Pattern Recognition Letters 178 (2024) 202–208

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Reducing redundancy in the bottleneck representation of autoencoders

Firas Laakom [a,*], Jenni Raitoharju [b,c], Alexandros Iosifidis [d], Moncef Gabbouj [a]

[a] Faculty of Information Technology and Communication Sciences, Tampere University, Finland
[b] Faculty of Information Technology, University of Jyväskylä, Finland
[c] Quality of Information, Finnish Environment Institute, Finland
[d] Department of Electrical and Computer Engineering, Aarhus University, Denmark

ARTICLE INFO

Editor: Alexandru C. Telea

MSC:
68T01
68T10
94A08
68T99

Keywords:
Autoencoders
Unsupervised learning
Diversity
Feature representation
Dimensionality reduction
Image denoising
Image compression

ABSTRACT

Autoencoders (AEs) are a type of unsupervised neural networks, which can be used to solve various tasks, e.g., dimensionality reduction, image compression, and image denoising. An AE has two goals: (i) compress the original input to a low-dimensional space at the bottleneck of the network topology using an encoder, (ii) reconstruct the input from the representation at the bottleneck using a decoder. Both encoder and decoder are optimized jointly by minimizing a distortion-based loss which implicitly forces the model to keep only the information in input data required to reconstruct them and to reduce redundancies. In this paper, we propose a scheme to explicitly penalize feature redundancies in the bottleneck representation. To this end, we propose an additional loss term, based on the pairwise covariances of the network units, which complements the data reconstruction loss forcing the encoder to learn a more diverse and richer representation of the input. We tested our approach across different tasks, namely dimensionality reduction, image compression, and image denoising. Experimental results show that the proposed loss leads consistently to superior performance compared to using the standard AE loss.

1. Introduction

With the progress of data gathering techniques, high-dimensional data are becoming available for training machine learning approaches. The impracticality of working in high dimensional spaces due to the *curse of dimensionality* and the understanding that the data in many problems reside on manifolds with much lower dimensions than those of the original space has led to the development of various approaches which try to learn a mapping of the data representations in the original space to more meaningful lower-dimensional representations.

Autoencoders (AEs) [1] are a powerful data-driven unsupervised approach used to learn a compact representation of a given input distribution. An autoencoder focuses solely on finding a low-dimensional representation, from which the input data can be reconstructed with minimal distortion. Autoencoders have been applied successfully in many tasks, such as transfer learning [2], anomaly detection [3], dimensionality reduction [4], and compression [5].

To accomplish these tasks, an autoencoder has two different parts: an encoder $g(\cdot)$, which maps an input $\boldsymbol{x} \in \mathcal{X}$ to a compact low-dimensional space $g(\boldsymbol{x})$, called the bottleneck representation, and a decoder $f(\cdot)$, which takes the output of the encoder as its input and uses it to reconstruct the original input $f \circ g(\boldsymbol{x})$. Given a distortion metric D: $\mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, which measures the difference between the original input and the reconstructed input. Autoencoders are trained in an end-to-end manner using gradient descent-based optimization [1] to minimize the loss L defined as the average distortion over the training data $\{\boldsymbol{x}_i\}_{i=1}^N$:

$$\min_{f,g} L\left(\{\boldsymbol{x}_i\}_{i=1}^N\right) \triangleq \min_{f,g} \frac{1}{N} \sum_{i=1}^{N} D(\boldsymbol{x}_i, f \circ g(\boldsymbol{x}_i)). \tag{1}$$

Several extensions and regularization techniques have been proposed to augment this loss [5,6] aiming at improving the mapping of the input to a compressed representation at the bottleneck of the autoencoder so that the original inputs can be better reconstructed from these compact representations using the decoder.

By controlling the size of the bottleneck, one can explicitly control the dimensionality of the representation and the compression rate [5]. A low size of the bottleneck increases the complexity of the task of the decoder risking a higher distortion rate. This trade-off forces the model to keep only those variations in the input data that are required

* Corresponding author.
E-mail addresses: firas.laakom@tuni.fi (F. Laakom), jenni.k.raitoharju@jyu.fi (J. Raitoharju), ai@ece.au.dk (A. Iosifidis), moncef.gabbouj@tuni.fi (M. Gabbouj).

https://doi.org/10.1016/j.patrec.2024.01.013
Received 28 November 2022; Received in revised form 5 January 2024; Accepted 10 January 2024
Available online 15 January 2024

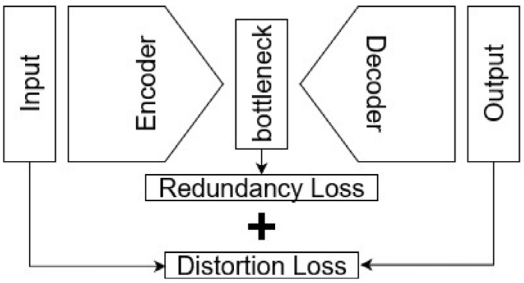

Fig. 1. An illustration of how the autoencoder loss is computed using our approach.

to reconstruct the input and to avoid redundancies and noise within the input. This is achieved implicitly by using error back-propagation for minimizing the reconstruction error, i.e., distortion D.

In the context of supervised neural networks, it has been shown that reducing redundancy improves generalization [7–9]. Approaches helping to reduce redundancy have been successfully applied, e.g., for pruning [10]. In this paper, we propose to model the feature redundancy in the bottleneck representation and minimize it explicitly. To this end, we propose augmenting the loss L using a redundancy term computed as the sum of the pairwise covariance between the bottleneck elements. The full scheme is illustrated in Fig. 1. We argue that explicitly penalizing the pairwise covariance between the different units in the bottleneck provides extra feedback for the encoder to avoid redundancy and to learn a richer representation of the input data.

The contributions of this paper can be summarized as follows:

- We propose a scheme to avoid redundant features in the bottleneck representation of autoencoders.
- We propose to augment the autoencoder loss to explicitly penalize the pairwise covariance between the features and learn a diverse compressed embedding of the training data.
- The proposed penalty term acts as an unsupervised regularizer on top of the encoder and can be integrated into any autoencoder-based model in a plug-and-play manner.
- The proposed method is extensively evaluated over three tasks: dimensionality reduction, image compression, and image denoising. Experimental results show consistent performance improvements compared to the standard approach.

The rest of this paper is organized as follows. Section 2 provides the background of autoencoders' training strategies and a brief review of different tasks considered in this work, i.e., dimensionality reduction, image compression, and image denoising. Section 3 describes the proposed approach. Section 4 reports experimental results for the dimensionality reduction task on the Madelon [11], ISOLET [12], and P53 Mutants [13] datasets. Section 5 reports experimental results for the image compression task on the MNIST [14] and CIFAR10 [15] datasets. Section 6 evaluates our approach on the image denoising task using the fashion MNIST [16] and CIFAR10 datasets. Section 7 concludes the paper.

2. Related work

Autoencoders are models trained to reconstruct their input, i.e., to approximate the identity function $f(x) \approx x$. While the identity function seems a particularly trivial function to learn, enforcing certain constraints on the network topology and particularly using a low number of units in the hidden layers [1] forces the model to learn to efficiently represent the data in a much lower-dimensional space compared to the original space [17,18]. This is a desired property in several tasks, e.g., dimensionality reduction [4], compression [5,19], and image denoising [20,21]. In [3,22–24], different extensions of autoencoders have been proposed to improve their performance in different contexts.

Several approaches based on reducing redundancy have been proposed recently in different contexts [25–29]. In particular, in the context of self-supervised learning, [26,28] proposed a training loss based on the pairwise correlation between the features of two perturbed variants of the same input. [9] proposed a data-dependent regularizer based on the L_2 distance between the units outputs in the last hidden layer of CNNs and showed that such approach can reduce overfitting in the context of supervised learning. In this paper, we explore a similar direction in the context of unsupervised learning with autoencoders. To the best of our knowledge, this is the first work that considers reducing redundancy in this context. We show that it helps improve the performance.

Dimensionality reduction refers to the problem of learning a mapping from a high-dimensional input space $\mathcal{X} \in \mathbb{R}^D$ into a lower-dimensional space $\mathcal{Z} \in \mathbb{R}^d$, where $d \ll D$, while preserving features of interest in the input data. Several linear [30–32] and non-linear [33–36] approaches have been proposed to solve this task. Some are supervised approaches, such as Linear Discriminant Analysis (LDA) and its extensions [37], others are unsupervised methods [38], such as Principal Component Analysis (PCA) [39]. Dimensionality reduction is the most straightforward application of autoencoders [4,40], as the mapping can be learned using an autoencoder by setting the size of the bottleneck to d units and training the model to reconstruct the input.

Image compression is an important task in many applications. Recent advances in deep neural networks [1] have enabled efficient modeling of high-dimensional data and led to outperforming traditional image compression techniques [41,42]. Recently, there has been interest in autoencoders to solve this task [5] due to their flexibility and easiness of training.

Image denoising [43] refers to the task of trying to restore a clean version of the image from its noisy corrupted counterpart. Due to their plug-and-play network architectures, CNN-based autoencoders have been widely adopted to solve this task [44,45]. In particular, an autoencoder is trained using pairs of noisy and clean images. By taking a noisy sample as input and setting its clean version as the target, the model learns to keep only the important information from the image and discard the noise.

3. Reducing the pairwise covariance within the bottleneck representation

Autoencoders are a special type of neural networks trained to achieve two objectives: (i) to compress an input into a low-dimensional space, (ii) to reconstruct the original input from the low-dimensional representation. This is achieved by minimizing the reconstruction loss over the training data, which implicitly forces learning a concise 'non-redundant' representation of the data. In this paper, we propose to augment the reconstruction loss with an additional term designed to explicitly minimize redundancy between the features learned at the bottleneck.

Given a training data set $\{\boldsymbol{x}_i\}_{i=1}^N$ and an encoder $g(\cdot) \in \mathbb{R}^D$, the covariance between the ith and jth features, g_i and g_j, can be expressed as follows:

$$C(g_i, g_j) = \frac{1}{N} \sum_n \Big(g_i(\boldsymbol{x}_n) - \mu_i \Big) \Big(g_j(\boldsymbol{x}_n) - \mu_j \Big), \tag{2}$$

where $\mu_i = \frac{1}{N} \sum_n g_i(\boldsymbol{x}_n)$ is the average output of the ith unit. Our aim is to minimize the redundancy of the bottleneck representations which corresponds to minimizing the pairwise covariance between different features. Thus, we augment the loss $L\big(\{\boldsymbol{x}_i\}_{i=1}^N\big)$ as follows:

$$\begin{aligned} L\big(\{\boldsymbol{x}_i\}_{i=1}^N\big)_{aug} &\triangleq L\big(\{\boldsymbol{x}_i\}_{i=1}^N\big) + \alpha \sum_{i \neq j} C(g_i, g_j) \\ &= \frac{1}{N} \sum_{i=1}^N D(\boldsymbol{x}_i, f \circ g(\boldsymbol{x}_i)) \end{aligned} \tag{3}$$

Table 1
Statistics of the three datasets used in the dimensionality reduction experiments. # Dim: dimensionality of the data. # Train: number of training samples. # Test: number of test samples. d: projection dimension.

Dataset	# Dim	# Train	# Test	d
Madelon [11]	500	2000	1800	10
ISOLET [12]	617	6238	1559	10
P53 Mutants [13]	5408	21 811	9348	50

$$+ \alpha \sum_{i \neq j} \Big(\frac{1}{N} \sum_{n} (g_i(\boldsymbol{x}_n) - \mu_i)(g_j(\boldsymbol{x}_n) - \mu_j) \Big), \tag{4}$$

where α is a hyper-parameter used to control the contribution of the additional term in the total loss. L_{aug} is composed of two terms, the first term is the standard autoencoder loss that depends on both the encoder and decoder parts to ensure that the autoencoder learns to reconstruct the input, while the second term depends only on the encoder and its aim is to promote the diversity of the learned features.

Intuitively, the proposed approach acts as an unsupervised regularizer on top of the encoder providing extra feedback during training to reduce the redundancy of the encoder's output. The proposed scheme can be embedded into any autoencoder-based model as a plug-in and optimized in a batch-manner, i.e., at each optimization step, we can compute the pairwise covariance using the mini-batch samples. Moreover, it is suitable for different learning strategies and different topologies.

4. Experiments on dimensionality reduction

In this section, we consider the dimensionality reduction task using an autoencoder. We test the proposed approach using three different tabular datasets, namely Madelon [11], ISOLET [12], and P53 Mutants [13]. The Madelon dataset [11] contains samples represented by 500-dimensional vectors grouped in 32 clusters placed on the vertices of a five-dimensional hypercube. The ISOLET dataset [12] is composed of alphabet-speech data from 150 different subjects. Each instance is represented by a 617-feature vector complied using spectral coefficients, contour features, sonorant features, pre-sonorant features, and post-sonorant features. The P53 Mutants dataset [13] is a large Biophysical dataset with more than 30k samples in total and 5408 attributes per instance. The feature representation is formed by combining the 2D electrostatic and surface-based attributes with the 3D distance-based attributes.

As the autoencoder topology, we use a simple architecture where the encoder maps the input using two intermediate fully-connected layers composed of 64 units with ReLU activation. Then, the bottleneck representation of size d is obtained using a fully-connected layer with d units and Leaky ReLU [1] activation. Symmetrically, the decoder is composed of two 64-dimensional fully-connected layers followed by ReLU activation and an output layer with the same size as the input using a sigmoid activation. The number of data dimensions, cardinalities of the training and test sets, and the value of d for each dataset is specified in Table 1. For training, we use the Adam optimizer with a learning rate of 10^{-2} and the mean square error as the standard training loss L. The number of epochs and the batch size are set to 50 and 32, respectively, in all experiments. Each experiment is repeated 10 times and the mean and standard deviation of the root mean square error (RMSE) on the test set are reported.

In Table 2, we report the experimental results obtained by training the autoencoder using the standard loss and our proposed augmented loss and different values for the hyper-parameter α, introduced in (4). It can be seen that, by explicitly penalizing redundancy in the bottleneck representations, the proposed approach consistently achieves lower errors compared to the standard approach on the three datasets. On the Madelon dataset, the best performance is achieved using $\alpha = 0.005$. On the ISOLET dataset, using $\alpha = 0.1$ leads to the highest improvement,

Table 2
Reconstruction error on the three datasets used in the dimensionality reduction experiments (average and standard deviation over 10 repetitions).

	Madelon	ISOLET	P53 Mutants
Standard	0.14027 ± 0.00023	0.13143 ± 0.00259	0.02777 ± 0.00159
Ours (0.1)	0.14022 ± 0.00016	**0.12993 ± 0.00283**	0.02717 ± 0.00087
Ours (0.05)	0.14024 ± 0.00038	0.13081 ± 0.00366	**0.02689 ± 0.00054**
Ours (0.01)	0.14022 ± 0.00043	0.13101 ± 0.00204	0.02709 ± 0.00052
Ours (0.005)	**0.14005 ± 0.00037**	0.13135 ± 0.00267	0.02694 ± 0.00051

whereas, on the P53 Mutants dataset, the best performance is achieved using $\alpha = 0.05$. It should be noted that while the performance gap is not large compared to the standard approach, the improvement is consistent on all the datasets and the different regularization rates.

In Fig. 2, we provide visualization results comparing the two approaches. We visualize the data in the projected space of the AE trained with the standard loss and the proposed augmented loss using t-SNE [35]. As can be seen, the AE trained with the augmented loss provides a more compact representation of the classes. We also note that by reducing redundancy, the learned embedding is more spread over the projection space and contains fewer empty regions.

Dimensionality reduction is typically applied as a pre-processing step to compile a compact feature representation that can be used to solve another task, such as classification. Intuitively, learning diverse and non-redundant features is crucial to achieve good performance on the task of interest. Here, to further assess the quality of the data representations learned using our approach, we conduct an extra experiment by applying the K-Nearest Neighbor (K-NN) classifier on top of the bottleneck features. In Table 3, we report the classification accuracy for $K = 3$ and $K = 5$. As can be seen, the bottleneck features obtained using our approach yield consistently higher accuracy on the three datasets. For example, for the Madelon dataset, using $\alpha = 0.005$ leads to 4.49% and 3.46% accuracy improvement compared to the standard approach when using $K = 3$ and $K = 5$, respectively. It is interesting to note also that using the augmented loss consistently leads to more stable performance compared to the standard approach, as shown by the lower variances.

Moreover, for comparative purposes, we report results obtained by using WLD-reg [7], i.e., replacing the proposed regularizer with the direct variant of WLD-reg on top of the bottleneck representation.[1] Consistently with our results, WLD-reg also boosts performance compared to the standard approach showing that reducing redundancy in the bottleneck representation indeed helps to learn better features. Compared to our approach, we also note that the use of pairwise covariance instead of the L_2 distance leads to higher performance on all three datasets.

To further study the effect of the number of dimensions at the bottleneck on performance, we conducted an additional experiment using the Isolet dataset. We plot the average accuracy over training the AE using 10 random seeds for different values of d in Fig. 2 (right). As can be seen, reducing redundancy improves performance for the different bottleneck sizes. It is interesting to note also that the performance gap is larger for small values of d. This can be explained by the fact that, when a smaller number of dimensions is used, it is more crucial to learn diverse features for solving the task.

5. Experiments on image compression

In this section, we consider the image compression task using an autoencoder. We start by testing the proposed approach on the MNIST

[1] We note that WLD-reg [7] is a diversity promoting regularizer designed to be added on top of the last intermediate layer of a neural network in a standard supervised learning setting and not designed for unsupervised learning on top of the bottleneck of an autoencoder.

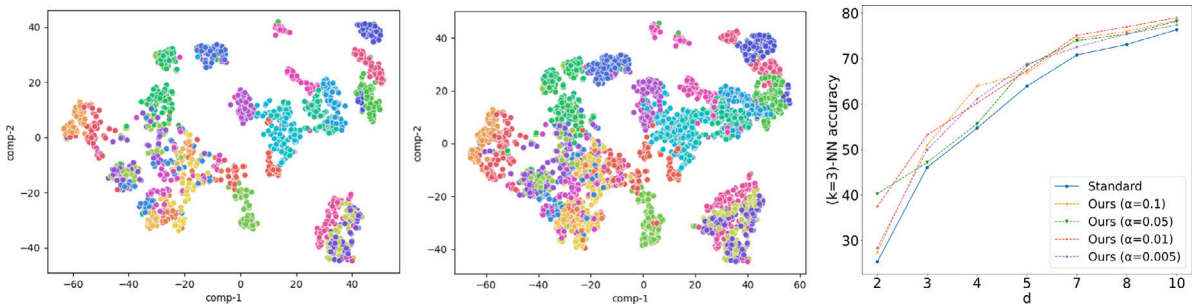

Fig. 2. t-SNE-based visualization of the ISOLET representations obtained by an AE trained by the standard approach (left) and the proposed approach (middle). Each color corresponds to data from a specific class. Average ($K = 3$)-NN accuracy as a function of the dimension of the bottleneck size d (right).

Table 3
Classification accuracy of Nearest Neighbor classifier applied on the bottleneck representations (average and standard deviation over 10 repetitions).

	Madelon	
	($K = 3$)-NN	($K = 5$)-NN
Standard	69.33% ± 2.71	71.32% ± 2.82
WLD-reg [7]	70.83% ± 2.08	72.45% ± 2.01
Ours (0.1)	72.51% ± 1.73	74.08% ± 1.63
Ours (0.05)	73.52% ± 1.91	74.53% ± 1.49
Ours (0.01)	72.65% ± 2.21	74.50% ± 1.87
Ours (0.005)	**73.82%** ± **1.83**	**74.78%** ± **1.78**
	ISOLET	
	($K = 3$)-NN	($K = 5$)-NN
Standard	76.32% ± 1.85	77.70% ± 1.60
WLD-reg [7]	78.22% ± 0.64	79.73% ± 0.70
Ours (0.1)	78.35% ± 0.46	79.82% ± 0.47
Ours (0.05)	78.18% ± 0.40	79.43% ± 0.44
Ours (0.01)	**78.96%** ± **0.56**	**79.83%** ± **0.54**
Ours (0.005)	77.34% ± 0.66	79.29% ± 0.66
	P53 Mutants	
	($K = 3$)-NN	($K = 5$)-NN
Standard	56.42% ± 0.60	54.99% ± 0.48
WLD-reg [7]	56.29% ± 0.36	54.86% ± 0.46
Ours (0.1)	**57.88%** ± **0.46**	**56.18%** ± **0.59**
Ours (0.05)	56.17% ± 0.46	55.39% ± 1.09
Ours (0.01)	57.22% ± 0.50	55.65% ± 0.46
Ours (0.005)	56.83% ± 0.41	55.92% ± 0.46

Table 4
RMSE, PSNR, and SSIM on the MNIST dataset (average and standard deviation over 5 repetitions).

	RMSE ↓	PSNR ↑	SSIM ↑
	784 → 256		
Standard	0.0518 ± 0.0005	0.9631 ± 0.0016	26.43 ± 0.09
Ours (0.1)	0.0508 ± 0.0005	0.9641 ± 0.0010	26.57 ± 0.11
Ours (0.05)	0.0508 ± 0.0005	0.9636 ± 0.0007	26.58 ± 0.08
Ours (0.01)	0.0513 ± 0.0005	**0.9647** ± **0.0013**	26.49 ± 0.09
Ours (0.005)	**0.0506** ± **0.0007**	0.9635 ± 0.0012	**26.61** ± **0.10**
	784 → 64		
Standard	0.0596 ± 0.0021	0.9597 ± 0.0022	25.25 ± 0.29
Ours (0.1)	**0.0584** ± **0.0010**	**0.9607** ± **0.0012**	**25.42** ± **0.16**
Ours (0.05)	0.0588 ± 0.0018	0.9604 ± 0.0017	25.38 ± 0.25
Ours (0.01)	0.0593 ± 0.0010	0.9599 ± 0.0012	25.30 ± 0.15
Ours (0.005)	0.0588 ± 0.0009	0.9602 ± 0.0013	25.35 ± 0.13

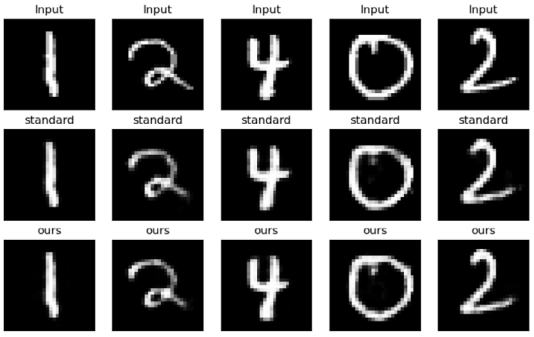

Fig. 3. Visualization of digits reconstructed by an AE trained by using the standard and the proposed training approaches. The first row contains the original inputs. Their reconstructed versions corresponding to the standard approach are shown in the second row, and the proposed approach in the third row.

dataset [14]. It contains grayscale images with resolution of 28 × 28 pixels, which are vectorized to form 784-dimensional vectors. The dataset is split in 50,000 training and 10,000 test images.

For the autoencoder model, we use a simple architecture. The encoder is composed of two fully-connected layers composed of 256 and 128 units, respectively. The final output of the encoder is composed of d units, where d is the size of the bottleneck. Similarly, the decoder part takes the encoder's output, maps it to an intermediate layer of 128 units, then 256 units, and outputs a 784-vector. In all the layers, we use ReLU activation except for the final layer, where sigmoid activation is used.

For training, we use the Adam optimizer with a learning rate of 10^{-2} and the mean square loss as the standard training loss L. We train using 80% of the images in the original training set and hold the remaining 20% of the images as a validation set. During training, the model with the lowest mean square error on the validation set is saved and used in the test phase. We repeat each experiment five times and report the mean and standard deviation of the root-mean-square error (RMSE) errors, the peak signal-to-noise ratio (PSNR), and structural index similarity (SSIM) scores on the test set for the different approaches. We experiment with two different bottleneck sizes, i.e., $d = 256$ and $d = 64$. The results for different bottleneck sizes are reported in Table 4.

We note that the proposed approach consistently improves performance compared to training with the standard loss, i.e., it leads to lower RMSE values and higher PSNR and SSMI scores. For $d = 256$, the lowest RMSE value is achieved using $\alpha = 0.005$, and the highest PSNR and SSMI scores are obtained using $\alpha = 0.01$ and $\alpha = 0.005$, respectively. For $d = 64$, using $\alpha = 0.1$ leads to the best performance across all the metrics. Fig. 3 provides visualization results of images reconstructed from the representations learned using our approach for $d = 64$. We note that using the proposed augmented loss to train the AE leads to reconstructed inputs with lower distortion.

We also evaluate our approach on the image compression task with a more challenging dataset, namely the CIFAR10 Dataset [15]. The dataset contains 32 × 32-pixel color images, which are vectorized to form 3,072-dimensional vectors. For the model topology, we use two hidden layers with 512 and 256 units and ReLU activation. For the bottleneck size d, we experiment with two configurations, $d = 128$ and $d = 256$. All the models are trained with Adam optimizer with a 10^{-2} learning rate and a batch size of 128 for 50 epochs. The average and standard deviation of the different metrics over 10 random seeds are provided in Table 5. Similar to the results on the MNIST dataset,

Table 5
RMSE, PSNR, and SSIM on the CIFAR10 dataset (average and standard deviation over 5 repetitions).

	RMSE ↓	PSNR ↑	SSIM ↑
	3072 → 256		
Standard	0.0888 ± 0.0022	0.6601 ± 0.0068	21.5547 ± 0.2470
Ours (0.01)	0.0882 ± 0.0012	0.6637 ± 0.0064	21.6168 ± 0.1314
Ours (0.005)	0.0882 ± 0.0006	0.6628 ± 0.0060	21.6271 ± 0.0593
Ours (0.001)	0.0882 ± 0.0011	0.6613 ± 0.0068	21.6347 ± 0.1154
Ours (0.0005)	**0.0877 ± 0.0011**	**0.6642 ± 0.0078**	**21.6829 ± 0.1156**
Ours (0.0001)	0.0885 ± 0.0014	0.6610 ± 0.0060	21.5927 ± 0.1518
	3072 → 128		
Standard	0.0929 ± 0.0015	0.6151 ± 0.0100	21.2830 ± 0.1455
ours (0.01)	0.0920 ± 0.0010	0.6210 ± 0.0078	21.3765 ± 0.0920
ours (0.005)	0.0927 ± 0.0014	0.6144 ± 0.0089	21.3093 ± 0.1280
ours (0.001)	**0.0917 ± 0.0009**	**0.6246 ± 0.0054**	**21.4150 ± 0.0888**
ours (0.0005)	0.0923 ± 0.0016	0.6190 ± 0.0130	21.3436 ± 0.1527
ours (0.0001)	0.0926 ± 0.0019	0.6182 ± 0.0072	21.3184 ± 0.2070

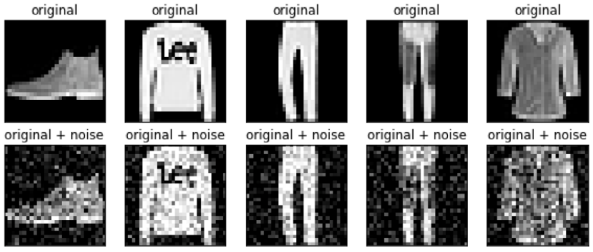

Fig. 4. Original samples from the fashion MNIST dataset (top), and their noisy versions using $\beta = 0.2$ (bottom).

we note that the proposed approach consistently leads to performance improvements. For $d = 256$, the best performance is achieved by using $\alpha = 0.0005$, whereas for $d = 128$, $\alpha = 0.001$ leads to the best performance.

6. Experiments on image denoising

In this section, we consider the image denoising task using an autoencoder. We test the proposed approach using the fashion MNIST dataset [16], which is an image dataset composed of 10 classes. Each sample is a 28×28 gray-scale image. The dataset has a total of 60,000 training samples and 10,000 test samples. To construct a noisy dataset, we add a random noise from the normal distribution $\beta \times \mathcal{N}(0, 1)$, where β is a hyper-parameter controlling the noise rate. In Fig. 4, we provide examples of original images and their noisy versions.

As the autoencoder model, we use a simple CNN-based architecture. The encoder is composed of two convolutional layers, each of which has 16 and 4 filters, respectively, with kernel size 3×3. Symmetrically, the decoder is composed of two transposed convolutional layers of sizes 4 and 16 and a final convolutional layer with one filter with kernel size 3×3. All the layers have ReLU activation function except for the last layer where we use a sigmoid activation. Each model is trained for 50 epochs using the mean square error loss and Adam optimizer. We repeat each experiment five times and report the mean and standard deviation of RMSE, PSNR, and SSMI scores for different noise rates.

In Table 6, we report the experimental results for three different noise rates, i.e., $\beta = 0.1$, $\beta = 0.2$, and $\beta = 0.4$. Except for the hyper-parameter $\alpha = 0.01$ with noise rates $\beta = 0.2$ and $\beta = 0.4$, we note that our approach by explicitly minimizing the redundancy constantly outperforms the standard approach across all metrics. For the noise rate $\beta = 0.1$, the lowest RMSE value and the highest SSMI score are achieved using our approach with $\alpha = 0.05$, while the best PSNR is achieved with $\alpha = 0.005$. For $\beta = 0.2$, the best scores across all metrics correspond to $\alpha = 0.005$. For the extreme level of noise case, i.e., $\beta = 0.4$, our approach

Table 6
RMSE, PSNR, and SSIM on the fashion MNIST dataset (average and standard deviation over 5 repetitions).

	RMSE ↓	PSNR ↑	SSIM ↑
	$\beta = 0.1$		
Standard	0.0796 ± 0.0016	0.7980 ± 0.0061	22.51 ± 0.19
Ours (0.1)	0.0786 ± 0.0009	0.8018 ± 0.0024	22.64 ± 0.13
Ours (0.05)	**0.0772 ± 0.0018**	0.8049 ± 0.0062	**22.84 ± 0.25**
Ours (0.01)	0.0779 ± 0.0019	0.8047 ± 0.0066	22.77 ± 0.27
Ours (0.005)	0.0774 ± 0.0012	**0.8058 ± 0.0044**	22.82 ± 0.18
	$\beta = 0.2$		
Standard	0.0941 ± 0.0026	0.7283 ± 0.0110	20.95 ± 0.25
Ours (0.1)	0.0934 ± 0.0021	0.7301 ± 0.0102	21.03 ± 0.19
Ours (0.05)	0.0933 ± 0.0020	0.7290 ± 0.0079	21.04 ± 0.19
Ours (0.01)	0.0975 ± 0.0034	0.7143 ± 0.1276	20.63 ± 0.31
Ours (0.005)	**0.0922 ± 0.0012**	**0.7357 ± 0.0058**	**21.14 ± 0.13**
	$\beta = 0.4$		
Standard	0.1262 ± 0.0021	0.5901 ± 0.0089	18.27 ± 0.16
Ours (0.1)	**0.1258 ± 0.0021**	**0.5954 ± 0.0095**	**18.30 ± 0.15**
Ours (0.05)	0.1260 ± 0.0016	0.5946 ± 0.0067	18.28 ± 0.12
Ours (0.01)	0.1266 ± 0.0014	0.5865 ± 0.0070	18.22 ± 0.09
Ours (0.005)	0.1260 ± 0.0017	0.5911 ± 0.0085	18.28 ± 0.13

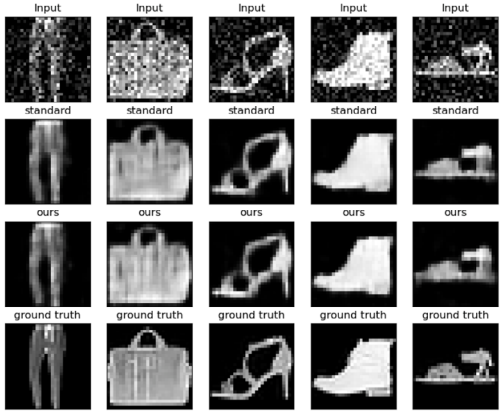

Fig. 5. Visualization of images denoised by AEs trained by using the standard and the proposed training approaches. The first row contains the original inputs. Their denoised versions corresponding to the standard approach are shown in the second row and the proposed approach in the third row. The last row contains the ground truth.

with $\alpha = 0.1$ achieves the best performance across the three metrics. In Fig. 5, we present visual outputs for our approach. As shown, our approach learns to efficiently discard the noise from the input images.

Next, we evaluate the performance of the proposed approach in image denoising with a more challenging dataset, i.e., CIFAR10. We use the same model topology and experimental protocol used for this dataset in Section 5. We experiment with two levels of noise $\beta = 0.1$ and $\beta = 0.2$. The results over 10 random seeds are presented in Table 7. As can be seen in Table 7, reducing features' redundancy in the bottleneck improves the performance of AE for both noise levels. For $\beta = 0.1$, using the augmented loss with $alpha = 0.005$ achieved the best performance, while for the high noise rate, i.e., $\beta = 0.2$, $\alpha = 0.0001$ led to the best performance across the three metrics. With the same hardware configuration, the standard autoencoder average training time is on average $1,297.7$ milliseconds per epoch, whereas using our approach takes on average $1,301.9$ milliseconds per epoch. So adding our regularizer leads to performance improvement with less than 0.33% additional time cost.

7. Conclusion

In this paper, we proposed a scheme for modeling redundancies at the bottleneck of an autoencoder. We proposed to complement

Table 7
RMSE, PSNR, and SSIM on the CIFAR10 dataset (average and standard deviation over 5 repetitions).

	RMSE ↓	PSNR ↑	SSIM ↑
	$\beta = 0.1$		
Standard	0.0954 ± 0.0019	0.6098 ± 0.0121	20.9243 ± 0.1734
Ours (0.01)	0.0948 ± 0.0016	0.6172 ± 0.0093	20.9703 ± 0.1550
Ours (0.005)	**0.0940 ± 0.0010**	**0.6227 ± 0.0056**	**21.0411 ± 0.1021**
Ours (0.001)	0.0952 ± 0.0018	0.6129 ± 0.0116	20.9325 ± 0.1651
Ours (0.0005)	0.0943 ± 0.0012	0.6190 ± 0.0085	21.0238 ± 0.1097
Ours (0.0001)	0.09489 ± 0.0012	0.6157 ± 0.0072	20.9644 ± 0.1102
	$\beta = 0.2$		
Standard	0.1001 ± 0.0012	0.5798 ± 0.0081	20.4497 ± 0.0972
Ours (0.01)	0.0996 ± 0.0013	0.5846 ± 0.0089	20.4900 ± 0.1155
Ours (0.005)	0.1000 ± 0.0015	0.5806 ± 0.0104	20.4597 ± 0.1118
Ours (0.001)	0.0999 ± 0.0014	0.5824 ± 0.0090	20.4626 ± 0.1118
Ours (0.0005)	0.0997 ± 0.0015	0.5814 ± 0.0111	20.4881 ± 0.1206
Ours (0.0001)	**0.0992 ± 0.0015**	**0.5884 ± 0.0081**	**20.5186 ± 0.1370**

the training loss with an extra regularization term, which explicitly penalizes the pairwise covariances of the units at the encoder's output and, thus, forces it to learn more diverse and compact representations for the input samples. The proposed approach can be interpreted as an unsupervised regularizer on top of the encoder and can be integrated into any autoencoder-based model in a plug-and-play manner. We empirically demonstrated the effectiveness of our approach across multiple tasks, namely dimensionality reduction, compression, and denoising. We showed that it improves performance compared to the standard approach, with minimal training time cost increase. Even though the proposed regularizer consistently improves the performance of autoencoders, its key limitation is the marginal improvement in certain tasks, as shown in the results, e.g., Table 2. Future directions include proposing more efficient redundancy modeling techniques to further improve the performance of autoencoders and exploring redundancy reduction strategies for variational autoencoders.

CRediT authorship contribution statement

Firas Laakom: Conceptualization, Methodology, Writing – original draft. **Jenni Raitoharju:** Supervision, Writing – review & editing. **Alexandros Iosifidis:** Supervision, Writing – review & editing. **Moncef Gabbouj:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work has been supported by the Academy of Finland Awcha project DN 334566 and NSF-Business Finland Center for Big Learning project AMALIA. The work of Jenni Raitoharju was supported by the Academy of Finland (projects 324475 and 333497).

References

[1] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep Learning, MIT Press, 2016.
[2] F. Zhuang, X. Cheng, P. Luo, S.J. Pan, Q. He, Supervised representation learning: Transfer learning with deep autoencoders, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[3] C. Zhou, R.C. Paffenroth, Anomaly detection with robust deep autoencoders, in: 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017.
[4] S. Petscharnig, M. Lux, S. Chatzichristofis, Dimensionality reduction for image features using deep learning and autoencoders, in: The 15th International Workshop on Content-Based Multimedia Indexing, 2017.
[5] L. Theis, W. Shi, A. Cunningham, F. Huszár, Lossy image compression with compressive autoencoders, 2017, arXiv preprint arXiv:1703.00395.
[6] G.D. Cavalcanti, L.S. Oliveira, T.J. Moura, G.V. Carvalho, Combining diversity measures for ensemble pruning, Pattern Recognit. Lett. (2016).
[7] F. Laakom, J. Raitoharju, A. Iosifidis, M. Gabbouj, WLD-reg: A data-dependent within-layer diversity regularizer, in: the 37th AAAI Conference on Artificial Intelligence, 2023.
[8] M. Cogswell, F. Ahmed, R.B. Girshick, L. Zitnick, D. Batra, Reducing overfitting in deep networks by decorrelating representations, in: International Conference on Learning Representations, 2016.
[9] F. Laakom, J. Raitoharju, A. Iosifidis, M. Gabbouj, On feature diversity in energy-based models, in: Energy Based Models Workshop-ICLR, 2021.
[10] H. Ide, T. Kobayashi, K. Watanabe, T. Kurita, Robust pruning for efficient CNNs, Pattern Recognit. Lett. (2020).
[11] I. Guyon, Madelon, 2008, UCI Machine Learning Repository.
[12] R. Cole, M. Fanty, ISOLET, 1994, UCI Machine Learning Repository.
[13] R. Lathrop, p53 Mutants, 2010, UCI Machine Learning Repository.
[14] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE (1998).
[15] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, Technical report, University of Toronto, 2009.
[16] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017, arXiv preprint arXiv:1708.07747.
[17] J. Guo, X. Yuan, P. Xu, H. Bai, B. Liu, Improved image clustering with deep semantic embedding, Pattern Recognit. Lett. (2020).
[18] Y. Sang, J. Sang, M.S. Alam, Image encryption based on logistic chaotic systems and deep autoencoder, Pattern Recognit. Lett. (2022).
[19] A. Golinski, R. Pourreza, Y. Yang, G. Sautiere, T.S. Cohen, Feedback recurrent autoencoder for video compression, in: Asian Conference on Computer Vision, 2020.
[20] X. Ye, L. Wang, H. Xing, L. Huang, Denoising hybrid noises in image with stacked autoencoder, in: 2015 IEEE International Conference on Information and Automation, IEEE, 2015.
[21] L. Gondara, Medical image denoising using convolutional denoising autoencoders, in: 2016 IEEE 16th International Conference on Data Mining Workshops, ICDMW, IEEE, 2016.
[22] M. Patacchiola, P. Fox-Roberts, E. Rosten, Y-autoencoders: Disentangling latent representations via sequential encoding, Pattern Recognit. Lett. (2020).
[23] J. Deng, Z. Zhang, E. Marchi, B. Schuller, Sparse autoencoder-based feature transfer learning for speech emotion recognition, in: Humaine Association Conference on Affective Computing and Intelligent Interaction, 2013.
[24] P. Baldi, Autoencoders, unsupervised learning, and deep architectures, in: ICML Workshop on Unsupervised and Transfer Learning, JMLR Workshop and Conference Proceedings, 2012.
[25] A. Jeffares, T. Liu, J. Crabbé, F. Imrie, M. van der Schaar, TANGOS: Regularizing tabular neural networks through gradient orthogonalization and specialization, 2023, arXiv preprint arXiv:2303.05506.
[26] J. Zbontar, L. Jing, I. Misra, Y. LeCun, S. Deny, Barlow twins: Self-supervised learning via redundancy reduction, in: The 38th International Conference on Machine Learning, 2021.
[27] F. Laakom, J. Raitoharju, A. Iosifidis, M. Gabbouj, Efficient CNN with uncorrelated bag of features pooling, in: 2022 IEEE Symposium Series on Computational Intelligence, SSCI, IEEE, 2022.
[28] A. Bardes, J. Ponce, Y. LeCun, Vicreg: Variance-invariance-covariance regularization for self-supervised learning, 2021, arXiv preprint arXiv:2105.04906.
[29] F. Laakom, J. Raitoharju, A. Iosifidis, M. Gabbouj, Learning distinct features helps, provably, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2023.
[30] W. Zhao, R. Chellappa, P.J. Phillips, Subspace Linear Discriminant Analysis for Face Recognition, Citeseer, 1999.
[31] Y. Koren, L. Carmel, Robust linear dimensionality reduction, IEEE Trans. Vis. Comput. Graph. (2004).
[32] F. Laakom, J. Raitoharju, N. Passalis, A. Iosifidis, M. Gabbouj, Graph embedding with data uncertainty, IEEE Access (2022).
[33] D. DeMers, G.W. Cottrell, Non-linear dimensionality reduction, in: Advances in Neural Information Processing Systems, Citeseer, 1993.
[34] Y.-R. Yeh, S.-Y. Huang, Y.-J. Lee, Nonlinear dimension reduction with kernel sliced inverse regression, IEEE Trans. Knowl. Data Eng. (2008).
[35] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. (2008).
[36] L. McInnes, J. Healy, J. Melville, Umap: Uniform manifold approximation and projection for dimension reduction, 2018, arXiv preprint arXiv:1802.03426.

[37] A. Iosifidis, A. Tefas, I. Pitas, On the optimal class representation in linear discriminant analysis, IEEE Trans. Neural Netw. Learn. Syst. (2013).
[38] A.C. Kumar, Analysis of unsupervised dimensionality reduction techniques, Comput. Sci. Inf. Syst. (2009).
[39] S. Wold, K. Esbensen, P. Geladi, Principal component analysis, Chemometr. Intell. Laboratory Syst. (1987).
[40] S.A. Thomas, A.M. Race, R.T. Steven, I.S. Gilmore, J. Bunch, Dimensionality reduction of mass spectrometry imaging data using autoencoders, in: IEEE Symposium Series on Computational Intelligence, SSCI, 2016.
[41] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, M. Covell, Full resolution image compression with recurrent neural networks, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017.
[42] J. Ballé, V. Laparra, E.P. Simoncelli, End-to-end optimization of nonlinear transform codes for perceptual quality, in: 2016 Picture Coding Symposium, PCS, IEEE, 2016.
[43] K. Gupta, S. Gupta, Image denoising techniques-a review paper, IJITEE (2013).
[44] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, C.-W. Lin, Deep learning on image denoising: An overview, Neural Netw. (2020).
[45] J. Garcia-Gonzalez, J.M. Ortiz-de Lazcano-Lobato, R.M. Luque-Baena, M.A. Molina-Cabello, E. López-Rubio, Foreground detection by probabilistic modeling of the features discovered by stacked denoising autoencoders in noisy video sequences, Pattern Recognit. Lett. (2019).