Autoencoder for Synthetic to Real Generalization: From Simple to More Complex Scenes

Supplementary Material

S1. Dataset details

If not specified otherwise, all images have been centre cropped and resized to 128 pixels.

A. MPI3D

We used the synthetic realistic and toy images as well as the real images, but we restricted the dataset to use only large objects, since even for humans the small objects cannot always be distinguished reliably. The dataset can be downloaded from Github.

B. SVIRO

We only used the grayscale training images from the SVIRO dataset. We considered everyday objects as background and removed all images containing empty child and infant seats. For the classification evaluation we used all the images from all the different vehicles, but we used training images only. Occupancy classification is performed on the entire image such that all three seats need to be classified simultaneously. Since four classes are available per seat (empty, infant seat, child seat and adult) this results in a total of $4^3 = 64$ classes. The dataset can be downloaded from our website.

C. SVIRO-Illumination

For the classification evaluation we used all the training and test images from all the different vehicles. We used all the variations per scenes, i.e. not just a single variation per illumination variation. The dataset can be downloaded from our website.

D. Our newly released dataset

We created 2938 training and 2981 test sceneries where each scenery is rendered with 10 different backgrounds out of a pool of 450 backgrounds. The background and the corresponding illumination conditions were defined using high dynamic range images (HDRI). The latter were downloaded from https://hdrihaven.com/. Human models, child seats and infant seats were randomly placed as if they were located inside a vehicle, but no vehicle is visible. There are four possible classes for each seat position (empty, infant seat, child seat and adult) leading to a total of $4^3 = 64$ classes for the whole image. We created randomly 172 adults using http://www.makehumancommunity.org/ and we used 6 child seats and 7 infant seats which were textured using randomly one out of five textures. Since the dataset is synthetic, there are no consent and privacy concerns. The dataset can be downloaded from our website. Examples are visualized in Fig. S1. We noticed that a larger number of different human models increases the transferability to real images.

Fig. S1. Examples of sceneries with different backgrounds from the newly generated dataset.

E. TICaM

We used all training and test images and also flipped the images for the classification evaluation. This was done, because otherwise the class variability is quite low and there is a strong bias towards people sitting on the right driver seat. Moreover, the steering wheel would always be placed at the same right position. We also needed to perform some pre-processing to make the real TICaM images compatible with the synthetic images. First, we adapted the labels: we extracted the labels for the left and right seat from the filename. The file name is split at the character _ after which the third (right seat) and ninth (left seat) part is responsible for the class definition. If the latter was a 0 or contained an o, we kept it as a 0. If it contained a p, it was changed into a 3. We changed the value to 2 if it was one of the child seats $s03$, $s13$, $s04$, $s14$ or the variation $g00$ for the child seats $s01$, $s11$, $s02$, $s12$. In all other cases, it was transformed to a 1, i.e. for the child seats $s05$, $s15$, $s06$, $s16$ and variations $g01$ $g11$ $g10$ for

$s01$, $s11$, $s02$, $s12$. Second, the illumination of the images was normalized using a histogram equalization. After that the images were cropped at height position 120 with height 300 and left position 106 with width 300. Finally, the images were resized to 128 pixels. The dataset can be downloaded from their website.

S2. Training details

All our experiments were conducted using PyTorch 1.8. Predefined and models pre-trained on Imagenet were taken from torchvision 0.9.0. The code implementation can be found on Github.

We used the same hyperparameters for all training experiments and for all autoencoder and classification models respectively. We used the AdamW optimizer with a learning rate of $1e-4$ and weight decay of $1e-5$. We used a batch size of 64 and the only augmentation performed was a random horizontal flip. All models were trained for 100 epochs on MPI3D and 250 epochs for the other datasets.

Both the extractor autoencoder and the classification models used the same layer for extracting the features from the pre-trained models. In both cases and for all pre-trained models we used layer level -3 in our implementation: those features were used to fine-tune the rest of the pre-trained classification model or to train from scratch our added autoencoder layers. In all cases, we interpolated the input images to be of size 224 and copied the single grayscale image channel twice along the channel dimension.

For the autoencoder training, we used the structural similarity index measure (SSIM) [1] or the binary cross entropy (BCE) to measure the error between reconstruction and target image. We used PyTorch MS-SSIM [2] to compute the SSIM. In Eq. 5 we chose $\alpha = 1$ and $\beta = 1$. We used a latent space dimension of 64 for all models trained on the vehicle interior and a latent space dimension of 10 for the MPI3D dataset. Further, we used the ReLU activation function. In case of a triplet loss, we used the swap parameter of Pytorch to make the negative mining more challenging [3]. As a positive sample, we selected an image of a different scenery of the same class, i.e. the same objects are at the same seat position. For the negative sample we selected a scenery which differs in a single seat position and we did not allow sceneries with empty seats only. In case the partially impossible reconstruction loss was used, the target images for the positive and negative samples are chosen to be partially impossible as well.

A. Model details

The autoencoder model architecture details are provided in Table S1, S3 and S4. Regarding the pre-trained models, we used the output of the following layers to retrieve the extracted features. The notations is according to the torchvision model definitions:

```
VGG-11
(16): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(17): ReLU(inplace=True)

Resnet-50
(5): Bottleneck(
  (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
)

Densenet-121
(denselayer24): _DenseLayer(
  (norm1): BatchNorm2d(992, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu1): ReLU(inplace=True)
  (conv1): Conv2d(992, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu2): ReLU(inplace=True)
  (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
)
```

TABLE S1
MODEL ARCHITECTURE FOR AE, VAE AND β-VAE ON MPI3D

Encoder	Decoder
Input: 3 x 64 x 64	Input: 10
Conv, 4x4, 32, padding 1, stride 2 ReLU	FC, 256, bias True ReLU
Conv, 4x4, 32, padding 1, stride 2 ReLU	FC, 1024, bias True ReLU
Conv, 4x4, 64, padding 1, stride 2 ReLU	ConvTranspose, 4x4, 64, padding 1, stride 2 ReLU
Conv, 4x4, 64, padding 1, stride 2 ReLU	ConvTranspose, 4x4, 32, padding 1, stride 2 ReLU
FC, 256, bias True ReLU	ConvTranspose, 4x4, 32, padding 1, stride 2 ReLU
FC, 10, bias True (twice in case of VAE)	ConvTranspose, 4x4, 3, padding 1, stride 2 Sigmoid

TABLE S2
MODEL ARCHITECTURE FOR FACTORVAE ON MPI3D. THE MODEL IS EXACTLY THE SAME AS THE VAE MODEL AND USES THE FOLLOWING DISCRIMINATOR.

Discriminator
Input: 10
FC, 1000, bias True LeakyReLU(0.2)
FC, 1000, bias True LeakyReLU(0.2)
FC, 1000, bias True LeakyReLU(0.2)
FC, 1000, bias True LeakyReLU(0.2)
FC, 1000, bias True LeakyReLU(0.2)
FC, 2, bias True

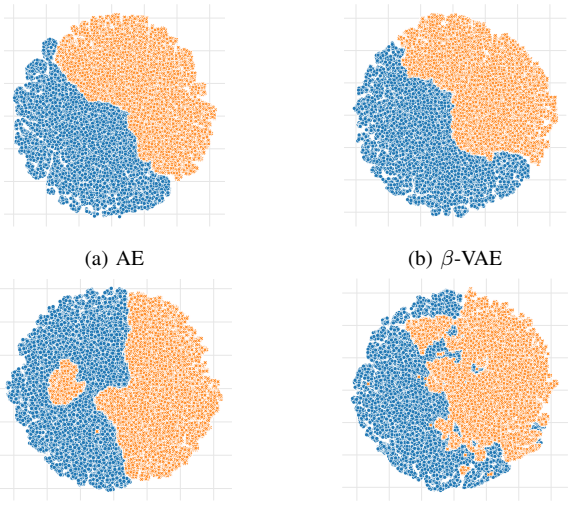
TABLE S3
MODEL ARCHITECTURE FOR E-AE ON MPI3D. THE EXTRACTOR IS FIXED DURING TRAINING.

Extractor + Summarizer + Encoder	Decoder
Input: 3 x 224 x 224	Input: 10
VGG-11 extractor after 7th Conv layer + ReLU Avgpool, 2x2, stride 2, padding 0	FC, 256, bias True ReLU
Conv, 4x4, 256, padding 0, stride 1 ReLU	FC, 1024, bias True ReLU
FC, 256, bias True ReLU	ConvTranspose, 4x4, 64, padding 1, stride 2 ReLU
FC, 10, bias True	ConvTranspose, 4x4, 32, padding 1, stride 2 ReLU
	ConvTranspose, 4x4, 32, padding 1, stride 2 ReLU
	ConvTranspose, 4x4, 3, padding 1, stride 2 Sigmoid

TABLE S4
MODEL ARCHITECTURE FOR E-AE ON SVIRO, SVIRO-ILLUMINATION AND TICAM. C IS THE CHANNEL DIMENSION WHICH IS 1 FOR ALL DATASETS. THE EXTRACTOR IS FIXED DURING TRAINING.

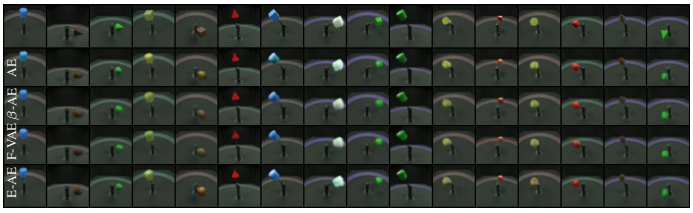
Extractor + Summarizer + Encoder	Decoder
Input: C x 224 x 224	Input: 64
VGG-11 extractor after 7th Conv layer + ReLU Avgpool, 2x2, stride 2, padding 0	FC, 256, bias True ReLU
Conv, 4x4, 256, padding 0, stride 1 ReLU	FC, 4096, bias True ReLU
FC, 256, bias True ReLU	ConvTranspose, 4x4, 64, padding 1, stride 2 ReLU
FC, 64, bias True	ConvTranspose, 4x4, 32, padding 1, stride 2 ReLU
	ConvTranspose, 4x4, 32, padding 1, stride 2 ReLU
	ConvTranspose, 4x4, C, padding 1, stride 2 Sigmoid

S3. ADDITIONAL RESULTS

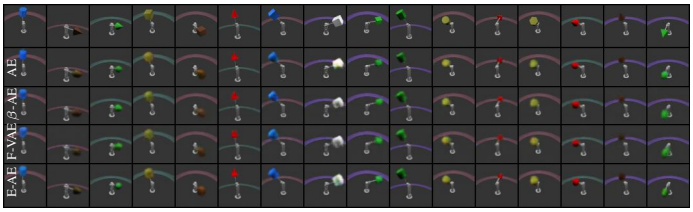


(a) AE (b) β-VAE (c) FactorVAE (d) E-AE

Fig. S2. t-SNE projection of the 10 dimensional latent space representation of the toy training (blue circle) together with the real (orange cross) images. Autoencoder (AE), β Variational Autoencoder (β-VAE), FactorVAE and Extractor Autoencoder (E-AE). When trained on toy images, our extractor approach performs still best although the synthetic-real distributions are not as overlapped as if trained on realistic images.



(a) Reconstruction of training data when being trained on realistic data.



(b) Reconstruction of training data when being trained on toy data.

Fig. S3. Reconstruction of realistic and toy training data for different autoencoders: Autoencoder (AE), β Variational Autoencoder (β-VAE), FactorVAE (F-VAE) and Extractor Autoencoder (E-AE).

TABLE S5

WE REPORT THE SSIM AND LPIPS [4] NORM BETWEEN THE RECONSTRUCTIONS OF THE REAL IMAGES (UNKNOWN) AND THE CORRESPONDING SYNTHETIC TRAINING IMAGES (REALISTIC OR TOY). WE REPORT THE MEAN OF THE NORMS ACROSS THE ENTIRE REDUCED DATASET: FOR SSIM LARGER ↑ AND FOR LPIPS SMALLER ↓ IS BETTER. E-AE PERFORMS BEST. SOME MODELS USED SSIM, OTHERS BCE DURING TRAINING.

Trained on	Model	Variant	SSIM ↑	LPIPS ↓
Toy	AE	BCE	0.559	0.412
Toy	AE	SSIM	0.558	0.347
Toy	E-AE (ours)	BCE	**0.896**	**0.095**
Toy	E-AE (ours)	SSIM	**0.899**	**0.103**
Toy	VAE	BCE	0.497	0.338
Toy	β-VAE	BCE, $\beta = 4$	0.527	0.311
Toy	β-VAE	BCE, $\beta = 8$	0.709	0.258
Toy	FactorVAE	BCE, $\gamma = 10$	0.660	0.262
Toy	FactorVAE	BCE, $\gamma = 30$	0.710	0.344
Toy	FactorVAE	BCE, $\gamma = 50$	0.712	0.221
Realistic	AE	BCE	0.841	0.211
Realistic	AE	SSIM	0.832	0.195
Realistic	E-AE (ours)	BCE	**0.917**	**0.071**
Realistic	E-AE (ours)	SSIM	**0.921**	**0.081**
Realistic	VAE	BCE	0.740	0.197
Realistic	β-VAE	BCE, $\beta = 4$	0.810	0.176
Realistic	β-VAE	BCE, $\beta = 8$	0.794	0.189
Realistic	FactorVAE	BCE, $\gamma = 10$	0.880	0.151
Realistic	FactorVAE	BCE, $\gamma = 30$	0.862	0.161
Realistic	FactorVAE	BCE, $\gamma = 50$	0.779	0.164

TABLE S6

FOR EACH EXPERIMENT, THE BEST PERFORMANCE (IN PERCENTAGE) ON REAL VEHICLE INTERIOR IMAGES (TICAM) ACROSS ALL EPOCHS IS TAKEN AND THEN THE MEAN AND MAXIMUM OF THOSE VALUES ACROSS ALL 10 RUNS IS REPORTED. FOR THE SAME BACKBONE MODEL EXTRACTOR, OUR APPROACH OUTPERFORMS THE VANILLA CLASSIFICATION MODELS SIGNIFICANTLY. THE MODEL WEIGHTS ACHIEVING THE MAXIMUM PERFORMANCE PER RUN ARE ALSO EVALUATED ON SVIRO WHERE THEY PERFORM BETTER AS WELL.

	Dataset	TICaM		SVIRO	
	Dataset size	13356		11959	
Model	Variant	Mean	Max	Mean	Max
VGG-11	Scratch	58.5 ± 4.0	64.6	65.6 ± 5.4	72.7
Resnet-50	Scratch	53.3 ± 3.5	60.4	56.4 ± 2.6	59.3
Densenet-121	Scratch	56.3 ± 5.5	62.1	68.8 ± 2.4	74.9
VGG-11	Pre-trained	75.5 ± 1.5	78.0	78.7 ± 2.9	84.0
Resnet-50	Pre-trained	78.1 ± 1.7	80.4	83.5 ± 2.7	88.1
Densenet-121	Pre-trained	72.2 ± 4.2	77.4	85.0 ± 2.3	88.0
VGG-11	E-TAE	76.7 ± 2.3	81.5	78.6 ± 2.6	82.3
Resnet-50	E-TAE	$\mathbf{83.8 \pm 1.3}$	**86.0**	85.8 ± 2.4	89.1
Densenet-121	E-TAE	78.5 ± 2.4	81.8	86.7 ± 1.3	88.2
VGG-11	I-E-TAE	79.7 ± 2.1	**82.2**	$\mathbf{80.9 \pm 4.0}$	**85.6**
Resnet-50	I-E-TAE	83.5 ± 1.3	85.6	89.2 ± 1.0	90.3
Densenet-121	I-E-TAE	77.2 ± 1.7	79.3	$\mathbf{90.4 \pm 1.3}$	92.1
VGG-11	II-E-TAE	$\mathbf{81.0 \pm 0.6}$	82.0	79.1 ± 3.9	84.8
Resnet-50	II-E-TAE	$\mathbf{83.7 \pm 0.5}$	84.5	$\mathbf{93.0 \pm 0.8}$	**94.1**
Densenet-121	II-E-TAE	$\mathbf{79.3 \pm 1.3}$	**81.5**	89.9 ± 1.8	**92.3**

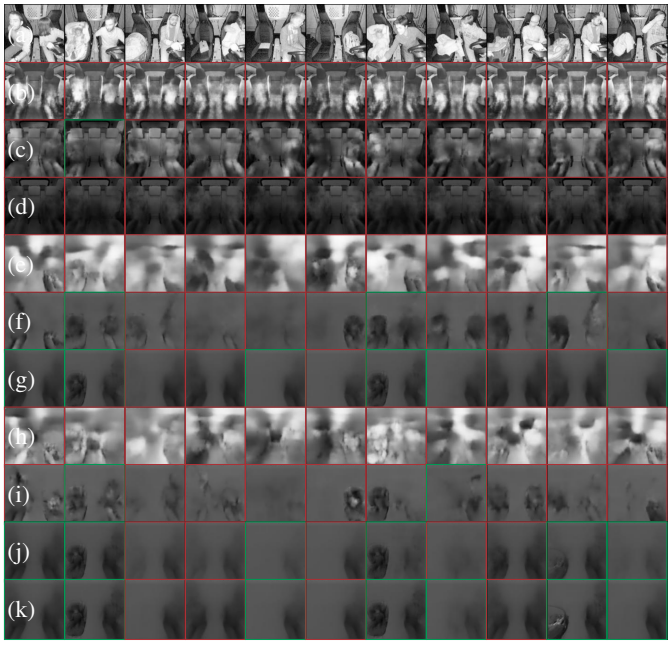
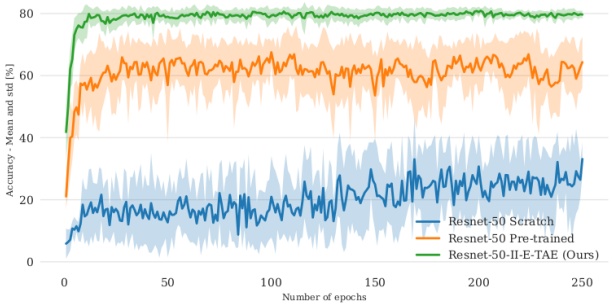
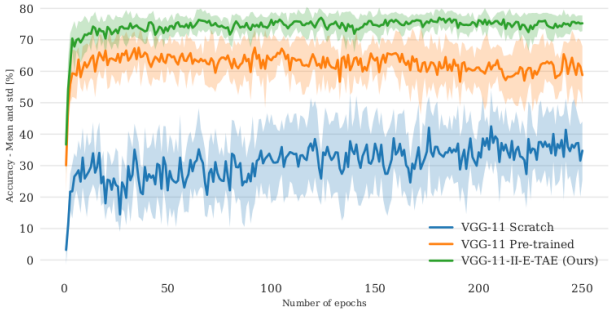


Fig. S4. Reconstruction results of unseen real data (a) from the TICaM dataset: (b) E-AE Trained on Tesla SVIRO, (c) E-AE Trained on Kodiaq SVIRO-Illumination , (d) I-E-AE Trained on Kodiaq SVIRO-Illumination , (e) E-AE, (f) I-E-AE, (g) II-E-AE, (h) E-TAE, (i) I-E-TAE, (j) II-E-TAE and (k) Nearest neighbour of (j). Examples (e)-(k) are all trained on our new dataset. A red (wrong) or green (correct) box highlights whether the semantics are preserved by the reconstruction.



(a) Resnet-50



(b) VGG-11

(c) Densenet-121

Fig. S5. Comparison of the training performance distribution for each epoch over 250 epochs. II-E-TAE is compared against training the corresponding extractor from scratch or fine-tuning the layers after the features which are used by the extractor in our autoencoder approach.

TABLE S7
DIFFERENT MODEL ARCHITECTURE VARIATIONS TRAINED ON MNIST. THEN DIFFERENT CLASSIFIERS WERE TRAINED ON THE LATENT SPACE REPRESENTATION OF THE TRAINING DATA AND EVALUATED ON REAL IMAGES OF DIGITS. MODELS WERE TRAINED FOR 20 EPOCHS USING A LATENT DIMENSION OF 64 AND MSE RECONSTRUCTION LOSS. SEE FIG. S6 FOR THE CORRESPONDING RECONSTRUCTION RESULTS AND INPUT IMAGES.

Model	KNN	RForest	SVM
AE	15.7	12.5	11.6
TAE	11.1	11.6	8.4
II-AE	27.8	20.2	23.6
II-TAE	21.8	17.9	23.9
E-AE	27.3	23.1	26.5
E-TAE	26.1	19.1	23.3
II-E-AE	**65.0**	61.9	**65.6**
II-E-TAE	64.1	**63.7**	63.7

Fig. S6. Reconstruction of real input images of digits by models trained on MNIST. Similar to the vehicle interior, the II-PIRL loss provides the best class preserving reconstructions. The latter is supported by the quantiative results in Table S7.

REFERENCES

[1] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger, "Improving unsupervised defect segmentation by applying structural similarity to autoencoders," *arXiv preprint arXiv:1807.02011*, 2018.

[2] F. Gongfan, "Pytorch ms-ssim," https://github.com/VainF/pytorch-msssim, 2019.

[3] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks." in *British Machine Vision Conference (BMVC)*, 2016.

[4] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.