Marginalized CNN: Learning Deep Invariant Representations

Jian Zhao[1][2]
zhaojian90@u.nus.edu
Jianshu Li[1]
jianshu@u.nus.edu
Fang Zhao[1]
elezhf@nus.edu.sg
Shuicheng Yan[1][3]
eleyans@nus.edu.sg
Jiashi Feng[1]
elefjia@nus.edu.sg

[1] National University of Singapore
[2] National University of Defense Technology
[3] Qihoo/360 AI Institute

Abstract

Training a deep neural network usually requires sufficient annotated samples. The scarcity of supervision samples in practice thus becomes the major bottleneck on performance of the network. In this work, we propose a principled method to circumvent this difficulty through marginalizing all the possible transformations over samples, termed as **m**arginalized **C**onvolutional **N**eural **N**etwork (mCNN). mCNN *implicitly* considers *infinitely many* transformed copies of the training data in every training epoch and therefore is able to learn representations invariant for transformation in an end-to-end way. We prove that such marginalization can be understood as a classic CNN with a special form of regularization and thus is efficient for implementation and not restricted to the CNN module used. Experimental results on the MNIST and affNIST digit number datasets demonstrate that mCNN can match or outperform the original CNN with much fewer training samples. Besides, mCNN also performs well for face recognition on the recently released large-scale MS-Cele-1M dataset and outperforms state-of-the-arts. Moreover, compared with the traditional CNNs which use data augmentation to improve their performance, the computational cost of mCNN is reduced by a factor of 26.

1 Introduction

Deep learning methods, and in particular **C**onvolutional **N**eural **N**etwork (CNNs) [14], have achieved very good performance on various computer vision tasks, from generic object recognition [13, 15, 23] to object detection [3, 5, 19] and face recognition [6, 12, 21]. The performance gain roots in the multiple layered model and a large amount of available training data. The layered architecture enables the model to extract high level visual patterns for describing the visual properties of images and a large number of training data provide supervision for optimizing the huge number of inherent parameters.

CNNs are powerful for learning comprehensive image representations. However, it is notoriously known that the representation is sensitive to the input image transformations, such

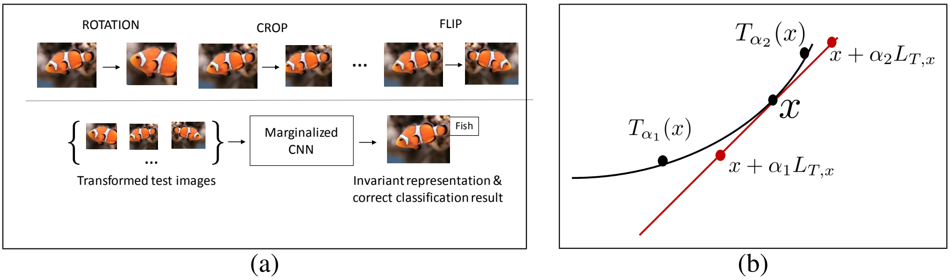


Figure 1: (a) Illustration of the motivation for learning invariant presentations. There are various image transformations (shown in the upper panel) which may change the output representations from CNN significantly and damage the classification performance. The proposed mCNN (shown in the bottom panel) can produce invariant representations and correct classification results for images with transformations. (b) Lie derivative: locally linear approximation to the image transformation manifold in the image space. Here x denotes the original image. T_{α_1} and T_{α_2} denote the transformation with two different parameters α_1 and α_2. $L_{T,x}$ denotes the derivative of the transformation *w.r.t.* the image x.

as translation, scaling, rotation, *etc*. Therefore, the performance of CNN will dramatically drop when applied to testing samples with *unseen* transformations.

To remedy this issue, a popular approach is to do data augmentation, *i.e.*, generating some synthetic samples by performing explicit transformations over the available training images. However, doing such augmentation significantly increases the size of the training set as well as the training time cost. Also, the ad hoc data augmentation cannot cover every possible transformation over the images.

In this work, we aim to develop a particular CNN model that is "robust" or explicitly "invariant" to the image transformations. The proposed model can learn invariant representations and correct classification results without explicit and expensive data augmentation. To better show the motivation of this work, we consider an example shown in Figure 1 (a). There are multiple images of the same single object with various transformations. Traditional CNN may fail in recognizing the transformed object correctly if it does not see such transformation in the training stage. Szegedy *et al.* [22] also confirm the intriguing property of CNN, which further verifies that the representation of CNN is quite fragile to specific noise or transformation on the images.

Towards training a CNN invariant to data transformation, we propose to explicitly consider minimizing the loss over the training images undergoing *all* the possible transformations (which could be infinitely many). This cannot be achieved by hand-crafted data augmentation but is possible through performing marginalization over the transformation parameters when training the CNNs. In this way, the inherent sensitiveness of the output representations *w.r.t.* the image transformations can be significantly alleviated. We call the model trained by such a new approach the ***m**arginalized **C**onvolutional **N**eural **N**etwork* (mCNN).

A similar idea was also investigated by LeCun *et al.* in their tangent vector work [20]. However, in that work, they mostly developed the regularization based on their intuition. In this work, we focus on how to obtain the transformation invariant representation from CNN with a more rigorous theoretical motivation. Moreover, we find that the transformation invariance property actually depends on the sample properties. The proposed mCNN will focus on certain samples difficult to classify, which are quite similar to the support vector samples in **S**upport **V**ector **M**achine (SVM), though they are motivated from different viewpoints.

Experimental results on the MNIST and affNIST digit number datasets demonstrate that mCNN can match or outperform the original CNN with much fewer training samples. Besides, we also conduct experiments on the recently released challenging large-scale MS-Cele-1M public dataset, demonstrating mCNN can generalize well to the face recognition task with superiority over other state-of-the-art methods. Moreover, compared with the traditional CNNs which use data augmentation to improve their performance, the computational cost of our scheme is only 3.85%.

2 Related Work

Recently, deep learning algorithms with invariant features have emerged as promising tools for classification and related applications [1, 2, 8, 9, 16, 18]. Here the term "invariant" is used in a loose manner: a learning system is invariant if, given an object, the predicted label remains unchanged for all possible variations of images of the class. Learning deep invariant representations is a challenging issue for deep learning and classification tasks, through which a more complex and intelligent network can be constructed with the robustness against variability in the high-dimensional input data.

There are two major types of deep invariant representation learning methods: (1) data augmentation, *i.e.*, applying finite predefined transformations on the original image dataset and then training the whole network using these augmented data; (2) embedded invariant feature representations, *i.e.*, adding transformations and their regulations directly on the internal equations of the network.

Data augmentation augments the training dataset with transformed versions of the original images. This is a well-known approach which works directly at the image level to enforce robustness of a learning system to variations of the input. In contrast to embedded invariant feature representations, this strategy is more intuitive and easier to implement. To develop a more adaptive data augmentation scheme, Image Transformation Pursuit (ITP) was proposed in [17] which selects a set of transformations from a pre-defined set through optimizing the training loss. However, given a large set of possible transformations, selecting a compact subset is still challenging and computationally expensive. In addition, not all the transformations are equally informative and adding uninformative transformations increases training time without gain in performance. Therefore, defining the proper transformation set itself is also challenging.

Embedded invariant feature representation methods try to learn invariant feature representations by embedding the invariance into structures of the learning system. For instance, Simard *et al.* [20] proposed a tangent vector network, whose development is purely based on their intuition. Szegedy *et al.* discussed the intriguing properties of neural networks in [22], and found that some subtle corruptions of the images may manipulate the output significantly. Thus they propose to harness adversarial samples to improve performance of a CNN. Our method attempts to alleviate such an undesired intriguing property through the extra invariance regularization.

It is worth mentioning that most previous works only perform transformations on the training dataset or enforce a particular regularizing function based on an ambiguous intuition. Our approach can obtain the transformation invariant representations and provide correct classification results without explicitly doing data augmentation.

3 Formulation of mCNN Loss

In this section we elaborate on the proposed ***marginalized CNN*** (mCNN). We start with a local approximation to linear transformation and then explain how to marginalize the joint distribution of classification loss and transformations. Finally we discuss the architecture of the mCNN.

3.1 Approximation to Local Transformation

Throughout the paper, we use T to denote the (affine) transformation imposed on a sample x. A transformation is specified by its parameter α, and we use $T_\alpha(x)$ to denote the mapped sample after transformation. A transformation of magnitude ε can be approximated locally by its Lie derivatives as $T_\alpha(x) = x + \varepsilon L_{T,x}$ [20], where we endow a normal distribution with the magnitude $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. $L_{T,x}$ is also known as the tangent vector [20], which can be computed in advance. Figure 1 (b) shows the first order approximation to the tangent plane of manifold $T_\alpha(x)$ at x, where

$$L_{T,x} = \left.\frac{\partial T_\alpha(x)}{\partial \alpha}\right|_{\alpha=0} = \left[\frac{\partial T_\alpha(x)}{\partial \alpha_1}, \ldots, \frac{\partial T_\alpha(x)}{\partial \alpha_m}\right]_{\alpha=0}.$$

3.2 Loss Function

In this work, the provided n training images are collectively denoted as $X = \{x_1, \ldots, x_n\}$. We define a family of transformations $\{T_\alpha | \alpha \in \mathcal{A}\}$ parameterized by vectors $\alpha \in \mathcal{A}$, where $\mathcal{A}$ is the set of all possible parameter vectors. The details of the considered transformations are provided in Section 4. Suppose we virtually impose m different transformations on the images, denoted as $T_{\alpha_1}, \ldots, T_{\alpha_m}$. The classification loss to minimize over the whole dataset is then

$$\mathcal{L}(x) = \frac{1}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} \ell(y_i, g_w(T_{\alpha_j}(x_i))). \tag{1}$$

Here the function $\ell(\cdot,\cdot)$ accounts for the loss caused by misclassification. Let $g_w(\cdot)$ denote the CNN parameterized by w.

Let $z_i = g_w(T_\alpha(x_i))$ be the learned representation of sample x_i, and assume α is from a uniform distribution over the set $\mathcal{A}$. Then according to the law of large number, taking $m \to \infty$, the resulted loss function becomes the expectation *w.r.t.* the parameter α:

$$\lim_{m\to\infty} \frac{1}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} \ell(y_i, z_i) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_\alpha \ell(y_i, z_i). \tag{2}$$

Hence, the loss function to minimize *w.r.t.* the CNN (parameterized by w) becomes

$$\mathcal{L}(X) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_\alpha \left[\ell(y_i, z_i)\right]. \tag{3}$$

Regarding the loss function, a common choice in practice is the multinomial negative log likelihood of the network output:

$$\ell(y_i, z_i) = -\langle y_i, \log h(z_i) \rangle, \tag{4}$$

where z_i usually goes through a softmax function $h(\cdot)$ for normalization purpose. In particular, the function $h(\cdot)$ is defines as

$$h(z) \triangleq \frac{\exp(z)}{\|\exp(z)\|_1}. \tag{5}$$

Note that z can be a vector and $\exp(\cdot)$ operates in a element-wise manner. Therefore, the loss function becomes

$$\begin{aligned}\mathcal{L}(X) &= \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_\alpha \left[-\langle y_i, z_i \rangle\right] + \mathbb{E}_\alpha \log \|\exp(z_i)\|_1 \\ &= \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_\alpha \left[-\langle y_i, z_i \rangle\right] + \log \|\exp\{\mathbb{E}_\alpha z_i\}\|_1 + \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_\alpha \log \|\exp(z_i)\|_1 - \log \|\exp\{\mathbb{E}_\alpha z_i\}\|_1.\end{aligned}$$

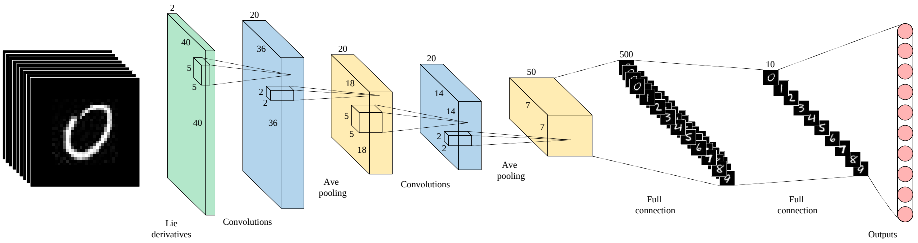


Figure 2: Illustration of the architecture of our proposed mCNN for learning deep invariant representations with classification task on MNIST and AffNIST digit number datasets as an example. Here the green box represents the Lie derivative layer, blue boxes represent the convolutional layers, yellow boxes represent the ave-pooling layers, and red circles represent the network outputs. Best viewed in color.

First order approximation to $g_w(\cdot|\alpha)$ If the activation function $g_w(\cdot|\alpha)$ is locally linear w.r.t. α for a small value of α, we can approximate $g_w(\cdot|\alpha)$ at the point of $\alpha = 0$ by its first order Taylor expansion (recall $z_i = g_w(x_i)$):

$$z_i \approx g_w(x_i) + \nabla_\alpha z_i|_{\alpha=0}\alpha.$$

Therefore,

$$\mathbb{E}_\alpha z_i \approx g_w(x_i) + \nabla_\alpha z_h|_{\alpha=0}\mathbb{E}\alpha \overset{(A1)}{=} g_w(x_i).$$

Then, the loss function can be written as

$$\mathcal{L}(X) = \frac{1}{n}\sum_{i=1}^{n} -\langle y_i, z_i\rangle + \log\|\exp(z_i)\|_1 + \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_\alpha \log\|\exp\{z_i\}\|_1 - \log\|\exp(z_i)\|_1.$$

It can be seen that the first two terms correspond to the loss function for the non-transformed images, and the rest two terms form a regularization on the CNN to augment its invariance. The expectation is hard to optimize directly. We adopt the following approximations to simplify the regularization term.

We approximate the loss $\mathbb{E}_\alpha \log\|\exp\{z_i\}\|_1$ by its second order Taylor expansion at $\alpha = 0$. Here we assume $\mathbb{E}\alpha = 0$ which is a mild requirement on the distribution of α. Then the first order term in the expansion term is zero.

$$\begin{aligned}\mathbb{E}_\alpha \log\|\exp\{z_i\}\|_1 &\approx \log\|\exp\{g_w(T_0(x_i))\}\|_1 + \frac{1}{2}\mathrm{Tr}\left[\Sigma_\alpha \nabla^2_{\alpha=0}\log\|\exp\{z_i\}\|_1\right] \\ &= \log\|\exp\{z_i\}\|_1 + \frac{1}{2}\sigma_\alpha^2 \mathrm{Tr}\left[\nabla^2_{\alpha=0}\log\|\exp\{z_i\}\|_1\right],\end{aligned}$$

where Σ_α is the covariance matrix of the parameter α: $\Sigma_\alpha = \mathbb{E}\alpha\alpha^\top$. We assume that the element in the α, *i.e.*, each type of transformation, is independent. Thus the covariance matrix is a diagonal one. If we further assume that the variance of each element is equal to σ_α^2, then we can pull the covariance matrix out of the trace in the second order term as σ_α^2.

Thus the regularization term becomes

$$\mathcal{R} = \frac{1}{2}\sigma_\alpha^2 \mathrm{Tr}\left[\nabla^2_{\alpha=0}\log\|\exp\{z_i\}\|_1\right]. \tag{6}$$

Suppose the number of transformation types is K, thus the dimension of the vector α is also K. For the computation efficiency, we calculate the above Hessian by only considering the diagonal elements:

$$\mathrm{Tr}\left[\nabla^2_{\alpha=0}\log\|\exp\{z_i\}\|_1\right] = \sum_{k=1}^{K}\left(\frac{\partial z_i}{\partial \alpha_k}\right)^\top \frac{\partial^2 \ell}{\partial z_i^2}\frac{\partial z_i}{\partial \alpha_k} \approx \sum_{k=1}^{K}\sum_{c=1}^{C}\frac{\partial^2 \ell}{\partial z_c^2}\left(\frac{\partial z_c}{\partial \alpha_k}\right)^2\Bigg|_{\alpha_k=0},$$

where $\ell = \log \|\exp\{z_c\}\|_1$. C is the number of channels at the final layer, which in this work is equal to the number of classes. z_c denote the c-th element in z_i. Several simple calculations give

$$\frac{\partial^2 \ell}{\partial z_c^2} = \frac{\exp(z_c)}{\|\exp(z_i)\|_1}\left(1 - \frac{\exp(z_c)}{\|\exp(z_i)\|_1}\right).$$

Thus, the final form of the regularization is

$$\mathcal{R} = \sum_{k=1}^{K}\sum_{c=1}^{C} \frac{\exp(z_c)}{\|\exp(z_i)\|_1}\left(1 - \frac{\exp(z_c)}{\|\exp(z_i)\|_1}\right)\left(\frac{\partial z_c}{\partial \alpha_k}\right)^2\Bigg|_{\alpha_k=0}.$$

Note that the $\frac{\exp(z_c)}{\|\exp(z)\|_1}$ is the score for the c-th class after a softmax function. When $\frac{\exp(z_c)}{\|\exp(z)\|_1} = \frac{1}{2}$, the weight achieves the maximal regularization. In other words, the regularization term highlights the samples, which gives an equal probability of belonging or not belonging to the c-th class. The second term in the regularization is a measure of the sensitiveness of the image representation *w.r.t.* the transformation. If there is ambiguity in the belongingness of the image to a category, the sensitiveness to the transformation is enforced to be small.

Now, we proceed to calculate the gradient of the generated representation *w.r.t.* the transformation parameter:

$$\frac{\partial z_c}{\partial \alpha_k}\Bigg|_{\alpha=0} = \frac{\partial z_c}{\partial T_\alpha(x)}\frac{\partial T_\alpha(x)}{\partial \alpha_k}\Bigg|_{\alpha=0} = \nabla_x z \cdot \frac{\partial T_\alpha(x)}{\partial \alpha_k}\Bigg|_{\alpha=0}.$$

Note that $T_\alpha(x) = x$ if $\alpha = 0$. Here $\nabla_x z$ is the Jacobian matrix of $g_w(x)$ for the image x, and $\partial T_\alpha(x)/\partial \alpha_k$ is the tangent vector associated with transformation T_α.

The theory of Lie algebra [4] ensures that the composition of local (small) transformations corresponds to linear combinations of the corresponding tangent vectors. Therefore, if the CNN is successfully trained to be locally invariant *w.r.t.* several transformations, it will also be invariant *w.r.t.* their composition.

The final objective function is

$$\mathcal{L}(X) = \frac{1}{n}\left\{\sum_{i=1}^{n} -\langle y_i, z_i\rangle + \log\|\exp\{z_i\}\|_1 + \frac{\lambda\sigma_\alpha^2}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\frac{\partial^2 \ell}{\partial z_c^2}\left(\frac{\partial z_c}{\partial \alpha_k}\right)^2\Bigg|_{\alpha_k=0}\right\}, \tag{7}$$

where λ is a trade-off coefficient to control the effect of the regularization. It is feasible to devise an efficient algorithm for performing the filter update, which is analogous to ordinary Back Propagation (BP). However, in addition to propagating neuron activations, it also propagates the tangent vectors. This is a unique feature of mCNN.

3.3 Network Architecture

The architecture of mCNN is depicted in Figure 2. Considering the classification scenario on the MNIST and AffNIST digit number datasets, the network contains seven layers with weights: the first layer is the Lie derivative layer, the second and fourth layers are convolutional layers, the third and fifth layers are ave-pooling layers, and the remaining two are fully-connected layers. The **Re**ctified **L**inear **U**nit (ReLU) non-linearity is applied to the output of the first fully-connected layer. The output of the last fully-connected layer is fed to a 10-way softmax which produces a distribution of the 10 class labels.

The Lie derivative layer carries out Lie derivatives upon the input images to approximate local transformation. The first convolutional layer filters the $40 \cdot 40 \cdot 2$ input image from the first layer with 20 kernels of size $5 \cdot 5 \cdot 2$ with a stride of 1 pixel (this is the distance between the receptive field centers of neighboring neurons in a kernel map). The first ave-pooling layer has 20 kernels of size $2 \cdot 2 \cdot 20$ with a stride of 2 pixels connected to the outputs of the first convolutional layer. The second convolutional layer has 20 kernels of size $5 \cdot 5 \cdot 20$ connected to the outputs of the first ave-pooling layer. The second ave-pooling layer has 50 kernels of size $2 \cdot 2 \cdot 20$ with a stride of 2 pixels connected to the outputs of the second convolutional layer. The fully-connected layers have 500 and 10 neurons, respectively.

Note that our approach is not restricted to the CNN module used, and can be generalized to other state-of-the-art architectures (*i.e.* ResNet [7], ResNext [24], DenseNet [10], *etc.*) for performance boosting.

4 Experiments

4.1 Experimental Settings

Datasets We use MNIST[1] and affNIST[2] digit number datasets to evaluate the performance of our proposed mCNN. Besides, the recently released challenging large-scale MS-Celeb-1M[3] public dataset is used to test the generalization capability of mCNN for face recognition.

MNIST and affNIST Datasets. The original MNIST handwritten digit dataset contains 60K training samples and 10K testing samples. The digits have been size-normalized and centered in a fixed-size image. affNIST is made by taking images from MNIST and applying various reasonable affine transformations to the images. In the process, the images are resized to 40×40 pixels large, with significant transformations involved, so much of the challenge for the models is to learn that a digit means the same thing in the upper right corner as it does in the lower left corner. Some example images in the affNIST dataset are shown in Figure 3 (a). The affNIST testing data are created by transforming the 10K original MNIST testing data; the affNIST training data come from 50K MNIST training data; the affNIST validation data come from the remaining 10K MNIST training data. The evaluation system measures the classification Top-1 accuracy.

MS-Celeb-1M Dataset. MS-Celeb-1M is a large-scale real-world face dataset, along with the protocol for evaluation of classification. The training set contains about 10M web images from the top 76,674 entities sampled from the 1M celebrity list in terms of their popularities, with approximately 100 images for each celebrity. The noise in the training data has not been manually removed to allow data cleaning, noisy label removal, and learning with noisy data proposed by the possible solutions. The validation set includes two tracks, Dev1 (Hard set) and Dev2 (Random set), each of which contains 500 images. The face images in the Hard set have large variations, while those in Random set are randomly selected and subjects therein highly likely to be covered by the training data. The evaluation system measures the recognition recall at a given precision 95%.

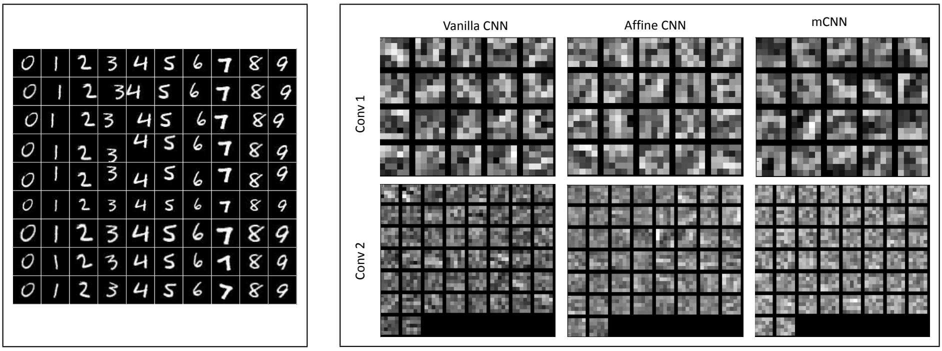
Implementation Details The proposed mCNN is implemented based on the publicly available Caffe platform [11], which is trained on a single NVIDIA GeForce GTX TITAN X GPU with 12G memory. The weights of all convolutional layers are initialized by normal distribution with an std of 0.001. During training, the learning rate is initialized to 0.01. We train our model using **S**tochastic **G**radient **D**escent (SGD) algorithm with a batch size of 128, momentum of 0.9, and weight decay of 0.0005.

For digit classification, we use a Vanilla CNN trained on MNIST and an Affine CNN trained on the affNIST dataset with data augmentation corresponding to the specific trans-

[1] http://yann.lecun.com/exdb/mnist/.

[2] http://www.cs.toronto.edu/ tijmen/affNIST/.

[3] https://www.microsoft.com/en-us/research/project/ms-celeb-1m-challenge-recognizing-one-million-celebrities-real-world/.



(a) (b)

Figure 3: (a) Example images in the affNIST dataset: top row shows the original centered and non-transformed digits; row 2-3 show the x-axis translated digits, with the tangent vectors: $\{\pm10, \pm7.5, \pm5, \pm2.5\}/\{\pm5\}$; row 4-5 show the y-axis translated digits, with the tangent vectors: $\{\pm10, \pm7.5, \pm5, \pm2.5\}/\{\pm5\}$; row 6-7 show the scaled digits, with the multiplication factors: $\{0.8\}/\{1.2\}$; bottom 2 rows show the rotated digits, with the angles: $\{5, 10, 15, 20\}/\{-5, -10, -15, -20\}$. (b) Visualization of the filters from two convolutional layers, (top) conv1 layer and (bottom) conv2 layer, in three CNNs: (left) Vanilla CNN trained on MNIST data, (middle) Affine CNN trained on affNIST data, and (right) mCNN trained on MNIST data.

formation as two baselines. Our proposed mCNN is trained on MNIST dataset for roughly 40 epochs and it takes 1.5-2 hours for the final model. The performance with 7 different network settings (*i.e.* λ ranging from 0.0 to 10.0 discretely) will be compared in the following experiments. We test our models on affNIST with two complementary sub-experimental settings where the elementary transformations of translation, scaling, rotation, and the corresponding combination of all the elementary transformations are applied respectively. Specifically, in sub-experimental setting I (transformations with small variance)/II (transformations with large variance), for translation, two different local transformations T_α are applied by horizontal (x-axis) translation and vertical (y-axis) translation with distances in $\{-5, 5\}/\{\pm10, \pm7.5, \pm5, \pm2.5\}$; for scaling, the local transformations T_α are conducted by multiplying the patch scale with the factors $\{0.9, 1.1\}/\{0.8, 1.2\}$; for rotation, the local transformations T_α are conducted by rotating the image with the angles $\{-5, 5\}/\{\pm5, \pm10, \pm15, \pm20\}$; for combination, all of the above mentioned elementary local transformations are ensembled.

For face recognition, for fair comparison, we use the modern architecture ResNet-101 [7] as the backbone CNN module of mCNN. Since the provided training dataset is crawled from the Internet without manually checking, there are a certain percentage of data which are actually noisy. For example, some of the faces are given with false labels and some images even do not contain faces. Thus, we propose a two-stage method to learn robust and disambiguated deep facial representations for effectively classifying celebrity faces at large scale. The first stage is to find outliers from each celebrity to clean the noisy data in the provided training set. We implement this by pretraining mCNN on the CASIA-WebFace dataset with a final 10,575-way classification layer and a softmax loss for 100 epochs. Then, we use the pre-trained mCNN model to extract deep features from all the aligned face images in MS-Celeb-1M. For all deep features corresponding to one celebrity, we calculate the median of all these deep features as the cluster centroid and Euclidean distance of each deep feature to the centroid. Based on the distances, we remove 12% of the data as outliers. Built on the cleaned dataset, the second stage is the task-specific robust and disambiguated deep feature

Method	Top-1 acc @ x-axis Translation I/II (%)	Top-1 acc @ y-axis Translation I/II (%)	Top-1 acc @ Scaling I/II (%)	Top-1 acc @ Rotation I/II (%)	Top-1 acc @ Combination I/II (%)
Vanilla CNN	46.32/42.18	26.28/22.62	90.25/85.42	94.40/89.98	36.68/32.52
Affine CNN	88.93/83.72	67.29/63.51	97.95/93.53	98.11/94.06	86.79/82.11
mCNN$_{\lambda=0.00}$	46.30/42.06	26.26/22.60	90.26/85.43	94.38/89.86	36.71/32.49
mCNN$_{\lambda=0.01}$	50.94/46.85	31.05/27.66	92.31/88.67	94.91/90.02	41.05/37.43
mCNN$_{\lambda=0.10}$	59.37/53.29	40.13/36.58	95.19/90.02	95.97/92.35	52.68/48.76
mCNN$_{\lambda=0.50}$	71.41/66.01	54.25/49.12	96.22/91.96	97.16/92.88	63.23/60.85
mCNN$_{\lambda=1.00}$	83.12/77.32	**66.93**/58.86	97.51/**93.10**	97.52/93.10	75.19/72.96
mCNN$_{\lambda=5.00}$	**87.89/82.87**	66.75/**62.97**	**97.82**/92.79	**98.02/93.96**	**86.11/81.95**
mCNN$_{\lambda=10.00}$	86.54/81.95	64.92/61.22	95.23/90.18	96.43/91.34	84.23/81.04

Table 1: Comparison of our mCNN digit classification performance with two baseline methods when evaluating on sub-experimental setting I/II. We also report results of the proposed mCNN with 7 different network settings in terms of the regularization factor λ. Our best results are given in bold.

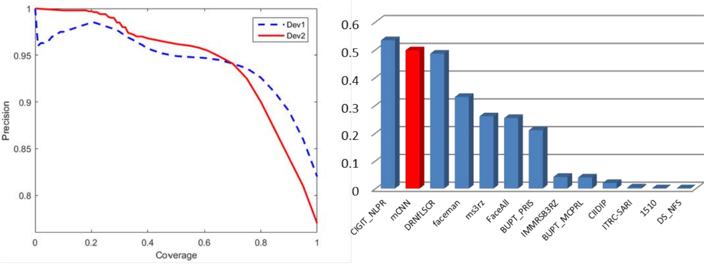
learning. We implement this by further fine-tuning the pre-trained network ended with a new 76,674-way classification layer and a softmax loss for 100 epochs. The whole process takes 11-12 days for the final specific model. As the testing data are not provided, we use the validation set instead for evaluation.

4.2 Results and Comparisons

Results on MNIST and affNIST: As shown in Table 1, as λ increases, the classification accuracy of mCNN improves consistently, which demonstrates that the proposed regularization can effectively enhance the robustness of our model to data transformation. However, starting from $\lambda = 5.00$, further increasing λ does not bring performance improvement any more. We even observe performance drop by further increasing λ. The reason is that a large value of λ will enforce the learned filters to be all zero ones, which always produce invariant performance without any discriminative information. Thus, in this work, we set $\lambda = 5.00$ for trade-off. mCNN$_{\lambda=5.00}$ outperforms the Vanilla CNN with significant gains for the classification task on MNIST and affNIST datasets. mCNN$_{\lambda=5.00}$ can even match the Affine CNN with the specific data augmentation for both the sub-experimental setting I with small variance and the sub-experimental setting II with large variance. This superior performance indicates that by minimizing the mCNN loss over all possible transformations on training images through doing marginalization over the transformation parameters, the sensitiveness of the output representations *w.r.t.* the image transformations is effectively decreased. Moreover, for the combination scenarios, our proposed mCNN can achieve the performance comparable to doing 26 different transformations on each training image, at the computational cost of only 3.85% (*i.e.* $1/26$) of that used by the augmentation. Besides, the training and testing phases of mCNN are all conducted for the classification task in an end-to-end way, thus the deep feature learning in our work is more efficient than previous methods. We also visualize the learned filters in three different CNNs in Figure 3 (b). It is interesting that similar to Affine CNN, mCNN also learns more diverse filter kernels than Vanilla CNN, which can help handle various transformations.

Results on MS-Celeb-1M: We further verify the generalization capacity of the proposed mCNN on the two tracks Dev1 (Hard set)/Dev2 (Random set) of MS-Celeb-1M validation set. As illustrated in Figure 4 and Table 2, on Random set the proposed mCNN reaches the Coverage 65.4% when Precision=95%, and on Hard set the proposed mCNN reaches the Coverage 49.8% when Precision=95%.

Generally, mCNN shows higher performance than other recent state-of-the-art methods in terms of Coverage at Precision=95% on MS-Celeb-1M Hard set. This demonstrates that the proposed mCNN can be generalized well to other computer vision tasks, such as face recognition. Note that we only utilize a single model here for evaluation. We believe that the performance of our model can be further improved with more ensembled models specified



(a) **(b)**

Figure 4: Results on the MS-Celeb-1M dataset. (a) The Precision-Coverage curve of mCNN on the two tracks of validation set. (b) The evaluation performance of our mCNN (highlighted in red) in MS-Celeb-1M benchmark (Dev1 of the validation set) in terms of Coverage at Precision=95%. Best viewed in color.

Method	**Coverage @ P = 0.95% Dev1/Dev2**	**Coverage @ P = 0.99% Dev1/Dev2**
NII-UIT-KAORI*	-/0.001	-/0.001
BUPT_MCPRL*	0.040/0.064	0.007/0.006
CIIDIP*	0.020/0.154	0.018/0.025
IMMRSB3RZ*	0.042/0.171	0.039/0.104
BUPT_PRIS*	0.210/0.421	0.117/0.216
faceman*	0.330/0.461	**0.211**/0.339
FaceAll*	0.254/0.554	0.142/**0.417**
1510*	0.001/0.570	0.001/0.065
CIGIT_NLPR*	**0.534/0.684**	0.026/0.045
mCNN	0.498/0.654	0.136/0.316

(* indicates corresponding result is reported by MS-Celeb-1M leaderboard[4])

Table 2: Performance comparison of mCNN with state-of-the-arts on the two tracks Dev1 (Hard set)/Dev2 (Random set) of large-scale MS-Celeb-1M face recognition. Symbol “-" implies that the result is not reported for that method. A large number means better performance. The best performance is highlighted in bold.

with different loss functions, and we would like to examine this in the future.

5 Conclusion

We proposed a **m**arginalized **C**onvolutional **N**eural **N**etwork (mCNN) that learns transformation invariant representations and thus performs better on testing data with unseen transformation. The mCNN minimizes the empirical loss over all possible transformations on the training images through doing marginalization over the transformation parameters. Experimental results on the MNIST and affNIST datasets demonstrate that mCNN can match or outperform the original CNN using much fewer training samples. Its good generalization capability to other tasks (*e.g.* face recognition) is also verified by experiments on the large-scale MS-Celeb-1M public dataset. Moreover, compared with the traditional CNNs which use data augmentation to improve performance, the computational cost of our scheme is only 3.85%.

Acknowledgement

The work of Jian Zhao was partially supported by **C**hina **S**cholarship **C**ouncil (CSC) grant 201503170248.

The work of Jiashi Feng was partially supported by National University of Singapore startup grant R-263-000-C08-133 and Ministry of Education of Singapore AcRF Tier One grant R-263-000-C21-112.

References

[1] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[2] Jun-Cheng Chen, Vishal M Patel, and Rama Chellappa. Unconstrained face verification using deep cnn features. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016.

[3] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2154, 2014.

[4] Robert Gilmore. *Lie groups, Lie algebras, and some of their applications*. Courier Dover Publications, 2012.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013.

[6] Tal Hassner, Iacopo Masi, Jungyeon Kim, Jongmoo Choi, Shai Harel, Prem Natarajan, and Gerard Medioni. Pooling faces: template based face recognition with pooled face images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 59–67, 2016.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[8] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[9] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[10] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[12] Fang Zhao Hao Liu Jing Li Shengmei Shen Jiashi Feng Jianshu Li, Jian Zhao and Terence Sim. Robust face recognition with deep multi-view representation learning. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 1068–1072. ACM, 2016.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[15] Vinod Nair and Geoffrey E Hinton. 3d object recognition with deep belief nets. In *Advances in Neural Information Processing Systems*, pages 1339–1347, 2009.

[16] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, volume 1, page 6, 2015.

[17] Mattis Paulin, Jerome Revaud, Zaid Harchaoui, Florent Perronnin, Cordelia Schmid, et al. Transformation pursuit for image classification. In *CVPR 2014-IEEE Conference on Computer Vision & Pattern Recognition (2014)*, 2014.

[18] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2006.

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[20] Patrice Y Simard, Yann A LeCun, John S Denker, and Bernard Victorri. Transformation invariance in pattern recognitionâĂŤtangent distance and tangent propagation. In *Neural networks: tricks of the trade*, pages 239–274. Springer, 1998.

[21] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Hybrid deep learning for face verification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1489–1496, 2013.

[22] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[23] Yida Wang and Weihong Deng. Self-restraint object recognition by model based cnn learning. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 654–658. IEEE, 2016.

[24] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.