Machine Vision and Applications (2025) 36:70
https://doi.org/10.1007/s00138-025-01689-6

RESEARCH

Traversing the subspace of adversarial patches

Jens Bayer[1,2] · Stefan Becker[1,2] · David Münch[1,2] · Michael Arens[1,2] · Jürgen Beyerer[1,2,3]

Received: 12 April 2024 / Revised: 21 December 2024 / Accepted: 6 April 2025

Abstract
Despite ongoing research on the topic of adversarial examples in deep learning for computer vision, some fundamentals of the nature of these attacks remain unclear. As the manifold hypothesis posits, high-dimensional data tends to be part of a low-dimensional manifold. To verify the thesis with adversarial patches–a special form of adversarial attack that can be used to fool object detectors in the physical world–this paper provides an analysis of a set of adversarial patches and investigates the reconstruction abilities of five different dimensionality reduction methods. Quantitatively, the performance of reconstructed patches in an attack setting is measured and the impact of sampled patches from the latent space during adversarial training is investigated. The evaluation is performed on two publicly available datasets for person detection. The results indicate that more sophisticated dimensionality reduction methods offer no advantages over a simple principal component analysis.

Keywords Adversarial attacks · Manifold learning · Object detection · Adversarial patches

1 Introduction

Adversarial patch attacks in deep learning for computer vision are a well-researched topic, yet, some fundamentals of the nature of these powerful attacks remain unclear. Due to the complexity of deep neural networks, where these patches are optimized on, there are no straightforward explanations why a generated pattern, appears the way it does. Moreover, there are no explainability methods like those used for deep neural networks that provide a human-understandable explanation of why these patches prevent a network from working properly. To further understand this high-dimensional data, this paper follows the manifold hypothesis [1]. For adversarial attacks, this states, that adversarial patterns are part of a lower-dimensional manifold. This paper builds on our recently published work "Eigenpatches– Adversarial Patches from Principal Components"[2]. By applying a principal component analysis (PCA) on a set of trained adversarial patches, it could already be shown that linear combinations of Eigenpatches can be used to successfully attack the investigated YOLOv7 object detector [3]. However, a PCA does not explicitly model the manifold structure of the data and does not take the non-linear relationships or intrinsic geometry that may exist in the data into account. Consequently, this paper investigates multiple manifold learning techniques and evaluates them in an attack scenario and when used as augmentation tool in adversarial training.

To be more specific, the contributions of this paper are:

(i) A more in-depth analysis of the crafted set of adversarial patches by analyzing the impact on the object detector and the detector backbone.
(ii) An evaluation of the performance of patches sampled from different low-dimensional manifolds and further analyze their generalization ability by evaluating them across varying detection models and datasets.
(iii) An evaluation of the impact of the usage of sampled adversarial patches from those manifolds when used in adversarial training.

The three conducted sets of experiments support the manifold hypothesis and show, that commonalities of the investigated set of adversarial patches could be captured successfully. Yet, despite the high-quality reconstructions,

✉ Jens Bayer
jens.bayer@iosb.fraunhofer.de

[1] Fraunhofer IOSB, Karlsruhe, Germany

[2] Fraunhofer Center for Machine Learning, Karlsruhe, Germany

[3] Karlsruhe Institute of Technology, Karlsruhe, Germany

Published online: 25 April 2025

the performance of the reconstructed patches remains far behind that of the original patches. Nonetheless, the reconstructions are sufficient to harden an object detector in adversarial training.

The structure of the paper is as follows: In Sect. 2, the similarities and distinctions between related work and this paper are presented. Section 3 covers the fundamentals of Eigenpatches and presents the alternative manifold learning methods that are evaluated: Isomap, locally linear embedding, convolutional autoencoder, and conditional variational autoencoder. The experiments and evaluation are described in Sect. 4. A discussion, followed by a brief conclusion and outlook, is given in Sect. 5.

2 Related work

Despite being an active research area, the focus of analyzing adversarial patterns is rarely [4] on object detector methods and adversarial patches. Instead, the investigations are performed mostly on image classifiers and attacks that induce high-frequent noise across the whole image [5–11]. Therefore only some selected works on analyzing adversarial attack patterns and sampling from low-dimensional embeddings are presented. For an overview of the different approaches to adversarial attacks, we refer to these surveys [12–15].

Wang et al. [6] propose a fast black-box adversarial attack that identifies key differences between different classes using a PCA. The principal components are then used to manipulate a sample into either a target class or the nearest other class.

Similarly, yet different, *Energy Attack* from Shi et al. [7] leverages PCA to obtain the energy distribution of perturbations generated by white-box attacks on a surrogate model. This transfer-based black-box adversarial attack samples patches according to the energy distribution, tiles them, and applies them to the target image. The extracted patches are high-frequent noise and are used to attack image classifiers. Despite the name, they should therefore not be confused with adversarial patches that are used to attack object detectors in a physical world attack.

Another approach uses the singular value decomposition and is presented by Weng et al. [10]. The authors combine the top-1 decomposed singular value-associated features for computing the output logits with the original logits, used to optimize adversarial examples. This results in an improvement in the transferability of the attacks.

Regarding the subspace of adversarial examples, researches have explored different methods to estimate the dimensionality of the space of adversarial inputs.

Dohmatob et al. [8], for example, investigate the vulnerability of neural networks to black-box attacks, specifically examining low-dimensional adversarial perturbations. They found that adversarial perturbations are likely to exist in low-dimensional subspaces that are much smaller than the dimension of the image space, supporting the manifold hypothesis.

An explicit estimation of the dimensionality of shared adversarial subspaces of, e.g., two fully connected networks trained on two different datasets, is presented in [5]. By examining untargeted misclassification attacks they demonstrate that manipulating a data point to cross a model's decision boundary is likely to result in similar performance degradation when applied to other models.

Tarchoun et al. [4] recently studied adversarial patches from an information theory perspective, measuring the entropy of random crops of the patches. Their findings indicate that the mean entropy of adversarial patches is higher than in natural images. Based on these results, they developed a defense mechanism against adversarial patches.

Moreover, theoretical limits on the susceptibility of classifiers to adversarial attacks are demonstrated by Shafahi et al. [9] using a unit sphere and unit cube. They suggest that these bounds may be potentially bypassed by employing extremely large values for the class densitiy functions. Furthermore, their findings suggest that the fundamental limits of adversarial training for specific datasets with complex image classes in high-dimensional spaces are far worse than one expects.

Godfrey et al. [16] conducted further research on the relationship between adversarial vulnerability and the number of perturbed dimensions. Their findings support the hypothesis that adversarial examples are a result of the locally linear behavior of neural networks with high-dimensional input spaces.

While the related works mainly focus on adversarial examples for image classifiers, this work explores commonalities of adversarial patch attacks against object detectors. The investigated dimensionality reduction methods are evaluated in an attack setting and also tested if sampled patches can be used in adversarial training.

3 Processing pipeline

The following section covers the necessary theoretical backgrounds of the investigated dimensionality reduction methods and manifold learning techniques. To be more precise, Eigenpatches, Isomap, locally linear embedding and autoencoders are investigated. The general processing pipeline is depicted in Fig. 1: The investigated methods are used to embed a set of given adversarial patches onto a manifold.

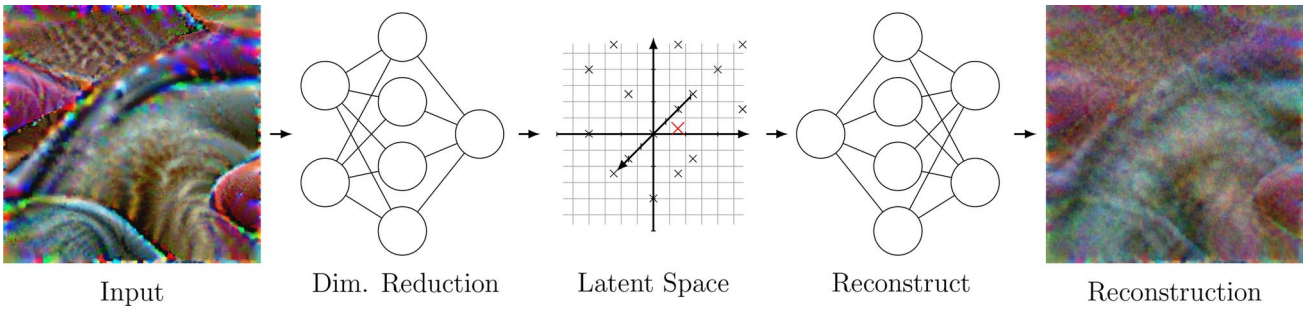

Fig. 1 A schematic overview on the reconstruction process. In general, the input (left) is a prime patch, which will be encoded by a dimensionality reduction method. Based on the encoding in the latent space (middle), the input is reconstructed (right)

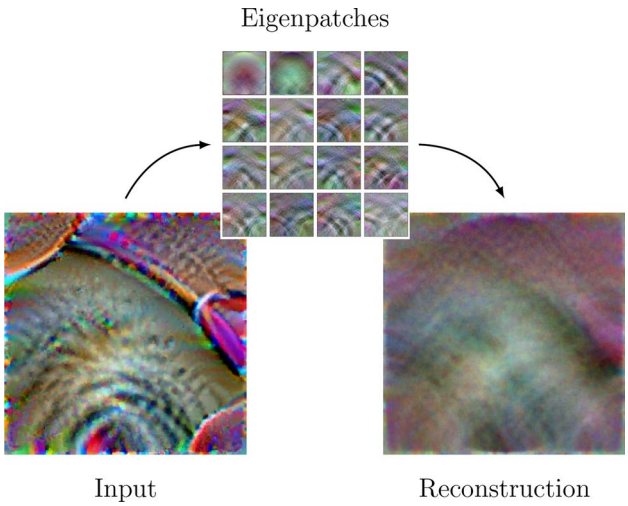

Fig. 2 Reconstruction (right) of an adversarial patch (left) using the first 16 eigenpatches (middle) [2]

Based on the encoding in the latent space, the patches are then reconstructed. In addition, the manifold is used to sample new patches. Both, the reconstructions and sampled patches are used in the experiments, either to measure the performance in the attack setting or for adversarial training.

3.1 PCA - eigenpatches

Eigenpatches or Eigenimages [17] are calculated on a set of adversarial patches [2]. The term Eigenimages is the name of the eigenvectors that can be derived from a set of training images when a PCA is applied. In general, Eigenimages can be used to represent the original training images and recreate these through a linear combination of the low-dimensional representation (see Fig. 2)

Given a set of adversarial patches

$$\mathcal{P} = \{\boldsymbol{P}_i | i = 1, \ldots, n\}, \quad \boldsymbol{P}_i \in [0,1]^{C \times H \times W} \tag{1}$$

where H is the height in pixel, W is the width in pixel and C is the number of channels. A principal component analysis is performed on $\mathcal{P}$. With the top k principal components $\boldsymbol{E}_j, j \in \{1, \ldots, k\}$ and the weights $\lambda_{i,j}$, the set

$$\hat{\mathcal{P}} = \{\hat{\boldsymbol{P}}_i | i = 1, \ldots, n\} \quad \hat{\boldsymbol{P}}_i = \sum_{j=1}^{k} \lambda_{i,j} \boldsymbol{E}_j \tag{2}$$

can be generated that consists of linear combinations of the principal components and is a recreation of $\mathcal{P}$ [2].

In some cases, especially when the data lies on or near a low-dimensional manifold within the high-dimensional space, a PCA can be considered a way to approximate that manifold. However, a PCA does not explicitly model the manifold structure of the data. It operates under the assumption, that the principal components capture the most important directions in the data, but it doesn't take into account the non-linear relationships or intrinsic geometry that may exist in the data.

To also take non-linearities into account, alternative methods are required.

3.2 Isomap

Isomap [18] is a nonlinear dimensionality reduction method that builds on classical Multidimensional scaling (MDS) [19] but seeks to preserve the intrinsic geometry of the data. First, the algorithm constructs a neighborhood graph over all data points, by connecting each data point to its neighbors based on some distance measure (e.g., Euclidean distance). Afterward, the geodesic distances between all pairs of points on the manifold are estimated by computing their shortest path distances in the graph. Finally, the low-dimensional embedding is constructed via MDS using the matrix of graph distances.

3.3 Locally linear embedding

Unlike Isomap, Locally Linear Embedding (LLE) [20] eliminates the need to estimate pairwise distances between widely separated data points. Instead, the algorithm first

assigns neighbors to each data point using K nearest neighbors. In a second step, weights are computed in such a way, that a linear reconstruction of a data point via its neighbors has a minimal reconstruction error. In a final step, each high dimensional data point is mapped to a low dimensional vector representing global internal coordinates on the manifold by choosing coordinates that minimizes an embedding cost function. The cost function is similar to the reconstruction error but instead of optimizing the weights, the coordinates are optimized.

3.4 Sampling from Isomap and LLE

There are various ways on how to inverse the embedding of data points lying within the manifold generated with Isomap and LLE. For the following experiments, Random Forest (RF) [21] and K-Neighbors (KN) regression are used.

3.5 Autoencoders

Autoencoders are particularly valuable in manifold learning due to their ability to learn compact and meaningful representations of high-dimensional data [22]. By compressing the data into a lower dimensional latent space, autoencoders effectively capture the essential features and structure of the data manifold. In general, autoencoders first encode the input data and decode them after they were propagated through a bottleneck. During training, the reconstruction loss is utilized to train the network weights. To capture spatial relation within image data, convolutional autoencoders are used, as they replace the fully connected layers with convolutional layers in both the encoder and decoder [22].

An extension that introduces a probabilistic approach on learning the latent space representation are variational autoencoders (VAE). They model the latent space as a probability distribution, allowing for more flexible generation of new data points [23]. In addition to the reconstruction loss, the Kullback–Leibler divergence is calculated and used as a regularization term to encourage the model to not focus on perfect reconstructions but rather be good at creating new data [24].

The experiments described in the next section use a convolutional autoencoder and a conditional variational autoencoder to generate reconstructions of the training data and sample new adversarial patches. The conditional variational autoencoder is a variant of the variational autoencoder that uses additional information during the encoding and decoding to condition the probability distribution [22].

Table 1 Different parameterization for the patch generation

ID	Epochs	Scheduler	Resize range	Rotation
A	125	StepLR	[0.5, 0.75]	45
B	100	StepLR	[0.75, 1.0]	45
C	100	Cos. annealing	[0.75, 1.0]	30
D	125	StepLR	[0.5, 0.75]	30
E	100	StepLR	[0.75, 1.0]	30

The resize range is the range of the scaling factor, relatively to the bounding box [2]

4 Evaluation

4.1 Experimental setup

If not stated otherwise, all experiments use the YOLOv7 tiny model as architecture. Furthermore, this paper uses the same patch set as [2]. They are referred as *prime patches* in the following. The prime patches are trained with different combinations (A-E) of rotation, scale and lr-scheduler parameters as described in [2] (see Table 1). They are optimized on the YOLOv7 tiny model with the provided pretrained weights and the *INRIA Person* dataset.

To measure the impact on the performance of the detector, the mean average precision (mAP) is used. The up/down facing arrows (↑/↓) in the tables indicate that a higher/lower score is more desirable. This is particularly important to keep in mind, since a high mAP is desired in the case of adversarial training and a low mAP in the case of a successful reconstruction of a prime patch.

4.1.1 Object detector

As in [2], YOLOv7 is used as a reference object detector. For the evaluation of adversarial training with patch reconstructions, the same training procedure is used for each trained model. SGD with the default parameters to train the model from scratch provided by the official git repository[1] is used. The batch size is set to 32 (128) and the probability, for a bounding box to contain a patch, is set to $\pi = 0.25$ ($\pi = 0.05$).

The single-patched network is trained with only a single prime patch and has not seen other patches during training. The multi-patched networks are trained with multiple patches and have therefore seen various patches during training. If an image is patched in a multi-patched network, all boxes in the image share the same patch.

The prime multi-patched network is trained with 10 randomly predefined prime patches (see supplementary material).

The PCA multi-patched networks are trained with linear combinations of Eigenpatches. The weights for the linear combination are sampled from normal distributions with

[1] https://github.com/WongKinYiu/yolov7.

means and standard deviations calculated according to the encoded prime patches.

Similarly, the patches used in the training of the conditional variational autoencoder multi-patched network are sampled. Here, values in the latent space are sampled from the underlying normal distribution, conditioned to a randomly chosen parameter group.

Patches used in the convolutional autoencoder, Isomap and LLE multi-patched network training are sampled by uniformly selecting a random point in the latent space, bounded by the values of the encoded prime patches.

4.1.2 Datasets

For the experiments, the *INRIA Person* dataset [25] and the *Crowdhuman* dataset [26] are used. Both datasets contain images of persons in various environments.

The *INRIA Person* dataset is a small dataset where the selected subset contains a total of 585 train images with 3317 bounding boxes and 288 test images with 904 bounding boxes. The dataset has also been used to generate the prime patches.

To verify the experiments, the *Crowdhuman* dataset is used, which is a benchmark dataset for person detection. The dataset contains 15,000 images in the training set, 4370 in the validation set and 5000 images in the test set.

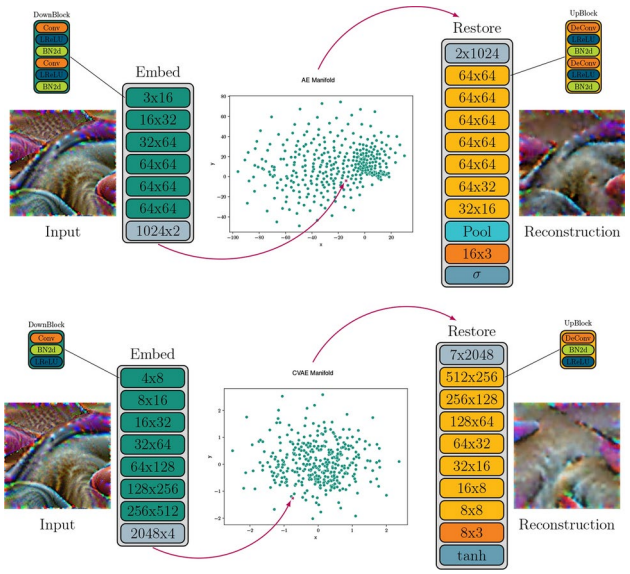

Fig. 3 Architectures of the used autoencoder networks. The convolutional autoencoder (upper) propagates the input through 6 "down blocks", followed by a fully connected layer. The reconstruction is performed by propagating a two-dimensional vector through a fully connected layer, followed by 7 "up blocks", an adaptive average pooling layer as well as a final convolution and activation layer. For the conditional variational autoencoder (lower), the encoder consists of 7 "down blocks" and a fully connected layer. The decoder consists of a fully connected layer, followed by 7 "up blocks", a convolutional layer and a final tanh activation layer

The total number of human instances in all sets is 470,000. For our experiments, we only use the train set. Since there are numerous small bounding boxes, each bounding box with less than 4096 pixels is filtered and removed from the dataset. This reduces the number of bounding boxes from 332,914 to 109,836.

4.1.3 Model parameters

Autoencoder: A schematic of the architecture of both autoencoder models can be found in Fig. 3. Both models are trained over 2000 epochs on the prime patches with an initial learning rate of 0.01 and the *AdamW* [27] optimizer. Each 100 epochs, the learning rate is reduced by a factor of 10. The batch size for both models is set to 64 and the bottleneck size is set to 2. For both models, the mean squared error between the input and output is optimized. The variational autoencoder also optimizes the KL-Div loss.

Isomap and LLE: The number of regression trees for reconstructions with the random forest is set to 5. For K-Neighbors, the neighborhood size matches the size used for Isomap and LLE and is also set to 5.

4.2 Experiments

4.2.1 Detector and backbone impact

The first experiment is about the t-distributed stochastic neighbor embeddings (t-SNE) of the prime patches. This set of qualitative experiments provides a more general overview of the impact of the patches on the pretrained object detector.

The prime patches and their influence on the activation of the last convolutional layer of the backbone in the YOLOv7 object detector can be seen in Fig. 4. The different colors in the plots correspond to the mean average precision the detector achieves after the bounding boxes in the image are attacked with a patch. The bright yellow dot is the activation of the original image without any patches present. The marker symbol of each data point corresponds to the parameter set used to optimize the patch. In addition to the five different prime patch parameter sets, a total number of 100 random noise patches and 100 grayscale patches are also shown.

Right next to the original activation of the unpatched image $(x = 30, y = -13)$, the activations of the different grayscale levels form a c-shape. The colors of the data points and the position in the neighborhood of the original activations indicate that the impact of the grayscale patches is low $(mAP = 0.79 \pm 0.02)$.

The second cluster with a relatively high mAP is given by the random patches. The center of this cluster is above

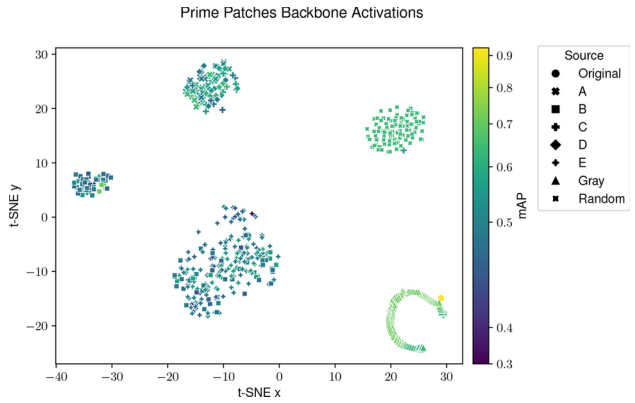

Fig. 4 t-SNE plot of the model backbone activation when attacked with a certain patch. Each marker represents a single patch, embedded at the same position in the same image. The color indicates the impact of the patch on the mean average precision. The parameter set of the patch is given by its marker symbol. The plot includes the prime patches, 100 random sampled patches as well as 100 grayscale patches. The parameter settings (A-E) used to train a certain prime patch correspond to the parameters presented in [2]

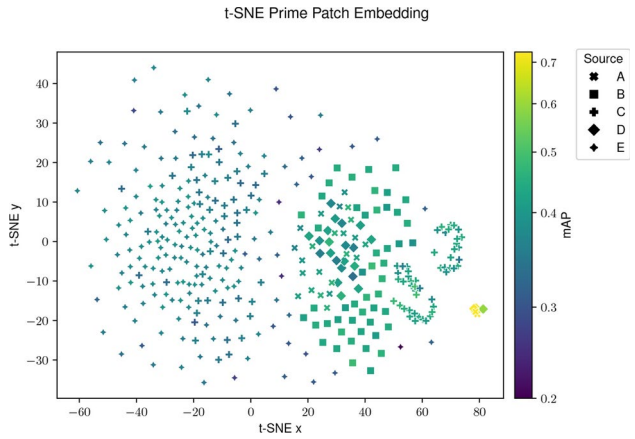

Fig. 5 t-SNE plot of the pixel values of each prime patch. The color indicates the impact of the patch on the mean average precision measured with the *INRIA Person* test set. The parameter set of a patch is given by its marker symbol

the first cluster at around (20, 15). In contrast to the grayscale patches, the random patches have a slightly deeper shade, which indicates that the random noise has a slightly stronger negative impact on the detector than the grayscale patches (0.75 ± 0.01).

All except for one data points of parameter set C are mixed with patches of parameter sets A and D in the upper left cluster, with the center at around $(-12, 25)$. The shade of the cluster is darker than the shade of the previous two clusters (0.69 ± 0.06).

Both remaining clusters contain data points of the parameter set B. The cluster at $(-35, 5)$ is a pure cluster of data points from parameter set B and is much denser compared to the remaining one. Both clusters have similar mAP of 0.63 ± 0.08 and 0.63 ± 0.05, yet the last remaining cluster with the center at around $(-10, -10)$ contains the data points with the lowest mAP values.

These observations demonstrate that noise and grayscale patches have a comparable low impact on the object detector on this specific input image. Moreover, the proximity of the activation of the grayscale patches to the original activation indicate only minor changes in the detector backbone activation. Additionally, patches that share an optimization parameter set alter the backbone activations similarly.

As this form of representation only shows the impact of a patch on a single image, another representation is given in Fig. 5. Here, each data point of the t-SNE plot corresponds to the patch itself. The colors of the data points in this plot encode the overall mean average precision drop in detection performance for the *INRIA Person* test set. Again, the marker symbol corresponds to the parameter set used to optimize the prime patch.

Similar to the previous observations, patches that share a parameter group form clusters and lead to a similar overall mAP drop on the *INRIA Person* test set.

4.2.2 Attack performance

The next experiment covers the attack performance on the pretrained object detector when reconstructed patches are used. In Fig. 6, the data points correspond to reconstructions of prime patches and prime patches themselves. Again, the color encodes the mAP value of the corresponding patch. The marker symbol, on the other hand, provides information about the used dimensionality reduction method.

CVAE is the conditional variational autoencoder. As the positions of the markers indicate, the method fails the reconstruction of the patches in most cases. Instead, multiple patches are alike. The mean Euclidean distance of the embedded CVAE patches to its corresponding prime embedding is 49.79 ± 30.59 and the highest among all three investigated methods.

The embeddings of the convolutional autoencoder reconstructions (AE) are more scattered and less focused on a few spots. The mean Euclidean distance of 19.30 ± 21.76 supports this.

As a representation of the PCA methods, the PCA(64) with 64 components is chosen. The mean Euclidean distance of 14.42 ± 19.45 is smaller than both autoencoders. Its reconstructions also form less dense clusters.

Nonetheless, compared to the reconstructions produced with random forest regression for Isomap and LLE, the PCA(64) performs much worse. The mean Euclidean distance for Isomap(RF) and LLE(RF) is with 5.62 ± 15.01 and 7.68 ± 18.34 much lower. When K-neighbors is used as reconstruction strategy, the mean Euclidean distances for Isomap(KN) and LLE(KN) are 25.32 ± 26.28 and

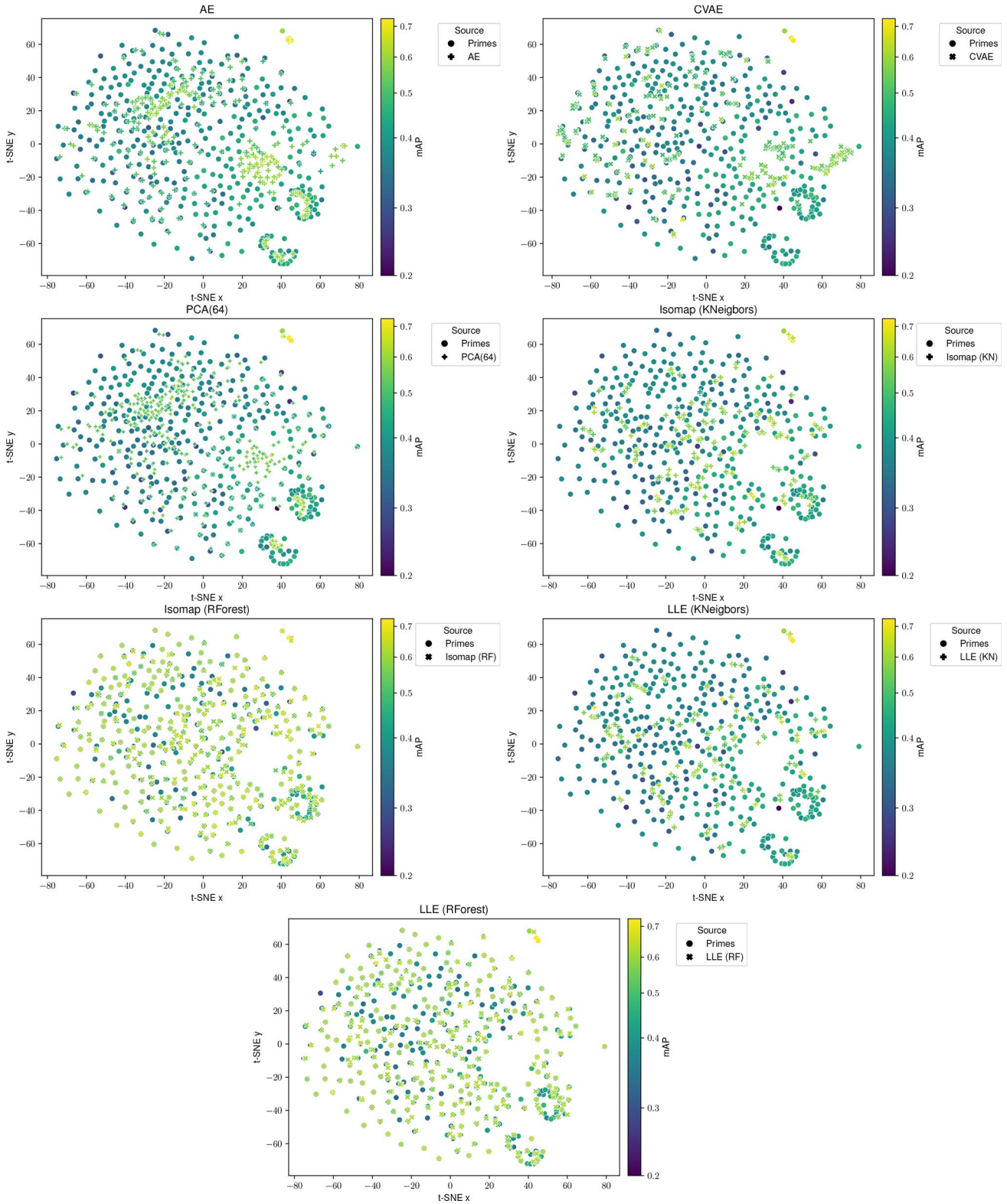

Fig. 6 t-SNE plots of the different reconstruction methods: autoencoder (AE), conditional variational autoencoder (CVAE), principal component analysis with 64 components (PCA(64)) and Isomap/LLE with random forest/K-neighbors. Again, the color indicates the impact on the mean average precision measured with the *INRIA Person* test set. The marker symbol shows, whether it is a reconstruction from the corresponding construction method or a prime patch

Table 2 Mean average precision of the COCO-pretrained YOLOv7 tiny model, on the *INRIA Person* test set

Patch mode	n	mAP 0.5 ↓	mAP 0.5:0.95↓
None	1	0.96	0.90
Grayscale	11	0.84 ± 0.02	0.71 ± 0.01
Prime patches	375	0.57 ± 0.06	0.39 ± 0.06
PCA (64)	375	$\mathbf{0.70 \pm 0.04}$	0.54 ± 0.04
AE	375	0.73 ± 0.04	0.55 ± 0.05
CVAE	375	0.72 ± 0.04	$\mathbf{0.53 \pm 0.05}$
Isomap (RF)	375	0.77 ± 0.03	0.62 ± 0.03
Isomap (KN)	375	0.75 ± 0.04	0.59 ± 0.05
LLE (RF)	375	0.75 ± 0.02	0.61 ± 0.02
LLE (KN)	375	0.75 ± 0.04	0.59 ± 0.05

The lowest mAP is marked as bold

The patch mode indicates, which patches are used during the attack: *None* uses no patches and Grayscale uses different grayscale levels as patches

25.26 ± 27.09 and therefore even higher than the convolutional autoencoder.

Interestingly, despite having the lowest mean Euclidean distance to their corresponding prime patches, the color of the data points for the random forest reconstructions already reveal, that the performance in the attack setting is lower compared to the rest. This is confirmed by Table 2, which contains the performance of the reconstructed patches in an attack setting. The prime patches and grayscale patches in the table can be considered an upper and lower bound for the patch performance. Almost all methods are able to reconstruct patches that result in a mean decline of the detector performance by 0.2. Compared to the prime patches, the reconstructions lack of certain details, as these cause an mAP drop to 0.57.

4.2.3 Adversarial training

Tables 3 and 4 show the results of the adversarial training. Each row represents a trained network. The patch mode indicates, which method was used to provide the adversarial patches during training. *None* is the trained network without any adversarial patches present during training. The entries in the following table show the mean value and standard deviation of the mAP for the respective patch set examined.

In Table 3, the problem with training on a small data basis as the selected subset of the *INRIA Person* dataset becomes visible. The induced adversarial patches during training can be considered as a form of data augmentation, resulting in a higher overall mAP. Interestingly, the networks are prone to grayscale patches, while at the same time, have learned to ignore or rather profit from adversarial patches on bounding boxes of interest. A possible solution to this could be the inclusion of grayscale patches during training.

The highest mAP when attacked with prime patches achieves the PCA (128) network. The best performance without any patches visible has the PCA (16) network. The PCA (64) network achieves the highest mAP when grayscale patches are visible.

When trained on the Crowdhuman dataset, the results are as expected (see Table 4): The induced adversarial patches during training affect the network performance when no patches are present, and further improve the performance when patches are present. The latter finding supports the thesis that adversarial-trained networks profit from the patches by using them as a guidance. An adjustment of the probability that a bounding box contains a patch during adversarial training to only $\pi = 0.05$ instead of $\pi = 0.25$ improves the

Table 3 Mean average precision for the testset of the *INRIA Person* dataset

	No patches (n=1)		Grayscaled (n=11)		Prime patches (n=375)	
Patch mode	mAP 0.5 ↑	mAP 0.5:0.95 ↑	mAP 0.5 ↑	mAP 0.5:0.95 ↑	mAP 0.5 ↑	mAP 0.5:0.95 ↑
None	0.80	0.66	0.61 ± 0.09	0.46 ± 0.12	0.71 ± 0.04	0.57 ± 0.04
Single-patched	0.85	0.71	0.71 ± 0.06	0.57 ± 0.08	0.89 ± 0.03	0.75 ± 0.02
Multi-patched	0.84	0.70	0.71 ± 0.07	0.57 ± 0.08	0.89 ± 0.02	0.76 ± 0.02
PCA (16)	**0.85**	**0.72**	0.69 ± 0.07	0.56 ± 0.08	0.86 ± 0.02	0.72 ± 0.03
PCA (32)	0.82	0.69	0.68 ± 0.07	0.53 ± 0.08	0.85 ± 0.02	0.71 ± 0.02
PCA (64)	0.85	0.70	$\mathbf{0.73 \pm 0.06}$	$\mathbf{0.59 \pm 0.06}$	0.88 ± 0.02	0.75 ± 0.02
PCA (128)	0.85	0.70	0.72 ± 0.06	0.59 ± 0.07	$\mathbf{0.90 \pm 0.02}$	$\mathbf{0.76 \pm 0.02}$
AE	0.84	0.70	0.72 ± 0.06	0.59 ± 0.07	0.87 ± 0.02	0.75 ± 0.02
CVAE	0.84	0.70	0.70 ± 0.08	0.56 ± 0.10	0.87 ± 0.02	0.75 ± 0.02
Isomap (KN)	0.81	0.65	0.66 ± 0.08	0.52 ± 0.09	0.82 ± 0.01	0.67 ± 0.02
Isomap (RF)	0.81	0.65	0.66 ± 0.07	0.53 ± 0.07	0.82 ± 0.01	0.67 ± 0.01
LLE (KN)	0.82	0.65	0.68 ± 0.08	0.54 ± 0.08	0.81 ± 0.03	0.66 ± 0.03
LLE (RF)	0.76	0.61	0.61 ± 0.08	0.49 ± 0.08	0.80 ± 0.02	0.66 ± 0.02

Bold values represent the highest (and thus best) mAP score of the given column

The model first was trained on the train data and later attacked with the given set of patches. While the YOLOv7 model has not seen any patches during the training, the other models had a 25 percent chance, that a bounding box contained a patch during training. The patch was either a single predefined patch, a random patch from the patch set or a sampled patch with the PCA or an autoencoder method

Table 4 Mean average precision for the testset of the *INRIA Person* dataset

	No Patches (n=1)		Grayscaled (n=11)		Prime Patches (n=375)	
Patch Mode	mAP 0.5 ↑	mAP 0.5:0.95 ↑	mAP 0.5 ↑	mAP 0.5:0.95 ↑	mAP 0.5 ↑	mAP 0.5:0.95 ↑
None	**0.95**	**0.91**	0.85 ± 0.01	$\mathbf{0.78 \pm 0.02}$	0.84 ± 0.03	0.75 ± 0.05
Single-patched	0.89	0.79	0.76 ± 0.04	0.66 ± 0.04	0.93 ± 0.04	0.84 ± 0.05
Multi-patched	0.84	0.74	0.72 ± 0.02	0.62 ± 0.02	0.95 ± 0.04	0.88 ± 0.05
PCA (16)	0.87	0.78	0.66 ± 0.05	0.56 ± 0.05	0.90 ± 0.04	0.80 ± 0.05
PCA (32)	0.88	0.78	0.71 ± 0.04	0.60 ± 0.04	0.94 ± 0.05	0.85 ± 0.06
PCA (64)	0.83	0.75	0.69 ± 0.03	0.60 ± 0.02	0.93 ± 0.08	0.86 ± 0.09
PCA (128)	0.84	0.76	0.70 ± 0.03	0.61 ± 0.03	0.93 ± 0.07	0.86 ± 0.08
AE	0.88	0.76	0.77 ± 0.05	0.65 ± 0.06	0.95 ± 0.02	0.84 ± 0.02
CVAE	0.88	0.77	0.66 ± 0.06	0.53 ± 0.07	0.92 ± 0.05	0.84 ± 0.06
Isomap (KN)	0.85	0.74	0.84 ± 0.06	0.74 ± 0.07	0.96 ± 0.01	0.88 ± 0.01
Isomap (RF)	0.82	0.70	$\mathbf{0.86 \pm 0.05}$	0.75 ± 0.06	0.96 ± 0.01	0.87 ± 0.01
LLE (KN)	0.83	0.73	0.80 ± 0.07	0.70 ± 0.08	0.94 ± 0.02	0.86 ± 0.03
LLE (RF)	0.82	0.71	0.80 ± 0.06	0.70 ± 0.07	$\mathbf{0.97 \pm 0.00}$	$\mathbf{0.89 \pm 0.01}$

Bold values represent the highest (and thus best) mAP score of the given column

The model first was trained on the crowdhuman train data with $\pi = 0.25$ and later attacked with the given set of patches

Table 5 Mean average precision for the testset of the *INRIA Person* dataset

	No patches (n=1)		Grayscaled (n=11)		Prime patches (n=375)	
Patch mode	mAP 0.5 ↑	mAP 0.5:0.95 ↑	mAP 0.5 ↑	mAP 0.5:0.95 ↑	mAP 0.5 ↑	mAP 0.5:0.95 ↑
None	**0.94**	**0.89**	0.82 ± 0.01	0.74 ± 0.02	0.83 ± 0.04	0.71 ± 0.07
Single-patched	0.92	0.86	0.83 ± 0.01	0.74 ± 0.02	0.92 ± 0.02	0.85 ± 0.03
Multi-patched	0.94	0.88	0.83 ± 0.01	0.74 ± 0.01	$\mathbf{0.95 \pm 0.02}$	$\mathbf{0.89 \pm 0.02}$
PCA (16)	0.91	0.81	0.77 ± 0.03	0.66 ± 0.03	0.92 ± 0.03	0.83 ± 0.04
PCA (32)	0.93	0.87	0.82 ± 0.01	0.74 ± 0.01	0.93 ± 0.03	0.86 ± 0.04
PCA (64)	0.93	0.87	0.82 ± 0.01	0.72 ± 0.01	0.93 ± 0.04	0.86 ± 0.05
PCA (128)	0.94	0.88	0.80 ± 0.01	0.72 ± 0.02	0.93 ± 0.04	0.85 ± 0.05
AE	0.93	0.87	$\mathbf{0.85 \pm 0.01}$	$\mathbf{0.77 \pm 0.01}$	0.95 ± 0.02	0.88 ± 0.02
CVAE	0.93	0.87	0.81 ± 0.01	0.73 ± 0.01	0.93 ± 0.03	0.86 ± 0.04
Isomap (KN)	0.92	0.84	0.84 ± 0.04	0.75 ± 0.05	0.93 ± 0.01	0.84 ± 0.02
Isomap (RF)	0.92	0.83	0.81 ± 0.04	0.72 ± 0.04	0.93 ± 0.02	0.85 ± 0.02
LLE (KN)	0.92	0.83	0.82 ± 0.05	0.73 ± 0.06	0.92 ± 0.02	0.83 ± 0.02
LLE (RF)	0.92	0.84	0.81 ± 0.05	0.71 ± 0.05	0.94 ± 0.01	0.85 ± 0.02

Bold values represent the highest (and thus best) mAP score of the given column

The model first was trained on the crowdhuman train data with $\pi = 0.05$ and later attacked with the given set of patches

unpatched performance while maintaining the performance on the prime patches (see Table 5).

In comparison to the results of Table 3, the corresponding mAP differences between the networks differ. The best performance, when no patches or grayscale patches are present, is given by the unpatched network. This is also true for Table 5. When attacked with prime patches, LLE(RF) surpass the remaining networks, followed by both Isomap networks. The convolutional autoencoder and the multi-patched network are on third place. The PCA networks (32, 64, 128) are on par with the single-patched network, when attacked with prime patches, yet, they perform worse when no patches or grayscale patches are present. The lowest overall performance is given by the PCA (16) network, followed by the CVAE network.

Due to computational intensity, the worth of training an auto encoder is questionable. Especially, since a similar performance can be achieved with a single patch.

5 Conclusion

This paper provides an in-depth analysis of adversarial patches, used to fool object detectors. To be more specific, a set of so-called prime patches to evade a YOLOv7 based person detector is analyzed. A qualitative insight into the activations of the backbone network is given, when the detector is attacked with these patches. In a series of experiments, prime patches are processed by multiple dimensionality reduction methods, and the mAP drop of the attacked object detector for their reconstructions is measured. Furthermore,

the resulting manifolds of the dimensionality reduction methods are sampled and used in adversarial training. The results indicate that the training of more sophisticated manifold learning methods as the investigated autoencoders does not provide a significant better or more varying way to sample adversarial patches. The inclusion of a small set of prime patches or sampled patches using a PCA, LLE or Isomap is sufficient for adversarial training. Moreover, relying on a diverse set of sampled patches using a learned representation results only in a small improvement compared to naive adversarial training. The computational overhead for the use of deep-learning systems that require far more data to provide good results is therefore questionable.

The results also show that the investigated dimensionality reduction methods are able to capture some of the necessary features that are required to fool an object detector. Overall, this can be seen as a further indication that the manifold assumption applies to adversarial patches. Future work should investigate, how manifold learning methods could be used to build new or enhance protection mechanisms against this kind of adversarial attacks.

Acknowledgements This work was developed in Fraunhofer Cluster of Excellence "Cognitive Internet Technologies".

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Fefferman, C., Mitter, S., Narayanan, H.: Testing the manifold hypothesis. J. Am. Math. Soc. **29**(4), 983–1049 (2016). https://doi.org/10.1090/jams/852. arXiv:1310.0425
2. Bayer, J., Becker, S., Münch, D., Arens, M.: Eigenpatches—adversarial patches from principal components. In: Advances in Visual Computing, pp. 274–284. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-47966-3_21
3. Wang, C.-Y., Bochkovskiy, A., Liao, H.-Y.M.: Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: CVPR, pp. 7464–7475 (2023). https://doi.org/10.1109/CVPR52729.2023.00721
4. Tarchoun, B., Khalifa, A.B., Mahjoub, M.A., Abu-ghazaleh, N.: Jedi: Entropy-based localization and removal of adversarial patches. In: CVPR, pp. 4087–4095 (2023). https://doi.org/10.1109/CVPR52729.2023.00398
5. Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., McDaniel, P.: The space of transferable adversarial examples. In: arXiv Prepr., pp. 1–15 (2017). https://doi.org/10.48550/arXiv.1704.03453
6. Wang, Z.M., Gu, M.T., Hou, J.H.: Sample based fast adversarial attack method. Neural Process. Lett. **50**(3), 2731–2744 (2019). https://doi.org/10.1007/s11063-019-10058-0
7. Shi, R., Yang, B., Jiang, Y., Zhao, C., Ni, B.: Energy attack: on transferring adversarial examples. In: arXiv Prepr. (2021). https://doi.org/10.48550/arXiv.2109.04300
8. Dohmatob, E., Guo, C., Goibert, M.: Origins of low-dimensional adversarial perturbations. In: Ruiz, F., Dy, J., Meent, J.-W. (eds.) AISTATS, pp. 9221–9237 (2023). https://doi.org/10.48550/arXiv.2203.13779
9. Shafahi, A., Huang, R., Studer, C., Feizi, S., Goldstein, T.: Are adversarial examples inevitable? In: ICLR (2019) https://doi.org/10.48550/arXiv.1809.02104arXiv:1809.02104
10. Weng, J., Luo, Z., Lin, D., Li, S., Zhong, Z.: Boosting adversarial transferability via fusing logits of top-1 decomposed feature (2023) https://doi.org/10.48550/arXiv.2305.01361arXiv:2305.01361
11. Garcia, W., Chen, P.-Y., Clouse, H.S., Jha, S., Butler, K.R.B.: Less is more: Dimension reduction finds on-manifold adversarial examples in hard-label attacks. In: SaTML, pp. 254–270 (2023). https://doi.org/10.1109/SaTML54575.2023.00025
12. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access **6**, 14410–14430 (2018). https://doi.org/10.1109/ACCESS.2018.2807385
13. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: A survey on adversarial attacks and defences. CAAI Trans. Intell. Technol. **6**(1), 25–45 (2021). https://doi.org/10.1049/cit2.12028
14. Pauling, C., Gimson, M., Qaid, M., Kida, A., Halak, B.: A tutorial on adversarial learning attacks and countermeasures (2022). https://doi.org/10.48550/arXiv.2202.10377
15. Wang, S., Veldhuis, R., Strisciuglio, N.: A survey on the robustness of computer vision models against common corruptions. arXiv Prepr. 1–23 (2023) https://doi.org/10.48550/arXiv.2305.06024arXiv:2305.06024
16. Godfrey, C., Kvinge, H., Bishoff, E., Mckay, M., Brown, D., Doster, T., Byler, E.: How many dimensions are required to find an adversarial example? In: CVPRW, pp. 2353–2360 (2023). https://doi.org/10.1109/CVPRW59228.2023.00232
17. Sirovich, L., Kirby, M.: Low-dimensional procedure for the characterization of human faces. J. Opt. Soc. Am. A **4**(3), 519 (1987). https://doi.org/10.1364/josaa.4.000519
18. Tenenbaum, J.B., Silva, V.D., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000). https://doi.org/10.1126/science.290.5500.2319
19. Cox, M.A.A., Cox, T.F.: Multidimensional Scaling, pp. 315–347. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-33037-0_14
20. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500), 2323–2326 (2000) https://doi.org/10.1126/science.290.5500.2323
21. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001) https://doi.org/10.1023/A:1010933404324

22. Berahmand, K., Daneshfar, F., Salehi, E.S., Li, Y., Xu, Y.: Autoencoders and their applications in machine learning: a survey. Artif. Intell. Rev. **57**(2), 28 (2024). https://doi.org/10.1007/s10462-023-10662-6
23. Sohn, K., Yan, X., Lee, H.: Learning structured output representation using deep conditional generative models, 3483–3491 (2015)
24. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014). https://doi.org/10.48550/arXiv.1312.6114
25. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR, vol. 1, pp. 886–893 (2005). https://doi.org/10.1109/CVPR.2005.177
26. Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., Sun, J.: CrowdHuman: A benchmark for detecting human in a crowd, 1–9 (2018) https://doi.org/10.48550/arXiv.1805.00123. arXiv:1805.00123
27. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019) https://doi.org/10.48550/arXiv.1711.05101. arXiv:1711.05101

Jens Bayer holds an M.Sc. in Computer Science from the Karlsruhe Institute of Technology (KIT), awarded in 2020. He is currently doing his Ph.D. at KIT and is also a researcher at the Fraunhofer Institute for Optronics, System Technologies, and Image Exploitation (IOSB) within the "Video Content Analysis" group. His research focuses on adversarial attacks and robustness in computer vision with a particular emphasis on object detection.

Stefan Becker received his Ph.D. (Dr.-Ing) in computer science and his diploma in electric engineering from the Karlsruhe Institute of Technology (KIT). Currently, he is working as a post-doctoral researcher in the "Video Content Analysis" group at the Fraunhofer Institute for Optronics, System Technologies, and Image Exploitation (IOSB). He participated in and contributed to several projects in industry, government, and the EU.

David Münch holds a Ph.D. (Dr.-Ing.) and a Diploma in Computer Science. He is a post-doctoral researcher at the Fraunhofer IOSB in the "Video Content Analysis" group. His work focuses on object detection, event recognition, and semantic video understanding, combining machine learning with classical computer vision. He also has a teaching assignment at the Baden-Württemberg Cooperative State University.

Michael Arens received the Diploma degree in computer science and the Ph.D. degree (Dr.rer.nat.) from the University of Karlsruhe, in 2001 and 2004, respectively. He is currently the Head of the Department of the Object Recognition (OBJ), Fraunhofer Institute for Optronics, System Technology, and Image Exploitation (IOSB).

Jürgen Beyerer has been a full professor (Dr.-Ing.) sfor informatics at the Institute for Anthropomatics and Robotics at the Karlsruhe Institute of Technology KIT since March 2004 and director of the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB in Ettlingen, Karlsruhe, Ilmenau, Görlitz, Lemgo, Oberkochen and Rostock. Research interests include automated visual inspection, signal and image processing, variable image acquisition and processing, active vision, metrology, information theory, fusion of data and information from heterogeneous sources, system theory, autonomous systems and automation.