Autoencoders - A Comparative Analysis in the Realm of Anomaly Detection

Sarah Schneider[1,2]
sarah.schneider@ait.ac.at

Doris Antensteiner[1]
doris.antensteiner@ait.ac.at

Daniel Soukup[1]
daniel.soukup@ait.ac.at

Matthias Scheutz[2]
matthias.scheutz@tufts.edu

[1]Center for Vision, Automation and Control, Austrian Institute of Technology
Vienna, Austria

[2]Human-Robot Interaction Lab, Tufts University
Medford, MA, USA

Abstract

We applied convolutional versions of a "standard" autoencoder (CAE), a variational autoencoder (VAE) and an adversarial autoencoder (AAE) to two different publicly available datasets and compared their anomaly detection performances. We used the MNIST dataset [14] as a simple anomaly detection scenario. The CIFAR10 dataset [13] was used to examine the autoencoders in a more complex anomaly detection task. The anomaly detection performance of our different autoencoder types is compared in a qualitative and quantitative manner. The time needed for training the models is measured to capture their complexity. The simplest model demanding the simplest training, the CAE, computes results which are nearly as accurate and for some cases even better than results achieved by the VAE and AAE. We show that all three autoencoder types computed convincing anomaly detection results for the more simple-structured MNIST scenario. However, none of the autoencoder types proved to capture a good representation of the relevant features of the more complex CIFAR10 dataset, leading to moderately good anomaly detection performances.

1. Introduction

An anomaly deviates from what is regarded as normal or usual. Anomalies appear with great diversity, therefore their detection represents a complex problem [18]. The detection of defects in industrial settings [22], fraud detection in bank transfer processes [6], the localization of diseased tissue in medical imaging [8] and many other problems can be formulated as an anomaly detection framework, thus the detection of anomalies is also highly relevant. Autoencoders [20] can be used to detect anomalous samples. The autoencoder (AE) model consists of an encoder and a decoder. The encoder encodes data to a compressed latent representation. The decoder decodes the latent representation back to the original dimensions. The parameters of the AE are optimized such that the computed output is as similar to the input as possible.

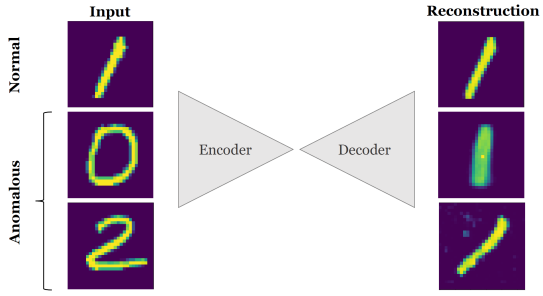

Figure 1. Anomaly detection based on an AE. By training the AE exclusively on images of a class defined as normal (digit 1), it will learn the profile of this normal class and it will reconstruct its images accurately. The reconstruction of an image belonging to an anomalous class (digit 0, digit 2) deviates from the input image as the mapping was learned from normal data.

By training the AE on normal samples, it will learn to reconstruct normal data accurately. As the profile of anomalous data differs from the learned profile of normal data, the trained AE model will compute a worse reconstruction for anomalous samples. The reconstruction error can be used as an indicator to distinguish normal from anomalous. The reconstruction error computed for anomalies will be higher

than the reconstruction error computed for normal samples.

The concept of autoencoders for detecting anomalies was used in [21] in a medical context, namely on stained histological images and chest X-rays. Autoencoders were applied to discriminate schizophrenic from healthy individuals in [23]. Autoencoders have also been used to detect defects in industrial settings, e. g., [7] used autoencoders to detect manufacturing defects on metal boxes and [17] applied the concept on images of defective rail surfaces.

We investigated the quality of anomaly detection results of three convolutional versions of AEs - a "standard" autoenocoder (CAE) [9], a variational autoencoder (VAE) [12] and an adversarial autoencoder (AAE) [15]. We used two publicly available dataset, *MNIST* [14] and *CIFAR10* [13]. Images of one class, e.g. class "*1*" for *MNIST* and class "*dog*" for *CIFAR10*, were considered as normal. The images of all other classes were considered anomalous. The reconstruction error was interpreted as an indicator to distinguish images of the normal class from images of the anomalous classes. The results are compared based on the area under curve of Receiver-Operator-Characteristic and Precision-Recall curve as presented in [4] and [5].

Our experiments on AEs for anomaly detection show that all convolutional autoencoders compute very good results for *MNIST*. However, we could also observe that the *CIFAR10* dataset represents a far more challenging task. The distinction between normal and anomalous class samples was made only moderately accurate. To capture the "cost-benefit ratio" of our different autoencoder types, we evaluated the quality of their anomaly detection results alongside with their measured training times.

Our contribution comprises:

- A thorough review and suggestions for improved designs based on the given evidence,
- A quantitative and qualitative comparison of different convolutional autoencoder types (CAE, VAE, AAE) based on their ability to make a per-image "normal-or-anomalous" decision,
- An analysis of the "cost-benefit ratio" for the different AE types based on their training times and anomaly detection quality.

2. Autoencoders for Anomaly Detection

An AE encodes an input sample $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$ to a latent representation $\mathbf{z} \in \mathbb{R}^{C_b\times H_b\times W_b}$ (see Fig. 1). The latent representation is then decoded back such that it is as similar to the original input as possible. The dimensions of the latent representation are usually chosen to be significantly smaller than the dimensions of the input ("bottleneck"). This leads to a compression of the data and the model is forced to focus on the most relevant features.

If the AE is trained on images regarded as normal, it will learn to compress and reconstruct these normal samples. The trained AE will then fail to reconstruct anomalous data accurately since it uses the mapping learned from normal images.

The reconstruction error $\mathbf{R}(\mathbf{x},\tilde{\mathbf{x}}) \in \mathbb{R}^{C\times H\times W}$ is defined to be the mean squared error (MSE, see Eq. 2), e.g. the averaged squared pixel-wise difference of the network input $\mathbf{x}$ and its reconstruction $\tilde{\mathbf{x}}$ computed by the trained network. A normal image sample leads to a lower reconstruction error than an anomalous image, hence $\mathbf{R}(\mathbf{x},\tilde{\mathbf{x}})$ is utilized to distinguish normal from anomalous samples [2,3,16].

2.1. "Standard" Convolutional Autoencoder

A "standard" convolutional AE (CAE) is composed of a convolutional encoder E_{CAE} and a convolutional decoder D_{CAE}. The encoder compresses the input data $\mathbf{x}$ into a latent representation $\mathbf{z}$. The decoder decodes $\mathbf{z}$ back to input dimensions.

$$\tilde{\mathbf{x}} = D_{CAE}(E_{CAE}(\mathbf{x})) = D_{CAE}(\mathbf{z}) \tag{1}$$

The parameters of the CAE are optimized by minimizing the mean squared error (MSE) function which is defined as follows:

$$\mathcal{L}_{MSE}(\mathbf{x},\tilde{\mathbf{x}}) = \frac{1}{H}\frac{1}{W}\sum_{m=1}^{H}\sum_{n=1}^{W}(\mathbf{x}(m,n)-\tilde{\mathbf{x}}(m,n))^2, \tag{2}$$

where the scalar-filled matrices $\mathbf{x}(m,n)$ and $\tilde{\mathbf{x}}(m,n)$ denote the intensity at the pixel locations (m,n) with $m \in [1,...,H]$ and $n \in [1,...,W]$ of image $\mathbf{x}$ and $\tilde{\mathbf{x}}$, respectively [2,3,16,20].

2.2. Variational Autoencoder

The variational autoencoder (VAE, [12], [19]) is structured similarly to the CAE. It is composed of an encoder E_{VAE} and a decoder D_{VAE}. The encoder encodes the input to a lower dimensional latent representation, the decoder decodes the latent representation back to input dimensions. The VAE constrains the latent representation $\mathbf{z}$ of an input $\mathbf{x}$ to be a random variable, distributed according to a prior distribution $p_\theta(\mathbf{z})$. Hence, the VAE not only compresses the input to a latent representation, but encodes it as a distribution over the latent space. The true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable for a continuous latent space but it can be approximated in a deterministic manner. The approximation $q_\Phi(\mathbf{z}|\mathbf{x})$ can be determined by applying the variational interference technique. The underlying principle is to set up a parametrised family of distributions and choosing the distribution that gives the least approximation error. The parameters of the VAE are optimized as follows:

$$\begin{aligned}&\mathcal{L}(\theta,\phi,\mathbf{x}) = \\ &\mathbb{E}_{q_\Phi(\mathbf{z}|\mathbf{x})}[p_\theta(\mathbf{x}|\mathbf{z})] - \mathcal{D}_{KL}(q_\Phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{x})).\end{aligned} \tag{3}$$

The first term is equivalent to the loss function of the CAE - it is the reconstruction error, measuring the reconstruction quality. The second term is the Kullback–Leibler divergence. This loss term ensures that the latent distribution stays close to the prior distribution, i.e. it evaluates how much information is lost if $p_\theta(\mathbf{x})$ is approximated. If it is assumed that the approximated posterior distribution $q_\Phi(\mathbf{z}|\mathbf{x})$ is a Gaussian distribution, the reconstruction error simplifies to the MSE loss (Eq. 2). When both prior approximation $p_\theta(\mathbf{z})$ and posterior approximations $p_\Phi(\mathbf{z}|\mathbf{x})$ are assumed to be distributed according to a Gaussian distribution, the Kullback-Leibler divergence can be solved analytically as described by [12]. Variance σ and mean μ will be determined according to Eq. 4. J is the number of elements in the vectors σ and μ.

$$-\mathcal{D}_{KL}(q_\Phi(\mathbf{z}|\mathbf{x})||p_\theta) = \frac{1}{2}\sum_{j=1}^{J}(1+\log((\sigma_j)^2)-(\mu_j)^2-(\sigma_j)^2) \tag{4}$$

2.2.1 Re-parametrization Trick

The decoder samples from $q_\Phi(\mathbf{z}|\mathbf{x})$ randomly in order to compute a latent vector $\mathbf{z}$. However, backpropagation can not flow through this random sampling operation. [12] therefore propose the re-parametrization trick. The problematic random sampling operation is bypassed by describing $\mathbf{z}$ as a differentiable transformation through a function g of another random variable denoted as ε. By assuming $q_\Phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z};\mu,\text{diag}(\sigma^2))$ and re-parametrization with $\varepsilon \sim \mathcal{N}(0,\mathbf{I})$, the latent representation $\mathbf{z}$ can be computed as follows:

$$\mathbf{z} = \mu + \sigma \odot \varepsilon, \tag{5}$$

where the scalars μ and $\log(\sigma)$ are determined previously by the encoder network (Eq. 4) and $\odot$ denotes the Hadamard product.

2.3. Adversarial Autoencoder

The adversarial autoencoder (AAE) was proposed in [15] and functions in a probabilistic manner. Similar to the VAE, a prior distribution is imposed on the latent representation $\mathbf{z}$. The VAE applies the Kullback-Leibler divergence in order to match the posterior of the latent vector with a prior distribution. The AAE uses the concept of Generative Adversarial Networks (GAN) [10].
Our AAE is composed of a convolutional encoder, a convolutional decoder and an additional GAN component, which regularizes the latent vector $\mathbf{z}$. The components are trained jointly in two phases.

During the *reconstruction* phase, the encoder E_{AAE} and decoder D_{AAE} are trained exactly as the encoder and decoder of a "standard" CAE. Their parameters are optimized by minimizing the MSE loss (Eq. 2). The components are forced to compute a reconstruction as similar to the input as possible.

During the *regularization* phase, the encoder E_{AAE} operates as a generator for the adversarial network. The encoder is trained to "fool" the discriminative adversarial network C_{AAE} "into thinking" that the latent code it generated emerged from the true prior distribution. The discriminative network C_{AAE} is trained to determine if a latent vector is "real" or "fake" by computing the Wasserstein distance [1]. A "real" vector $\mathbf{z}$ originates from the "real" training data distribution, a "fake" vector $\tilde{\mathbf{z}}$ originates from the "fake" distribution generated by E_{AAE}. The Lipschitz constraints are maintained by clipping weights to a fixed box at each weight update [1, 15].

3. Comparison of Different Autoencoder Types for Anomaly Detection

The architecture of the encoder and the decoder of the CAE, the VAE and the AAE are implemented identically. The bottleneck of the VAE consists of two innermost layers, one for the latent vector of standard deviations, one for the latent vector of mean values. These values are then reparametrized before being processed by the decoder of the VAE. The architectures of the implemented networks are explicitly given in 3.1.1. The convolutional kernels used for feature extraction were of size 3×3. All AE types of identical architectures were trained in the exact same manner, i.e. for the same number of epochs, with the same learning rate and minibatch size for each set of experiments. The Adam optimization algorithm [11] was used to optimize the parameters of the AEs. Fig. 2 shows an illustration of the similarities and differences between the different AEs.

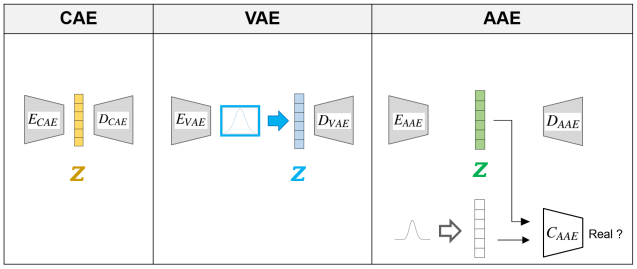

Figure 2. Illustration of the different types of AEs used. The encoder and decoder of the AEs are implemented in an identical manner, the computations of the latent vector in the bottleneck layer vary.

3.1. Experiments on MNIST and CIFAR10

In order to investigate the autoencoders also on publicly available datasets, we applied them on *MNIST* [14] and *CIFAR10* [13]. For the *MNIST* dataset, we defined images of

the class "1" to be normal. For *CIFAR10*, class "*dog*" was considered as normal. Our three autoencoder models were trained on the training subset of the normal class predefined by [13] and [14]. All other classes were treated as anomalies. The trained models were then applied to unseen images of the normal class and images of all other classes. The resulting reconstruction errors (see Eq. 2) were averaged and thresholded in order to make a binary novel-or-non-novel decision.

3.1.1 Network Architectures

In this section we show our self-designed autoencoder architectures. We designed architectures for *MNIST* and *CIFAR10* and evaluated them for all three autoencoder types which are shown in Fig. 2. The encoder and decoder of the CAE, VAE and AAE are identical. The type of autoencoder can be switched by changing the bottleneck layer according to the network type (CAE , VAE , AAE). Tab. 1 shows encoder and decoder architecture used for the *MNIST* dataset. The architecture of the encoder and decoder implemented for the *CIFAR10* dataset is shown in Tab. 2. Tab. 3 represents the network architecture of the bottlenecks of the different autoencoder types used for both datasets. The architecture of the discriminative network C_{AAE} of the AAE which was used for *MNIST* and *CIFAR10* is given in Tab. 4.

Table 1. Encoder and decoder architecture used for *MNIST*.

	Layer	Resolution	Channels	Input	Activ. func.
Encoder	conv1	$W \times H / W \times H$	1/16	Image	Leaky ReLU
	conv2	$\frac{W}{2} \times \frac{H}{2}$ / $\frac{W}{2} \times \frac{H}{2}$	16/32	conv1	Leaky ReLU
	conv3	$\frac{W}{4} \times \frac{H}{4}$ / $\frac{W}{4} \times \frac{H}{4}$	32/64	conv2	Leaky ReLU
	conv4	$\frac{W}{4} \times \frac{H}{4}$ / $\frac{W}{7} \times \frac{H}{7}$	64/64	conv3	Leaky ReLU
	conv5	$\frac{W}{7} \times \frac{H}{7}$ / $\frac{W}{7} \times \frac{H}{7}$	64/16	conv4	Leaky ReLU
Bottleneck (CAE , VAE , AAE)					
Decoder	deconv1	$\frac{W}{7} \times \frac{H}{7}$ / $\frac{W}{7} \times \frac{H}{7}$	16/64	lin4 (reshaped)	Leaky ReLU
	deconv2	$\frac{W}{7} \times \frac{H}{7}$ / $\frac{W}{4} \times \frac{H}{4}$	64/64	deconv1	Leaky ReLU
	deconv3	$\frac{W}{4} \times \frac{H}{4}$ / $\frac{W}{2} \times \frac{H}{2}$	64/32	deconv2	Leaky ReLU
	deconv4	$\frac{W}{2} \times \frac{H}{2}$ / $W \times H$	32/16	deconv3	Leaky ReLU
	deconv5	$W \times H$ / $W \times H$	16/1	deconv4	Sigmoid

3.1.2 Evaluation Metrics

The quality of the novel-or-non-novel decisions was examined by computing Receiver-Operator-Characteristic (ROC) curves and Precision-Recall (PR) curves as described in [5]. Each data point of the curves corresponds to a threshold value used for thresholding the reconstruction error. The comparison of anomaly detection performances is based on the area under curve (AUC) [5] for both the ROC and PR curve.

Table 2. Encoder and decoder architecture used for *CIFAR10*.

	Layer	Resolution	Channels	Input	Activ. func.
Encoder	conv1	$W \times H / W \times H$	3/8	Image	Leaky ReLU
	conv2	$\frac{W}{2} \times \frac{H}{2}$ / $\frac{W}{2} \times \frac{H}{2}$	8/16	conv1	Leaky ReLU
	conv3	$\frac{W}{4} \times \frac{H}{4}$ / $\frac{W}{4} \times \frac{H}{4}$	16/32	conv2	Leaky ReLU
	conv4	$\frac{W}{4} \times \frac{H}{4}$ / $\frac{W}{4} \times \frac{H}{4}$	32/16	conv3	Leaky ReLU
Bottleneck (CAE , VAE , AAE)					
Decoder	deconv1	$\frac{W}{4} \times \frac{H}{4}$ / $\frac{W}{4} \times \frac{H}{4}$	32/32	lin4	Leaky ReLU
	deconv2	$\frac{W}{4} \times \frac{H}{4}$ / $\frac{W}{2} \times \frac{H}{2}$	32/16	deconv1	Leaky ReLU
	deconv3	$\frac{W}{2} \times \frac{H}{2}$ / $W \times H$	16/8	deconv2	Leaky ReLU
	deconv4	$W \times H$ / $W \times H$	8/3	deconv3	Sigmoid

Table 3. *Bottleneck* architecture used for *MNIST* (conv_b = conv5, $r = 100$, $n_z = 20$) and *CIFAR10* (conv_b = conv4, $r = 100$, $n_z = 20$). Layer lin1 gets the flattened output of the encoder, conv_b, as its input. For the VAE, lin3 takes the re-parametrization (see 2.2.1 for details) of lin21 and lin22 as its input.

AE type	Layer	Resolution	Input	Activ. func.
AAE / CAE	lin1	$1 \times \frac{W}{16} * \frac{H}{16} * 16$ / 1×1024	conv_b (flattened)	Leaky ReLU
	lin2	$1 \times r$ / $1 \times n_z$	lin1	-
	lin3	$1 \times n_z$ / $1 \times r$	lin2	Leaky ReLU
	lin4	$1 \times r$ / $1 \times \frac{W}{16} * \frac{H}{16} * 16$	lin3	Leaky ReLU
VAE	lin1	$1 \times \frac{W}{16} * \frac{H}{16} * 16$ / 1×1024	conv_b (flattened)	Leaky ReLU
	lin21	$1 \times r$ / $1 \times n_z$	lin1	-
	lin22	$1 \times r$ / $1 \times n_z$	lin1	-
	lin3	$1 \times n_z$ / $1 \times r$	lin21, lin22 (reparam.)	Leaky ReLU
	lin4	$1 \times r$ / $1 \times \frac{W}{16} * \frac{H}{16} * 16$	lin3	Leaky ReLU

Table 4. Architecture of the C_{AAE} used for both datasets (see Fig. 2 AAE).

Layer	Resolution	Channels	Input	Activ. func.
lind1	$1 \times n_z$ / 1×10	-	lin2	Leaky ReLU
lind2	1×10 / 1×1	-	lind1	Leaky ReLU

4. Results

In this section, we demonstrate the quantitative and qualitative performance of our three autoencoder types as introduced in Sec. 2.

4.1. MNIST

In this subsection, we evaluate the quantitative and qualitative performance of our three autoencoder types on the well-known *MNIST* dataset [14].

4.1.1 Qualitative Results

Examples of our qualitative results are shown in Fig. 3, which gives an insight into the autoencoder's "way of thinking". The first row shows us how the trained models reconstruct an (unseen) image of the class they were trained on. The resulting reconstructed images look very similar to

the input image. The second and third row illustrate what happens if we apply the trained models on an image of an anomalous class - the autoencoders reconstruct it as if it was an image of the class they were trained on. Interestingly, one can also observe that the choice of the convolutional autoencoder type leads to severe differences in reconstruction results, especially the VAE seems to try to reconstruct the image as if it was a composite of ellipsoids. One can also see that, as all numbers are placed in the center of the images, all models learned that the intensity of the pixel at the center of the image is high for most cases.

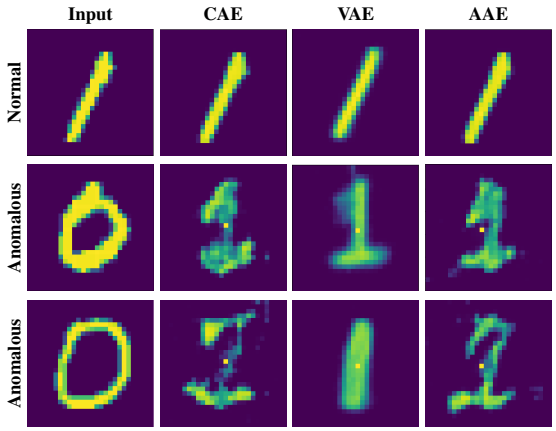

Figure 3. Illustration of the behaviour of autoencoders on unknown classes on the *MNIST* dataset. The first row shows an (unseen/untrained) image of the trained non-anomalous class 1 and its reconstruction by the CAE, the VAE and the AAE. The second and third row show two examples of the anomalous class 0 and their reconstructions by the CAE, the VAE and the AAE.

4.1.2 Quantitative Results

Our quantitative comparisons given in Tab. 5 show that all our tested autoencoders manage to distinguish images of the normal class form images of the anomalous class quite effortlessly. Each autoencoder type seems to perform better than others for some classes, but none of them outperforms the others consistently over all classes. The training of the CAE takes less time than training the VAE, the AAE trains the longest. Again, this is totally reasonable since the training procedure of the CAE is the least complex one, there is no re-parametrisation (VAE) or adversarial component training (AAE) necessary.

4.2. CIFAR10

In this subsection, we show the performance evaluation results on the *CIFAR10* dataset [13].

Table 5. Area under curve of the ROC (AUC_{ROC}) and PR (AUC_{PR}) curve for all autoencoder types on the *MNIST* dataset. Class *1* is considered to be normal, all others anomalous. The last row shows the time needed for training the models ($time_{tr}$, in seconds).

	Type / Class	CAE		CAE		AAE	
		AUC_{ROC}	AUC_{PR}	AUC_{ROC}	AUC_{PR}	AUC_{ROC}	AUC_{PR}
AUC_{ROC}	0	**0.9999**	**0.9999**	**0.9999**	**0.9989**	**0.9999**	**0.9999**
	2	0.9933	0.9949	0.9989	0.9975	0.9974	0.9975
	3	0.9964	0.9970	0.9986	0.9977	0.9966	0.9972
	4	0.9980	0.9980	0.9984	0.9975	0.9974	0.9975
	5	0.9979	0.9977	0.9990	0.9982	0.9982	0.9982
	6	0.9942	0.9956	0.990	0.9984	0.9959	0.9966
	7	0.9664	0.9776	0.9971	0.9957	0.9711	0.9809
	8	0.9985	0.9982	0.9988	0.9980	0.9992	0.9988
	9	0.9876	0.9914	0.9980	0.9969	0.9893	0.9921
$time_{tr}$	-	**1015.53**		1055.02		1748.8	

4.2.1 Qualitative Results

We present some illustrative images and their reconstructions in Fig. 4 for a qualitative comparison. All our three autoencoders compute very blurry versions of the input images and put their focus mainly on the colours in the images. The models are not able to capture more representative features at a higher level of abstraction, e.g. that dogs have two ears and a snout. One can see that the autoencoders reconstruct the image of class *truck* as if it was an image of the class they were trained on. The models show to neglect the red colours of the given *truck* image and just reconstruct it by using colours they have seen during training. As the colors of dogs deviate from the "truck-colours" more significantly than they deviate from the "frog-colours", distinguishing trucks from dogs is accomplished better than distinguishing frogs from dogs.

4.2.2 Quantitative Results

Our quantitative comparisons are given in Tab. 6 and show that the anomaly detection for the *CIFAR10* dataset is substantially more challenging than for the *MNIST* dataset. Images of anomalous class *truck* and *car* (*CIFAR10*) are distinguished best form images of normal class *dog* by all autoencoders. The models perform worst for the classes *bird*, *cat*, *deer* which is reasonable if one considers that their colours are rather similar to the colour in the images of dogs. Surprisingly, the class *ship* was distinguished poorly as well. This is most likely due to the fact that, although there is a comparably high extent of blue colours in the images, the ships themselves are coloured in white, gray and brown colour tones. The AAE was outperformed by either the CAE or the VAE for all classes. The CAE performs best for all classes except for class *ship*. The training of the AAE takes the longest, followed by VAE and CAE training.

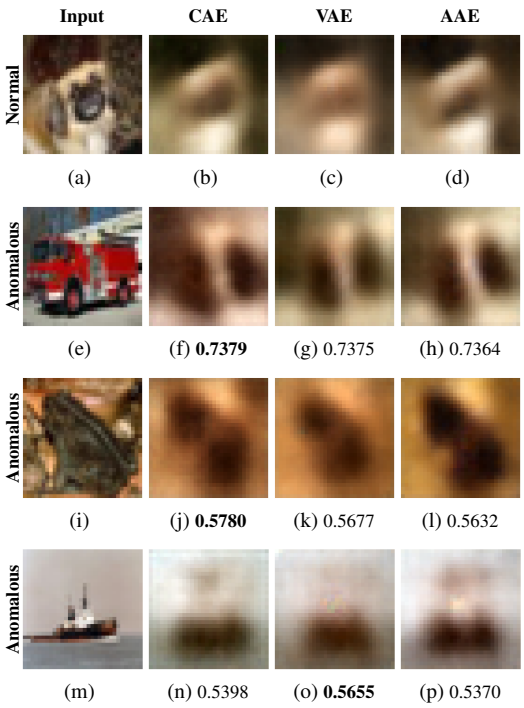

Figure 4. Illustration of the behaviour of our autoencoders on unknown classes of the *CIFAR10* dataset. The first row shows (unseen/untrained) images of non-anomalous class *dog*, the second includes images of anomalous class *truck*, the third row images of anomalous class *frog* and the fourth row images of anomalous class *ship*. In the first column are input images, in the second column are their reconstructions computed by the CAE. The reconstructions of VAE and AAE are in the third and fourth column, respectively. The area under curve for the ROC curve AUC_{ROC} is computed over all (unseen) images of normal class *dog* and images of the anomalous class to capture the quality of the distinction between normal and anomalous images and is given below the reconstructed images.

Table 6. AUC_{ROC} and AUC_{PR} for all autoencoder types on the *CIFAR10* dataset. Class *dog* is considered to be normal, all others anomalous. The last row shows the time needed for training the models ($time_{tr}$, in seconds).

	Type / Class	CAE		CAE		AAE	
		AUC_{ROC}	AUC_{PR}	AUC_{ROC}	AUC_{PR}	AUC_{ROC}	AUC_{PR}
	Plane	0.6343	0.6280	0.6448	0.6338	0.6338	0.6254
	Car	**0.7689**	**0.7389**	**0.7615**	**0.7259**	**0.7510**	**0.7180**
	Bird	0.5405	0.5482	0.5328	0.5353	0.5315	0.5363
	Cat	0.5525	0.5489	0.5421	0.5321	0.5447	0.5380
	Deer	0.5463	0.5459	0.5404	0.5344	0.5328	0.5246
	Frog	0.5780	0.5616	0.5677	0.5474	0.5632	0.5495
	Horse	0.6612	0.6345	0.6470	0.6185	0.6435	0.6255
	Ship	0.5398	0.5445	0.5655	0.5692	0.5370	0.5462
	Truck	0.7379	0.7132	0.7375	0.7073	0.7364	0.7120
$time_{tr}$	-	**89704.82**		99882.32		105906.03	

5. Conclusions

For this paper, we compared three types of convolutional autoencoders - a standard AE, a variational AE and an adversarial AE - on the well known *MNIST* and *CIFAR10* datasets. The averaged reconstruction error can be seen as an "anomaly score", as it will be higher for an image of an anomalous class than for an image of a normal class. By thresholding the averaged reconstruction error, we can classify an image to be normal or anomalous. The results computed by the different AE types are compared in a quantitative and qualitative manner.

For the *MNIST* dataset, using the proposed autoencoder types for detecting anomalous classes led to highly accurate results over all autoencoder types and all classes. However, the models struggle to distinguish normal from anomalous classes for the far more complex *CIFAR10* dataset. The results show systematic difficulties, which can be explained by the fact that the images of class *dog* are mainly composed of differently sized rounded regions with mostly white, brown, grey and black colours. As a result, the trained networks are relatively good at distinguishing trucks and cars from dogs as the vehicles appear in more colourful versions, but are unable to distinguish frogs from dogs since dogs and frogs appear in similar colours.

The CAE demands the simplest implementation and training procedure. This leads to the fastest measured training times. Training the VAE takes longer than training a CAE (see Tab. 5 and 6), due to the re-parametrization step and two innermost bottleneck layers. The AAE took the longest to train (see Tab. 5 and 6). This is of course reasonable, because we do not only train the encoder and decoder during the *reconstruction* phase, but the adversarial components during the *regularization* phase. Despite extended model and training complexity, the AAE and VAE models achieve only slightly better results than the CAE model for very few classes of *MNIST* and *CIFAR10* (Tab. 5 and Tab. 6).
The increase in anomaly detection performance is only marginal and not even present for some classes. Nevertheless, Tab. 3 provides insight on the fact that, although all three autoencoder types are based on the same principle of encoding and decoding information, their underlying concepts vary. The reconstruction results differ from each other (Fig. 3 and 4). This might be utilized in some way and is definitely an interesting starting point for further investigation. However, for some applications one might need to consider if it is justified to use a far more complex model if the results it computes are not significantly better.

The autoencoder showed high versatility and the underlying principle straightforward. For flawless ("good")

data, it is straightforward to compose a strong and diverse training set and the approach is therefore easily applicable. Although AEs are trained in an unsupervised manner, the compilation of a good representative training set can be demanding for real-word data. The results of the *CIFAR10* dataset show nicely that we depend on the fact that the network learns to extract the "relevant distinction features" - if the model does not capture that dogs have a snout, four legs and two ears, it will fail to properly separate dogs from other classes.

Our currently ongoing future work and follow-up experiments with autoencoders do not only treat images of one class as normal, but extend our approach and vary between different choices of classes to be considered normal. Additionally, we will investigate the defect detection performances of autoencoders in industrial inspection settings thoroughly.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 70:214–223, 2017. 3

[2] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *CoRR*, abs/2003.05991, 2020. 2

[3] Paul Bergmann, Sindy Löwe, Michael Fauser, David Sattlegger, and Carsten Steger. Improving unsupervised defect segmentation by applying structural similarity to autoencoders. *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, 2019. 2

[4] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997. 2

[5] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. *Proceedings of the 23rd International Conference on Machine Learning, ACM*, 06, 2006. 2, 4

[6] Alexander Diadiushkin, Kurt Sandkuhl, and Alexander Maiatin. Fraud detection in payments transactions: Overview of existing approaches and usage for instant payments. *Complex Systems Informatics and Modeling Quarterly*, pages 72–88, 2019. 1

[7] Oumayma Essid, Chafik Samir, and Laga Hamid. Automatic detection and classification of manufacturing defects in metal boxes. *PLOS ONE*, 2018. 2

[8] Tharindu Fernando, Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Deep learning for medical anomaly detection – a survey. *ACM Computing Surveys*, 54:1–37, 2021. 1

[9] Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 241–246, 2016. 2

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Y. Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3, 2014. 3

[11] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2014. 3

[12] Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. *Computing Research Repository (CoRR)*, abs/1312.6114, 2014. 2, 3

[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 1, 2, 3, 4, 5

[14] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. 1, 2, 3, 4

[15] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *ArXiv*, abs/1511.05644, 2016. 2, 3

[16] Ilja Manakov, Markus Rohm, and Volker Tresp. Walking the tightrope: An investigation of the convolutional autoencoder bottleneck. *ArXiv*, abs/1911.07460, 2019. 2

[17] Manpreet Singh Minhas and John Zelek. Semi-supervised anomaly detection using autoencoders. 2020. 2

[18] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection. *ACM Computing Surveys*, 54(2):1–38, 2021. 1

[19] Joseph Rocca. Understanding variational autoencoders (vaes) - building, step by step, the reasoning that leads to vaes. 2019. 2

[20] David E. Rumelhart and James L. McClelland. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, pages 318–362, 1987. 1, 2

[21] Nina Shvetsova, Bart Bakker, Irina Fedulova, Heinrich Schulz, and Dmitry V. Dylov. Anomaly detection in medical imaging with deep perceptual autoencoders. *IEEE Access*, 9:118571–118583, 2021. 2

[22] Jing Yang, Shaobo Li, Zheng Wang, Hao Dong, Jun Wang, and Shihao Tang. Using deep learning to detect defects in manufacturing: A comprehensive survey and current challenges. *Materials*, 13:5755, 2020. 1

[23] Ling-Li Zeng, Huaning Wang, Panpan Hu, Bo Yang, Weidan Pu, Hui Shen, Xingui Chen, Zhening Liu, Hong Yin, Qingrong Tan, Kai Wang, and Dewen Hu. Multi-site diagnostic classification of schizophrenia using discriminant deep learning with functional connectivity mri. *EBioMedicine*, 30:74–85, 2018. 2