Mining the Displacement of Max-pooling for Text Recognition

Yuchen Zheng[a,*], Brian Kenji Iwana[a], Seiichi Uchida[a]

[a]*Department of Advanced Information Technology, Kyushu University, Fukuoka, Japan*

Abstract

The max-pooling operation in convolutional neural networks (CNNs) downsamples the feature maps of convolutional layers. However, in doing so, it loses some spatial information. In this paper, we extract a novel feature from pooling layers, called displacement features, and combine them with the features resulting from max-pooling to capture the structural deformations for text recognition tasks. The displacement features record the location of the maximal value in a max-pooling operation. Furthermore, we analyze and mine the class-wise trends of the displacement features. The extensive experimental results and discussions demonstrate that the proposed displacement features can improve the performance of the CNN based architectures and tackle the issues with the structural deformations of max-pooling in the text recognition tasks.

Keywords: Convolutional neural networks, Max-pooling, Displacement feature, Text recognition

1. Introduction

In recent years, convolutional neural networks (CNNs) have had excellent performance in the field of document analysis and recognition. The max-pooling operation in modern CNNs combines the maximal responses of the feature maps

*Corresponding author

Email addresses: yuchen@human.ait.kyushu-u.ac.jp (Yuchen Zheng), brian@human.ait.kyushu-u.ac.jp (Brian Kenji Iwana), uchida@kyushu-u.ac.jp (Seiichi Uchida)

into a summarized joint distribution of the features over some region of interest [1]. The objective of max-pooling in traditional CNNs is to reduce the size of the parameter space by removing redundant information while preserving the relevant responses from the convolutional feature maps. In other words, the role of pooling layers is to merge semantically similar features into one [2]. The introduction of max-pooling layers has helped CNNs be somewhat spatially invariant to the position of features [3]. Many popular architectures with max-pooling layers have been proposed in recent years, such as Alexnet [4], GoogleNet [5], Fast-RCNN [6], ResNet [7], and so on, which demonstrates the effectiveness of max-pooling operation for building deep neural networks.

However, max-pooling only preserves the maximal value and discards information about the position of maximal value which may introduce problems, such as the coordinate transform problem [8] and losing spatial information [9]. Due to the max-pooling operation, the proceeding layers only have knowledge of the useful features but do not know where they came from. Therefore, in order to solve problems of the traditional max-pooling operation, it is better to know the maximum and how the maximum moves to the center point simultaneously.

The motivation of this paper is to improve the max-pooling operation with position information obtained from the max-pooling operation. In the first max-pooling layer of a CNN, we first extract the maximums (pooling features) and their positions in the pooling windows. In the next step, we transform the position information into the displacement features which describes the distance of the selected maximal value from the center point in the pooling windows. Fig. 1 shows how to extract the displacement features from a pooling window. Here, the arrows with different colors represent the displacements of the center of the pooling windows. We can use a two dimensional vector to represent the displacement features in horizontal and vertical directions. Fig. 2 presents the pooling features and displacement features of samples from the HASY dataset [10] based on a Hue-Saturation-Value (HSV) color model whose color and intensity denote the direction and average length of displacement feature.

To improve the performance of the max-pooling, we explore two approaches

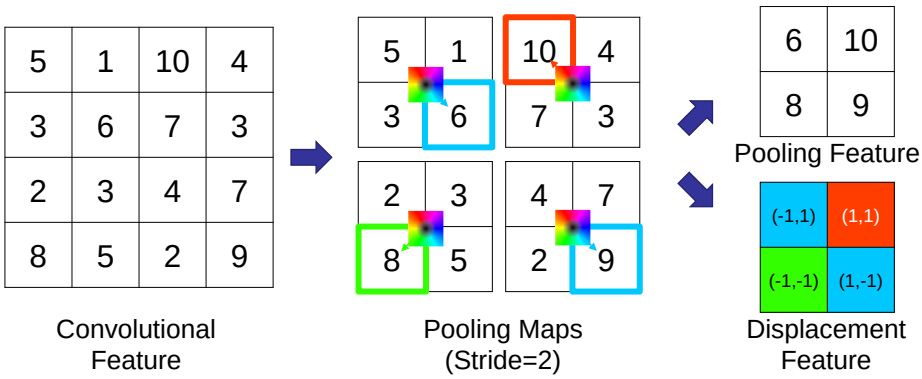

Figure 1: Extracting displacement features from a 2×2 pooling window.

to combine the pooling features and the displacement features for text recognition tasks. The first one is to use a CNN without the first convolutional layer to address the displacement features and combine the pooling features and the displacement features in the fully connected layers. The second one is to transform the displacement features into cosine features based on Principal Component Analysis (PCA). Then, we combine the cosine features with the pooling features in the fully connected layers for classification tasks.

Furthermore, in order to discover the class-wise trends of the max-pooling operation, we analyze the behaviors of the displacement features. First, we visualize the displacement features based on an HSV color model to observe the behaviors of the displacement features from the same and different categories. Next, we summarize the displacement features on all feature maps in each class, and illustrate the cumulative histograms to understand the distribution of the displacement features. Then, we introduce PCA to further analyze the properties of the displacement features and compare the similarity of class-wise subspaces spanned by the first N largest principal components between different categories. Finally, we compare the confusion matrices that are obtained by the recognition only using the pooling features and the proposed method to observe which samples are improved or degraded by combining the displacement features. The contributions of this paper are summarized as follows.

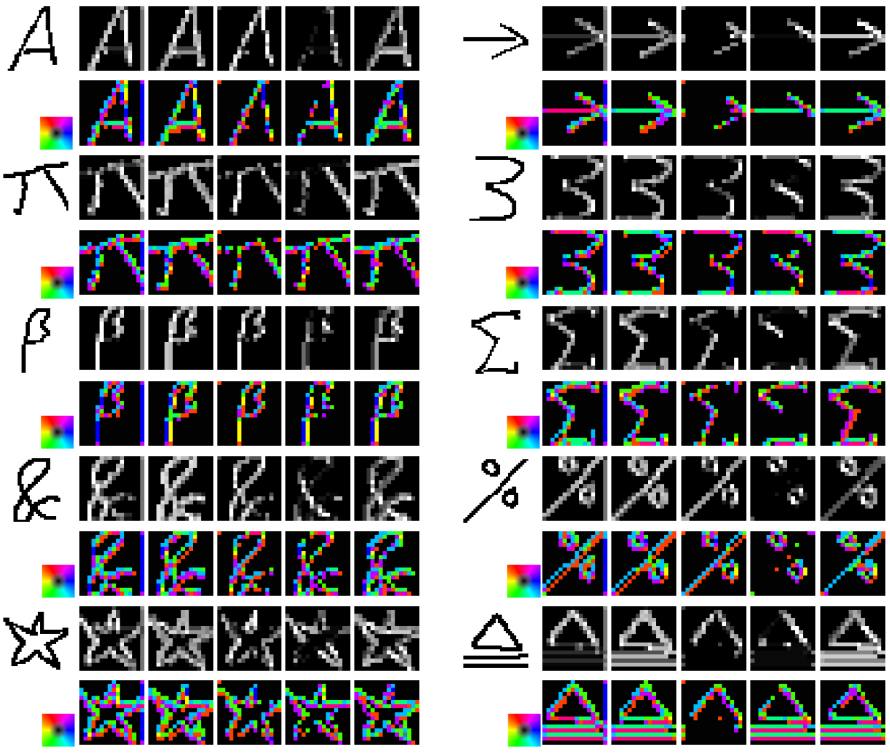

Figure 2: Visualization of the pooling features and the displacement features on the HASY dataset [10]. Here, the pooling size is 2×2, the first row represents the original image and corresponding pooling features, the second row represents the displacement features, and each column represents one convolutional filter. The visualization of displacement features is based an HSV color model whose color and intensity denote the direction and average length of displacement feature.

- We extract and represent the displacement features from pooling windows and explore how to use the position information.

- This paper explores methods of combining the pooling features and displacement features for text recognition tasks. We show that the proposed method achieves state-of-the-art results on MNIST, HASY, and Chars74k-font datasets.

- This paper analyzes and mines the class-wise trends of max-pooling operation, which includes analysis of the visualizations and distribution of the displacement features, and comparison of the similarity matrix and confusion matrix on displacement features. Through the analysis, we find that the displacement features could capture some structural deformations which are useful for some specific tasks.

This work is an extension of [11]. In the previous work, we only introduce the concept of the displacement features and analyze the class-wise behaviors of the displacement features on the MNIST dataset. Compared to the previous work, first, this paper gives a detailed introduction of how to extract the displacement features from the max-pooling operation. Then, we introduce the proposed method of fusing the displacement features with the pooling features and transforming the displacement features into cosine features. In addition, we conduct many experiments on the MNIST, HASY, and Chars74k-font datasets to evaluate the proposed method and achieve the state-of-the-art results on these datasets.

The rest of this paper is organized as follows. In Section 2, we give an overview of related work. In Section 3, we present how to extract the displacement features from pooling layers. In Section 4, the proposed method of combining the pooling features and displacement features is described in detail. The experimental settings and results are reported in Section 5. Section 6 concludes this paper with remarks and future work.

2. Related Work

Recently, many robust models have been proposed based on the max-pooling operation for text recognition tasks [12, 13, 14, 15], image classification tasks [16, 17, 18] and other fields of computer vision [19, 20, 21, 22, 23, 24]. In [25], Er et al. proposed an attention pooling-based CNN for sentence modelling. In [26], Gong et al. presented a simple but effective scheme called multiscale orderless pooling (MOP-CNN) based on Vectors of Locally Aggregated Descriptor (VLAD) encoding [27]. The orderless nature of VLAD helps to build a more invariant representation. Hinton et al. presented a multi-layer capsule system, which replaces the scalar-output feature detectors of CNNs with vector-output capsules and max-pooling with routing-by-agreement [28, 29]. In addition, DeepFlow [30], DeepMatching [31] and EpicFlow [32] also improve the pooling operation in many ways.

In the field of document analysis and recognition, compared to many traditional feature learning techniques [33, 34, 35], CNN based models have shown a powerful performance in different challenging tasks, such as handwriting recognition [36, 37, 38], scene text localization and detection [39, 40, 41, 42], script identification [43, 44, 45, 46], etc. In [47], Wu et al. applied CNN shape models to over-segmentation, geometric context modeling and character recognition. Then, they integrated this model with neural network LMs (NNLMs) and obtained the state-of-the-art performance on handwritten Chinese text recognition task. In [48], Liu et al. proposed a real-time scene text recognition method, called "SqueezedText", which combined a binary convolutional encoder-decoder neural network (B-CEDNet) and a bidirectional recurrent neural network (Bi-RNN). The 2×2 max-pooling is applied in binary convolutional encoder to obtain the strongest activation which will be binarized by the binarized activations. However, in these works, the position information of max-pooling is not combined with the pooling features to further improve the max-pooling operation, they just change some strategies in the pooling layers.

A few studies also focus on the position information and especially the dis-

placement information in pooling operation. Qian et al. proposed max-pooling positions (MPPs) as an effective discriminative feature to predict category labels for traffic sign recognition [49]. For example, in a 2×2 pooling window, they defined a quaternary encoding, '1000, 0100, 0010, 0001', where the '1' represents the position of maximal value in pooling windows. Through experiments, they found that MPPs demonstrate the ideal characteristics of small inter-class variance and large intra-class variance. Compared to our proposed displacement features, MPPs are kind of a variation. In [50], Zhao et al. proposed a novel architecture, the stacked what-where auto-encoders (SWWAE) which represents the position of maximum as the displacement features in a pooling window. The position information in this paper is similar with our work. However, it also uses this information just for an "unpooling" operation, but our work combines the displacement features with pooling features for the classification tasks.

By just preserving the maximum value of each window, max-pooling introduces problems, such as coordinate transform and structural deformation problems. To solve the coordinate transform problem, Liu et al. defined a new operation in CNNs, called "CoordConv" [8], which allows convolutional filters to know where they are by adding extra input channels that contain coordinates of the original data. The "CoordConv" operation can learn perfect translation invariance and varying degrees of translation dependence at the same time. However, this work only adds the "CoordConv" operation between the convolutional layer and max-pooling layer, which does not change the max-pooling operation in deep insight. To overcome the limitation of modeling geometric transformations in CNNs, Dai et al. proposed a new pooling operation, called "deformable RoI pooling" [9], which adds an offset to each bin position in the regular bin partition of the previous RoI pooling [6, 51]. Compared to our proposed method, it just converts a rectangular region input of arbitrary size into fixed size features but does not combine this information with the pooling features for classification tasks. The purpose of the proposed method is to improve the max-pooling operation by combining the position, or displacement information of the maximal value for recognition tasks.

3. Extracting Displacement Features from Pooling Layers

In this section, we introduce the proposed method. First, we present how to extract the position information and pooling features simultaneously from a pooling layer. Then, we introduce the displacement features based on the position information. Finally, we present how to address the problem if there are more than one maximal value in a pooling window.

3.1. Extracting Displacement Features from a Pooling Layer

The traditional max-pooling operation obtains the maximum value from the pooling windows. The end result is a down-sampling of the convolutional feature map which retains the maximum response (*pooling features*). However, in this step, the max-pooling operation does not record where the maximal value is. In this situation, the max-pooling operation may lose crucial spatial information. In other words, we only know the specific maximal features in the feature map, but we do not know where they came from. In order to address this problem, we extract the pooling features and their position information simultaneously. When we obtain the position of maximum, we divide the position information into two parts, $\underset{s}{\operatorname{argmax}}\,\mathbf{F}_{(s,t)}$ and $\underset{t}{\operatorname{argmax}}\,\mathbf{F}_{(s,t)}$ which represent the horizontal and vertical coordinates of the maximal value in the pooling window, respectively. The size of $\underset{s}{\operatorname{argmax}}\,\mathbf{F}_{(s,t)}$ and $\underset{t}{\operatorname{argmax}}\,\mathbf{F}_{(s,t)}$ are same as the pooling features. If the pooling size is $n \times n$, the $\underset{s}{\operatorname{argmax}}\,\mathbf{F}_{(s,t)}$ and $\underset{t}{\operatorname{argmax}}\,\mathbf{F}_{(s,t)} \in \{0, 1, \cdots, n-1\}$.

Based on the position information, we transform it into the displacement features. Fig. 1 and Fig. 3 show the cases of the even and odd pooling sizes. Therefore, in even cases, all units have displacements in both horizontal and vertical directions. In the odd case, if the maximal value in central row or column, the displacement would be zero in the horizontal and vertical directions, respectively. In the same way as the position information, we divide the displacement features into two parts, $\mathbf{X}_d$ and $\mathbf{Y}_d$. The dimensionality of $\mathbf{X}_d$ and $\mathbf{Y}_d$ also as same as the pooling features.

Now, we will discuss how to transform the position information into displacement features in this part. For a $n \times n$ pooling window, if n is even number, x_d

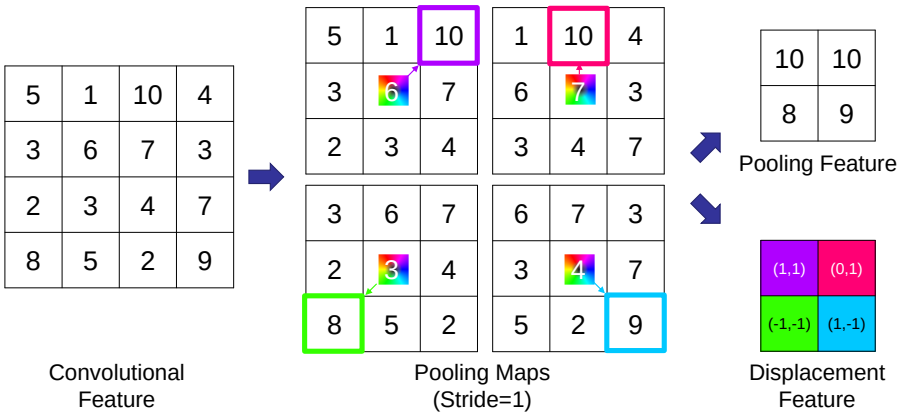

Figure 3: Extracting displacement features from a 3×3 pooling window.

and y_d are the elements of the $\mathbf{X}_d$ and $\mathbf{Y}_d$. The displacement features $\mathbf{X}_d$ and $\mathbf{Y}_d$ are,

$$x_d = \begin{cases} \underset{t}{\operatorname{argmax}}\, \mathbf{F}^{n\times n}_{(s,t)} - \frac{n}{2}; & \underset{t}{\operatorname{argmax}}\, \mathbf{F}^{n\times n}_{(s,t)} < \frac{n}{2} \\ \underset{t}{\operatorname{argmax}}\, \mathbf{F}^{n\times n}_{(s,t)} - \frac{n}{2} + 1; & \underset{t}{\operatorname{argmax}}\, \mathbf{F}^{n\times n}_{(s,t)} \geq \frac{n}{2} \end{cases} \tag{1}$$

$$y_d = \begin{cases} -\underset{s}{\operatorname{argmax}}\, \mathbf{F}^{n\times n}_{(s,t)} + \frac{n}{2}; & \underset{s}{\operatorname{argmax}}\, \mathbf{F}^{n\times n}_{(s,t)} < \frac{n}{2} \\ -\underset{s}{\operatorname{argmax}}\, \mathbf{F}^{n\times n}_{(s,t)} + \frac{n-2}{2}; & \underset{s}{\operatorname{argmax}}\, \mathbf{F}^{n\times n}_{(s,t)} \geq \frac{n}{2} \end{cases} \tag{2}$$

Here, argmax is the function to obtain the position of the maximum value, $\mathbf{F}$ is a feature map, s and t are the horizontal and vertical coordinates. For an odd case, if the pooling size is $n \times n$, n is odd number, the displacement $\mathbf{X}_d$ and $\mathbf{Y}_d$ are,

$$x_d = \underset{t}{\operatorname{argmax}}\, \mathbf{F}^{n\times n}_{(s,t)} - \frac{n-1}{2} \tag{3}$$

$$y_d = -\underset{s}{\operatorname{argmax}}\, \mathbf{F}^{n\times n}_{(s,t)} + \frac{n-1}{2} \tag{4}$$

The displacement features represent positions and directions of the pooling features.

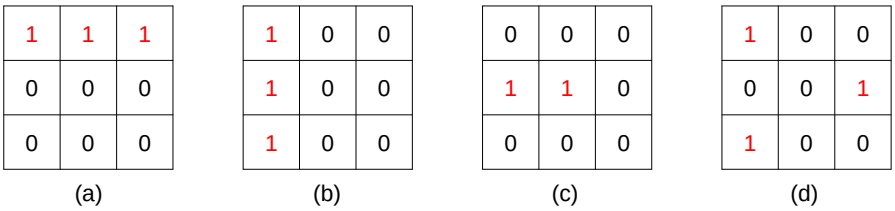

Figure 4: Different cases of multiple maximum. (a) the displacement features are (-1,1),(0,1),(1,1), so the mean displacement feature is (0,1). (b) Displacement features: (-1,1),(-1,0),(-1,-1). Mean displacement feature: (-1,0). (c) Displacement features: (-1,0),(0,0). Mean displacement feature: $(-\frac{1}{2},0)$. (d) Displacement features: (-1,1),(-1,-1),(1,0). Mean displacement feature: $(-\frac{1}{3},0)$.

3.2. Multiple Maximal Value Condition

The previous section only discusses a single maximum in a pooling window. However, for some examples of extracting $\mathbf{X}_d$ and $\mathbf{Y}_d$ from the displacement information, pooling windows may contain several maximums. Fig. 4 presents some cases with this problem. For an extreme example, if a pooling window contains the single background, all the units in the pooling window are same. If we apply the traditional max-pooling operation, only the first unit would be recorded. In this situation, the displacement should not occur because of the single value background. To solve this problem, firstly, we extract all the displacement features of the maximums in one pooling window. Then, we calculate the mean value of the displacement features as the final displacement features. The results would be displacement features with possibly non-integer values.

4. Combining the Displacement Features with Pooling Features

In this section, we introduce how to combine the displacement features with the pooling features to improve the performance of CNNs. The first architecture fuses the two features in a multi-modal CNN by having a second set of convolutional layers and combining them in the fully connected layer. The second architecture transforms the displacement features into the new cosine features

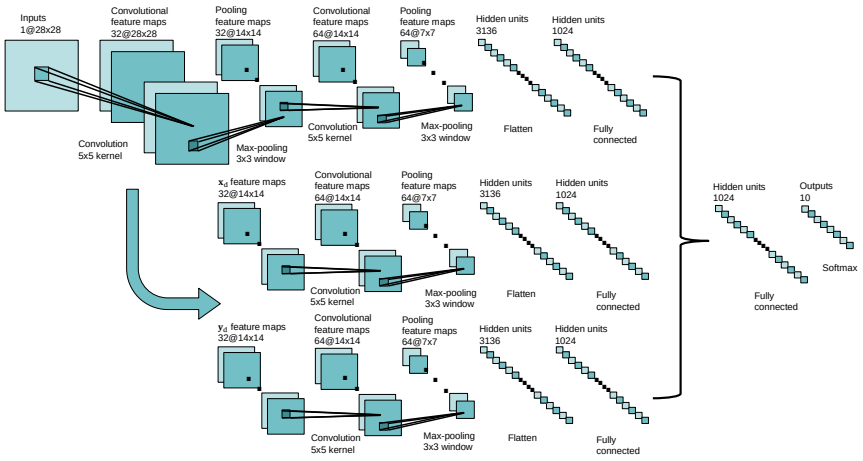

Figure 5: The architecture of the proposed method.

based on PCA and combines the new cosine features with the pooling features in a similar way as the first architecture.

4.1. Combining the Displacement Features with Pooling Features Based on C-NNs

Because the dimensionality of the displacement features is same as the pooling features from the same convolutional layers, we can use a CNN with similar hyper-parameters as the original CNN to process the pooling features and displacement features simultaneously. First, we use a standard CNN on one batch of training samples. Then, we extract the displacement features from the first convolutional layer. The next step is to use another same CNN without the first convolutional and pooling layers on the displacement features. The final step is to fuse the pooling features and the new displacement features in a fully connected layer. To train the proposed method, first we train a CNN to obtain the displacement features, then we use the combined CNN based on the pre-trained values. The structure of the network is illustrated in Fig. 5. This architecture contains two convolutional and pooling layers, three fully connected layers, and the final layer uses the softmax for classification.

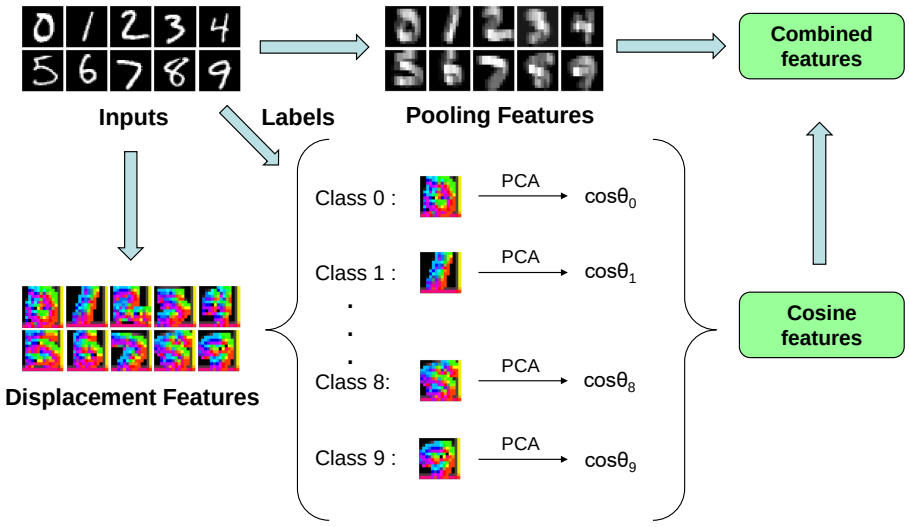

Figure 6: The training process of the cosine features based architecture.

4.2. Extracting the Cosine Features Based on Displacement Features and Principal Component Analysis (PCA)

To further explore more information of the displacement features, we propose a new feature based on PCA. The goal of PCA is to build the low-dimensional representation that describes high-dimensional data with as much of the variance in the data as possible. Due to the high dimensionality of the displacement features, the PCA can reduce some redundancy and preserve the primary information of the displacement features.

In order to apply PCA to the displacement features, we design the architecture as in Fig. 6. For the training data, we first obtain their corresponding displacement features and divide them into C groups where C is the number of classes. For each group, we use a PCA model to get the cosine features, or,

$$\cos\theta = \frac{\mathbf{D} \cdot \sum_{i=1}^{k} \lambda_i \nu_{\mathbf{i}}}{|\mathbf{D}||\sum_{i=1}^{k} \lambda_i|}, \tag{5}$$

where $\mathbf{D}$ is the displacement features both in horizontal ($\mathbf{X}_d$) and vertical ($\mathbf{Y}_d$) directions, λ_i is the i-th eigenvalue of $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_i, \ldots, \lambda_k)$, k the number of largest eigenvalues that we want to use in the PCA model, and ν_i is the i-th

eigenvector corresponding to λ_i. If a displacement feature space is similar to the PCA space, then the value of $\cos\theta$ will close to 1. For a test sample, if the label is same as one sample in training data, it should have a very similar $\cos\theta$. Combining the cosine features with pooling features is similar to have a penalty for the pooling features.

5. Experimental Results

To evaluate the effectiveness of the proposed frameworks, we design a series of experiments on three text datasets, MNIST dataset [1], HASY dataset [2], and Chars74K-font dataset [3].

5.1. Dataset Descriptions

MNIST is an isolated handwritten digit dataset has a training set of 60,000 samples (55,000 samples for training and 5,000 samples for validation), and a test set of 10,000 samples with the size of 28×28 pixels. The HASY dataset is a dataset of handwritten mathematical symbols. It contains 168,233 instances of 369 classes with size 32×32. The Chars74K dataset is composed from different sub-datasets. We use the sub-dataset which includes characters from computer fonts with different variations (combinations of italic, bold and normal). We refer to this sub-dataset as Chars74K-font and it contains 62,992 samples of 62 classes with the size 128×128.

5.2. Classification on MNIST Dataset

To evaluate the proposed displacement features and cosine features, we design a series of experiments on MNIST dataset. We use different pooling sizes from 2×2 to 6×6 to evaluate the performance of the displacement features. For example, if the pooling size is 2×2, the corresponding displacement features

[1] http://yann.lecun.com/exdb/mnist/
[2] https://zenodo.org/record/259444#.XB8r6PZuJPY
[3] http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/

Table 1: Classification accuracy on the MNIST dataset. Here, '$\mathbf{P}$' represents the pooling features, '$\mathbf{X}_{d_1}$' and '$\mathbf{Y}_{d_2}$' represents the displacement features in the horizontal and vertical direction from the first layer. '$\mathbf{D}_1$' and '$\mathbf{D}_2$' represent the displacement features from the first and second layers. '$\mathbf{S}_1$' and '$\mathbf{S}_2$' represent the cosine features from the first and second layers.

Pooling Window	$\mathbf{P}$	$\mathbf{X}_{d_1}$	$\mathbf{Y}_{d_1}$	$\mathbf{X}_{d_1}$	$\mathbf{Y}_{d_1}$	$\mathbf{P}+\mathbf{D}_1$	$\mathbf{P}+\mathbf{D}_2$	$\mathbf{P}+\mathbf{S}_1$	$\mathbf{P}+\mathbf{S}_2$	$\mathbf{P}+\mathbf{S}_1+\mathbf{S}_2$
2×2	0.9932	0.9785	0.9804	0.9758	0.9725	0.9932	0.9922	**0.9940**	0.9928	**0.9940**
3×3	0.9928	0.9843	0.9835	0.9768	0.9753	0.9940	0.9920	**0.9943**	0.9930	0.9938
4×4	0.9929	0.9821	0.9818	0.9813	0.9802	0.9934	0.9920	**0.9934**	0.9929	0.9932
5×5	0.9912	0.9895	0.9889	0.9823	0.9811	0.9923	0.9902	**0.9930**	0.9908	**0.9930**
6×6	0.9911	0.9872	0.9896	0.9822	0.9815	0.9920	0.9911	**0.9928**	0.9911	0.9925

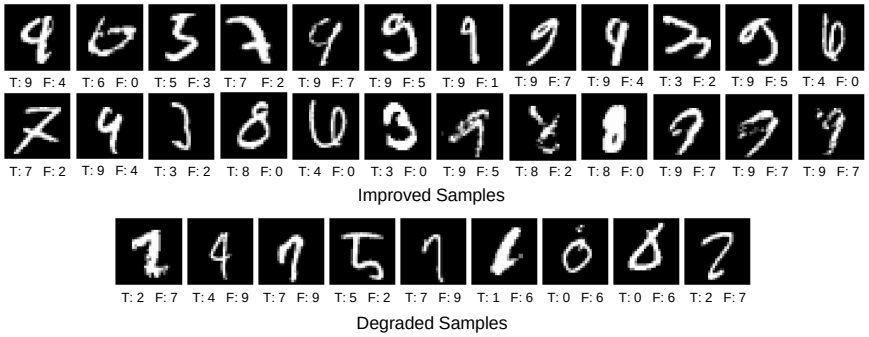

Figure 7: Example of improved and degraded samples by using the pooling+cosine features with 3×3 pooling size. Here, 'T' represents the true class and 'F' represents the false class by prediction.

$\mathbf{X}_d$ and $\mathbf{Y}_d$ belong to $[-1,1]$. We use cross entropy as the loss function, and minimize it by Adam [52] with a 1×10^{-4} learning rate, the batch size and number of iterations are set to 50 and 20,000, respectively.

The second architecture is based on the first model. We compute the cosine features from the displacement features to further penalize the pooling features. We just use 10% of eigenvalues to build the cosine features. This is because only the largest eigenvalues retain the crucial information of the original data, and other information are redundancy or noise in most cases. Then, we combine the cosine features with the pooling features in the fully connected layer.

The experimental results are shown in Table 1. From Table 1, we can see

that with the size of the pooling windows increasing, the performance of the displacement features increases. It is easy to understand because the larger pooling windows will provide more displacement information for learning procedure. Using only the displacement features for classification can also obtain a high accuracy. In addition, combining the displacement features from the first convolutional layer with the pooling features improves the classification accuracies for all pooling sizes. However, combining the displacement features from the second layer with the pooling features does not improve the accuracy. The reason for this could be that the displacement features from the second layer have already lost the spatial information of the first layer. We also find that only combining the cosine features from the second layer with the pooling features can not get the better results by combining the cosine features from the first layer. Furthermore, combining the displacement features from the first layer with the pooling features always improves the performance, especially when the pooling size is 3×3.

Fig. 7 presents some improved samples (misclassifications by the only pooling features trial but correct classification by the pooling+cosine features trial) and degraded samples (correct classifications by the pooling features trial but misclassifications by the pooling+cosine features trial). There are many '9's that are improved by the cosine features. These '9's are recognized as '4's or '7's by the pooling features alone. It is easy to understand that '9', '7', and '4' have similar local features and traditional CNNs might down sample those features with max-pooling so much that the discriminating information is lost. Therefore, introducing the cosine features act like a penalty for the final decision and introduce the position information of the local features. However, introducing the cosine features also brings the problem that the network will more sensitive to the local features.

The confusion matrix are shown in Fig. 8. We can see that combining the displacement features with the pooling features improved the recognition accuracies on class '3', '4', '5', '7', '8', and '9', and just degraded a little on class '2', and '6'. For the class '9', the proposed method improved a lot when some

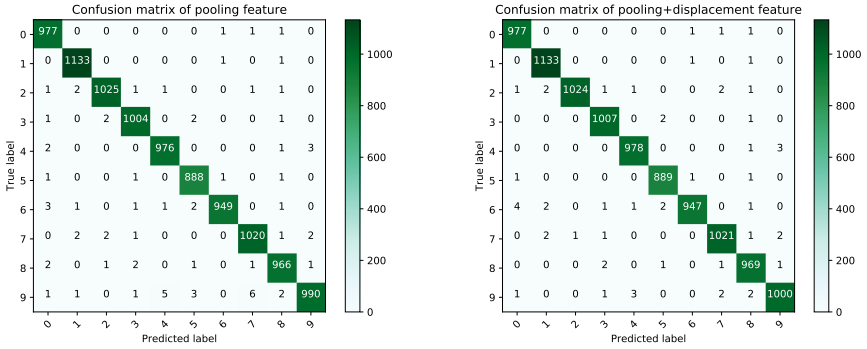

Figure 8: Confusion matrix by only using the pooling features and combining the displacement features with the pooling features.

test samples are misclassified to class '4', '5', and '7'.

5.2.1. Visualization of the Displacement Features

In order to analyze the class-wise trends of the displacement features, we visualize them based on an HSV color model whose color and intensity denote the direction and average value of displacement feature. The visualization results are shown in Fig. 9. We can see that each corresponding displacement feature records the direction information that describes the displacement of the maximums. The displacement features of the different classes have large dissimilarities. In addition, we can see that the samples in the same class often have the similar behaviors. For instance, in the first filter of all class '3' samples, the top left corners are blue, the top right corners are in green and the bottom right corners are in red. The reason that only using the displacement features can also obtain the high accuracies for classification tasks is due to the large differences in the class behaviors.

5.2.2. The Distribution of the Displacement Features

In order to further observe the behaviors of the displacement features on different categories, we illustrate the cumulative histograms of the displacement features, which accounts the coordinate points of the displacement features of

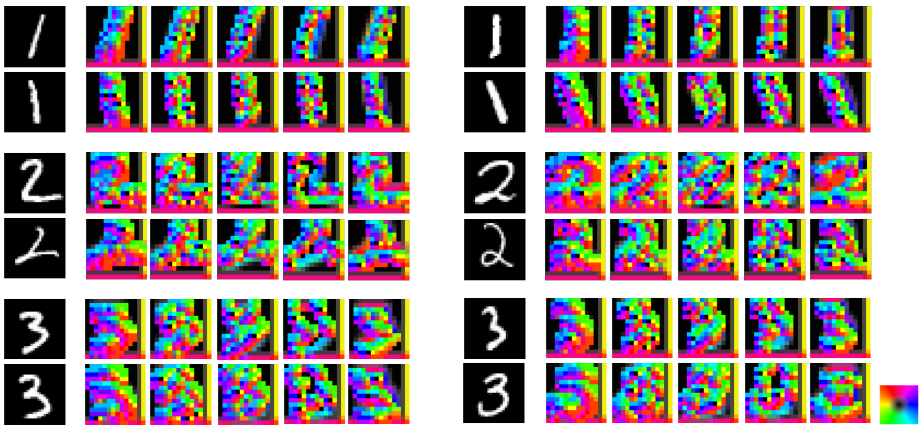

Figure 9: Visualization of the displacement features on the different samples that are in the same class on MNIST dataset. Here, the pooling size is 3×3. Each column represents one convolutional filter

different samples on all feature maps. Here, the pooling size of the first pooling layer is 3×3 with stride 2.

The distribution of the displacement features in the first pooling layer is shown in Fig. 10. In Fig. 10, Due to the pooling size being 3×3, the displacement features $\mathbf{X}_d$ and $\mathbf{Y}_d$ belong to $[-1, 1]$. We can see that the samples from the same class have similar distributions and from different classes have very different distributions. For the class '1', there is a relatively horizontal line in the distribution for the two samples. They are very similar because the displacement information is rare in vertical direction (the shape of class '1' goes from top to bottom). For the class '2', the distribution is more scattered than the class '1', and '3'. For the class '3', there are more points around (0,0) than class '1', and '2' and the distribution is more compact. In addition, most values in all of the samples is (0,0) because the most features in original images are background.

5.2.3. Class-wise Similarity of the Displacement Features in the PCA Subspaces

We also measure the displacement features in PCA subspaces to compare the similarities between different categories. First, we adopt ten different PCA

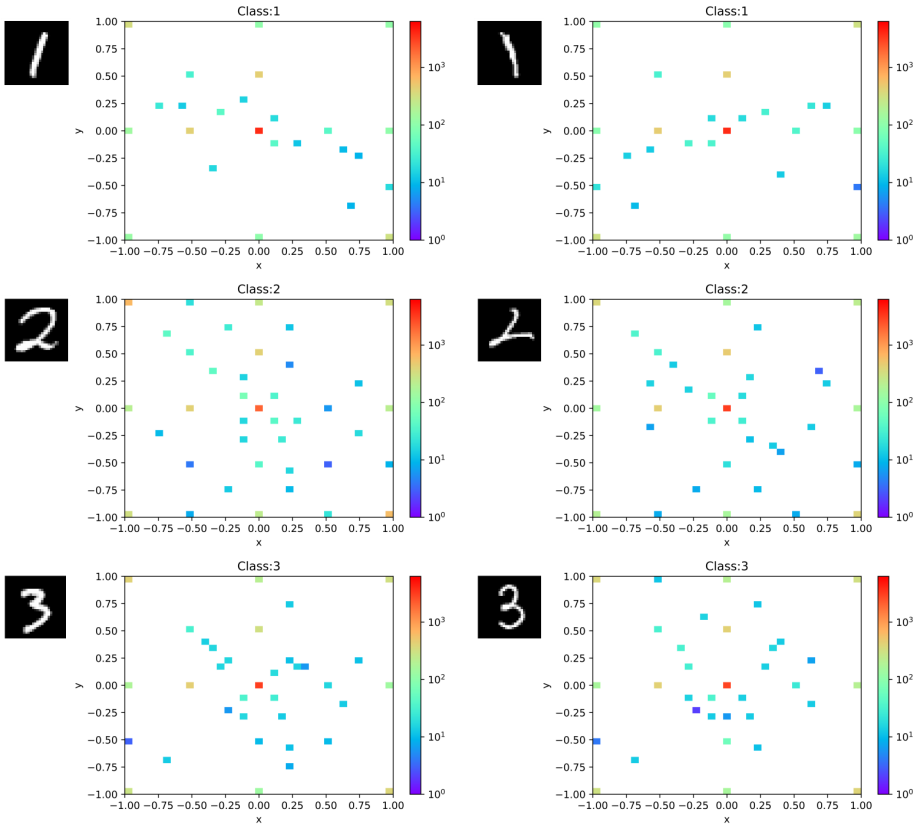

Figure 10: The distribution of displacement features from class '1', '2', and '3' (each row represents 4 different samples in the same class). Here, the pooling size is 3×3, the color histograms represent the number of the displacement features on corresponding coordinate points.

models on the samples, one for each class. Then, we preserve 10% of eigenvalues for each PCA model to build up the PCA subspaces. Based on these subspaces, we calculate the class-wise similarity matrix for each feature map. Then, the similarity can be defined by using the canonical angles,

$$\cos\delta_i = \sup_{\substack{\boldsymbol{\alpha}_i \perp \boldsymbol{\alpha}_j, \boldsymbol{\beta}_i \perp \boldsymbol{\beta}_j \\ 1 \leq i,j \leq p}} \frac{\boldsymbol{\alpha}_i^T \boldsymbol{\beta}_i}{\|\boldsymbol{\alpha}_i\| \|\boldsymbol{\beta}_i\|} \tag{6}$$

Here, $\boldsymbol{\alpha} \in \mathbf{P}, \boldsymbol{\beta} \in \mathbf{Q}$, $\mathbf{P}$ and $\mathbf{Q}$ are two PCA subspaces, $\mathbf{P}, \mathbf{Q} \in \mathbb{R}^n$, $\dim \mathbf{P} = p \leq \dim \mathbf{Q} = q$. Then the similarity can be defined as,

$$S = \frac{1}{p} \sum_{i=1}^{p} \cos\delta_i^2 \tag{7}$$

If two PCA subspaces completely coincide with each other, all canonical angles will be 0 and S equals to 1. The similarity gets smaller as the two spaces separate. Finally, the similarity is zero when the two subspaces are orthogonal to each other [53]. The similarity matrix in PCA subspaces are shown in Fig. 11. We can see that the similarities in different classes are relatively small, which means that the displacement features are discriminative in different classes. In addition, the similarity of class '4', and '9', '7', and '9' are larger than others. It means that it is easier to misclassify class '4', '7', and '9'. Our classification results also showed it. However, the similarities are all far away from 1, which means that the displacement features are also discriminative for recognition tasks.

5.3. Classification on HASY Dataset

HASY is a more challenging dataset than the MNIST dataset because the number of the classes in HASY dataset is larger than MNIST [10]. Furthermore, there are many classes that are very similar in the HASY dataset, such as, \xi (ξ) and \zeta (ζ) whose shapes are similar when writing them, \varphi (φ) and \phi (ϕ) who have the different glyphs can correspond to the same semantic entity, \sum ($\sum$) and \Sigma (Σ) who have the same glyph but different semantics and hence they are different symbols.

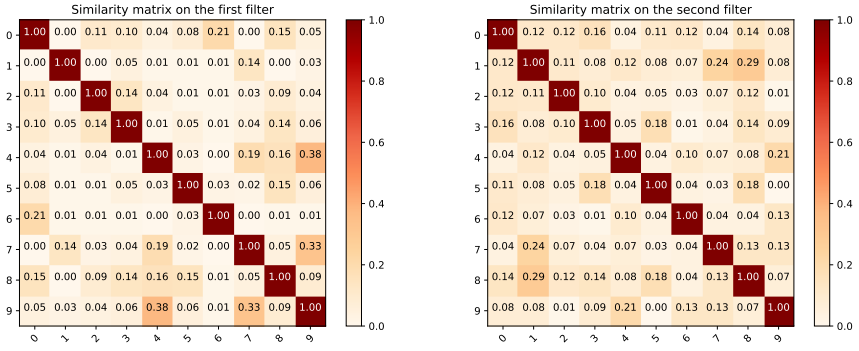

Figure 11: Similarity matrix on the first and second filters in PCA subspaces.

We deploy our experiments on 10 pre-defined folds for 10-fold cross-validation. 3-layer (CNN-3) and 4-layer (CNN-4) CNNs and the CNN-4a architecture [10] are used in our experiments. We also compare our proposed method with Random Forest [54, 55], MLP [56, 55] and, LDA [57, 55]. For all CNN based architectures, Adam optimizer are used for training procedure. The hyper-parameters are same as in [10]. The description of the compared models are listed as follow.

- **Random Forest** [54, 10]: Random forests use a large number of decision trees in an ensemble classifier.

- **MLP** [56, 10]: The multilayer perceptron (MLP) is a fully-connected feed forward neural network.

- **LDA** [57, 10]: Linear Discriminant Analysis (LDA) is a classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes rule.

- **CNN-3** [10]: CNN-3 is a 3-layer CNN model, which contains a convolutional layer with 32 filters of 3×3 kernel size is followed by a 2×2 max-pooling layer with stride 2. The next layer is another convolutional layer with 64 filters of 3×3 kernel size followed by a 2×2 max-pooling layer with stride 2. The output layer is a fully connected softmax layer

Table 2: Classification results on HASY Dataset

Models	Accuracy
Random Forest [54, 10]	0.624
MLP [56, 10]	0.622
LDA [57, 10]	0.468
CNN-3 [10]	0.784
CNN-4 [10]	0.805
CNN-4a [10]	0.810
CNN-3+ours	0.788±0.005
CNN-4+ours	0.814±0.008
CNN-4a+ours	**0.823±0.002**

with 369 nodes.

- **CNN-4** [10]: CNN-4 is a 4-layer CNN model. Like the CNN-3, it adds another convolutional layer with 128 filters followed by a 2×2 max-pooling layer before the softmax layer.

- **CNN-4a** [10]: CNN-4a is also similar with CNN-3. The convolutional layer is same as the 3-layer CNN. Then it contains a fully connected layer with 1024 nodes and tanh activation function, a dropout layer with probability 0.5 and a softmax layer.

Table 2 shows the 10-fold cross-validation results for three architectures. We conduct the t-test with confidence value 0.05 on the results. From Table 2, we can see that, the proposed method (combining the cosine features with pooling features) improves the performance of all CNN based architectures and performs significantly better than other methods. The CNN-4 and CNN-4a models with our method obtain the best results.

Fig. 12 shows an example that is improved by the proposed method. The ground truth label of this example is \zeta (ζ). If we only use the pooling features for classification, it would be classified to \xi (ξ) (in the upper right of the

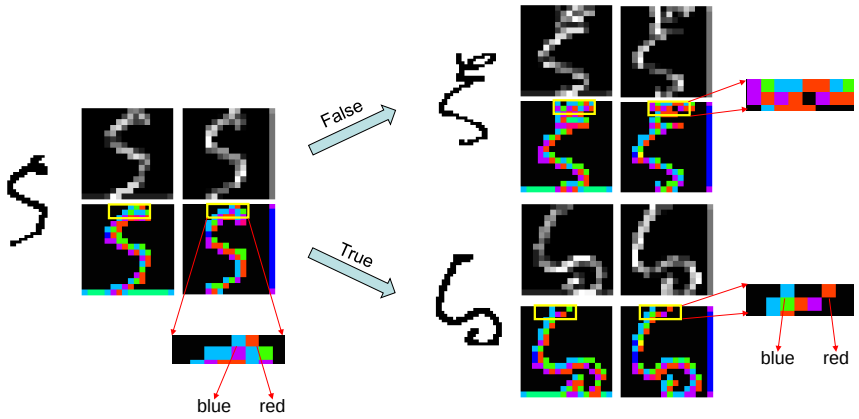

Figure 12: One improved sample by using our proposed method. The left is a test sample that is improved by our method, the upper right is a sample whose label is same as the misclassified label, the lower right is another sample whose label is same as test sample.

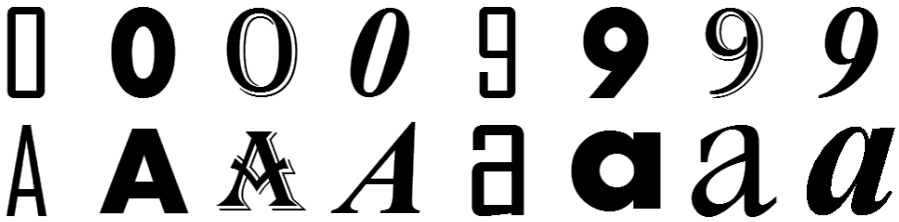

Figure 13: The Samples from Chars74k-font dataset.

figure). The reason is that the pooling features of the \xi (ξ) and \xi (ξ) is very similar. For our proposed displacement features, there is a clear discrimination between these two samples if they are from different classes. For example, the top of the image is in blue and red direction in two \zeta (ζ) samples, and the left edge is in blue direction. These good characteristics help for improving the performance of the max-pooling operation.

5.4. Classification on Chars74k-font dataset

The Chars74K-font dataset is also a more challenging dataset than MNIST dataset. It includes 62 categories from the numbers 0 to 9, capital letters A to Z and lowercase letters a to z which are synthesised by different computer

Table 3: Classification results on Chars74K-font dataset

Method	Accuracy
Global DCT [55]	0.6630
ART [55]	0.4920
Shape Context [35, 60]	0.6483
SIFT [61, 60]	0.4694
Block DCT [55]	0.6640
Neumann et.al [62]	0.7160
Geometric Blur [63, 60]	0.6971
I2CDML [64]	0.7300
Lenet [58]	0.8839
Lenet-5 [58]	0.8553
SPnet [59]	0.9056
Lenet+ours	0.8896±0.002
Lenet-5+ours	0.8593±0.004
SPnet+ours	**0.9145±0.005**

fonts. Fig. 13 shows examples from the Chars74k-font dataset. We can see that the samples in the same category also undergo deformations. The architectures that we use in this dataset are Lenet [58], Lenet-5 [58] and SPnet [59]. For SPnet, we use Adam optimizer with a 1×10^{-5} learning rate, the batch size and iterations are set to 64 and 200,000, respectively. For our experiments, we use 55% (34,658) samples for training, 20% (12,586) samples for validation and 25% (15,748) samples for test procedure. The original images are resized to 28×28 for Lenet, 32×32 for Lenet-5, 60×60 for SPnet. The experiments are repeated 10 times and the results are shown in Table 3. We also compare the proposed method with many state-of-the-art methods in our experiments. The methods are listed as follows.

- **Global DCT** [55]: This method uses Discrete Cosine Transform (DCT)

to extract the features from character after normalization for scale and translation. Then nearest neighbor classifier is used for recognition task.

- **ART** [55]: This method uses Angular Radial Transform (ART) descriptor to obtain the feature vectors. Then, it uses a nearest neighbor classifier for classification.

- **Shape Context** [35, 60]: This method uses Sobel edge detector to extract the features. Then, it uses a nearest neighbor classifier.

- **SIFT** [61, 60]: This method used Harris Hessian-Laplace detector to extract the Scale Invariant Feature Transform (SIFT). Then it used the nearest neighbor classifier for classification.

- **Block DCT** [55]: This method is based on Global DCT. It performs 8×8 block-based DCT and retains 15% coefficients for every block.

- **Neumann et.al** [62]: This method uses directional feature vectors for training multi-class support vector machines (SVMs) classifier with Radial Basis Function (RBF) kernel.

- **Geometric Blur** [63, 60]: This method is similar to Shape Context method. The region around an interest point is blurred according to the distance from this point. Then, it uses a nearest neighbor classifier.

- **I2CDML** [64]: This method rotated the original images through a range of angles to form the 3-mode tensor. Then, it uses a rank-1 Tucker decomposition to get holistic feature descriptor for the character image and applied the image-to-class distance metric learning for classification.

- **Lenet** [58]: Lenet contains 2 convolutional layers, 2 pooling layers and 3 fully connected layers.

- **Lenet-5** [58]: Lenet-5 contains 3 convolutional layers, 2 pooling layers and 3 fully connected layers.

- **SPnet** [59]: SPnet contains 3 convolutional layers, 3 pooling layers and 3 fully connected layers.

We conduct the t-test with confidence value 0.05 on the results in Table 3. We can see that the proposed methods (combining the cosine features with pooling features) are significant better and obtain the state-of-the-art results compared with other methods. Using our method improves the performance of Lenet, Lenet-5 and SPnet (CNN based models). It means that the proposed method is robust for different CNN architectures.

6. Conclusion

In this work, a new feature named a displacement feature is extracted from the max-pooling operation, which records the location information of the maximum values in pooling windows. To improve the performance of the traditional CNN based architectures, we fused the displacement features with the pooling features to capture the structural deformations between different samples. Then, cosine features based on the displacement features are proposed to further improve the performance for different text recognition tasks.

To evaluate the performance of the proposed method, we conducted the experiments on three text datasets, MNIST, HASY and Chars74K-font and compared the proposed method with traditional CNN based models and the state-of-the-art models. Extensive experimental results demonstrate that, the displacement features can improve the CNN based architectures and achieve the state-of-the-art results. In addition, we found that the displacement features can capture the structural deformations between the samples from different classes, which is very useful for some specific text recognition tasks.

For the future work, we plan to adopt some other techniques to enhance the performance of the displacement features and applied it to larger deep learning models and other applications. It is expected that the displacement features can greatly improve the performance of different architectures.

References

[1] Y.-L. Boureau, J. Ponce, Y. LeCun, A theoretical analysis of feature pooling in visual recognition, in: Proceedings of the ICML, 2010, pp. 111–118.

[2] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444. doi:10.1038/nature14539.

[3] M. Jaderberg, K. Simonyan, A. Zisserman, et al., Spatial transformer networks, in: Proceedings of the NIPS, 2015, pp. 2017–2025.

[4] A. Krizhevsky, I. Sutskever, G. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the NIPS, 2012, pp. 1097–1105. doi:10.1145/3065386.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al., Going deeper with convolutions, in: Proceedings of the CVPR, 2015, pp. 1–9. doi:10.1109/CVPR.2015.7298594.

[6] R. Girshick, Fast r-cnn, in: Proceedings of the ICCV, 2015, pp. 1440–1448. doi:10.1109/ICCV.2015.169.

[7] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the CVPR, 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.

[8] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, J. Yosinski, An intriguing failing of convolutional neural networks and the coordconv solution, arXiv preprint arXiv:1807.03247, 2018.

[9] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, Y. Wei, Deformable convolutional networks, in: Proceedings of the ICCV, 2017, pp. 764–773. doi:10.1109/ICCV.2017.89.

[10] M. Thoma, The hasyv2 dataset, arXiv preprint arXiv:1701.08380, 2017.

[11] Y. Zheng, B. K. Iwana, S. Uchida, Discovering class-wise trends of max-pooling in subspace, in: Proceedings of the ICFHR, 2018, pp. 98–103. doi:10.1109/ICFHR-2018.2018.00026.

[12] Z. Xie, Z. Sun, L. Jin, H. Ni, T. Lyons, Learning spatial-semantic context with fully convolutional recurrent network for online handwritten chinese text recognition, TPAMI 40 (8) (2018) 1903–1917. doi:10.1109/TPAMI.2017.2732978.

[13] I.-J. Kim, X. Xie, Handwritten hangul recognition using deep convolutional neural networks, IJDAR 18 (2015) 1–13. doi:10.1007/s10032-014-0229-4.

[14] D. C. Ciresan, U. Meier, L. M. Gambardella, J. Schmidhuber, Convolutional neural network committees for handwritten character classification, in: Proceedings of the ICDAR, 2011, pp. 1135–1139. doi:10.1109/ICDAR.2011.229.

[15] C. Wu, W. Fan, Y. He, J. Sun, S. Naoi, Handwritten character recognition by alternately trained relaxation convolutional neural network, in: Proceedings of the ICFHR, 2014, pp. 291–296. doi:10.1109/ICFHR.2014.56.

[16] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, W. Xu, Cnn-rnn: A unified framework for multi-label image classification, in: Proceedings of the CVPR, 2016, pp. 2285–2294. doi:10.1109/CVPR.2016.251.

[17] H. Lee, H. Kwon, Going deeper with contextual cnn for hyperspectral image classification, TIP 26 (2017) 4843–4855. doi:10.1109/TIP.2017.2725580.

[18] M. Zhang, W. Li, Q. Du, Diverse region-based cnn for hyperspectral image classification, TIP 27 (2018) 2623–2634. doi:10.1109/TIP.2018.2809606.

[19] C.-Y. Lee, P. W. Gallagher, Z. Tu, Generalizing pooling functions in cnns: Mixed, gated, and tree, TPAMI 40 (2018) 863–875. doi:10.1109/TPAMI.2017.2703082.

[20] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al., Deepid-net: Deformable deep convolutional neural networks for object detection, in: Proceedings of the CVPR, 2015, pp. 2403–2412. doi:10.1109/CVPR.2015.7298854.

[21] M. D. Zeiler, G. W. Taylor, R. Fergus, Adaptive deconvolutional networks for mid and high level feature learning, in: Proceedings of the ICCV, 2011, pp. 2018–2025. doi:10.1109/ICCV.2011.6126474.

[22] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Proceedings of the ECCV, 2014, pp. 818–833. doi:10.1007/978-3-319-10590-1_53.

[23] M. D. Zeiler, D. Krishnan, G. W. Taylor, R. Fergus, Deconvolutional networks, in: Proceedings of the CVPR, 2010, pp. 2528–2535. doi:10.1109/CVPR.2010.5539957.

[24] Y. Zheng, Y. Cai, G. Zhong, Y. Chherawala, Y. Shi, J. Dong, Stretching deep architectures for text recognition, in: Proceedings of the ICDAR, 2015, pp. 236–240. doi:10.1109/ICDAR.2015.7333759.

[25] M. J. Er, Y. Zhang, N. Wang, M. Pratama, Attention pooling-based convolutional neural network for sentence modelling, Information Sciences 373 (2016) 1339–1351. doi:10.1016/j.ins.2016.08.084.

[26] Y. Gong, L. Wang, R. Guo, S. Lazebnik, Multi-scale orderless pooling of deep convolutional activation features, in: Proceedings of the ECCV, 2014, pp. 392–407. doi:10.1007/978-3-319-10584-0_26.

[27] H. Jégou, M. Douze, C. Schmid, P. Pérez, Aggregating local descriptors into a compact image representation, in: Proceedings of the CVPR, 2010, pp. 3304–3311. doi:10.1109/CVPR.2010.5540039.

[28] S. Sabour, N. Frosst, G. E. Hinton, Dynamic routing between capsules, in: Proceedings of the NIPS, 2017, pp. 3857–3867.

[29] G. E. Hinton, A. Krizhevsky, S. D. Wang, Transforming auto-encoders, in: Proceedings of the ICANN, 2011, pp. 44–51. doi:10.1007/978-3-642-21735-7_6.

[30] P. Weinzaepfel, J. Revaud, Z. Harchaoui, C. Schmid, Deepflow: Large displacement optical flow with deep matching, in: Proceedings of the ICCV, 2013, pp. 1385–1392. doi:10.1109/ICCV.2013.175.

[31] J. Revaud, P. Weinzaepfel, Z. Harchaoui, C. Schmid, Deepmatching: Hierarchical deformable dense matching, IJCV 120 (2016) 300–323. doi:10.1007/s11263-016-0908-3.

[32] J. Revaud, P. Weinzaepfel, Z. Harchaoui, C. Schmid, Epicflow: Edge-preserving interpolation of correspondences for optical flow, in: Peoceedings of the CVPR, 2015, pp. 1164–1172. doi:10.1109/CVPR.2015.7298720.

[33] K. Santosh, Character recognition based on dtw-radon, in: Proceedings of the ICDAR, 2011, pp. 264–268. doi:10.1109/ICDAR.2011.61.

[34] K. Santosh, B. Lamiroy, L. Wendling, Dtw–radon-based shape descriptor for pattern recognition, IJPRAI 27 (03) (2013) 1350008. doi:10.1142/S0218001413500080.

[35] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, TPAMI 24 (4) (2002) 509–522. doi:10.1109/34.993558.

[36] A. Poznanski, L. Wolf, Cnn-n-gram for handwriting word recognition, in: Proceedings of the CVPR, 2016, pp. 2305–2314. doi:10.1109/CVPR.2016.253.

[37] X. Xiao, Y. Yang, T. Ahmad, L. Jin, T. Chang, Design of a very compact cnn classifier for online handwritten chinese character recognition using dropweight and global pooling, in: Proceedings of the ICDAR, Vol. 1, 2017, pp. 891–895. doi:10.1109/ICDAR.2017.150.

[38] X. Xiao, L. Jin, Y. Yang, W. Yang, J. Sun, T. Chang, Building fast and compact convolutional neural networks for offline handwritten chinese character recognition, Pattern Recognition 72 (2017) 72–81. doi:10.1016/j.patcog.2017.06.032.

[39] M. Bušta, L. Neumann, J. Matas, Deep textspotter: An end-to-end trainable scene text localization and recognition framework, in: Proceedings of the ICCV, 2017, pp. 2223–2231. doi:10.1109/ICCV.2017.242.

[40] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, X. Xue, Arbitrary-oriented scene text detection via rotation rroposals, TMM 20 (2018) 3111–3122. doi:10.1109/TMM.2018.2818020.

[41] X. Rong, C. Yi, Y. Tian, Unambiguous text localization and retrieval for cluttered scenes, in: Proceedings of the CVPR, 2017, pp. 3279–3287. doi:10.1109/CVPR.2017.349.

[42] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, J. Liang, East: An efficient and accurate scene text detector, in: Proceedings of the CVPR, 2017, pp. 2642–2651. doi:10.1109/CVPR.2017.283.

[43] L. Gomez, A. Nicolaou, D. Karatzas, Improving patch-based scene text script identification with ensembles of conjoined networks, Pattern Recognition 67 (2017) 85–96. doi:10.1016/j.patcog.2017.01.032.

[44] A. K. Bhunia, A. Konwer, A. K. Bhunia, A. Bhowmick, P. P. Roy, U. Pal, Script identification in natural scene image and video frames using an attention based convolutional-lstm network, Pattern Recognition 85 (2019) 172–184. doi:10.1016/j.patcog.2018.07.034.

[45] B. Shi, X. Bai, C. Yao, Script identification in the wild via discriminative convolutional neural network, Pattern Recognition 52 (2016) 448–458. doi:10.1016/j.patcog.2015.11.005.

[46] S. Ukil, S. Ghosh, S. M. Obaidullah, K. Santosh, K. Roy, N. Das, Improved word-level handwritten indic script identification by integrating s-

mall convolutional neural networks, Neural Computing and Applications (2019) 1–16doi:10.1007/s00521-019-04111-1.

[47] Y.-C. Wu, F. Yin, C.-L. Liu, Improving handwritten chinese text recognition using neural network language models and convolutional neural network shape models, Pattern Recognition 65 (2017) 251–264. doi:10.1016/j.patcog.2016.12.026.

[48] Z. Liu, Y. Li, F. Ren, W. L. Goh, H. Yu, Squeezedtext: A real-time scene text recognition by binary convolutional encoder-decoder network, in: Proceedings of the AAAI, 2018, pp. 7194–7201.

[49] R. Qian, Y. Yue, F. Coenen, B. Zhang, Traffic sign recognition with convolutional neural network based on max pooling positions, in: Proceedings of the ICNC-FSKD, 2016, pp. 578–582. doi:10.1109/FSKD.2016.7603237.

[50] J. Zhao, M. Mathieu, R. Goroshin, Y. Lecun, Stacked what-where auto-encoders, in: Proceedings of the ICLR Workshop, 2015.

[51] J. Dai, Y. Li, K. He, J. Sun, R-fcn: Object detection via region-based fully convolutional networks, in: Proceedings of the NIPS, 2016, pp. 379–387.

[52] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[53] Y. Igarashi, K. Fukui, 3d object recognition based on canonical angles between shape subspaces, in: Proceedings of the ACCV, 2010, pp. 580–591. doi:10.1007/978-3-642-19282-1_46.

[54] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32. doi:10.1023/A:101093340.

[55] D. Kumar, A. Ramakrishnan, Recognition of kannada characters extracted from scene images, in: Proceedings of the DAR, 2012, pp. 15–21. doi:10.1145/2432553.2432557.

[56] H. Bourlard, C. J. Wellekens, Links between markov models and multilayer perceptrons, in: Proceedings of the NIPS, 1989, pp. 502–510.

[57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, Journal of machine learning research 12 (Oct) (2011) 2825–2830.

[58] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324. doi:10.1109/5.726791.

[59] S. B. Driss, M. Soua, R. Kachouri, M. Akil, A comparison study between mlp and convolutional neural network models for character recognition, in: Proceedings of the RIVP, Vol. 10223, 2017, p. 1022306. doi:10.1117/12.2262589.

[60] T. E. De Campos, B. R. Babu, M. Varma, et al., Character recognition in natural images., in: Proceedings of the VISAPP, Vol. 7, 2009, pp. 273–280. doi:10.5220/0001770102730280.

[61] D. G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the ICCV, Vol. 2, 1999, pp. 1150–1157. doi:10.1109/ICCV.1999.790410.

[62] L. Neumann, J. Matas, A method for text localization and recognition in real-world images, in: Proceddings of the ACCV, 2010, pp. 770–783. doi:10.1007/978-3-642-19318-7_60.

[63] A. C. Berg, T. L. Berg, J. Malik, Shape matching and object recognition using low distortion correspondences, in: Proceedings of the CVPR, Vol. 1, 2005, pp. 26–33. doi:10.1109/CVPR.2005.320.

[64] M. Ali, H. Foroosh, Character recognition in natural scene images using rank-1 tensor decomposition, in: Proceedings of the ICIP, 2016, pp. 2891–2895. doi:10.1109/ICIP.2016.7532888.