Affine Equivariant Autoencoder

Xifeng Guo[1] , **En Zhu**[1*] , **Xinwang Liu**[1*] and **Jianping Yin**[2]
[1]College of Computer, National University of Defense Technology, Changsha 410073, China
[2]Dongguan University of Technology, Dongguan, China
guoxifeng1990@163.com, {enzhu, xinwangliu}@nudt.edu.cn, jpyin@dgut.edu.cn

Abstract

Existing deep neural networks mainly focus on learning transformation invariant features. However, it is the equivariant features that are more adequate for general purpose tasks. Unfortunately, few work has been devoted to learning equivariant features. To fill this gap, in this paper, we propose an affine equivariant autoencoder to learn features that are equivariant to the affine transformation in an unsupervised manner. The objective consists of the self-reconstruction of the original example and affine transformed example, and the approximation of the affine transformation function, where the reconstruction makes the encoder a valid feature extractor and the approximation encourages the equivariance. Extensive experiments are conducted to validate the equivariance and discriminative ability of the features learned by our affine equivariant autoencoder.

1 Introduction

Deep neural networks (DNNs) have brought breakthroughs in different machine learning domains and applications. To accomplish the task at hand, DNNs learn task-specific features and discard the information that is irrelevant to the current task. For example, applying local transformation to an image example will not change the feature representation in a DNN for classification. This kind of feature is called transformation invariant feature. The transformation invariance property improves the robustness of the feature and favors the task at hand. However, in some cases where there's no specific task to achieve, especially in representation learning field [Bengio *et al.*, 2013], the feature is required to preserve as much information as possible. To capture the transformation information which is directly discarded by traditional DNNs, the feature should change accordingly with the transformation of the input image. We call the feature with the above property *transformation equivariant feature*.

The transforming autoencoder [Hinton *et al.*, 2011] is the first work to mention about the importance of the equivariant feature. But it is only an approximation of the given transformation function. Neither theoretical guarantee nor empirical validation is provided to prove the equivariance. Capsule network [Sabour *et al.*, 2017] encodes the instantiation parameters to a group of neurons but can not control what parameters to learn. The rotation equivariant network [Li *et al.*, 2018] is the only work that explicitly incorporates the equivariance property. But it can only achieve the equivariance to rotation transformation by defining a series of complex layers.

In this paper, we propose an affine equivariant autoencoder (AEAE) to learn features that are equivariant to the affine transformation in an unsupervised manner. We first derive an objective to directly satisfy the definition of the equivariance. Then we relax the objective to avoid instability. The resulting objective consists of three parts: 1) self-reconstruction of the original example, 2) self-reconstruction of the affine transformed example, and 3) approximation of the affine transformation function. The first two parts ensure the content information is captured by the encoder network. The third part incorporates the equivariance property to the encoder. We conduct experiments to validate the equivariance of the learned features both quantitatively and qualitatively. We also perform clustering and classification on the learned features to prove that the feature is discriminative.

Our main contributions are summarized as follows.

- We propose an affine equivariant autoencoder (AEAE) to learn deep features that are equivariant to the affine transformation. To the best of our knowledge, this is the first work to explicitly learn equivariant features in an unsupervised manner.
- We design experiments to validate the equivariance of the learned features both quantitatively and qualitatively.
- Our model achieves the state-of-the-art clustering performance by simply running spectral clustering on the learned features.

2 Affine Equivariant Autoencoder

2.1 Equivariance and Invariance

Given two sets $\mathcal{X}$ and $\mathcal{Z}$, and a mapping function $f : \mathcal{X} \to \mathcal{Z}$. Formally, the mapping f is equivariant to a family of transformations $\mathscr{T}$, if for any transformation $\mathcal{T} \in \mathscr{T}$ there exists

[*]Corresponding Authors

a corresponding transformation $\mathcal{T}'$, such that

$$f(\mathcal{T}(x)) = \mathcal{T}'(f(x)), \forall x \in \mathcal{X}. \tag{1}$$

And the mapping f is invariant to $\mathscr{T}$, if for all $\mathcal{T} \in \mathscr{T}$ the following equation holds:

$$f(\mathcal{T}(x)) = f(x), \forall x \in \mathcal{X}. \tag{2}$$

In this paper, we only consider image datasets and the affine transformation. Specifically, let $\mathcal{X} \subset \mathbb{R}^{c \times w \times h}$ be an image dataset in which each element x is an image represented by a $c \times w \times h$ matrix where c, w, h are channels, width and height, respectively. The $\mathcal{Z} \subset \mathbb{R}^d$ is comprised of d-dimensional feature vectors of images in $\mathcal{X}$. Let $\mathcal{T}_\sigma$ be parameterized by $\sigma \in \mathbb{R}^5$ where σ_0 represents the degree of rotation anticlockwise, σ_1 the pixels of translation rightward, σ_2 the pixels of translation downward, σ_3 the scaling factor, and σ_4 the degree of shear transformation.

To disentangle affine parameters from feature vectors, we force the transformation $\mathcal{T}'$ to be parameterized by σ and to take a simple form:

$$\mathcal{T}'(z) = \mathcal{T}'_\sigma(z) = z + t(\sigma), \tag{3}$$

where t linearly transforms the elements in σ to a reasonable range and expands the size of σ from 5 to d by padding zeros. Therefore, we aim to find a feature extractor f that satisfies

$$f(\mathcal{T}_\sigma(x)) = \mathcal{T}'_\sigma(f(x)) = f(x) + t(\sigma), \forall x \in \mathcal{X}. \tag{4}$$

2.2 Model Description

To find a feature extractor f that satisfies (4), we implement f by a deep neural network. Given a dataset with n examples $X = \{x_i\}_{i=1}^n \subset \mathcal{X} \subset \mathbb{R}^{c \times w \times h}$. We define an encoder network $f_\mathbf{w}(\cdot)$ to transform each example x_i to $z_i \in \mathcal{Z} \subset \mathbb{R}^d$ and a decoder network $g_\mathbf{u}(\cdot)$ to map z_i back to x_i where $\mathbf{w}, \mathbf{u}$ are the sets of trainable weights in encoder and decoder networks, respectively.

We need to make sure that the autoencoder can reconstruct all x and $\mathcal{T}_\sigma(x)$ in order to extract their features by $f_\mathbf{w}$. Therefore, the ideal objective is:

$$\begin{aligned}\min_{\mathbf{w},\mathbf{u}} \sum_{i=1}^{n} \Big(&\|x_i - g_\mathbf{u}(f_\mathbf{w}(x_i))\|^2 \\ &+ \|\mathcal{T}_\sigma(x_i) - g_\mathbf{u}(f_\mathbf{w}(\mathcal{T}_\sigma(x_i)))\|^2 \\ &+ \|f_\mathbf{w}(\mathcal{T}_\sigma(x_i)) - (f_\mathbf{w}(x_i) + t(\sigma))\|_2^2 \Big),\end{aligned} \tag{5}$$

where $\|\cdot\|^2$ denotes the sum of squared errors and $\|\cdot\|_2$ is the ℓ_2 norm. The first two terms enable $f_\mathbf{w}$ to be a valid feature extractor for both x and $\mathcal{T}_\sigma(x)$, and the last term makes $f_\mathbf{w}$ equivariant to affine transformation $\mathscr{T}$. If the above conditions are met, $z = f_\mathbf{w}(x)$ is called affine equivariant feature.

However, from the perspective of optimization, the error signal of the last term can hardly back-propagate to the encoder's layers through two paths. The typical solution is to block one path and back-propagate through another. But we find this solution is unstable and sometimes leads to divergence of the objective. To solve this problem, we propose an alternative objective:

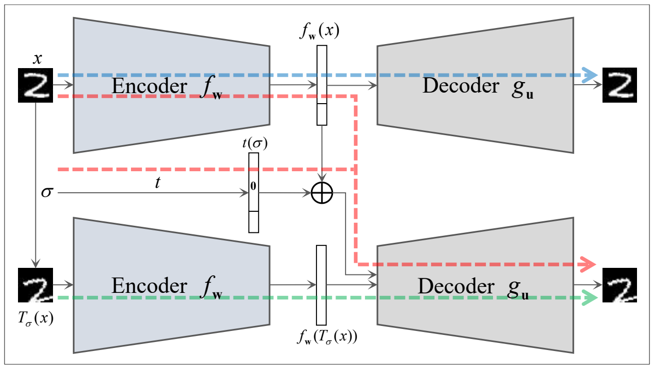

Figure 1: Affine equivariant autoencoder. The autoencoders at top and bottom share the same parameters **w** and **u**. There are three data flows denoted by blue, green, and red dashed lines, making the encoder $f_\mathbf{w}$ a valid feature extractor for clean example x, affine transformed example $\mathcal{T}_\sigma(x_i)$, and equivariant to affine transformations, respectively. In the red flow, the affine parameter $\sigma \in \mathbb{R}^5$ is scaled by element and then added to the last five elements of the feature vector $f_\mathbf{w}(x)$.

$$\begin{aligned}\min_{\mathbf{w},\mathbf{u}} \sum_{i=1}^{n} \Big(&\|x_i - g_\mathbf{u}(f_\mathbf{w}(x_i))\|^2 \\ &+ \|\mathcal{T}_\sigma(x_i) - g_\mathbf{u}(f_\mathbf{w}(\mathcal{T}_\sigma(x_i)))\|^2 \\ &+ \|\mathcal{T}_\sigma(x_i) - g_\mathbf{u}(f_\mathbf{w}(x_i) + t(\sigma))\|^2 \Big).\end{aligned} \tag{6}$$

We prove (6) is approximately equivalent to (5) under the condition that $t(\sigma)$ is in a small range near zero. When (6) reaches the minimal value 0, we have

$$\begin{cases} x_i = g_\mathbf{u}(f_\mathbf{w}(x_i)), \\ \mathcal{T}_\sigma(x_i) = g_\mathbf{u}(f_\mathbf{w}(x_i) + t(\sigma)). \end{cases} \tag{7}$$

Apply $f_\mathbf{w}$ to both sides, we get

$$\begin{cases} f_\mathbf{w}(x_i) = f_\mathbf{w}(g_\mathbf{u}(f_\mathbf{w}(x_i))), \\ f_\mathbf{w}(\mathcal{T}_\sigma(x_i)) = f_\mathbf{w}(g_\mathbf{u}(f_\mathbf{w}(x_i) + t(\sigma))). \end{cases} \tag{8}$$

The first equation in (8) implies $f_\mathbf{w}(g_\mathbf{u}(\cdot))$ is an identity mapping for $f_\mathbf{w}(x_i) \in \mathcal{Z}$. If we restrict $t(\sigma)$ to a small range near 0, we have

$$f_\mathbf{w}(\mathcal{T}_\sigma(x_i)) = f_\mathbf{w}(g_\mathbf{u}(f_\mathbf{w}(x_i) + t(\sigma))) \approx f_\mathbf{w}(x_i) + t(\sigma), \tag{9}$$

which means the equivariance of $f_\mathbf{w}$ can be approximately satisfied by optimizing (6). This completes the proof.

Another benefit of (6) is that the affine transformation $\mathcal{T}_\sigma(x_i)$ is explicitly approximated by $g_\mathbf{u}(f_\mathbf{w}(x_i) + t(\sigma))$. So we can validate whether the feature vector disentangles the affine factors by adding turbulence to the feature and observing the reconstructed image.

Finally, we depict the proposed model in Figure 1. There are three data flows denoted by blue, green, and red dashed lines, corresponding to three terms in (6). The blue data flow represents a standard autoencoder which takes an example x as input and tries to reconstruct x. The green data flow also defines a standard autoencoder with the input and target are

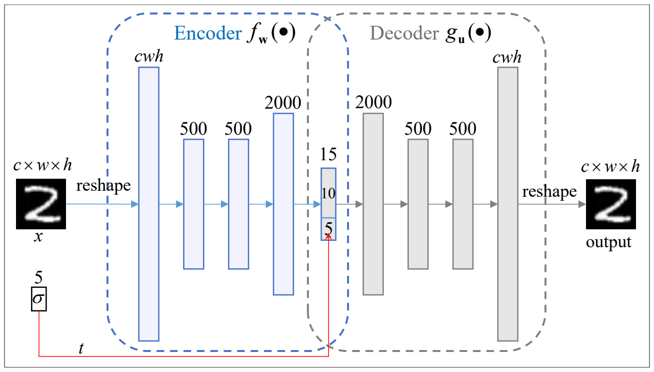

Figure 2: The detailed structure of the affine equivariant autoencoder (AEAE). The model consists of eight fully connected layers. The number of neurons in each layer is listed at top, where c, w, h are the number of channels, width, and height of the input image.

the affine transformed example $\mathcal{T}_\sigma(x)$. The blue and green autoencoders share the same weights $\{\mathbf{w}, \mathbf{u}\}$, enabling the encoder $f_\mathbf{w}$ to be a valid feature extractor for both x and $\mathcal{T}_\sigma(x)$. The red data flow tries to reconstruct $\mathcal{T}_\sigma(x)$ with the clean example x and affine parameter σ as inputs. The affine parameter $\sigma \in \mathbb{R}^5$ is scaled by element and then added to the last five elements of the feature vector $f_\mathbf{w}(x)$. The resulting feature is fed to the decoder to construct $\mathcal{T}_\sigma(x)$. This red flow learns an approximate mapping for the affine transformation $\mathcal{T}_\sigma$ and ensures the equivariance of $f_\mathbf{w}$ to affine transformations $\mathscr{T}$.

2.3 Implementation

The autoencoder's structure can either be fully connected or convolutional in theory. However, convolutional networks, especially those including pooling layers, are known to be invariant to local transformations, which is harmful to our goal of learning an equivariant mapping. Therefore, we implement the encoder $f_\mathbf{w}$ and decoder $f_\mathbf{u}$ by using fully connected layers instead of convolutional layers. The detailed structure is as shown in Figure 2. There are total eight fully connected layers with number of neurons $D - 500 - 500 - 2000 - 15 - 2000 - 500 - 500 - D$, where $D = cwh$ is the dimension of the vectorized input image. Given an image example $x \in \mathbb{R}^{c \times w \times h}$ which can be optionally transformed by affine transformation $\mathcal{T}_\sigma$, we flatten it to a vector with length cwh and then feed it into the proposed model. The output is also a vector with length cwh which is finally reshaped to $c \times w \times h$ in order to compute the distance $\|\cdot\|$ with x. Except the embedding (with 15 neurons) and output layers are followed by linear and sigmoid activations, respectively, all internal layers are activated by ReLU [Glorot *et al.*, 2011].

We limit the parameters σ of affine transformations to a reasonable range as shown in Table 1. The function t is defined as

$$t(\sigma) = \left[0, \ldots, 0, \frac{\sigma_0}{100}, \frac{\sigma_1}{10}, \frac{\sigma_2}{10}, \sigma_3 - 1, \frac{\sigma_4}{100}\right], \tag{10}$$

in order to keep $t(\sigma)$ in a small range near 0 which is a necessary condition of the equivalence of (5) and (6) as described in Section 2.2.

Meaning of σ_i	Range of σ_i	Range of $t(\sigma_i)$
Rotation degree	[-20, 20]	[-0.2, 0.2]
Translation rightward	[-3, 3]	[-0.3, 0.3]
Translation downward	[-3, 3]	[-0.3, 0.3]
Scaling factor	[0.9, 1.1]	[-0.1, 0.1]
Shear degree	[-20, 20]	[-0.2, 0.2]

Table 1: Allowed range of affine parameters $\sigma \in \mathbb{R}^5$ and $t(\sigma)$

Dataset	# Examples	Original size	New size
MNIST-full	70,000	28×28	32×32
MNIST-test	10,000	28×28	32×32
USPS	9,298	16×16	20×20
Fashion	70,000	28×28	32×32

Table 2: Dataset statistics. The new size is used in our experiment.

We initialize all weights by following [He *et al.*, 2015]. The AEAE is trained in an end-to-end manner by using Adam [Kingma and Ba, 2014] optimizer with initial learning rate 0.001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The maximum number of epochs is set to 100 for large datasets ($n > 10000$) and 500 for small ones ($n <= 10000$). The mini-batch size is fixed to 256. The source code is publicly available on https://github.com/XifengGuo/AEAE.

3 Experiments

We conduct extensive experiments to validate the effectiveness of the proposed AEAE. First, we demonstrate the equivariance property quantitatively and qualitatively. Then we assess the quality of features learned by our AEAE according to the clustering and classification performances. Four image datasets are used in our experiments:

- MNIST-full: A popular handwritten digit dataset with 70,000 examples [LeCun *et al.*, 1998].
- MNIST-test: A dataset that only contains the test set of MNIST-full, with 10,000 examples. We consider this a separate dataset because the number of examples may affect the clustering performance, by following [Yang *et al.*, 2016; Dizaji *et al.*, 2017].
- USPS[1]: A dataset contains 9298 gray digit images with size of 16x16 pixels divided into 10 categories.
- Fashion: A dataset of Zalando's article images [Xiao *et al.*, 2017], sharing the same statistics with MNIST-full but more challenging for machine learning algorithms.

Table 2 shows the statistics of these datasets. All image examples in these datasets are padded two pixels of zeros in each of four directions to avoid information loss during the affine transformation. The pixel values are scaled to $[0, 1]$.

3.1 Equivariance Validation

We empirically validate the equivariance of the encoder mapping $f_\mathbf{w}$ on MNIST-full and Fashion datasets. We directly

[1]http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html

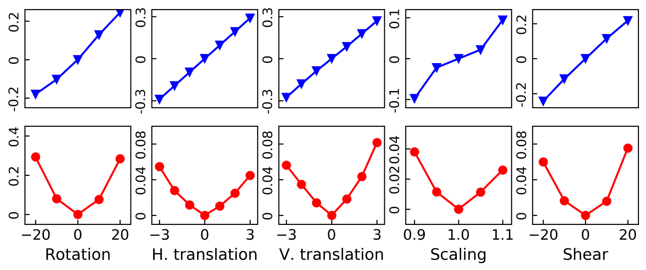

Figure 3: Equivariance validation by directly comparing the original example's feature $f_{\mathbf{w}}(x)$ and affine transformed example's feature $f_{\mathbf{w}}(\mathcal{T}_\sigma(x))$. The first row shows the average difference in one element of $f_{\mathbf{w}}(x)$ and $f_{\mathbf{w}}(\mathcal{T}_\sigma(x))$. The second row depicts the average distances between features, i.e., $\frac{1}{n}\sum_{i=1}^{n}\|f_{\mathbf{w}}(\mathcal{T}_\sigma(x_i)) - (f_{\mathbf{w}}(x_i)+t(\sigma))\|_2^2$.

check if (4) is satisfied after optimizing (6). To this end, we first compute the difference between features of original and affine transformed examples:

$$\Delta_i = f_{\mathbf{w}}(\mathcal{T}_\sigma(x_i)) - f_{\mathbf{w}}(x_i). \qquad (11)$$

The σ is restricted to have only one element active, i.e, only one of rotation, horizontal translation, vertical translation, scaling, and shear transformations is applied to all original examples. According to the definition of equivariance and invariance, if $\Delta_i = \mathbf{0}$ for all i, $f_{\mathbf{w}}$ will be invariant to the affine transformation. However, we observe Δ_i changes linearly with the affine parameter σ. As shown in the first row in Figure 3, the jth element of the average $\Delta = \sum_{i=1}^{n}\Delta_i$ is positively correlated to the kth element of σ where $j = 10+k, k = 0, 1, \ldots, 4$. To be exact, Δ is close to its expectation $t(\sigma)$. On the other hand, we record the average value of $\|f_{\mathbf{w}}(\mathcal{T}_\sigma(x_i)) - (f_{\mathbf{w}}(x_i)+t(\sigma))\|_2^2$ over i in the second row in Figure 3. For all transformations, this value is close to 0. To conclude, (4) is approximately satisfied when σ is limited by Table 1. That is, the empirical quantitative evidence supports that $f_{\mathbf{w}}$ is equivariant to the affine transformation.

In addition to the direct validation of the equivariance property, we give an indirect evidence as follows. As shown in the last term of (6), we attempt to reconstruct the affine transformed example $\mathcal{T}_\sigma(x)$ from the original example x with affine parameters σ added to the embedding feature of x. Therefore, if (6) is successfully optimized, the output of our model will be continuously transformed when adding turbulence to the last five elements of the feature of x. To cover the range of affine parameters σ as listed in Table 1, we sample a value from $\{-0.4, -0.3, \ldots, 0.4\}$ and add it to one or all of the last five elements (corresponding to rotation, horizontal translation, vertical translation, scaling, and shear) of $f_{\mathbf{w}}(x)$. The reconstructed images under different configurations are shown in Figure 4. The input examples are shown in the first row, denoted by "Input". The corresponding pure reconstructions are list in row "0". From row "-0.4" to "+0.4", the images are transformed continuously, even though the model only sees examples that are transformed in a range smaller than $[-0.3, 0.3]$. The last column "Affine" denotes adding noise to all the last five elements, which is an extreme test but still reflects the affine transformation. The result provides qualitative evidence that the encoder mapping $f_{\mathbf{w}}$ is equivariant to the affine transformation.

3.2 Clustering

This subsection focuses on evaluating the equivariant features by performing unsupervised clustering task. After training the AEAE, we extract all features $\{z_i = f_{\mathbf{w}}(x_i)\}_{i=1}^{n}$ from the embedding layer. Then we perform k-means and spectral clustering (SC) on the extracted features, denoted by AEAE+k-means and AEAE+SC, respectively. We run each method for five times and record the average performance in terms of clustering ACCuracy (ACC) and Normalized Mutual Information (NMI). The baseline methods include conventional shallow clustering and state-of-the-art deep clustering. We report the results by excerpting from the corresponding papers or by running their released code when available.

The results on four datasets are reported in Table 3. Our AEAE+k-means is comparable to deep clustering algorithms and AEAE+SC achieves the best performance. This demonstrates the AEAE can learn more discriminative features. It is possible to further improve the performance by jointly optimizing the clustering model and encoder network, as does existing deep clustering method.

We also compare with other autoencoders including vanilla autoencoder (AE) and denoising autoencoder (DAE). They share the same network structure (number of layers, number of neurons in each layer, activation function, etc.) and optimization setting (learning rate, batch size, maximum epochs, etc.) with our model. The only difference is the objective: AE tries to reconstruct x from x and DAE attempts to reconstruct x from $\hat{x}$ where $\hat{x}$ is the corrupted version of x by applying Dropout [Srivastava *et al.*, 2014] with probability 0.2. We report the SC result on the features of different autoencoders in Table 4. Our AEAE outperforms the other autoencoders by a large margin in terms of ACC and NMI. This proves that it is the proposed objective (6), not network structure, that leads to the better discriminative features.

As validated in Section 3.1, the affine factors are encoded in the last five neurons of the embedding layer. So the discriminative features are supposed to be captured by the rest neurons. To validate this point, we extract features from the first ten neurons of the embedding layer and report the SC result (appended by "-10") in Table 4. Our AEAE-10 is comparable to AEAE, but AE-10 and DAE-10 are much worse than AE and DAE, respectively. This well validates the above claim. This also provides guidance in how to select a part of features for the task at hand.

3.3 Classification

We further evaluate the quality of AEAE's features by performing classification task. Each dataset is divided into training and testing set by the ratio of 6 : 1. The features are normalized to zero mean and unit variance. The SVM classifier with RBF kernel is used. We select the best penalty parameter C of the error term from $[2^0, 2^1, \ldots, 2^9]$ by cross validation. We compare with the vanilla autoencoder (AE) and denoising autoencoder (DAE) which share the same network structure and optimization parameters with our AEAE.

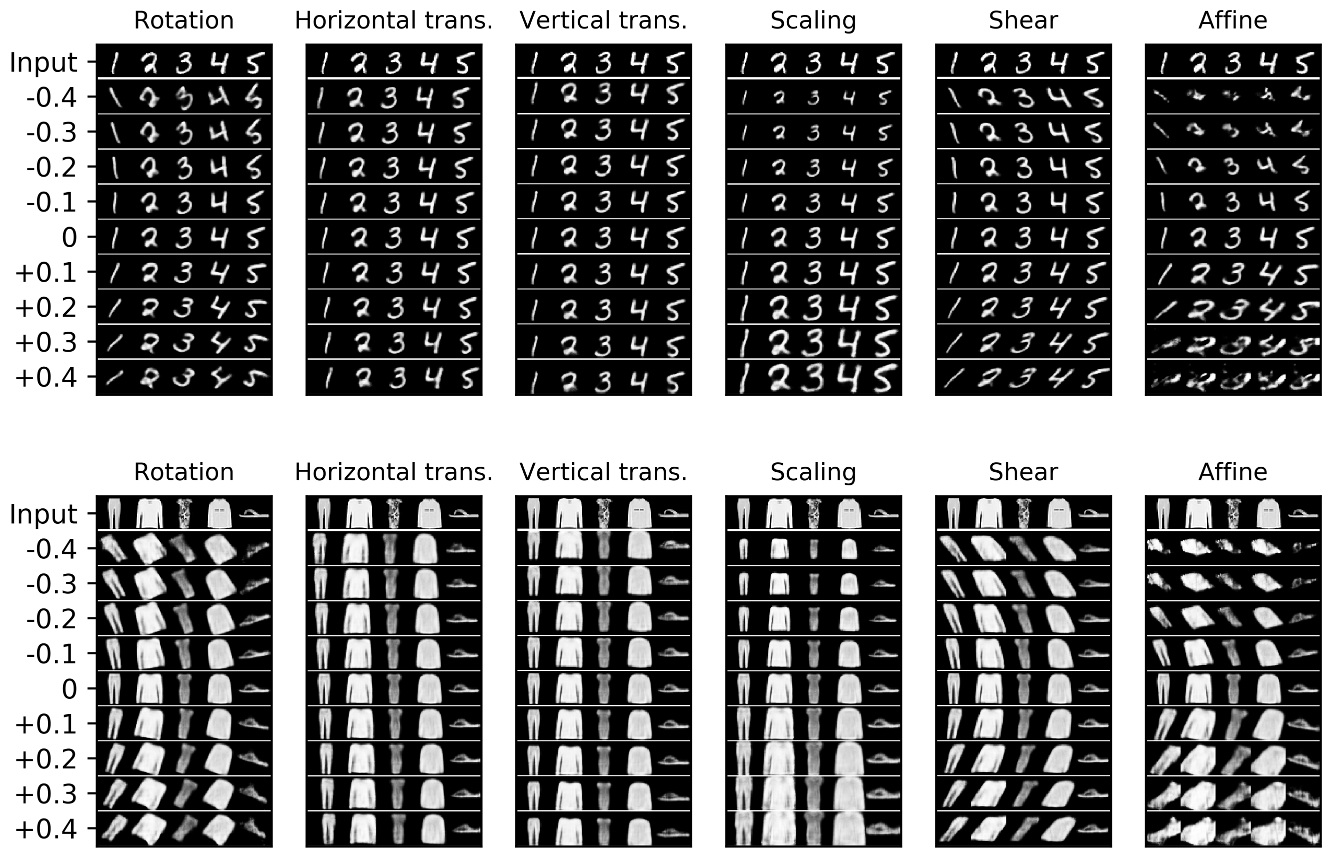

Figure 4: Reconstructions based on the sum of original example's feature and turbulence. By adding turbulence to one element of the embedding layer, the reconstruction behaves like being transformed by rotation, translation, scaling, or shear. This implies that affine factors are disentangled and encoded by different neurons of the embedding layer, and validates the equivariance property.

Method	MNIST-full		MNIST-test		USPS		Fashion	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
k-means [MacQueen, 1967]	0.532	0.500	0.546	0.501	0.668	0.627	0.474	0.512
SC [Shi and Malik, 2000]	0.656	0.731	0.660	0.704	0.649	0.794	0.508	0.575
AC [Jain, 2010]	0.621	0.682	0.695	0.711	0.683	0.725	0.500	0.564
NMF-LP [Cai *et al.*, 2009]	0.471	0.452	0.479	0.467	0.652	0.693	0.434	0.425
RCC [Shah and Koltun, 2017]	N/A	0.893*	N/A	0.828	N/A	0.742	N/A	0.614
DCN [Yang *et al.*, 2017]	0.830*	0.810*	0.802	0.786	0.688	0.683	0.501	0.558
DKM [Fard *et al.*, 2018]	0.840*	0.796*	N/A	N/A	0.757*	0.776*	N/A	N/A
DEC [Xie *et al.*, 2016]	0.863	0.834	0.856	0.830	0.762	0.767	0.518	0.546
IDEC [Guo *et al.*, 2017]	0.881*	0.867*	0.846	0.802	0.761*	0.785*	0.529	0.557
CSC [Peng *et al.*, 2017]	0.872*	0.755*	0.865*	0.733*	N/A	N/A	N/A	N/A
SR-k-means [Jabi *et al.*, 2018]	0.939*	0.866*	0.863*	0.873*	0.901	0.912	0.507	0.548
VaDE [Jiang *et al.*, 2017]	0.945	0.876	0.287	0.287	0.566	0.512	0.578	0.630
ClusterGAN [Mukherjee *et al.*, 2019]	0.950*	0.890*	N/A	N/A	N/A	N/A	0.630*	0.640*
JULE [Yang *et al.*, 2016]	0.964*	0.913*	0.961*	0.915*	0.950	0.913	0.563	0.608
DEPICT [Dizaji *et al.*, 2017]	0.965*	0.917*	0.963*	0.915*	0.899	0.906	0.392	0.392
AEAE+k-means (ours)	0.943	0.882	0.939	0.876	0.939	0.878	0.616	0.639
AEAE+SC (ours)	**0.971**	**0.940**	**0.974**	**0.941**	**0.978**	**0.941**	**0.632**	**0.701**

Table 3: Clustering performances of different algorithms. All results of baseline algorithms are reported by running their released code except those marked by ($*$) on top which are excerpted from the corresponding paper. N/A denotes that the result is unavailable from the paper or the code. The first group is traditional shallow clustering method, the second deep clustering, and the last our method.

Method	MNIST-full ACC	MNIST-full NMI	MNIST-test ACC	MNIST-test NMI	USPS ACC	USPS NMI	Fashion ACC	Fashion NMI
AE	0.832±0.002	0.893±0.002	0.791±0.003	0.814±0.005	0.716±0.043	0.775±0.016	0.602±0.010	0.682±0.003
DAE	0.824±0.002	0.883±0.004	0.790±0.002	0.817±0.006	0.773±0.067	0.829±0.019	0.605±0.010	0.680±0.003
AEAE	**0.971±0.002**	**0.940**±0.005	**0.974±0.005**	**0.941**±0.007	**0.978±0.001**	**0.941**±0.002	**0.632**±0.002	**0.701**±0.002
AE-10	0.650±0.043	0.701±0.039	0.620±0.073	0.590±0.035	0.627±0.020	0.620±0.039	0.575±0.016	0.582±0.041
DAE-10	0.656±0.024	0.721±0.033	0.621±0.015	0.602±0.028	0.639±0.058	0.653±0.023	0.569±0.021	0.591±0.020
AEAE-10	**0.971**±0.002	**0.939**±0.005	**0.974**±0.006	**0.942**±0.008	**0.977**±0.001	**0.939**±0.002	**0.631**±0.002	**0.701**±0.002

Table 4: Clustering performances of different autoencoders. AE, DAE, and AEAE denote the result of performing spectral clustering (SC) on the full features extracted from different autoencoders. That appended by "-10" is SC result on the first 10 dimensions of features.

	MNIST-full	USPS	Fashion
AE	0.989±0.001	0.977±0.000	**0.889±0.002**
DAE	0.988±0.000	0.982±0.002	0.886±0.001
AEAE	**0.990±0.001**	**0.989**±0.001	0.879±0.001
AE-10	0.941±0.009	0.907±0.022	0.840±0.011
DAE-10	0.943±0.013	0.929±0.020	0.840±0.006
AEAE-10	**0.990±0.001**	**0.988**±0.001	**0.874±0.001**

Table 5: Classification accuracy of SVM on the embedding features of different autoencoders. "X-10" denotes the accuracy on the first 10 dimensions of X's features.

We run all methods for five times and report the average and standard deviation of the classification accuracies.

The results are shown in Table 5. First, our AEAE outperforms AE and DAE, which validates that the AEAE can learn discriminative features. Second, by comparing AE-10 (DAE-10) with AE (DAE), the classification performance drops dramatically when removing the last five dimensions of the features. Third, AEAE-10 is comparable to AEAE in terms of accuracy, which implies that the discriminative features are mainly encoded by the first ten dimensions. These conclusions are consistent with that drawn in the last subsection.

4 Related Work

The proposed AEAE is closely related to the **transforming autoencoder** [Hinton *et al.*, 2011] which can also reconstruct the affine transformed image from the original image and affine parameters. However, there exists large differences between them as follows. The transforming autoencoder aims to solve the optimization problem of $\min_{\mathbf{w},\mathbf{u}} \sum_{i=1}^{n} \|\mathcal{T}_\sigma(x_i) - g_\mathbf{u}(f_\mathbf{w}(x_i) + \sigma)\|^2$ where $f_\mathbf{w}(x_i)$ and σ have the same number of dimensions. It can only learn a mapping to approximate the transformation $\mathcal{T}$ and can not make $f_\mathbf{w}$ a valid feature extractor for x_i and $\mathcal{T}_\sigma(x_i)$. Furthermore, the direct addition of $f_\mathbf{w}(x_i)$ and σ disables the representation $f_\mathbf{w}(x_i)$ to be discriminative, even if it is a valid feature of x_i. By contrast, our AEAE can learn discriminative features by forcing the encoder $f_\mathbf{w}$ to be equivariant to affine transformations. The network structure is also different. The transforming autoencoder only consists of one hidden layer, leading to the lack of capacity. While our multi-layer model has powerful ability to transform data.

Another related model is the **capsule network** [Sabour *et al.*, 2017] which extends the transforming autoencoder by incorporating dynamic routing mechanism. The capsule network also avoids to learn local transformation invariant features by encoding the instantiation parameters into a group of neurons termed capsule. However, the capsule network requires supervision provided to accomplish this goal. While our AEAE takes advantage of the affine transformation. In other words, our model is trained in an unsupervised manner. Another difference is that our model explicitly encodes the affine factors into specified neurons while the capsule network implicitly learns the instantiation parameters which are uncontrollable.

The **rotation equivariant network** [Li *et al.*, 2018] focuses on the rotation transformation which is only a specific type of affine transformation. Three new types of convolutional layers (cycle layer, isotonic layer, and decycle layer) were proposed in [Li *et al.*, 2018] to implement rotation equivariance. The rotation equivariant network is trained in supervised manner while our model is unsupervised. It can only learn rotation equivariant features and is hard to adapt to other affine transformations like translation, scaling and shearing. Our AEAE is easy to implement other transformation equivariance if the transformation can be manipulated in a known way.

5 Conclusion

The proposed affine equivariant autoencoder (AEAE) can learn a mapping that is equivariant to the affine transformation in an unsupervised manner. The mapping can serve as a valid feature extractor for images and the resultant features are proved to be very discriminative by performing clustering and classification tasks. The affine factors are disentangled and encoded by the specified neurons of the embedding layer, making the features more interpretable. Future work includes investigating other form of the transformation $\mathcal{T}'_\sigma$ in the feature space, incorporating the equivariance to more transformations other than the affine transformation, and extending our unsupervised model to semi-supervised scenario.

Acknowledgments

This work was supported by the National Key R & D Program of China 2018YFB1003203 and the National Natural Science Foundation of China (project no. 61773392 and 61672528).

References

[Bengio *et al.*, 2013] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[Cai *et al.*, 2009] Deng Cai, Xiaofei He, Xuanhui Wang, Hujun Bao, and Jiawei Han. Locality preserving nonnegative matrix factorization. In *IJCAI*, pages 1010–1015, 2009.

[Dizaji *et al.*, 2017] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *ICCV*, pages 5747–5756, 2017.

[Fard *et al.*, 2018] Maziar Moradi Fard, Thibaut Thonet, and Eric Gaussier. Deep k-means: Jointly clustering with k-means and learning representations. *ArXiv*, 2018.

[Glorot *et al.*, 2011] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. *Journal of Machine Learning Research*, 15:315–323, 2011.

[Guo *et al.*, 2017] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI*, pages 1753–1759, 2017.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Sun Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015.

[Hinton *et al.*, 2011] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming auto-encoders. In *ICANN*, 2011.

[Jabi *et al.*, 2018] Mohammed Jabi, Marco Pedersoli, Amar Mitiche, and Ismail Ben Ayed. Deep clustering: On the link between discriminative models and k-means. *ArXiv*, 2018.

[Jain, 2010] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[Jiang *et al.*, 2017] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *IJCAI*, pages 1965–1972, 2017.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ArXiv*, 2014.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Li *et al.*, 2018] Junying Li, Zichen Yang, Haifeng Liu, and Deng Cai. Deep rotation equivariant network. *Neurocomputing*, 290:26–33, 2018.

[MacQueen, 1967] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[Mukherjee *et al.*, 2019] Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. Clustergan : Latent space clustering in generative adversarial networks. In *AAAI*, 2019.

[Peng *et al.*, 2017] Xi Peng, Jiashi Feng, Jiwen Lu, Wei-Yun Yau, and Zhang Yi. Cascade subspace clustering. In *AAAI*, pages 2478–2484, 2017.

[Sabour *et al.*, 2017] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In *NIPS*, pages 3856–3866, 2017.

[Shah and Koltun, 2017] Sohil Shah and Vladlen Koltun. Robust continuous clustering. *Proceedings of the National Academy of Sciences of the United States of America*, 114(37):9814–9819, 2017.

[Shi and Malik, 2000] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[Xiao *et al.*, 2017] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[Xie *et al.*, 2016] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, pages 478–487, 2016.

[Yang *et al.*, 2016] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, pages 5147–5156, 2016.

[Yang *et al.*, 2017] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*, pages 3861–3870, 2017.