2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)
Bari, Italy. October 6-9, 2019

RC-CNN: Representation-Consistent Convolutional Neural Networks for Achieving Transformation Invariance

Jun Gu, Anfeng He, Xinmei Tian
CAS Key Laboratory of Technology in Geo-spatial Information Processing and Application System
University of Science and Technology of China, Hefei Anhui, China
Email: gujun110@mail.ustc.edu.cn, heanfeng@mail.ustc.edu.cn, xinmei@ustc.edu.cn

***Abstract*—Convolutional neural networks (CNNs) are powerful and have achieved state-of-the-art performance in many visual recognition tasks. Despite their impressive performance, CNNs are still unable to remain invariant while some spatial transformations are applied on images. Herein, we propose representation-consistent neural networks to solve this problem. By introducing consistent losses between the representations in different layers of transformed images, the recognition performance of transformed images is significantly improved. This model not only learns to map from the transformed images to the pre-defined labels but each layer also learns to generate invariant representations when the input images are transformed. All the characteristics of transformation invariance are embedded in the model, which means that no extra parameters or computations are introduced in the well-trained model. Comparative experiments demonstrate the superiority of our model when learning invariance to rotation, translation, and scaling on large-scale image recognition and retrieval tasks.**

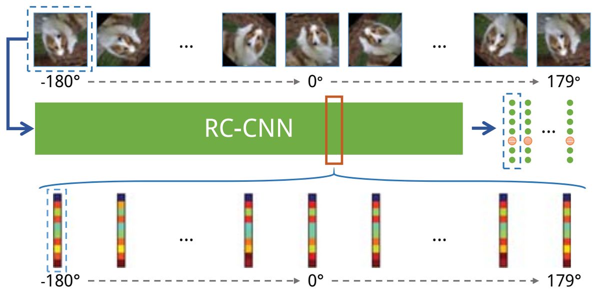

Fig. 1. Illustration of our RC-CNN. To push the recognition results to be the same as the ground truth for transformed images, we push the inner representations of the transformed images to be consistent by adding consistent losses in the network.

I. Introduction

Over the years, computer vision has rapidly advanced as a result of adopting convolutional neural networks (CNNs). A considerable number of CNN-based models have achieved state-of-the-art results in image recognition [1], [2], image retrieval [3], [4], [5], action recognition [6], [7] and localization [8], [9], [10] tasks. These advances are primarily due to the excellent discriminative description ability of deep neural networks, which can easily fit any nonlinear function according to the data space.

An ideal image recognition system should be robust to image transformations. Although distortions or translations of the input can cause the positions of salient features to vary, local receptive fields with shared weights in CNNs have the ability to detect invariant elementary features despite changes in the positions of salient features [11]. Furthermore, with the local max-pooling or average-pooling [12] layers, feature maps are downsampled layer by layer. Consequently, the model becomes increasingly more insensitive to small local translations and distortions. Nevertheless, when faced with large and global transformations, the traditional CNNs cannot be sufficiently insensitive due to the relatively small size of local receptive fields and pooling kernels. An easy approach for addressing the problem of large global transformations and the most popular one in deep learning is data augmentation [13]. Using this approach, any transformation can be applied to the original images. These transformed images are used as new training samples during learning. Rotation, translation and scale are the most common transformations in general computer vision tasks. By using the data augmentation technique, the models in CNNs are able to access numerous unseen images with different transformations. Therefore, the models can easily learn the mapping from the transformed images and the raw images to the predefined labels.

Although data augmentation is an easy choice when training CNNs, the transformation invariance is not good enough. This is because this approach only forces the model to learn a mapping from the transformed images to pre-defined labels one by one. As shown in Fig. 2, when training a model using data augmentation, the inner relationship between the transformed images is not sufficiently taken into account. However, the key to transformation invariance is to learn a model that outputs an invariant representation before or after transformation. Training a model using data augmentation is a simple approach to push the model to represent invariantly on high-level features. However, it is difficult to expect an

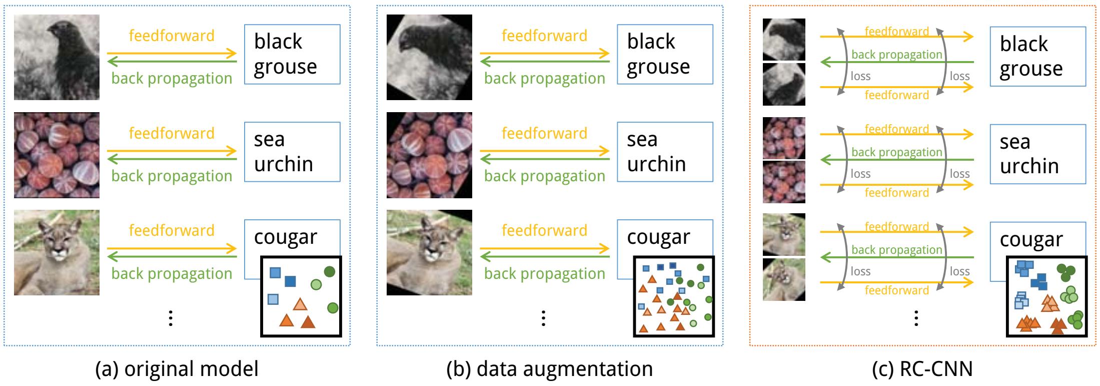

Fig. 2. Difference between original CNNs, CNNs with data augmentation, and our RC-CNN. The small patches on the lower right are demonstrations of the spatial distributions of the feature representation of transformed images. (a) Feature maps generated by the traditional CNNs. Points having the same shape but different brightness mean different images from the same category. (b) After transformation, although the model is trained with transformed images, some feature maps of the transformed images are still easily mixed together. Thus, the performance of the model will decrease. (c) By adding consistent losses, feature maps of images generated from the same image with different transformations will be closer. Therefore, all images can easily be recognized.

invariant representation by adding loss on top-layer features because low-level features are easily mixed together if there is no constraint on low-level features. However, as mentioned previously, with sufficient parameters and depth of multi-layer networks, CNNs have the ability to learn any mapping from any images to transformation-invariant representations. Thus, we should provide a global approach to push the CNN-based model to learn how to represent images invariantly to the transformation of images.

In this work, we train a representation-consistent convolutional neural network (RC-CNN) for image recognition tasks. As shown in Fig. 1, rather than simply training this model using data augmentation, we train our networks with consistent losses to push the model to learn how to represent invariant feature maps at both low-level features and high-level features. The characteristic of transformation invariance is embedded in the networks. Therefore, there are neither extra parameters in the model nor extra computation during testing. Intuitively, while inputting transformed images, if we push the representations in the low layers to be more consistent, the representations of the higher layers will consequently be consistent. Finally, a consistent representation at the label level will be promoted, which means that the model is able to achieve better performance. However, the effect of the lowest layers such as conv1 and conv2 is detecting some primary features of images, such as edges and corners. Thus, we want to impose invariance at the middle and higher layers of the network (see Section IV for details).

In contrast to data augmentation, in which the models only tend to learn the mapping from transformed images to the pre-defined labels, our representation-consistent model constrains the feature map of the convolution layers and fully connected layers to be stable and invariant after any type of transformation, including rotation, scale, and translation. Consistent losses push the model to be consistent on three levels. The highest one is the consistency of class labels, which means that the labels of any views of the same image should be consistent. The second one is consistency of high-level semantic features, which means that we want the high-level representation of the model to be consistent. The third one is consistency of low-level features, which means that the feature extractors such as convolution layers are pushed to generate consistent feature maps for transformed images generated from the same original image. In contrast to other models for transformation invariance, our model is better while transferring to other tasks of transformation invariance because representations are pushed to be consistent while images are transformed. Meanwhile, with standard back-propagation, this model can be trained end-to-end as with training normal CNNs. No extra optimization process is required to train the model. The model of our RC-CNN is the same as that of traditional CNNs. Thus, our model can be easily applied to any applications by simply replacing only the parameters.

II. Related Works

A. *Low-Level Transform-Invariant Features*

Previously, due to limited computational resources and technology, the invariant representation of transformed data was relatively low-level. Scale-invariant feature transformation (SIFT) [14] is the most famous and most effective approach. The first step of SIFT is to detect key points. The majority of the key points are corner points or blobs in images. Thus, SIFT is able to be invariant while the location of an object is changed. Then, the features of the key points are extracted. By detecting the orientation of the image patch near key points and using the difference of Gaussian (DoG) algorithm, the SIFT algorithm is able to obtain the ability of rotation and scale invariance. Raw pixels are directly operated by the SIFT algorithm; thus, the speed is fast when extracting features. However, although it has the great ability of transformation invariance and some additional benefits, the performances of

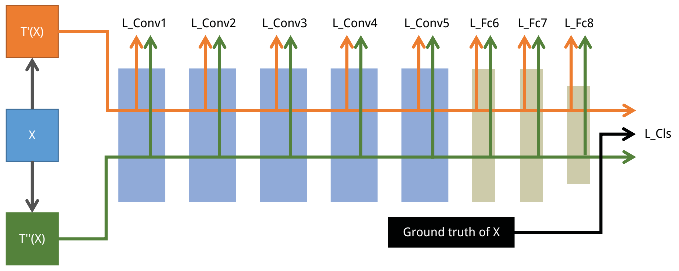

Fig. 3. The framework of our RC-CNN. The raw image X is transformed by two groups of random parameters. The consistent losses can be added in each layer to constrain the network representation consistently while any random transformation is added to the input raw images. Softmax losses are added at the ends of the networks to ensure that the classification performance is good.

these algorithms are not good enough due to the limited description ability of low-level features.

B. Invariance on CNNs

Recently, with the rapid development of parallel computing, deep convolutional neural networks (DCNNs) have become a hot topic in the field of computer vision. As previously mentioned, data augmentation is a popular approach to solve the global transformation-invariant problem. However, to obtain better performance, other approaches with a more complex model are proposed.

A novel way to address the scale-invariant problem is SI-CNN [15]. Different scale parameters in multiple levels are applied to the feature maps. Then, the feature map of all scales is convolved by the same kernels. Finally, after normalization and polling, an invariant representation for the local scale is obtained. However, extra learnable parameters are needed in this model. Furthermore, in this paper, only the scale-invariant problem was considered. In our model, we embed all characteristics of transformation invariance in the parameters of networks, which means that no extra parameters or computations are introduced in the well-trained models. Meanwhile, by constraining the feature maps of any type of transformation, the proposed model is able to handle all types of transformations.

One of the latest works to obtain the invariant model is the spatial transformer network [16]. A new layer is introduced in this model to transform the distorted images back to non-transformed images. However, the invariance of this model needs more learnable parameters. Meanwhile, the invariance of this model arises from additional spatial transformation modules. Thus, the invariance of the CNN model is not promoted. Another recent work to incorporate the transformation-invariant problem is TI-CNN [17]. By introducing transformations to the feature maps of CNNs during the training phase, the network is able to observe more combinations of transformations at different levels. This approach can be viewed as an enhanced data augmentation. However, the essence of this model is still to learn a mapping from transformed images to pre-defined labels.

In this paper, in contrast, extremely simple constraints are added between feature maps of raw images and transformed images during the training stage. With these constraints, our proposed model starts obtaining intrinsic invariance ability to variety of transformations by pushing the model representing consistent when transforming. No more parameters or layers of the CNNs are required which means that no extra time and space is consumed when using our proposed model. In addition, after training finished, the model is easy to applied to any occasions by simply replacing weights from the original model to the new model.

III. Representation-Consistent CNN

In this section, the formulation of our representation-consistent CNN model is introduced. The goal of our method is to learn a transformation-invariant model by learning invariant representations of transformed images layer by layer. The representation-consistent ability is achieved by introducing consistent losses between the feature representation of transformed images and original images. As shown in Fig. 3, consistent losses are added in each layer of the original CNNs. By adding consistent losses, the features of any two transformed images are consistent and invariant.

A. Convolutional Layers and Fully Connected Layers

Convolutional neural networks (CNNs) are multi-layer architectures. Different representations are learned in each layer by applying convolutional kernels as feature detectors to previous feature maps. A classifier or regressor with a target loss function is generally applied to the outputs of the final layer. Shared-weight convolutional kernels and spatial pooling layers are great ways to obtain some small local translations or scale-invariant ability. Meanwhile, the number of trainable parameters is reduced.

The feed-forward of CNNs is the bias plus the feature map of the input image convolved with the kernels. Then, a non-linear activation function is applied. These steps can be written as follows:

$$X_i^j = f\left(\left(W_i^j * X_{i-1}\right) + b_i^j\right) \tag{1}$$

where W_i^j is the j^{th} kernel of the convolutional layer and b_i^j is the bias of the j^{th} feature map X_i^j in the i^{th} layer (X_0 is the input image data). Both W_i^j and b_i^j are trainable weights. The symbol $*$ is the convolution operator, X_{i-1} are the feature maps of the previous layer, and $f(\cdot)$ is a non-linear function, e.g., ReLU, sigmoid, or tanh function.

The equation for the fully connected layer is quite similar to equation (1) except that the symbol $*$ is replaced by the matrix multiplication symbol $\times$:

$$X_i^j = f\left(\left(W_i^j \times X_{i-1}\right) + b_i^j\right) \tag{2}$$

B. Image Transformation

In this work, constraints are applied between the transformed images generated from the same original image. The first step in our method is to transform an image. The most frequently used three types of 2D transformation are rotation, translation, and scale [18]. We denote the location of a pixel in the original image and in the transformed images as (x, y) and (x', y'), respectively. The transformation of an image can be calculated as follows:

$$(x', y', 1) = (x, y, 1) \times \mathcal{T} \tag{3}$$

where $\mathcal{T}$ is a transformation matrix.

The transformation matrix of rotation can be written as follows:

$$\mathcal{T}_R = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where θ is the rotation angle.

The transformation matrix of translation can be written as follows:

$$\mathcal{T}_T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ d_x & d_y & 1 \end{bmatrix} \tag{5}$$

where d_x and d_y are the translation distances in the x and y axes, respectively.

The transformation matrix of scale can be written as follows:

$$\mathcal{T}_S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

where s_x and s_y are the scale rates in the x and y axes, respectively.

By using the transformation matrix, it is easy to add all transformations as follows:

$$\mathcal{T}_{RTS} = \mathcal{T}_R \times \mathcal{T}_T \times \mathcal{T}_S \tag{7}$$

C. Representation-Consistent CNN

As previously mentioned, although local receptive fields with shared weights and spatial pooling operations make the CNNs invariant to small local transformations in the traditional CNNs, it is difficult to be invariant, and the performance becomes worse significantly while the transformation is global and larger. In this section, a representation-consistent CNN is proposed. By introducing consistent losses between the feature maps of transformed images, the network starts to learn how to represent invariantly while the input image is transformed. In contrast to other related approaches, the transformation-invariant ability is learned in the network without any additional learnable parameters or extra models. Fig. 2 shows the difference between the original model and our model.

As shown in Fig. 3, in the training phase, the input of this network is raw image X. The two random transformed images X' and X'' can be written as follows:

$$X' = \mathcal{T}'(X), X'' = \mathcal{T}''(X) \tag{8}$$

where $\mathcal{T}'(\cdot)$ and $\mathcal{T}''(\cdot)$ are transformation operators with the two different transformation matrices mentioned previously.

Then, feed-forward propagation is applied to these two transformed images X' and X''. The equations for convolution and the fully connected layers are shown in Eq. 1 and Eq. 2, respectively. The consistent loss of the i^{th} layer $\mathcal{L}_i$ is added between the two transformed images' feature representations $Fea_i(X')$ and $Fea_i(X'')$, i.e., the output representation of convolutional layers or fully connected layers:

$$\mathcal{L}_i = \frac{1}{2} \parallel Fea_i(X') - Fea_i(X'') \parallel_2^2 \tag{9}$$

The loss function of the entire model can be written as follows:

$$\mathcal{L}_{All} = \lambda_{Cls} \times (\mathcal{L}'_{Cls} + \mathcal{L}''_{Cls}) + \sum_i \lambda_i \times \mathcal{L}_i \tag{10}$$

where λ_i and λ_{Cls} are the coefficients to trade off the consistent losses $\mathcal{L}_i$ in each layer and the classification losses $\mathcal{L}_{Cls}$. $\mathcal{L}'_{Cls}$ and $\mathcal{L}''_{Cls}$ are the classification losses corresponding to transformed images X' and X'' respectively. Suppose that N is the number of classes; the classification loss $\mathcal{L}_{Cls}$ is the loss of an N-way softmax layer.

During the testing phase, the testing images are pre-transformed. Thus, we directly input the transformed testing images into our RC-CNN model.

IV. Experiments

To compare our model with other state-of-the-art methods, our proposed RC-CNN is evaluated on two tasks. To evaluate its performance for large-scale image recognition tasks, extensive experiments are conducted on the ImageNet [19] dataset in Section IV-A. To verify the generalization ability of the trained RC-CNN model on other transformation-invariant vision tasks, corresponding image retrieval experiments are conducted on the UK-Bench [20] and UK-Bench+ MIRFlickr [21] datasets in Section IV-B.

A. Large-Scale Image Recognition

ILSVRC-12 [19] is a famous large-scale image dataset in the field of computer vision. This dataset consists of three subsets: the training set (1.3M images), the validation set (50K images), and the testing set (100K images). This dataset is a subset of ImageNet and contains 1,000 categories of images. In this work, the performance of our RC-CNN on large-scale image recognition tasks is evaluated on this dataset. The recognition performance is evaluated in terms of the top-1 accuracy and the top-5 accuracy. The top-1 accuracy is a multi-class classification accuracy. The top-5 accuracy is the

TABLE I
THE ACCURACIES (TOP-1/TOP-5) OF OUR PROPOSED RC-CNN AND OTHER CNNS ON TRANSFORMED VALIDATION SET. R, T, S, AND RTS REPRESENT THE TRANSFORMATIONS APPLIED TO THE IMAGE, WHICH ARE ROTATION, TRANSLATION, SCALE, AND ROTATION-TRANSLATION-SCALE, RESPECTIVELY.

Model	R	T	S	RTS
CNNs	36.7/57.7	46.5/70.8	53.3/76.9	33.3/55.2
CNNs+Data Augmentation	36.5/58.3	50.0/73.9	54.0/77.6	36.0/58.2
SI-CNN [15]	36.8/58.9	-	53.8/77.3	-
ST-CNN [16]	37.2/59.3	51.4/75.3	53.8/77.3	37.0/60.1
TI-CNN [17]	37.3/60.2	51.3/75.4	55.0/78.5	37.5/60.5
RC-CNN(Cls) (ours)	38.1/60.8	53.0/76.2	55.8/78.8	38.6/61.4
RC-CNN(Conv+Cls) (ours)	38.7/61.7	52.9/76.3	**56.8/79.7**	37.9/61.2
RC-CNN(FC+Cls) (ours)	38.7/61.4	**53.4/77.2**	56.5/79.3	**39.2**/61.8
RC-CNN(Conv+FC+Cls) (ours)	**39.1/61.9**	53.3/76.7	56.7/**79.7**	38.9/**62.1**

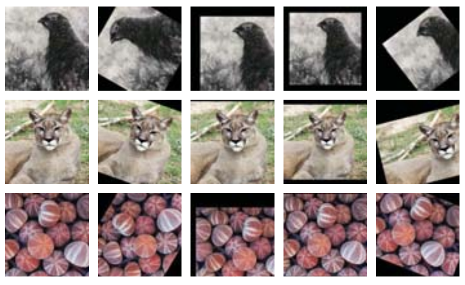

Fig. 4. Examples of transformed images. ORI: original images without transformation, R: rotation, T: translation, S: scale, and RTS: rotation-translation-scale.

fraction of test images for which the correct label is among the five labels predicted as being most probable.

Because the sizes of the raw images are not the same, all images in the training set are pre-processed to ensure that the sizes of the images are the same. The pre-processing is the same as that in AlexNet [1], in which images are rescaled to ensure that the shortest sides of the images are 256 pixels. Then, the center 256×256 region of the image is cropped.

When training the model, random transformations are applied to the original images. To evaluate the generalization ability to transformation-invariant tasks, rather than training the model with the same transformation settings as the testing phase such that the model learns to map from the original images to the ground-truth label, training images are generated with less distortion than the validation images. Training images of rotation (R) are generated by randomly rotating images in $\mathcal{N}\left(0,(\pi/12)^2\right)$. Training images of translation (T) are generated by randomly shifting images by a proportion of $\mathcal{N}\left(0, 0.075^2\right)$. The scaled training images (S) are generated by randomly scaling images in $\mathcal{N}\left(1, 0.1^2\right)$. The settings for the rotation-translation-scale (RTS) dataset are a combination of all the transformations mentioned above. Finally, a randomly located 227×227 region is cropped in the image.

To evaluate the transformation-invariant ability of our model, rather than evaluating our model on the original images, some random transformations are applied on the validation images to make the task more difficult. As previously mentioned, when testing the model, transformations applied to the validation images are more distorted than the training images. Thus, the method to generate validation images is similar to the method to generate training images except that the parameters for the transformed images follow a uniform distribution: $R : \theta \sim \mathcal{U}\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, $T : d_x, d_y \sim \mathcal{U}\left[-0.2, 0.2\right]$, and $S : s_x, s_y \sim \mathcal{U}\left[0.7, 1.2\right]$. The RTS dataset is a combination of all the transformations mentioned above. Some transformed images are presented in Fig. 4 as examples.

The architecture of each column in our model is based on AlexNet [1]. Consistent losses are placed after ReLU layers except for the conv1, conv2 and conv5 layers, where consistent losses are placed after polling layers. The weights and biases of the convolutional kernel are initialized by $\mathcal{N}\left(0, 0.01^2\right)$ and constant value 0.1, respectively. The weights and biases of the fully connected layers are initialized from $\mathcal{N}\left(0, 0.005^2\right)$ and constant value 0.1. We use a base learning rate of 0.01 and decay it by 0.1 every $100,000$ iterations. We use a momentum of 0.9, gradient clip of 35 and weight decay of 0.0005. The learning rate of the bias is set to be $2\times$ the learning rate for the weights. The RC-CNN model is trained for $450,000$ iterations. Our model is implemented by using the open source Caffe framework [22]. To solve this problem, the weights and biases of the model should be optimized for this network. In our work, stochastic gradient descent (SGD) [11] is used.

We set the coefficient $\lambda_{Cls} = 0.5$, since $\mathcal{L}'_{Cls}$ and $\mathcal{L}''_{Cls}$ contribute equally for the model. To set suitable values for the coefficients λ_i corresponding to consistent losses in five convolutional layers and three fully connected layers (as shown in Fig. 3), we calculated the consistent loss in traditional CNNs, as shown in Fig. 5. We can see that the orders of magnitude of these losses are very different. To ensure the consistent loss in each layer has equal contribution, we set λ_i according to the order of magnitude of these losses in CNNs. Specifically, we select λ_i for convolutional layers from $\{10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}\}$ and select λ_i for fully connected layers from $\{10^{-5}, 10^{-4}, 10^{-3}\}$. A randomly select subset of training data (10000 images) are used for these parameter selection. λ_{Conv1} and λ_{Conv2} are empirically set as zero, since these two layers extract very low-level features (e.g., edges and corners) and there is no need to put consistent constrains on

TABLE II
PERFORMANCE OF OUR PROPOSED RC-CNN AND OTHER CNNS ON THE ORIGINAL VALIDATION DATASET.

Model	Accuracy(Top-1/Top-5)
CNNs	57.1/80.2
SI-CNN [15]	57.3/80.3
ST-CNN [16]	59.1/81.7
TI-CNN [17]	57.6/80.5
RC-CNN (ours)	58.0/80.8

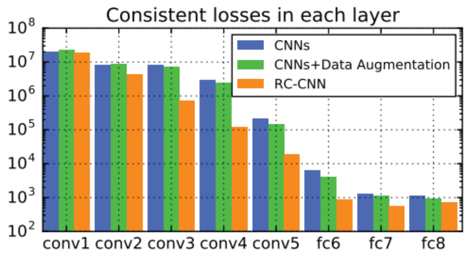

Fig. 5. Consistent losses $\mathcal{L}_i$ in each layer on the rotated validation dataset. A smaller value means that the feature map is more invariant to transformation. As consistent losses are introduced in the networks, the feature maps are becoming more robust to transformation.

them.

Table I summarizes the classification accuracies (top-1/top-5) of our proposed RC-CNN and the comparison with the original CNNs, original CNNs with data augmentation (CNNs+Data Augmentation), SI-CNN [15], ST-CNN [16] and TI-CNN [17] on distorted validation images. We evaluate several settings of our RC-CNN model, including RC-CNN(Cls), RC-CNN(Conv+Cls), RC-CNN(FC+Cls), and RC-CNN (Conv+FC+Cls). In RC-CNN(Cls), only the classification loss is applied and the λ_i for all convolutional layers and fully connected layers are zero. In RC-CNN(Conv+Cls), the classification loss and losses of three convolutional layers (Conv3, Conv4, and Conv5) are applied. In RC-CNN(FC+Cls), the classification loss and losses of three fully connected layers (FC6, FC7, and FC8) are applied. In RC-CNN(Conv+FC+Cls), all the three types of losses (the classification loss, losses of three convolutional layers, and losses of three fully connected layers) are applied.

From Table I, we can find that RC-CNN(Cls) achieves better performance than all baseline methods already. When convolutional layer loss or fully connected layer loss is further added, the performance is further improved, i.e., both RC-CNN(Conv+Cls) and RC-CNN(FC+Cls) outperforms RC-CNN(Cls) in most cases. Specifically, by introducing consistent losses in the convolution layers, the performance of the rotated images and scaled images improves significantly. However, the performance enhancement of translation and rotation-translation-scale is minimal. This result occurs because irrespective of the convolution filter, the feature map of translated images is the same but placed differently. Therefore, consistent losses always exist in translated images. Thus, adding consistent losses in convolution layers has no effect,

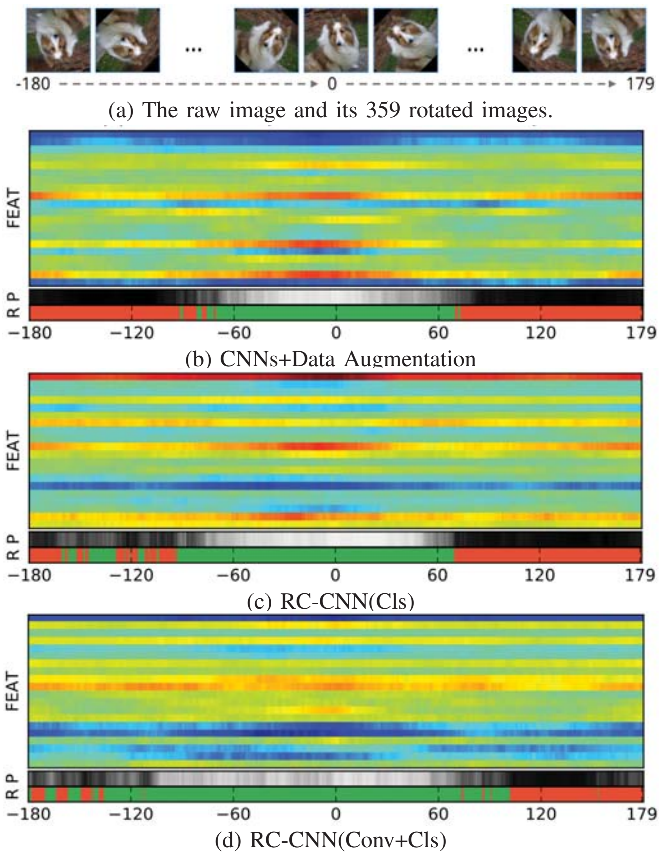

Fig. 6. Feature maps of the rotated images in the pool5 layer. (a) For a given raw image, we rotate it to obtain its transformed images. The angles of rotation range from $-180°$ to $179°$, resulting 360 images in total. For each image, we calculate the average value of its feature maps from the 256 channels in the pool5 layer, resulting a 256-dimensional vector. Then, the dimension of this vector is reduced from 256 to 20 by PCA. Thus, the $H \times W$ of the visualization image "FEAT" is 20×360. (b) (c) (d) Visualization of feature maps in the CNNs+Data Augmentation model, RC-CNN(Cls) and RC-CNN(Conv+Cls) separately. P: The confidence of correct prediction. A whiter color means that the model is more confident to the ground-truth label. R: Green color means that the prediction is correct, whereas red color means that the prediction is incorrect. These figures show that the feature maps of our RC-CNN model are more consistent. Moreover, our RC-CNN model is more confident in classifying the transformed image to a pre-defined ground truth. Finally, our RC-CNN model is able to acquire better performance.

and it will even make the situation worse. To improve the results of translated images, it is necessary to introduce consistent losses on a higher level. Therefore, we add consistent losses in the fully connected layers (RC-CNN(FC+Cls)) to push the model to be consistent at the semantic level. The results show that the performance of translated images improved significantly in this setting. However, the performance of rotated and scaled images is slightly worse. This result means that pushing representations in the middle layers of the network to be consistent is useful for achieving the goal of transformation invariance. We also attempt to place consistent losses in both convolution layers and fully connected layers - RC-CNN(Conv+FC+Cls). This model achieves a good balance and obtains a moderate result.

In addition, we also test our model on the original validation set without any tranformation. As shown in Table II, compared to the original CNN model, the performance of our model is better on the original validation images, which means that our model is generally better rather than over-fitting the transformed images.

To investigate the effect of the consistent losses on the CNN model, further visualization experiments are conducted. As shown in Fig. 6, the representation of our RC-CNN with consistent loss is more invariant to transformations. Consequently, the classification performance is improved. To quantitatively evaluate the performance of RC-CNN on the validation images, consistent losses are calculated in each layer, as shown in Fig. 5. The values in this figure are the average consistent losses between the feature maps of random transformed validation images. As shown in Fig. 5, by introducing consistent losses, the feature maps of random transformed validation images become more invariant to transformations.

Furthermore, the consistent representation of the middle layer will help the higher layers be consistent. As shown in Fig. 7, during the training phase, the softmax loss of the network without consistent losses is smaller than the loss of the network with consistent losses. However, during the testing phase, The softmax loss of the network without consistent losses is larger than the loss of the network with consistent losses. This result occurs primarily because networks without consistent losses are more apt to learn a model mapping from transformed images to only labels. By adding consistent losses, although the networks cannot map the transformed images as well as networks without consistent losses in the training dataset, the ability of RC-CNN is better while the network is optimized. This phenomenon indicates that by introducing consistent losses, the model can avoid over-fitting during the training phase.

In addition, to evaluate the ability of transformation invariance when extracting features, an invariance measure [23] is applied on our model. For more details of this invariance measure, please refer to [23]. Here, the transformed dataset is constructed by applying rotation ($[-\pi/4, \pi/4]$ with a step size of $\pi/20$), scale ($[0.7, 1.5]$ with a step size of 0.08), and translation ($[-0.2, 0.2]$ with a step size of 0.04) on the 50,000 validation images of ImageNet-2012 [19]. This measurement is applied at the end of each layer. As shown in Fig. 8, compared to the original CNNs, the invariance score of our proposed RC-CNN is significantly improved.

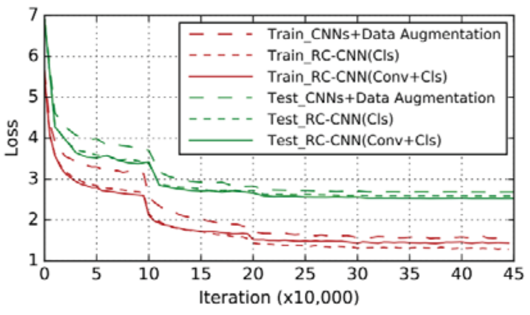

Fig. 7. Curves of classification losses during training and testing phases. The transformation applied to the images is rotation. A smaller classification loss at the end of the testing phase corresponds to better performance of the model. For better visualization, the curves in the training phase are averaged by 5,000 iterations. The results show that the optimization of the model is better by constraining the label-level representation to be consistent. In addition, by introducing the consistent losses in the lower level, although the classification losses in the training phase are larger than the model without consistent losses, the classification losses in the testing phase are smaller than the model without consistent losses. This result means that rather than simply learning the mapping from transformed images to the ground-truth label and over-fitting to the transformed training images by simply training the model with data augmentation, our RC-CNN model has more ability for generalization and consistent representations. Thus, our RC-CNN model has better performance in the testing phase.

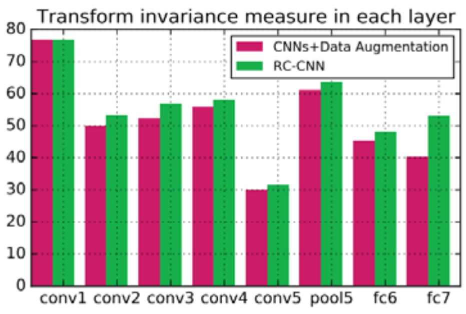

Fig. 8. Transformation invariance measurement in each layer. A larger value corresponds to better performance.

B. Image Retrieval

Our proposed model is also evaluated on the UK-Bench [20] dataset, which is an image retrieval task benchmark. This dataset is composed of $2,550$ groups of images. Each group contains 4 images of the same object or scene but from different viewpoints. The target of this dataset is using one image from among all images as the retrieval query to find the remaining 3 images from the same group. To evaluate our RC-CNN model in a large-scale dataset, an additional $1M$ images in MIR Flickr [21] are used as negative examples. Our model is directly applied without re-training or fine-tuning on the retrieval datasets.

The feature extraction method is similar to the technique in [24], where all images are fed into the RC-CNN and other CNN. On the last pooling layer, average pooling is applied. The size of the feature maps in the fifth pooling layer is $n \times 256 \times 13 \times 13$. To reduce the dimension of the feature representation, the average of the 256 channels is obtained. Then, 256-dimensional vectors are obtained for all images. Finally, the root value of each dimension is computed, and L2 normalization is performed for the 256-dimensional vectors. The final 256-dimensional vectors are used as features.

TABLE III
PERFORMANCE COMPARISON ON THE IMAGE RETRIEVAL DATASETS.

Model	UK-Bench	UK-Bench+MIRFlickr
CNNs	3.518	3.350
SIFT	3.350	3.295
SI-CNN [15]	3.531	3.398
ST-CNN [16]	3.540	3.412
TI-CNN [17]	3.572	3.483
RC-CNN (ours)	**3.727**	**3.639**

The image retrieval is processed after extracting the features. To evaluate our model on the UK-Bench dataset, the distances from the query image to all 10,200 images in the UK-Bench dataset are calculated. Then, the distances are sorted in ascending order. The distance metric is the Euclidean distance. The top four nearest images are used to calculate the NS-Score of the model. The NS-Score is the average accuracy of the top four images, which is generally used to evaluate the performance of models in retrieval tasks. If all four images in a group are correctly retrieved, then the model receives a score of 4.0.

To compare our proposed RC-CNN with other state-of-the-art CNNs, the NS-Score metric is applied to the results of all models. We compared our RC-CNN (Conv+FC+Cls) to the SIFT [14], CNNs [1], SI-CNN [15], ST-CNN [16] and TI-CNN [17]. As shown in Table III, the performance of our proposed RC-CNN achieves the best performance. The explanation for the success of our model is very simple: the original CNNs with or without data augmentation only learn the mapping from the images to the pre-trained labels. However, there is nothing to constrain the feature map to be transformation invariant when inputting transformed images. The essence of image retrieval is comparing the distance between the features of images. Consistent losses added between transformed images are able to push the representation of the image to be consistent and invariant to transformations.

V. CONCLUSION

In this paper, an effective and simple method is proposed to enhance the transformation-invariant ability of CNNs by introducing consistent losses. This proposed method is effective in reducing the dependency of the transformation. In contrast to other typical methods, our framework improves the performance without any extra learnable parameters or extra layers. The model is also different from the technique of data augmentation or other approximate methods by simply pushing the model to learn how to map from the transformed images to the pre-defined labels. The ability for consistent representations is learned layer by layer. The experiments show that our framework has achieved state-of-the-art performance in image recognition and image retrieval tasks.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[3] J. Yu, X. Yang, F. Gao, and D. Tao, "Deep multimodal distance metric learning using click constraints for image ranking," *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4014–4024, 2017.

[4] J. Yang, X. Shen, X. Tian, H. Li, J. Huang, and X.-S. Hua, "Local convolutional neural networks for person re-identification," in *ACM Multimedia*, 2018, pp. 1074–1082.

[5] Y. Li, X. Tian, X. Shen, and D. Tao, "Classification and representation joint learning via deep networks," in *IJCAI*, 2017, pp. 2215–2221.

[6] G. Gkioxari, R. Girshick, and J. Malik, "Contextual action recognition with r* cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1080–1088.

[7] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[8] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Advances in neural information processing systems*, 2014, pp. 1799–1807.

[9] C. Hong, J. Yu, J. Wan, D. Tao, and M. Wang, "Multimodal deep autoencoder for human pose recovery," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5659–5670, 2015.

[10] C. Hong, J. Yu, D. Tao, and M. Wang, "Image-based three-dimensional human pose recovery by multiview locality-sensitive sparse retrieval," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3742–3751, 2015.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 111–118.

[13] D. A. Van Dyk and X.-L. Meng, "The art of data augmentation," *Journal of Computational and Graphical Statistics*, 2012.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[15] A. Kanazawa, A. Sharma, and D. Jacobs, "Locally scale-invariant convolutional neural networks," in *Advances in neural information processing systems*, 2014.

[16] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[17] X. Shen, X. Tian, A. He, S. Sun, and D. Tao, "Transform-invariant convolutional neural networks for image classification and search," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 1345–1354.

[18] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[20] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2161–2168.

[21] M. J. Huiskes and M. S. Lew, "The mir flickr retrieval evaluation," in *ACM International Conference on Multimedia Information Retrieval*, 2008, pp. 39–43.

[22] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[23] I. Goodfellow, H. Lee, Q. V. Le, A. Saxe, and A. Y. Ng, "Measuring invariances in deep networks," in *Advances in neural information processing systems*, 2009, pp. 646–654.

[24] L. Xie, L. Zheng, J. Wang, A. L. Yuille, and Q. Tian, "Interactive: Inter-layer activeness propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 270–279.