Image Representation Learning by Transformation Regression

Xifeng Guo, Jiyuan Liu*, Sihang Zhou, En Zhu*
PRMI Group, College of Computer
National University of Defense Technology
Changsha 410073, China
Email: guoxifeng1990@163.com

Shihao Dong
College of Mathematics and Computer Science
Zhejiang Normal University
Jinhua 321004, China
Email: 1274149231@qq.com

Abstract—Self-supervised learning is a thriving research direction since it can relieve the burden of human labeling for machine learning by seeking for supervision from data instead of human annotation. Although demonstrating promising performance in various applications, we observe that the existing methods usually model the auxiliary learning tasks as classification tasks with finite discrete labels, leading to insufficient supervisory signals, which in turn restricts the representation quality. In this paper, to solve the above problem and make full use of the supervision from data, we design a regression model to predict the continuous parameters of a group of transformations, i.e., image rotation, translation, and scaling. Surprisingly, this naive modification stimulates tremendous potential from data and the resulting supervisory signal has largely improved the performance of image representation learning. Extensive experiments on four image datasets, including CIFAR10, CIFAR100, STL10, and SVHN, indicate that our proposed algorithm outperforms the state-of-the-art unsupervised learning methods by a large margin in terms of classification accuracy. Crucially, we find that with our proposed training mechanism as an initialization, the performance of the existing state-of-the-art classification deep architectures can be preferably improved.

I. Introduction

Recently, deep neural networks, especially Convolutional Neural Networks (CNNs), have witnessed tremendous progress on many computer vision tasks, such as image classification, segmentation and object detection. With the guidance of human annotations, the existing deep neural networks can already achieve satisfactory performance in many applications like face recognition, disease diagnose, etc. However, the success of these methods largely relies on massive amount of manually labeled data, which is expensive and impracticable to scale on the vast amount of visual data that are available today. Moreover, in some fields, like medical care and astronomy, only domain experts are able to provide reliable annotations, making it impossible to collect a large number of labels. To this end, unsupervised learning techniques, especially self-supervised methods, are becoming an increasingly important research direction.

Existing self-supervised learning methods usually define a label free surrogate task to provide a pretext supervision signal for feature learning. For example, Auto-Encoder (AE) [1] targets at reconstructing original data by minimizing the error between network output and corresponding data. Variational Auto-Encoder (VAE) [2] and Generative Adversarial Network (GAN) [3] are also taking the data content as supervision, though they have very different objectives. These methods focus too much on pixel details and thus are hard to learn high-level semantic representation. Some methods apply many groups of transformations to the original data and teach the CNN model to recognize which group the input data comes from. Fundamentally, they usually construct a classification task with finite discrete labels, resulting in insufficient supervisory signals, which in turn harms the performance of representation learning.

To overcome the mentioned problem, we propose a regression-based learning protocol which predicts continuous parameters of transformations, e.g, image rotation, translation, and scaling. Specifically, we apply a group of transformations with continuous parameters to the input image and can get infinite new images, theoretically. We expect a neural network to be fed a transformed image with parameters $\mathbf{\Theta}$ and output $\mathbf{\Theta}$. In other words, the neural network is asked to approximate the underling composition function of different types of transformations by fitting continuously transformed image data. In this way, the auxiliary supervisory signals are infinite and thereby help the neural network learn a better representation. It is also convenient to incorporate new type of transformations, by simply adding a new neuron to the output layer. In contrast, those classification-based self-supervised representation learning methods require adding more output neurons according to the sampling step during discretization. We perform classification task on CIFAR10, CIFAR100, STL10, and SVHN, which are all widely used in representation learning community, by using the feature representation learned by our proposed transformation regression. The result shows that our method outperforms other self-supervised representation learning methods and sets new state-of-the-art performance on all benchmarks. We also show that our model can be used for initialization of the classification model to further improve its performance.

Our main contributions lie in three aspects as follows.

- This is the first work to create sufficient auxiliary supervisory signals by regressing continuous transformation parameters.
- We achieve the state-of-the-art performance on four popular image benchmark datasets by performing classification on the representation learned through our proposed transformation regression method.
- Experimental results validate that taking our proposed training mechanism as an initialization, the performance of prior state-of-the-art deep classification models can be further improved.

II. Related Work

Current self-supervised representation learning methods can be roughly separated into two categories, i.e. content based and manipulation based representation learning.

A. Content based Representation Learning

The content based representation learning methods take advantage of data itself as the supervisory signal. They are also known as conventional unsupervised representation learning. Representative methods include Auto-Encoder (AE) [1], Variational Auto-Encoder (VAE) [2], and Generative Adversarial Network (GAN) [3].

Vanilla AE is designed as two parts, an encoder and a decoder, which are symmetric in most cases. It learns feature representation by minimizing the error between original input image and the corresponding reconstructed one. But it tends to easily overfit the data when the neural network has enough capacity. To overcome this problem and improve the robustness, additional restriction is applied on the input data or network model. For example, stacked autoencoder [4] uses the reconstruction loss to train the AE layer by layer. Denoising autoencoder [5] adds noise to input data and force the network to reconstruct the clean data. Sparse autoencoder [6] constrains the embedding layer to be sparse by adding an l_1 penalty. And contractive autoencoder [7] makes representation less sensitive to small variations in its training dataset by adding a regularizer, or penalty term, to the reconstruction loss.

VAE is a generative unsupervised learning model which consists of a classical auto-encoder component and a Bayesian regularization over the latent space, forcing the posterior of latent representation matches a prior distribution [8]. VAE and its variants have been widely applied to various representation learning tasks, including image understanding [9], [10] and sentence modeling [11].

Meanwhile, GAN [3] is also a popular and effective technique for unsupervised representation learning, though it is well known as generative model that can generate realistic images. Radford et al. [12] propose Deep Convolutional GAN (DCGAN) that largely improves its ability of learning representation. Tran et al. [13] propose the Disentangled Representation learning GAN (DR-GAN) to learn an interpretable and discriminative representation. InfoGAN [14] improves the interpretability of the representation by maximizing the mutual information between a subset of latent variables and the observations.

The underlying representations from aforementioned models are further adopted for image classification and object detection tasks and shown to be robust and effective. However, the main drawback is that they put too much emphasis on pixel details. As a result, there exists a non-ignorable gap with most supervised approaches by using the extracted representations on classification task.

B. Manipulation based Representation Learning

The manipulation based representation learning techniques define a group of manipulations (e.g., image rotation, color jitter, and inpainting) employed on images and generate corresponding labels. For example, Gidaris et al. [15] apply four rotations ($0°, 90°, 180°,$ and $270°$) to images and label them with class 1 to 4. It is worth noting that the labels can be generated automatically by following the pre-specified rules, which is a key distinction from supervised methods. Doersch et al. [16] randomly sample patch pairs from single image and feed them into a convolutional network, so as to predict the relative position within pairs in a supervised manner. Then, the captured image representations are tested on Pascal VOC 2011 with clustering methods, illustrating their effectiveness. The discriminative image features can also be extracted by training deep networks to colorize the gray scale images, which is well demonstrated in [17], [18]. Agrawal et al. [19] use the moving objects to train a Convolutional Neural Network with their ego-motions as the supervisory signal for feature learning. Norroozi and Favaro [20] build a deep network to solve Jigsaw puzzles as a pretext task without manual labeling and use the extracted features to perform object classification, showing a high quality performance. Huang et al. [21] adopt sample neighborhood information to train a deep network model so as to learn the underlying class decision boundaries at the same time.

Our proposed method belongs to manipulation based representation learning. Compared with content based learning methods, it forces CNNs to capture the intrinsic image structures by performing supervised learning to predict geometric transformations of images, producing a far more effective latent representation. At the same time, manipulation based methods generate discrete labels for representation learning, including certain amount of rotation angles [15], relative positions[16], pixel color values [17], etc. The disadvantage of this manner is that only partial labels are considered. For example, four rotation angles, i.e. 0, 90, 180, 270 degrees, are concerned in [15]. Our proposed method uses continuous rotation angles from 0 to 360 degrees and predicting the accurate values in regression manner. Continuous pixel translation and scaling are also adopted at the same time. With far richer

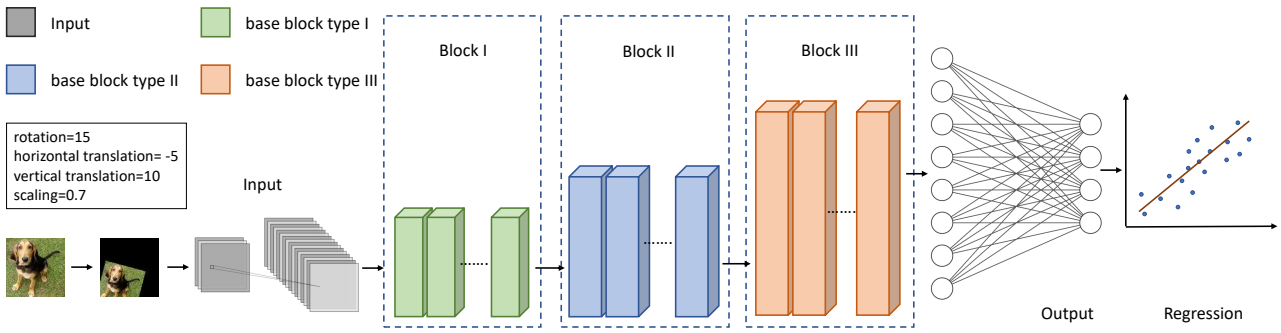

Fig. 1. Our proposed neural network consists of three blocks apart from input and output layers. Inside each block are six same base blocks which are made up with two convolutional layers. Before feeding images into the network, four transformations are performed in order to get continuous labels. Finally, regression techniques are applied on labels and network output to calculate loss for back propagation.

image transformations are considered in learning process, our model outperforms the state-of-the-art methods.

III. Transformation Regression Learning

This section elaborates the proposed transformation regression learning method. First, we give some basic definitions and our objective function. Then, we introduce the model architecture in detail. At last, the optimization algorithm is briefly presented.

A. Objective Function

Given an image space $\mathbb{X}$ and a data set with n samples $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{X}$, our primary goal is training a convolutional neural network $\mathcal{F}$ parameterized by $\mathbf{W}$ to map image samples into a feature space $\mathbb{Z}$ where $Z = \{\mathcal{F}(\mathbf{x}_i)\}_{i=1}^n$ can be readily categorized. Due to the absence of annotations, the network $\mathcal{F}$ is difficult to optimize. Although plenty of end-to-end unsupervised learning techniques are presented in recent literature, there is still a non-negligible gap in performance compared with supervised approaches.

By following the common practice of self-supervised learning, we construct a new dataset with automatically generated labels, $\hat{X} = \{\hat{\mathbf{x}}_i, y_i\}_{i=1}^n$, where $\hat{\mathbf{x}}_i = \mathcal{T}(\mathbf{x}_i; y_i)$ is an affine transformed version of the original image x_i with affine transformation parameters y_i. For simplicity, we only select four types of affine transformations, including rotation, horizontal translation, vertical translation, and scaling, parameterized by $\theta_1, \theta_2, \theta_3$, and θ_4, respectively. Then we have $y_i = [\theta_1, \theta_2, \theta_3, \theta_4]^\top$. And $\mathcal{T}(\mathbf{x}_i; y_i)$ denotes a new image that is obtained by applying these four transformations to the original image $\mathbf{x}_i$. Take Figure 1 as an example, given a dog image $\mathbf{x}_i$, randomly sampling a value of affine transformation parameters, $y_i = [15, -5, 10, 0.7]^\top$, resulting the affine transformed image and the corresponding label, $\{\hat{\mathbf{x}}_i, y_i\}$. When y_i varies in a specified range, from one unlabeled image x_i we can generate many labeled samples $\{\hat{\mathbf{x}}_i, y_i\}$. Details of the aforementioned affine transformations and their parameters are provided in Table I.

TABLE I
Details of geometric transformations and their parameters

Transformation t_i	Range of θ_i	Unit
Rotation	[-180, 180]	degree
Translation rightward	[-10, 10]	pixel
Translation downward	[-10, 10]	pixel
Scaling factor	[0.5, 1.5]	–

The neural network $\mathcal{F}$ can be trained in a supervised way by using the generated labeled dataset $\hat{X}$. The corresponding objective is

$$\min_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(\mathcal{F}\left(\hat{\mathbf{x}}_i; \mathbf{W}\right), y_i\right) \tag{1}$$

since the label y_i contains continuous values, we choose Mean Squared Error (MSE) to implement $\mathcal{L}$. So our final objective function is

$$\min_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^{n} \left\| \mathcal{F}\left(\mathcal{T}\left(\mathbf{x}_i; y_i\right); \mathbf{W}\right) - y_i \right\|_2^2 \tag{2}$$

According to the above objective, the neural network $\mathcal{F}$ is trying to predict the affine parameters y_i from the affine transformed image. To achieve this goal, the network has to emphasize on the salient objects in the foreground and learn their features. Because the background tends to only include simple features like the sky, grass, desk, etc. So the neural network $\mathcal{F}$ is expected to learn discriminative features from the images after training by (2).

B. Architecture

The architecture is shown in Figure 1. The neural network takes the affine transformed image $\hat{\mathbf{x}}_i$ as input and outputs the predicted affine parameters. The Wide Residual Network (WRN) [22] is chosen as the backbone network of our architecture. As seen in the Figure 1, the transformed images of size $3 \times 32 \times 32$ are fed as input, followed by a preprocessing layer of size $16 \times 32 \times 32$. Later, three neural network block, which is composed by six same base block each, are adopted to extract the

TABLE II
Dataset statistics

	#examples	#training examples	#testing examples	#classes	image size
CIFAR10	60,000	50,000	10,000	10	$32 \times 32 \times 3$
CIFAR100	60,000	50,000	10,000	100	$32 \times 32 \times 3$
STL10	13,000	10,000	3,000	10	$32 \times 32 \times 3$
SVHN	99,289	73,257	26,032	10	$32 \times 32 \times 3$

image latent representations. The base blocks of three types respectively consist of two convolutional layers of size $32 \times 32 \times 32, 64 \times 16 \times 16$ and $128 \times 8 \times 8$. Actually, outputs of each block are all able to be regarded as the latent representations of images, which means our representation extracting network $\mathcal{F}$ can be set as the part before any one of the three blocks, i.e. $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$. In the following section, representation effectiveness of $\{\mathcal{F}_i\}_{i=1}^3$ are thoroughly tested. The output layers are fully connected and of size 128 and 4, respectively. In our proposed network, the labels are continuous rather than discrete values. Therefore, regression techniques are performed.

C. Optimization

We initialize all weights by following [23]. The neural network is trained in an end-to-end manner by using Adam [24] optimizer with initial learning rate 0.001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The mini-batch size and epochs are fixed to 256 and 200, respectively. The whole algorithm is summarized in Algorithm 1.

Algorithm 1 Image Representation Learning Algorithm by Transformation Regression

Input: Image Dataset X; Transformation function $\mathcal{T}(\cdot, y)$
Neural Network $\mathcal{F}(\cdot, \mathbf{W})$.
Output: The parameters of the neural network $\mathbf{W}$.
Initialize the neural network parameter $\mathbf{W}$;
for t in 1 to T do
  for i in 1 to n do
    Random sample a value y_i;
    Apply transformation: $\hat{\mathbf{x}}_i = \mathcal{T}(\mathbf{x}_i; y_i)$.
    Forward pass to get the output $\mathcal{F}(\hat{\mathbf{x}}_i; \mathbf{W})$.
  end for
  Compute the loss $L = \frac{1}{n} \sum_{i=1}^{n} \|\mathcal{F}(\hat{\mathbf{x}}_i; \mathbf{W}) - y_i\|_2^2$.
  Update the W by gradient descent.
end for
return $\mathbf{W}$.

IV. Experiments

In this section, we conduct an extensive evaluation of our method on the most commonly used image datasets, i.e., CIFAR10, CIFAR100, STL10, and SVHN. The classification task is chosen to evaluate the discriminability of the representation learned by our transformation regression learning method. We also show the learned weights of our model can serve as an initialization for successive classification model. The datasets are introduced as follows.

- CIFAR10[1]: The dataset consists of 60,000 32×32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. It is widely used in deep learning community.
- CIFAR100: It is just like the CIFAR10, except it has 100 classes with each class containing 600 images. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs). We only use the fine label during evaluation.
- STL10[2]: The STL10 dataset is an image recognition dataset widely used for developing unsupervised feature learning algorithms. It contains 13,000 labeled color images with size of 96×96 pixels, divided uniformly into 10 classes, and 100,000 unlabeled

TABLE III
Classification accuracy (%) of a multiple layer perceptron (MLP) network on representations of different levels. The "Input" denotes the original pixel level, while "Block_i" the representation extracted from the ith block of the regression network for $i = 1, 2, 3$, respectively.

	Input	Block_1	Block_2	Block_3
CIFAR10	66.84	81.37	84.88	76.16
CIFAR100	39.60	53.19	57.06	44.07
STL10-10k	54.20	68.53	72.67	64.33
STL10-100k	54.20	71.00	77.03	67.80
SVHN	82.25	92.29	94.28	91.52

TABLE IV
Classification accuracy (%) of a one-block network on representations of different levels. The "Input" denotes the original pixel level, while "Block_i" the representation extracted from the ith block of the regression network for $i = 1, 2, 3$, respectively.

	Input	Block_1	Block_2	Block_3
CIFAR10	93.37	93.65	90.75	81.44
CIFAR100	70.84	71.42	65.69	49.01
STL10-10k	77.93	79.93	77.20	65.50
STL10-100k	77.93	81.03	79.60	67.17
SVHN	96.12	96.54	95.67	92.63

[1]http://www.cs.toronto.edu/ kriz/cifar.html
[2]https://cs.stanford.edu/ acoates/stl10

TABLE V
The result (accuracy in percentage) of training the same WRN as a classification model. The baseline is the random initialized WRN trained by supervised classification loss. (Freeze, First block) denotes the classification accuracy of the WRN with the first bock initialized by transformation regression and frozen during classification training. (Finetune, First block) is same with the supervised baseline except the first block is initialized by transformation regression.

		Supervised (baseline)	First block	First two blocks	All blocks
CIFAR10	Freeze	94.92	94.20	90.76	51.00
	Finetune		95.02	95.11	95.27
CIFAR100	Freeze	75.76	73.52	65.86	10.52
	Finetune		75.88	76.60	76.19
STL10-10k	Freeze	80.10	79.73	77.50	48.03
	Finetune		82.17	82.40	83.93
STL10-100k	Freeze	80.10	80.07	83.63	47.67
	Finetune		83.47	82.70	84.10
SVHN	Freeze	96.45	96.71	95.67	51.91
	Finetune		96.62	96.58	96.60

images. The labeled images are divided into training set with 10,000 images and testing set 3,000 images. All images are resized to 32×32 pixels. STL10-10k denotes training the regression model by 10,000 images and STL10-100k by 100,000 images.

- SVHN[3]: This is a real-world image dataset for developing machine learning algorithms. It has 10 classes, representing digit "0" to "9". There are 73,257 digits for training, 26,032 digits for testing. Each image is with size 32×32 pixels.

All datasets are normalized by rescaling the elements to [0,1] and then normalized to zero mean and unit standard deviation. We only use the training set during regression. A summary of dataset statistics is shown in Table II.

A. Evaluation Protocol

Our primary goal is to evaluate the discriminability of the neural network $\mathcal{F}$ trained by optimizing (2). To this end, we perform classification task on the features extracted from different layers of the neural network and use the classification accuracy as a metric to evaluate the discriminability. Since the neural network is a wide residual network (WRN) with 3 residual blocks, we extract features from each block and report the corresponding classification accuracy. As a baseline, we also report the classification accuracy on the original pixel level feature. We successively take a multiple layer perceptron (MLP) and a deep convolutional neural network (CNN) as the classifier. For all datasets except STL10-100k, both the regression model and classifier are trained on the training set but the former does not use the class labels and the latter does. For STL10-100k dataset, the unlabeled dataset with 100,000 images is used to train the regression model and the training set with 13,000 images for the classifier. After training the classifier, the classification accuracy on the testing set is reported.

[3]http://ufldl.stanford.edu/housenumbers

B. MLP classification on Learned Representation

We first evaluate the quality of the representation extracted from the WRN trained by transformation regression loss via feeding representations of different levels to a multiple layer perceptron (MLP) classifier. The MLP classifier is composed of three fully connected layers which have 400, 400, and K neurons, respectively, where K is the number of classes. Each layer, except the output layer, is followed by a batch normalization layer [25] and a ReLU activation function. We use cross entropy loss and SGD optimization algorithm with initial learning rate 0.1, Nesterov momentum 0.9, weight decay 0.0005. The learning rate is multiplied by 0.2 at the 60th, 120th, and 160th epoch, and the total number of training epochs is set to 200 with batch size 128. We only use random translation for at most 4 pixels in each direction and random horizontal flipping. The input data is preprocessed by first scaling the pixel values to [0,1] and then normalizing to zero mean and unit standard deviation.

The result is listed in Table III. The "Input" denotes the original pixel level feature which serves as the baseline classification performance. The representations extracted from WRN lead to much higher classification accuracies than the baseline. This validates that the proposed transformation regression method can learn meaningful and discriminative representation which favors simple MLP classifier. When the representation goes deeper, the MLP classification accuracy first increases and then decreases, and the "Block_2" representation achieves the highest performance. This result makes sense because the representation from network layer close to the output is prone to be specific to the transformation regression task and hence are not suitable for classification. The representation from shallower layer tends to be less task-specific and can be used for general purpose. This experiment validates that if the classifier at hand is not powerful enough, the representation learned by transformation regression can dramatically boost the classification performance.

C. CNN Classification on Learned Representation

We then perform classification by using a more powerful deep convolutional neural network (CNN) which consists of one WRN block, total 12 convolutional layers where the first convolutional layer is with kernel size 3, stride 2, and padding 1. The number of input and output channels is c_{in} and 128, respectively. The other layers are with stride 1 and padding 1. Each layer is followed by a batch normalization layer [26] and a ReLU activation function. The last convolutional layer is followed by a global average pooling layer and a fully connected layer. The number of channels c_{in} equals 3, 32, 64, and 128 for "Input", "Block_1", "Block_2", and "Block_3", respectively. The data augmentation, loss function and optimization parameters are same with the MLP classifier.

As shown in Table IV, the representation from the first block achieves the highest classification accuracy on all datasets. Due to the high capacity of the CNN classifier, the classification on the input pixel level representation is approaching the performance of the best supervised CNN model. Therefore the "Input" is more like an upper-bound baseline. However, our "Block_1" representation can still slightly outperforms "Input" which indicates that the first block of our regression model WRN can not only preserve enough information for training the powerful CNN classifier, but also enhance the discriminability of the learned representation compared with the input pixels. The representation from deeper block is less discriminative (losing more discriminative information) than the shallower ones since the accuracy drops gradually with representation changing from "Block_1" to "Block_3". This experiment further validates the effectiveness and discriminability of the representation learned by transformation regression method.

D. Pretraining by Transformation Regression

By replacing the fully connected layer of the regression model WRN with a K-neuron layer where K is the number of classes, we can construct a classification model. After training the WRN by transformation regression loss, we copy the weights in the first block, the first two blocks, and all blocks to the corresponding blocks in classification WRN, respectively. The copied weights can be frozen or finetuned during classification training. We also report the baseline result of classification WRN with all weights randomly initialized. In all settings, the data augmentation and optimization parameters are all same with that in the last subsection.

The results are shown in Table V. For CIFAR10 dataset, the first row "Freeze" gives the classification results of supervised training from scratch, copying and freezing the first block, the first two blocks, and all blocks. Freezing the first block leads to a comparable classification performance to the baseline supervised model, despite the former setting has only two blocks to train while the latter has three. Freezing all blocks results in inferior classification performance, because the representation close to the output is task-specific. This again validates the effectiveness of the first block representation learned by transformation regression, which is consistent with the conclusion drawn in the last subsection. The second row "Finetune" denotes the classification results when the weights in the first block, first two blocks, and all blocks are initialized by the corresponding weights in the regression WRN. As can be seen, using the pretrained weights by transformation regression can improve the baseline classification performance. This improvement is more significant on STL10 dataset where two to four percentage on accuracy improvement over baseline is observed.

The training statistics on STL10-130k are recorded in Figure 2. The losses of the pretrained model decrease faster than the baseline random initialized supervised model and reach the lower values. The classification accuracies are changing in the similar trend. This validates that the weights pretrained by transformation regression can used for initialization of classification model.

TABLE VI
Comparison with the state-of-the-art unsupervised learning methods. The result marked by * is excepted from [15] and that marked by † is from [21]. The other methods that have no markers share the same network structure and optimization parameters.

	CIFAR10	CIFAR100	STL10	SVHN
DCGAN* [12]	82.80	–	–	–
Split-Brain† [27]	67.10	39.00	–	77.30
Counting† [28]	50.90	18.20	–	63.40
AND† [21]	77.60	47.90	–	93.70
RotNet* [15]	91.16	–	–	–
TR (Ours)	93.65	71.42	79.93	96.54
Supervised	94.92	75.76	80.10	96.45

E. Comparison with the State-of-the-art Methods

We compare our proposed transformation regression learning method with state-of-the-art unsupervised learning methods including DCGAN [12], Split-Brain [27], Counting [28], AND [21], and RotNet [15]. The result of our method is the CNN classifier on the "Block_1" representation as shown in Table IV.

As shown in Table VI, our transformation regression (TR) method achieves the highest performance on all datasets. The classification accuracy on the unsupervised representation is approaching the supervised model. On SVHN, our method can even slightly outperforms the supervised model.

F. Ablation Study

Our regression WRN uses four types of transformations, namely rotation, horizontal translation, vertical translation, and scaling. We test the CNN classification performance on the first block when each type or combination of these transformations is used for regression. As Table VII shows, when the transformations are separately used (see

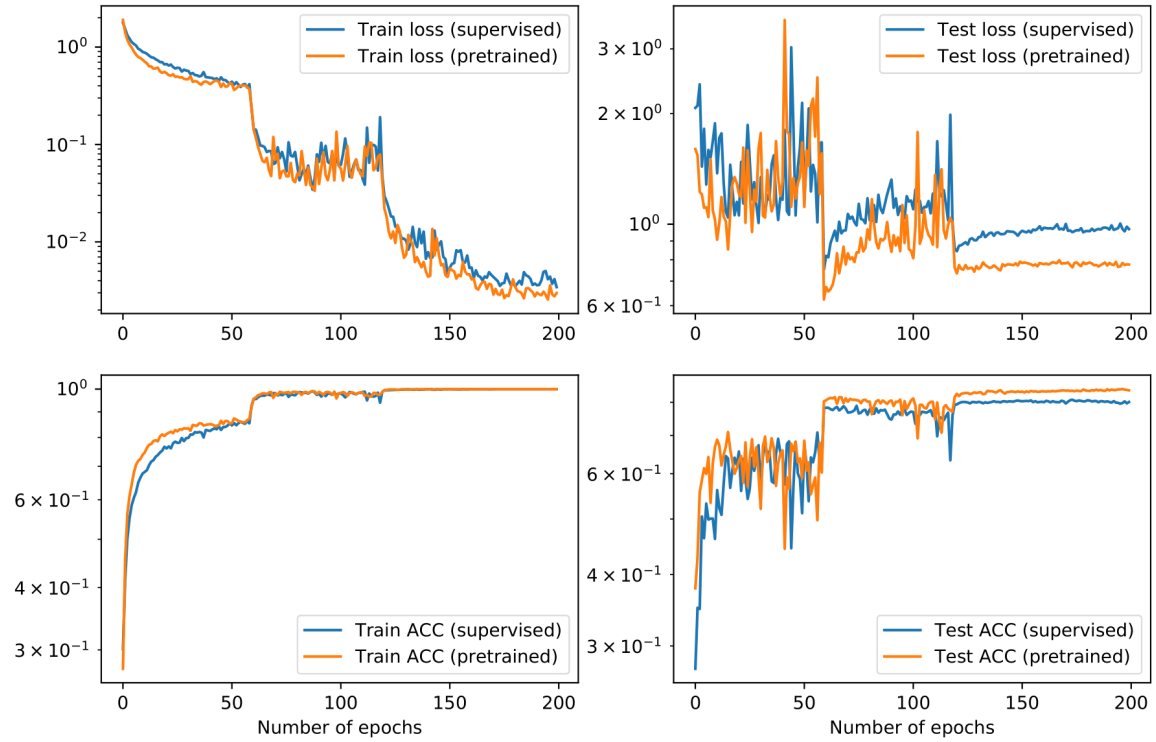

Fig. 2. The training statistics of the baseline supervised setting and all blocks pretrained setting on STL10. The classification WRN using the pretrained weights converges faster than random weights and the final losses (including training loss and testing loss) are also lower.

the first, second, and forth rows), the performances are very bad. This is because the neural network can easily learn to predict the parameters of only one transformation and thus tend to overfit the obvious pattern. For example, if an image is rotated by 45 degrees, the neural network can predict the angle by simply focusing on the edges. However, combining two or three types of transformations makes the network difficult to predict all transformation parameters precisely only by observing simple patterns like edges. Therefore, the network is forced to capture patterns hiding in the image content in order to successfully infer the transformation parameters. But the edge effect is still harmful to the learning process. How to eliminate the edge effect will be one of our future works. To conclude, this analysis quantitatively validate our assumption that image representation can be learned by regressing the parameters of a combination of transformations.

TABLE VII
Analysis on different transformations.

Rotation	Translation	Scaling	ACC (%)
0	0	[0.5, 1.5]	90.97
0	[-10, 10]	1.0	87.97
0	[-10, 10]	[0.5, 1.5]	91.48
[-180, 180]	0	1.0	87.99
[-180, 180]	0	[0.5, 1.5]	92.21
[-180, 180]	[-10, 10]	1.0	93.24
[-180, 180]	[-10, 10]	[0.5, 1.5]	**93.65**

V. Conclusion

We propose a new image representation learning method by constructing a regression task whose target is to predict the continuous parameters of some transformations applied to the input image. Extensive experiments on various image datasets validate the effectiveness and discriminability of representation learned by our proposed transformation regression method. Our method not only outperforms existing representation learning methods by a large margin, setting new state-of-the-art performance in terms of successive classification accuracy, but also is comparable to the supervised model. Besides, it can further improve the performance of supervised models by initializing their weights with our training mechanism. Future work includes: 1) exploring other types of transformations like image flipping, cropping, and color jitter; and 2) eliminating the edge effect when applying some transformations like image rotation.

VI. Acknowledgment

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

[1] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks." Science, vol. 313, no. 5786, pp. 504–507, 2006.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in 2nd International Conference on Learning Representations (ICLR), 2014.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in Neural Information Processing Systems (NeurIPS), 2014, pp. 2672–2680.

[4] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao, "Stacked convolutional denoising auto-encoders for feature representation," IEEE transactions on cybernetics, vol. 47, no. 4, pp. 1017–1027, 2016.

[5] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008, 2008, pp. 1096–1103.

[6] A. Makhzani and B. J. Frey, "k-sparse autoencoders," in 2nd International Conference on Learning Representations (ICLR), 2014.

[7] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011, 2011, pp. 833–840.

[8] P. Goyal, Z. Hu, X. Liang, C. Wang, and E. P. Xing, "Nonparametric variational auto-encoders for hierarchical representation learning," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 5094–5102.

[9] J. Walker, C. Doersch, A. Gupta, and M. Hebert, "An uncertain future: Forecasting from static images using variational autoencoders," in European Conference on Computer Vision. Springer, 2016, pp. 835–851.

[10] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," arXiv preprint arXiv:1502.04623, 2015.

[11] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017, pp. 1587–1596.

[12] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in 4th International Conference on Learning Representations (ICLR), 2016.

[13] L. Tran, X. Yin, and X. Liu, "Disentangled representation learning gan for pose-invariant face recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1415–1424.

[14] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in Advances in neural information processing systems (NeurIPS), 2016, pp. 2172–2180.

[15] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in 6th International Conference on Learning Representations (ICLR), 2018.

[16] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1422–1430.

[17] G. Larsson, M. Maire, and G. Shakhnarovich, "Learning representations for automatic colorization," in European Conference on Computer Vision (ECCV). Springer, 2016, pp. 577–593.

[18] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in European conference on computer vision (ECCV). Springer, 2016, pp. 649–666.

[19] P. Agrawal, J. Carreira, and J. Malik, "Learning to see by moving," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 37–45.

[20] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in European Conference on Computer Vision (ECCV). Springer, 2016, pp. 69–84.

[21] J. Huang, Q. Dong, S. Gong, and X. Zhu, "Unsupervised deep learning by neighbourhood discovery," in Proceedings of the 36th International Conference on Machine Learning (ICML), 2019, pp. 2849–2858.

[22] S. Zagoruyko and N. Komodakis, "Wide residual networks," in Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016, 2016.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, 2015, pp. 1026–1034.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in Proceedings of the 32nd International Conference on Machine Learning (ICML), 2015, pp. 448–456.

[26] ——, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.

[27] R. Zhang, P. Isola, and A. A. Efros, "Split-brain autoencoders: Unsupervised learning by cross-channel prediction," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 645–654.

[28] M. Noroozi, H. Pirsiavash, and P. Favaro, "Representation learning by learning to count," in IEEE International Conference on Computer Vision (ICCV), 2017, pp. 5899–5907.