	0.00	0.00
8	0.00	0.00	0.01	0.05	0.00	0.00	0.00	0.00	0.93	0.00
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

2 Additional results

2.1 3D Shapes disentanglement experiment

In Figure 1, the learning curves of the classifiers of the 3D Shapes attribute are displayed. As can be seen, using the shared representation of our model as input to train a classifier on a shared attribute is useful achieving a high accuracy while it provides no information when training a classifier on an exclusive attribute achieving a low accuracy. Using the exclusive representation shows a similar behavior.

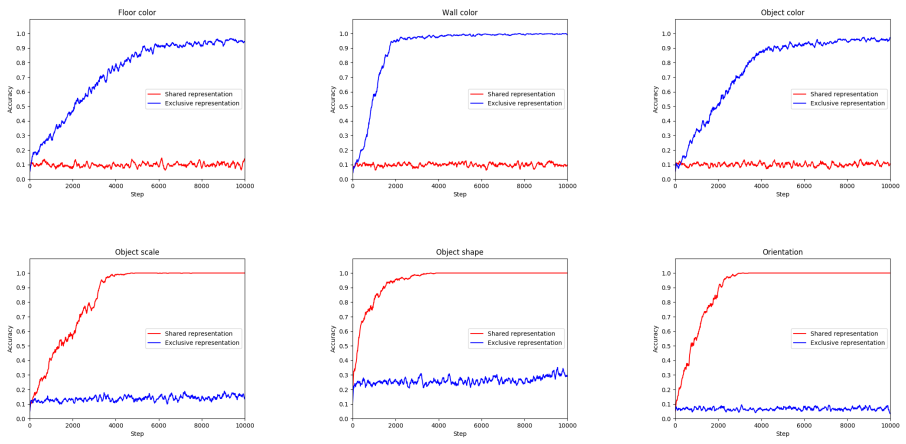


Fig. 1: Learning curves of the 3D Shapes factors of variation using the shared representations (red curves) and the exclusive representations (blue curves). Results are plotted in terms of factor accuracy over training steps. (a) Floor color (exclusive factor); (b) Wall color (exclusive factor); (c) Object color (exclusive factor); (d) Object scale (shared factor); (e) Object shape (shared factor); (f) Scene orientation (shared factor).

2.2 3D Shapes ablation study

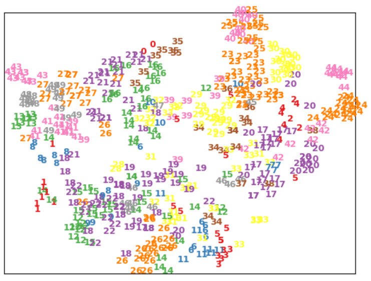
We include the results from the ablation experiment in the shared representation learning stage for the 3D Shapes in Table 22. Similar to the colored-MNIST and IAM dataset cases, switching the shared representations is an important element as it enforces to keep the common information and remove the exclusive information in the shared representation. As shown, when the shared representations are not switched, the accuracy on exclusive attributes drastically increases (for instance, the floor color accuracy increases from 9.96% to 93.09%) which means the presence of exclusive information in the shared representation.

Table 22: 3D Shapes ablation study. Accuracy using S_X.

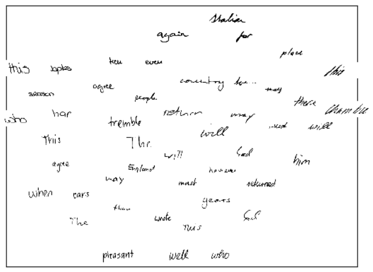
Method	Floor color	Wall color	Object color	Object scale	Object shape	Scene Orientation
Ideal feature S_X	10.00%	10.00%	10.00%	100.00%	100.00%	100.00%
Baseline	9.96%	10.08%	9.95%	99.99%	99.99%	99.99%
Baseline (non-SSR)	93.09%	99.66%	95.89%	45.65%	59.55%	29.34%
Baseline ($\gamma = 0$)	10.75%	13.21%	12.53%	99.95%	99.99%	99.99%
Baseline ($\alpha^{\text{sh}} = 0$)	10.06%	10.14%	10.56%	99.95%	99.99%	99.99%
Baseline ($\beta^{\text{sh}} = 0$)	9.97%	10.32%	9.99%	98.99%	98.98%	99.84%

2.3 IAM data organization

For the IAM dataset, in order to get the intuition of how the data is organized in the shared representation space, we perform a dimensionality reduction via the t-SNE algorithm. In Figure 2a, we plot the writer label corresponding to each reduced shared representation. As shown, the shared representations are organized in clusters of writers. Moreover, the shared representations seem to be organized in accordance with the writer style as can be seen in Figure 2b.



(a)



(b)

Fig. 2: A t-SNE visualization of the shared representations S_X. a) Displaying the writer label for 50 writers; b) Displaying the corresponding IAM image.

2.4 Mutual information distance

We train a model to learn the shared representations of our Sentinel-2 dataset using the setting described in Section 1.4 but using image patches of size $9 \times 9 \times 4$. In Figure 3, we show some additional examples of how the *crossed mutual information* objective can be used to measure the distance between the pixels of an image.

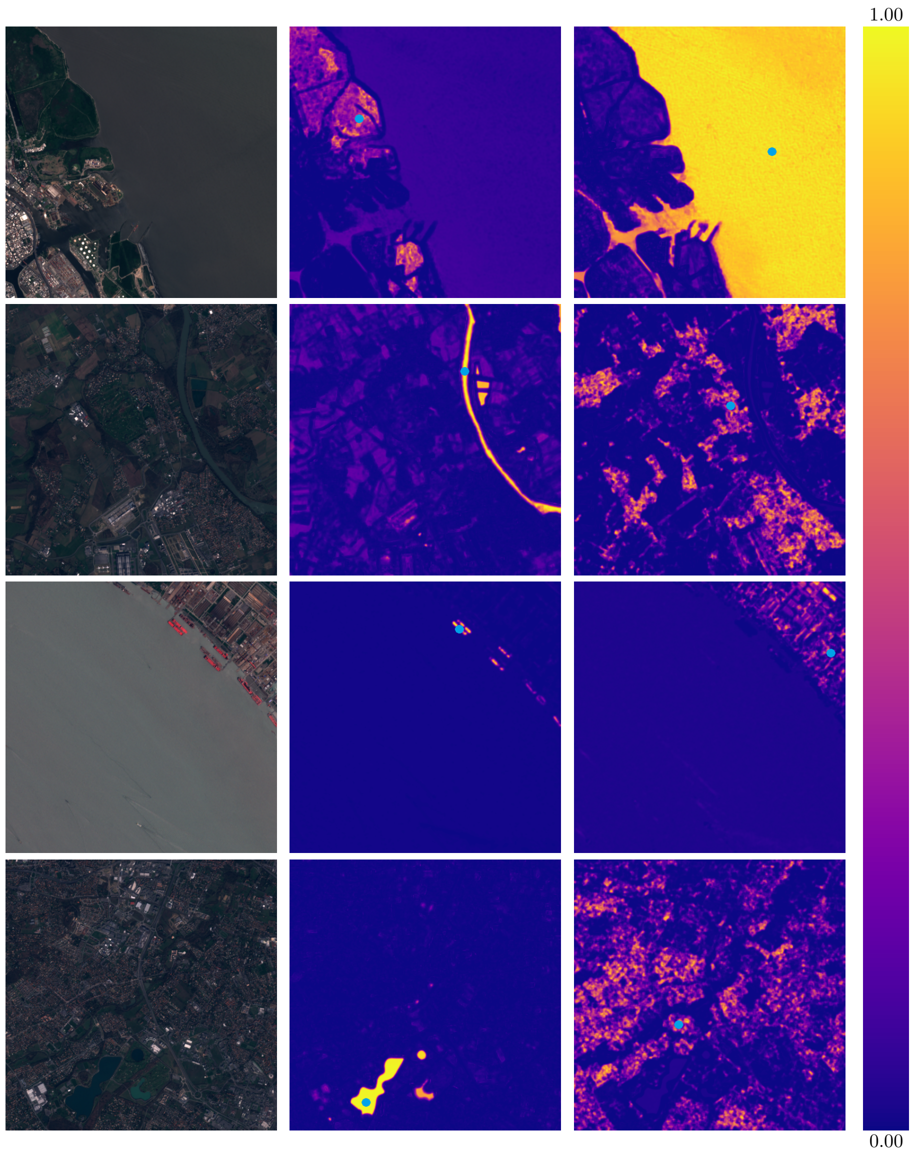


Fig. 3: Pixel similarity based on mutual information. The mutual information is computed between a given pixel (blue point) and the remaining image pixels.