Mining the Displacement by Max-pooling in Convolutional Neural Networks

鄭, 煜辰

Mining the Displacement by Max-pooling in Convolutional Neural Networks

Yuchen Zheng

Department of Advanced Information Technology
Doctor of Philosophy

KYUSHU UNIVERSITY

June 2020

Mining the Displacement by Max-pooling in Convolutional Neural Networks

Yuchen Zheng

Department of Advanced Information Technology
June 2020
Doctor of Philosophy

Abstract

The max-pooling operation is a common step in modern deep convolutional neural networks (CNNs), which is often introduced to obtain translation-invariant representations and downsample the feature maps of convolutional layers. However, in doing so, it loses the spatial information of the maximums. In this thesis, a novel feature is extracted from the max-pooling operation in CNNs, called displacement features. The displacement features record the location coordinates of the maximums in pooling windows of the max-pooling operation, which represents the "inter-class" or "intra-class" micro differences between different samples. Then, the class-wise trends and behaviors of the displacement features are discovered and analyzed in different ways. To verify the effectiveness of the displacement features, the displacement features are applied on two classical tasks, text recognition and offline signature verification. For text recognition tasks, the displacement features are extracted from the max-pooling layer and combined with the features resulting from max-pooling to capture the inter-class micro differences between the similar classes. The displacement features compensate for spatial information lost in the traditional max-pooling operation helps discriminate unnecessary absorptions from necessary absorptions. The extensive experiments and discussions on three text datasets, MNIST, HASY, and Chars74K-font datasets demonstrate that the proposed displacement features can improve the performance of the CNN based architectures and tackle the issues with the micro differences of max-pooling in the text recognition tasks. For offline signature verification tasks, the displacement features of the maximums in the max-pooling operation are extracted and fused with the pooling features to capture the intra-class micro differences between the genuine signatures and skilled forgeries as a feature extraction procedure. The displacement features represent the crucial differences between the genuine signatures and their corresponding skilled forgeries, which is useful for verification systems. The extensive experimental results and analysis on GPDS-150, GPDS-300, GPDS-1000, GPDS-2000, and GPDS-5000 datasets demonstrate that the proposed method can discriminate the genuine signatures and their corresponding skilled forgeries well and achieve state-of-the-art performance on these datasets.

Thesis Supervisor: Seiichi Uchida
Title: Professor

Acknowledgments

Looking back at these past three years of my Ph.D., it looks like a memorable journey with many challenges, uncertainties, opportunities, pressures, and happiness. From this journey, I learned a lot to handle various problems both in study and life. I am sincerely grateful to many people and organizations for all their support to help me pass through this journey.

At first, I would like to thank my supervisor, Prof. Seiichi Uchida, for his valuable guidance, constructive feedback, and comments from the very beginning till the completion of this Ph.D. Actually, he acts as many roles in my Ph.D. life. Sometimes, he likes a father, and always takes care of my life in Japan. Sometimes, he like a good friend. We always talk happily and I can tell my worry both in life and research to him. He is also an excellent colleague for me. We discuss many academic and technical problems periodically and share some novel and valuable ideas during each seminar and meeting. During the past three years, working and studying with Prof. Uchida is a wonderful experience. His wide knowledge, strong research enthusiasm, and hard-working attitude have inspired me during my life and study and will have a profound effect on my academic career forever. I still remember when I first met him with his kind guidance, help. I am very lucky that I have such a responsible, excellent, and nice supervisor. Without his support, guidance, patience, and encouragement, I could not imagine how I can finish my Ph.D. research. I would like to thanks my honorable supervisor, Prof. Seiichi Uchida again for his contribution during my Ph.D. life.

Next, I would like to thank my predecessor and friend, Associate Prof. Brian Kenji Iwana and his wife, Mrs. Yirong Zhao for their kindly care during these three years. When I joined our laboratory, Brian always likes an elder brother for me. He gave me many useful suggestions and guidance both in my life and research. He helps me schedule my Ph.D. life and reminds me of what should I do at the right time and places. When I encounter some difficulties, he always helps me to deal with them

and comforts me when I am feeling upset. In addition, Brian is an excellent colleague and partner during my research life. He inspired me a lot and always helps me revise my paper both in technique and language. I would like to say that he likes another supervisor during my Ph.D. research.

Then, I would like to thank my laboratory, Human Interface Laboratory at Kyushu University. Our laboratory likes a family for me. Each member of our lab creates a wonderful memory that makes me so happy as a member of our lab. I would like to thank Associate Prof. Ryoma Bise, Assistant Prof. Daiki Suehiro and Hideaki Hayashi, Prof. Wataru Ohyama (Now, he is in Intelligent Media Processing Laboratory, Saitama Institute of Technology), Secretary Mrs. Mika Maeda and Maki Fukutomi, and All Doctor, Master, Bachelor students in our lab. In this big family, I can intently focus on my own research without any trouble and worry. I really appreciate all the people who help me in our lab and I take our lab as my home forever.

Also, I would like to thank my parents who always accompany me and stand by me. I am so lucky that I was born into an open-minded family. My parents always support my decision unconditionally. During these three years, I can not take care of them because I need to complete my Ph.D. in a foreign country. However, they never complain about that. They are the greatest parents in the world. I also want to thank my friends, especially the Chinese friends in Fukuoka, such as Yan Zheng and Xiaotong Ji (they are my juniors in my lab), Kang Zhang and Wenbo Li (they are my roommates), Xueru Zhang and Wanju Wei (they are my neighborhoods). Thanks for their support and help.

Finally, I would like to thank Kyushu University and CSC (China Scholarship Council) for sponsoring my Ph.D. with a fully-funded scholarship. We know this year is very special because of the spread of COVID-19. The situation must go well after the efforts of all the people. Wish the world peace, economic go up, people live a happy life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Convolutional Neural Networks (CNNs)

In recent years, Convolutional Neural Networks (CNNs) have had excellent performance in the fields of image recognition and detection [2, 3, 4, 5, 6], natural language processing [7, 8, 9, 10], document analysis and recognition [11, 12, 13, 14, 15]. Many popular CNN-based architectures have been proposed in recent years, such as AlexNet [16], GoogleNet [17], Fast-RCNN [18], ResNet [19], and so on [20, 21, 22], which demonstrates the effectiveness of CNNs for addressing different real-world problems.

Typically, a CNN is composed of several modules. In the first few stages, it includes several convolutional layers, activation functions, and pooling layers. The role of the convolutional layer is to detect local conjunctions of features from the previous layer. After a convolutional layer, the features will pass through an activation function to obtain the nonlinear representations. Then, the pooling layer is applied to merge semantically similar features into one. After stacking several convolutional layers, activation functions, and pooling layers, fully-connected layers are introduced

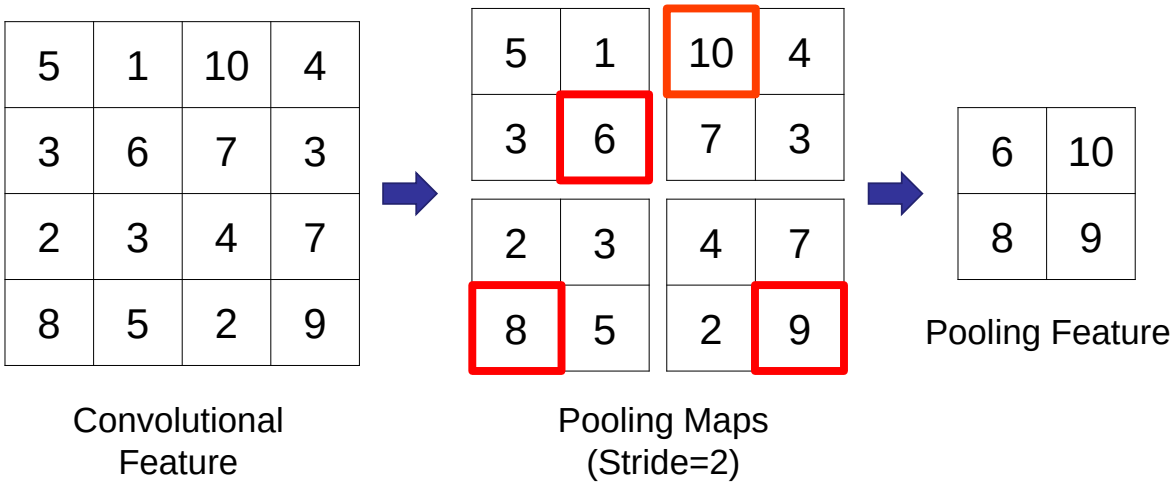

Figure 1-1: The procedure of doing a max-pooling operation on a convolutional feature. The pooling size is 2×2 with stride 2. The red blocks represent the maximums in pooling windows.

for different tasks.

1.1.2 The Roles of Max-pooling

The max-pooling operation in CNNs is a very essential module that combines the maximal responses of the feature maps into a summarized joint distribution of the features over some region of interest [23, 24]. The objective of max-pooling in CNNs is to reduce the size of the parameter space by removing redundant information while preserving the relevant responses from the convolutional feature maps. Fig. 1-1 shows the procedure of doing a max-pooling operation.

The first role of the max-pooling operation is information gathering. After several max-pooling layers, the size of the original images gradually reduces by preserving the maximal responses. The maximums in pooling windows represent important information that is kept by doing the max-pooling operation. In other words, the max-pooling operation merges semantically similar features into one, which is similar to dimensionality reduction.

The second role of the max-pooling operation is deformation compensation. The

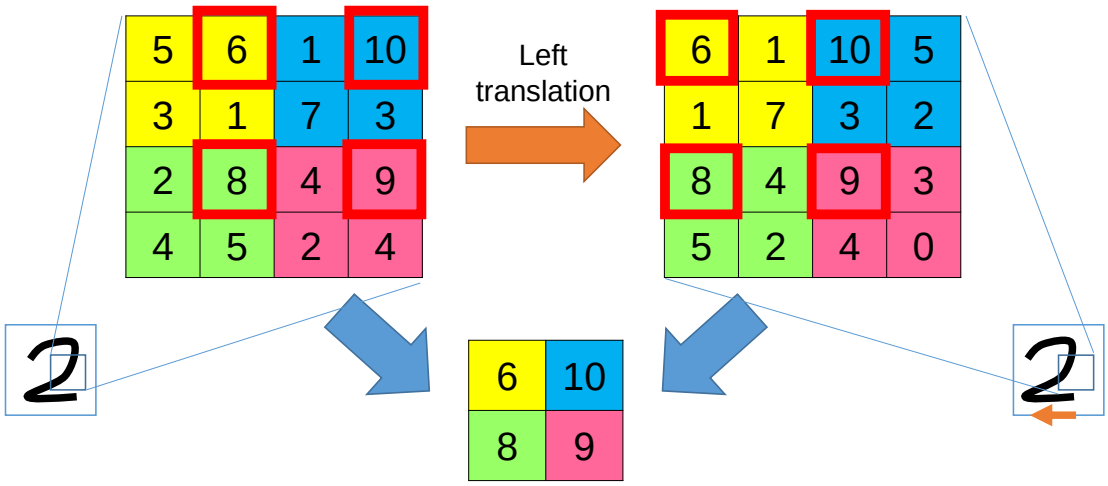

Figure 1-2: An example that the max-pooling operation helps compress information into a lower dimensional representation and absorb the left translation.

max-pooling operation helps CNNs be somewhat spatially invariant to the position of features [25]. In other words, if the original images have geometric deformations (such as translation or scaling), the features become insensitive to the geometric deformations in the original image because of the max-pooling operation. Fig. 1-2 presents an example that the max-pooling operation helps compress information into a lower-dimensional representation and absorb the left translation. In Fig. 1-2, the "intra-class" micro differences are eliminated between two samples. Before the max-pooling operation, the features are different because the strokes are in different positions, which may cause misclassification problems. After the max-pooling operation, the left translation is absorbed by preserving the maximums in pooling windows.

1.1.3 The Negative Effects of Max-pooling

However, because of the deformation compensation ability of the max-pooling operation, it also introduces several negative effects. The important micro differences are eliminated by preserving only maximal responses. The traditional max-pooling operation does not care about the importance of the difference. Therefore, the impor-

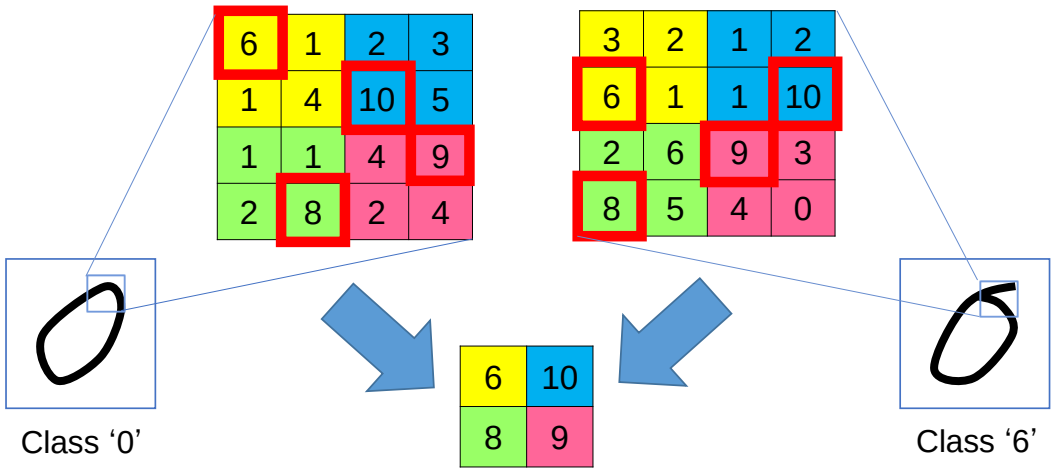

Figure 1-3: An example of the negative effect of the traditional max-pooling operation. After using the max-pooling operation, the features of the samples '0' and '6' are the same. Here, the pooling size is 2×2 with stride 2. The red blocks represent the maximums of each pooling window.

tant "inter-class" micro differences are eliminated along with unnecessary "intra-class" differences (i.e., deformations).

Fig. 1-3 shows that important inter-class micro differences are eliminated by the max-pooling operation. After the max-pooling operation, the same pooled features of samples '0' and '6' are obtained even the convolutional features are different. Then, these two samples might be classified in the same class. In fact, the max-pooling operation will absorb the important micro differences which often contribute to discriminative features.

Fig. 1-4 shows other character pairs that have important inter-class micro differences on MNIST, Chars74K-font, and HASY datasets. We can see that these samples are very similar except for some micro differences. After several downsampling operations, these micro differences might be absorbed by the max-pooling operation. Then, they might be classified in the same class. Therefore, sensing the risk of eliminating important micro differences between the different classes from the max-pooling operation is necessary for text recognition tasks.

The elimination of intra-class micro differences is more and more serious for i-

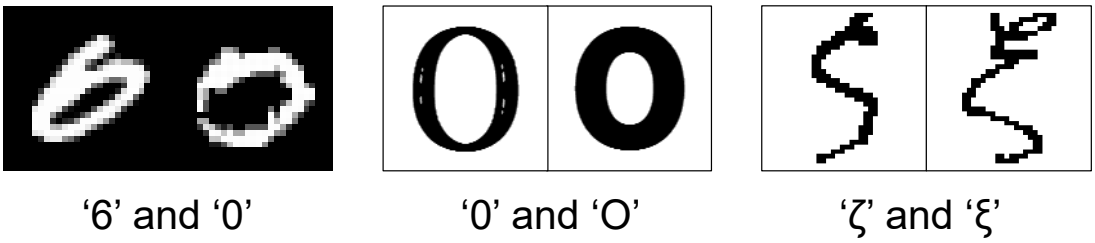

Figure 1-4: The similar samples from different classes in MNIST, Chars74K-font, and HASY datasets.

dentification tasks. Different from the recognition tasks, identification tasks need to be careful of “very micro” differences. Most behaviors of the forged samples are very similar to the genuine samples except for some micro differences. Therefore, discriminating theses micro differences between genuine samples and practiced forged samples plays an important role in identification tasks.

In this sense, bio-metrics scenarios, such as signature verification, will suffer from the unnecessary elimination of inter-individual (intra-class) differences. For example, skilled forgeries in signature verification try to mimic the genuine signatures, and therefore the intra-class differences are often tiny. Specifically, these intra-class micro differences between the genuine signatures and skilled forgeries can be described as small translations, transformations, or distortions of strokes, scaling in local regions of signatures, and special writing habits of different signers which are very important cues for signature verification tasks.

Fig. 1-5 shows examples of different micro differences that exist between genuine signatures and skilled forgeries. The left sample in Fig. 1-5 shows a vertical translation in the part of ‘F’ between the genuine signature and skilled forgery. The middle sample shows a scaling problem in the part of ‘A’. The right sample shows a specific writing habit of different signers. A tail has occurred at the bottom of ‘E’ in the skilled forgery, but it does not occur in the genuine signature.

Although these micro differences are very general between genuine signatures and their corresponding skilled forgeries in offline signature verification systems, they are

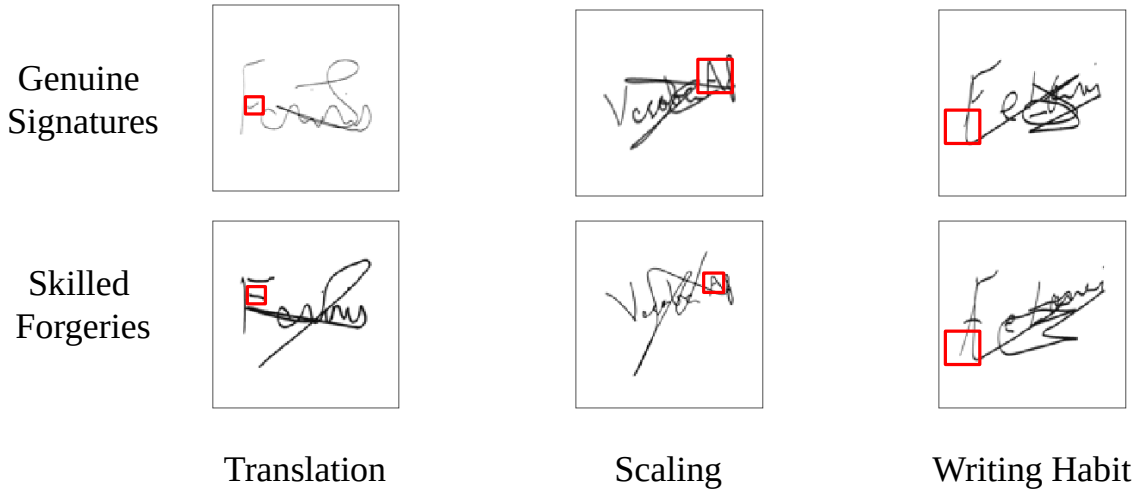

Figure 1-5: Examples of some specific "micro differences" occurred between genuine signatures and skilled forgeries. The red blocks represent the part with micro differences between genuine signatures and skilled forgeries.

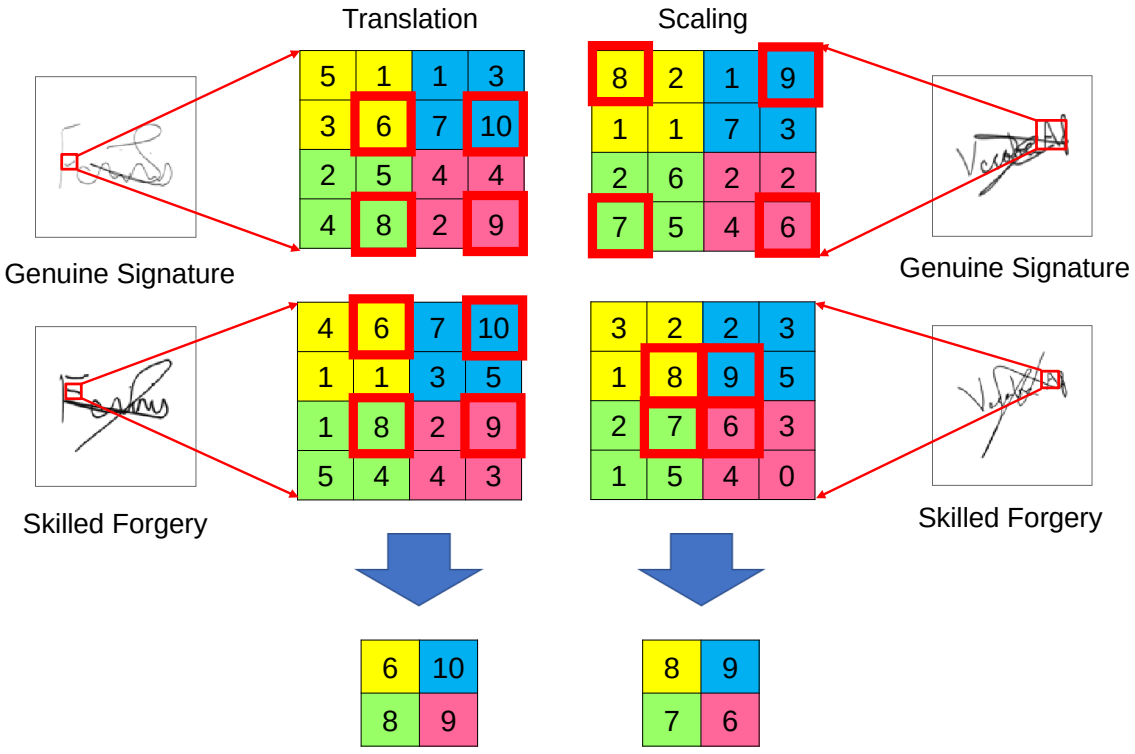

Figure 1-6: Examples of translation and scaling cases that are absorbed by the max-pooling operation between the genuine signatures and their corresponding skilled forgeries.

unexpectedly absorbed by the traditional max-pooling operation. Fig. 1-6 shows the translation and scaling cases that are absorbed by the traditional max-pooling operation between the genuine signatures and their corresponding skilled forgeries. This means that the traditional max-pooling operation degrades the performance of the verification systems by the elimination of the micro differences between the genuine signatures and their corresponding skilled forgeries.

1.1.4 Motivation

The motivation of this research is that how can we detect and penalize the cases when the max-pooling operation is going to remove the important inter-class and intra-class micro differences? In the traditional max-pooling operation, it wastes the spatial information that may represent the crucial cues of these important inter-class and intra-class micro differences. Due to the traditional max-pooling operation that could compensate for geometric deformations by preserving the maximal response in a pooling window, we want to keep this property and discriminate unnecessary absorptions from necessary absorptions by watching the behaviors of the max-pooling operation.

1.2 Thesis Objectives and Solutions

1.2.1 Avoiding the Negative Effects by Displacement Features

The key idea of avoiding the negative effects of the traditional max-pooling operation is to know "how" the max value is selected in a pooling window of the max-pooling operation. If we utilize the trends of the maximum selection for each class, we can understand the intra-class differences. This means that by observing the maximum selection, we can understand that the max-pooling operation is going to eliminate the inter-class or intra-class micro differences.

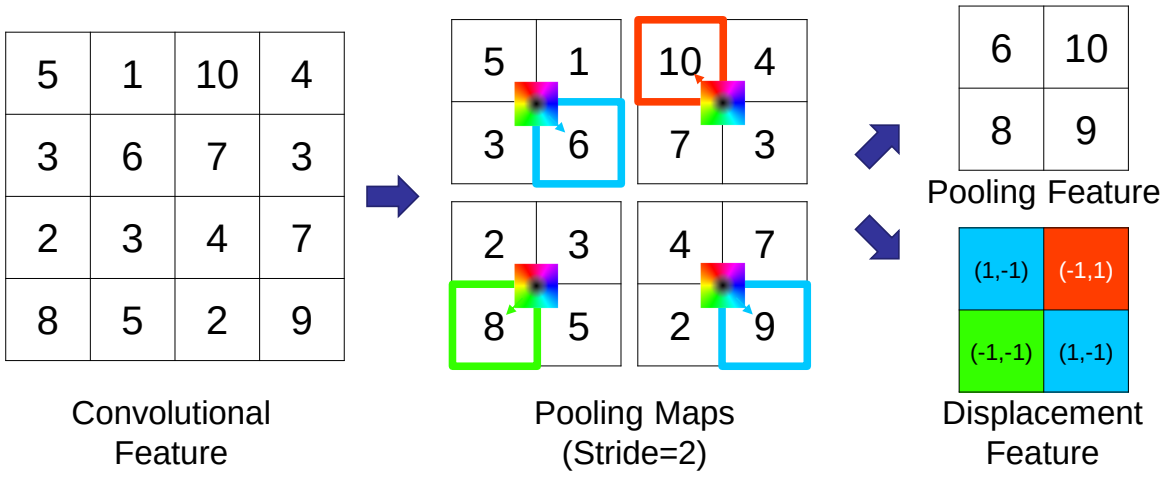

Figure 1-7: Extracting displacement features from a 2×2 pooling window.

To address this issue, memorizing the position of the maximum and utilizing it is one solution. In the traditional max-pooling operation, this spatial information may represent the inter-class or intra-class micro differences and the crucial cues that how the micro differences are eliminated. We believe that this phenomenon is not random. There must be inter-class or intra-class trends in this spatial information lost in the traditional max-pooling operation.

Fig. 1-7 shows how to extract this spatial information (displacement features) from the max-pooling operation. Here, the arrows with different colors represent the displacements of the center of the pooling windows. 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 describe the distance and direction of the selected maximal value from the center point in the pooling windows. Finally, We use a two-dimensional vector to represent the displacement features in horizontal and vertical directions.

Fig. 1-8 presents the pooling features and displacement features of samples from the HASY dataset [1] based on a Hue-Saturation-Value (HSV) color model whose color and intensity denote the direction and average length of the displacement features.

Different from the pooling features, the displacement features reflect the information from the spatial level. We can see that the displacement features reflect the class-wise trends between the inter-class samples. For example, in the first filter, the displacement features can capture the blue and green directions from the top of the sample '\pi (π)', but the red direction from the top of the sample '3'. In the second filter, the displacement features can capture the red direction from the bottom of the sample '\triangleq ($\triangleq$)', but the green direction from the bottom of the sample '3'. Theses examples give very clear class-wise trends captured by the displacement features to discriminate against the inter-class samples.

1.2.2 Utilization of Displacement Features in Text Recognition Tasks

For the text recognition tasks, to address the problem that the traditional max-pooling operation absorbs the inter-class micro differences between different samples, we explore two approaches to combine the pooling features and the displacement features. The idea is to compensate for the spatial information that is lost in the traditional max-pooling operation by only preserving the maximums in pooling windows.

In the first approach, we extract the displacement features from a pre-trained CNN. Then, we put the displacement features into another CNN and combine them with the pre-trained CNN in a fully connected layer. The purpose of this approach is to add the advantages of the displacement features and keep the good properties of the pooling features together.

The second approach is to transform the displacement features into cosine features based on Principal Component Analysis (PCA). First, we extract the displacement features from a pre-trained CNN. Then, for each independent class, we apply a PCA model to the displacement features that from the samples in this specific class. Finally, we combine the cosine features with the pooling features in the fully connected layers

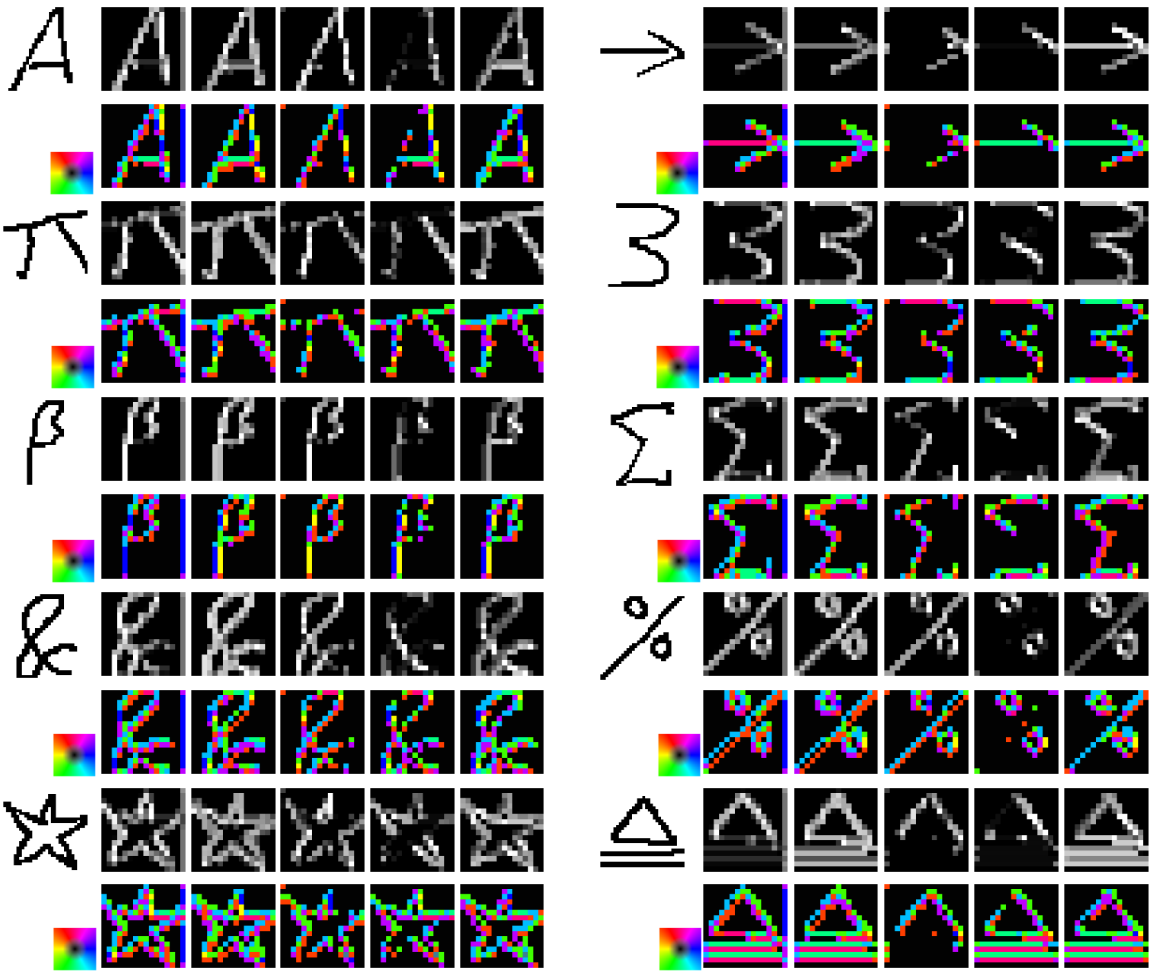

Figure 1-8: Visualization of the pooling features and displacement features on the HASY dataset [1]. Here, the pooling size is 2×2, the first row represents the original image and corresponding pooling features, the second row represents the displacement features, and each column represents one convolutional filter. The visualization of displacement features is based on an HSV color model whose color and intensity denote the direction and average length of displacement feature.

for different text recognition tasks. The purpose of this approach is that we hope PCA can catch only the trends of intra-class micro-differences captured by the displacement features and reduce the redundancy of the displacement features.

Furthermore, to discover the class-wise trends of the displacement features, we analyze the behaviors of the displacement features with different discussions. First, we visualize the displacement features based on an HSV color model to observe the inter-class and intra-class 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 the same PCA models as before 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, displacement features, and the proposed method to observe which samples are improved or degraded by combining the displacement features.

1.2.3 Utilization of Displacement Features in Offline Signature Verification Tasks

For the offline signature verification tasks, to address the problem that the traditional max-pooling operation absorbs the intra-class micro differences between different samples, we propose a novel CNN-based architecture that applies the displacement features to capture the micro differences between the genuine signatures and their corresponding skilled forgeries [26, 27]. Here, we apply the same strategy to apply the displacement features as the first approach in the text recognition tasks. We does not apply the PCA-based cosine features because there are many users (classes) in this task. We take the displacement features as the key features to represent the micro difference between the genuine signatures and their corresponding skilled forgeries

and fuse it with the pooling features to combine the merits both in the displacement features and pooling features as a feature extraction procedure.

We describe the proposed verification system in two phases. In the feature extraction phase, we train a CNN between the genuine signatures and skilled forgeries on a large scale dataset, named GPDS-10000 [28] to capture the general behaviors of genuine signatures and skilled forgeries. Then, we extract the displacement features from the first convolutional layer and fuse it with pooling features in another CNN to capture the micro differences between the genuine signatures and skilled forgeries. After the CNN training process, we take the trained CNN architecture as a feature extractor to obtain the discriminative features from original signature images. Then, we apply linear Support Vector Machines (SVMs) and RBF kernel SVMs as the writer-dependent classifiers for each user to build a complete signature verification system and evaluate the learned features.

1.3 Novelty and Contributions

The novelty and contributions of this thesis are summarized as follows.

- The position coordinates (displacement features) of the maximums are extracted and presented from pooling windows in the max-pooling operation and exploring how to use the displacement features to capture the "inter-class" or "intra-class" micro differences between different samples in the max-pooling operation with a similar strategy.

- The class-wise trends of the max-pooling operation are mined by using displacement features, which includes analysis of the visualizations and distribution of the displacement features, and comparison of the similarity matrices and confusion matrices on displacement features. Through the analysis, the displacement features could capture some micro differences which are useful for some specific tasks.

- Exploring the methods of combining the pooling features and displacement features for text recognition tasks. The proposed method achieves state-of-the-art results on MNIST, HASY, and Chars74k-font datasets.

- Applying the proposed displacement features to capture micro differences between genuine signatures and their corresponding skilled forgeries for building offline signature verification systems. The extensive experimental results and analysis on GPDS-150, GPDS-300, GPDS-1000, GPDS-2000, and GPDS-5000 datasets demonstrate that the proposed method can achieve drastic improvement in the offline signature verification task.

1.4 Thesis Structure

This thesis is organized as follows.

- **Chapter 1**: This chapter gives the general introduction of this thesis. At first, It describes the background and the problems that are existed in the traditional max-pooling operation. Then, it introduces the objectives, solutions, and two real-world applications (text recognition and offline signature verification tasks) that apply the proposed displacement features to capture the "inter-class" or "intra-class" micro differences in different samples. Finally, it summarizes the novelty and contribution of this thesis.

- **Chapter 2**: This chapter gives an overview of related work. First, it introduces some pooling based methods in modern deep neural networks and discusses the merits and drawbacks of different pooling methods. Then, it summarizes some CNN and max-pooling based models in the field of text recognition and compares them with the proposed method. Finally, it summarizes some handcrafted feature extractors, deep learning-based feature extractors, and novel offline signature verification systems and discusses their advantages and disadvantages

compared with the proposed verification system.

- **Chapter 3**: This chapter defines the proposed displacement features in detail. First, it introduces how to extract the displacement features and pooling features simultaneously from the max-pooling operation in a pre-trained CNN-based architecture. Then, it introduces how to address the multiple maximal values in a pooling window.

- **Chapter 4**: Since the inter-class micro differences are very common in text recognition tasks, this chapter introduces how to apply the proposed displacement features to capture the inter-class behaviors of the max-pooling operation in text recognition tasks. First, it introduces how to combine the displacement features and pooling features in a CNN-based architecture. Then, to further improve the recognition performance, it presents how to transfer the displacement features into consine features in PCA subspaces. Finally, it presents the experiment results on MNIST, HASY, and Chars74k-font datasets and designs a series of analysis to discover the class-wise trends of the displacement features.

- **Chapter 5**: Since the forged signatures are obtained by practiced imitators, the differences between the genuine signatures and skilled forgeries are very tiny. This chapter introduces a similar strategy with the text recognition tasks to apply the displacement features to capture the micro differences (intra-class behaviors) between the genuine signatures and their corresponding skilled forgeries for offline signature verification systems. First, it introduces how to train a CNN between genuine signatures and skilled forgeries and extract the displacement features from this trained CNN. Next, it presents how to train a CNN-based feature extractor by using the displacement features. Then, it introduces the procedure of training the writer-dependent classifiers for building a complete verification system. Finally, it presents the verification results on GPDS-150, GPDS-300, GPDS-1000, GPDS-2000 and GPDS-5000 datasets.

- **Chapter 6**: This chapter summarizes the conclusion, followed by reflections and future recommendations as well as concluding remarks of this thesis.

Chapter 2

Related Work

CNN, as one of the most popular deep learning-based architectures, achieves great success in text recognition [29, 30, 31, 32, 33] and offline signature verification [34, 35, 36, 37, 38] tasks. A pooling operation after a convolutional layer is very common to build deep architectures. In recent years, many researchers focus on different pooling operations to improve the performance of deep learning-based architectures [39, 40, 41]. The pooling operations can reduce the dimension of input features and absorb some structural deformations from small shifts and distortions [24], which makes the deep learning models more efficient.

2.1 Pooling Operations in Deep Learning Models

2.1.1 Improved Traditional Pooling Operations

To address some problems in conventional pooling operations, many researchers focus on extending and improving the max-pooling or average pooling [42, 43, 44, 45, 46]. In [47], Zhai *et al.* proposed S3Pool that extends the standard max-pooling operation by decomposing max-pooling into two steps: max-pooling with stride one and a non-deterministic spatial downsampling step by randomly sampling rows and columns from a convolutional feature map. They observed that this general stochasticity acts

as a strong regularizer, and can also be seen as doing implicit data augmentation by introducing distortions and deformations in the convolutional feature maps.

To handle the problems that the MP2-pooling (2×2 max-pooling operation) reduces the size of the hidden layers so quickly and the disjoint nature of the pooling regions can limit generalization, Graham [42] proposed Fractional Max-Pooling (FMP) to reduce the spatial size of the image by a factor of α with $1 < \alpha < 2$. FMP introduces a degree of randomness to the pooling process, which is very useful for overcoming the overfitting in the traditional CNNs.

To regularize CNN-based architectures, Yu *et al.* [43] proposed mixed pooling that was inspired by the random Dropout [48] and DropConnect [49] methods. Similarly, Wei *et al.* [44] proposed an intermediate form between max and average pooling called Polynomial pooling (P-pooling) to provide an optimally balanced and self-adjusted pooling strategy for semantic segmentation. Although these researches improve the traditional pooling operations for different tasks, the spatial information lost in the max-pooling operation is not considered to capture the micro differences between different samples.

2.1.2 Novel Pooling Operations

Considering the limitation of the traditional pooling methods, many novel pooling operations or layers are proposed to address the problems of the traditional pooling methods in some specific applications such as image detection and classification [50, 51, 52, 53, 54], handwriting and text recognition [55, 42, 56, 57, 58], semantic segmentation [59, 60, 61, 44, 62], and other challenging computer vision tasks [63, 64, 65, 66, 67]. In [50], He *et al.* introduced a Spatial Pyramid Pooling (SPP) layer to remove the fixed-size constraint of the network, which is robust to object deformation. In [52], Kobayashi proposed a novel trainable local pooling function guided by the global features beyond the local ones. The parameterized pooling form is derived from a probabilistic perspective to flexibly represent various types of

pooling and then the parameters are estimated by using statistics in the input feature map.

More recently, Gao *et al.* proposed Local Importance-based Pooling (LIP) that can automatically enhance discriminative features during the downsampling procedure by learning adaptive importance weights based on inputs [53]. LIP addressed the problem that the traditional downsample layers might prevent discriminative details from being well preserved, which is crucial for recognition and detection tasks. Although these novel pooling methods enhance the performance of the CNN-based architectures in different applications, to compensate for the spatial information lost in the traditional pooling operations is not considered.

2.1.3 Spatial Information in Pooling Operations

A few studies also focus on the position information and especially the displacement information in pooling operation [68, 69, 70, 71, 72]. Qian *et al.* proposed the Max-Pooling Positions (MPPs) as an effective discriminative feature to predict category labels for traffic sign recognition [68]. For example, in a 2×2 pooling window, they defined a quaternary encoding, '1000, 0100, 0010, 0001', where the '1' represents the position of the 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.

To further explore the spatial information in the max-pooling operation, Zhao *et al.* proposed a novel architecture, the Stacked What-Where Auto-Encoders (SWWAE) which represents the position of maximum in a pooling window [69]. The position information in this paper is similar to our work. However, it used this information just for an "unpooling" operation, but our work combines the displacement features with pooling features for address the problem that the micro differences are absorbed by the max-pooling operation. In [70], Sun *et al.* weighted sum bilinear pooling that considers the spatial location of the convolutional features. In this research,

the authors applied the spatial location as the weight to multiply the original features. Then, the average pooling operation is used to reduce redundancy. Compared with our method, we combine the spatial location with the pooling features in one framework, but this research separately obtained these two features.

By just preserving the maximum value of each window, the max-pooling operation 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" [73], 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" [74], which adds an offset to each bin position in the regular bin partition of the previous RoI pooling [18, 75]. Compared to the 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.

Compared to the existing pooling operations, the proposed method in this thesis considers spatial information lost in the traditional max-pooling operation. This spatial information can capture micro differences between different samples. Although this is the reverse of the ordinary usage of the max-pooling for absorbing the micro differences, it plays an important role in text recognition and offline signature verification task.

2.2 Literature Review for Text Recognition

Text recognition is an active research area that attempts to develop a computer application with the ability to automatically read the text from images [76]. The text recognition systems are very common in our daily life, such as financial purposes [77, 78, 79], transportation [80, 81, 82], healthcare [83, 84, 85], etc. Therefore, to build a robust and safe text recognition systems is very important for specific real-world applications [86, 87, 88].

2.2.1 CNN-based Models for Text Recognition

In the field of text recognition, compared to many traditional feature learning techniques [89, 90, 91, 92, 93], CNN based models have shown a powerful performance in different challenging tasks, such as handwriting recognition [94, 95, 96, 97, 98], scene text recognition and detection [99, 100, 101, 102, 103], script recognition [104, 105, 106, 107, 108], etc. Generally, the CNN-based architectures are often applied to build end-to-end recognition systems or feature extractors for extracting the discriminative features in different applications.

In [29], 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 [109], 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). In [13], Wang *et al.* proposed a novel writer-aware CNN based on parsimonious HMM (WCNN-PHMM) to address the large vocabulary and the diversity of writing styles in offline Handwritten Chinese Text Recognition (HCTR) tasks. This method is the first study of writer adaptation for offline HCTR.

2.2.2 Pooling Operations in Text Recognition

Recently, Many pooling based architectures are proposed for different text recognition tasks [110, 111, 95, 55, 112]. In [110], Gao *et al.* proposed a graph Pooling (gPool) layer in Graph Convolutional Networks (GCN), which applies a trainable projection vector to measure the importance of nodes in graphs. By selecting the most k important nodes to form the new graph, gPool achieves the same objective as regular max-pooling layers operation on images and texts. In [111], Zhang *et al.* proposed the Deep Contextual Stroke Pooling (DCSP) for scene text recognition. The proposed DCSP discovers the most prominent stroke information by using stroke detectors and captures the spatial context of discriminative strokes by learning contextual factors. In [95], Xiao *et al.* applied global pooling for building very compact online Handwritten Chinese Character Recognition (HCCR) systems. In this research, They also proposed DropWeight for pruning redundant connections in the CNN architecture to improve the performance.

More recently, Kim *et al.* claimed that pooling operations lose information regarding spatial relationships and are likely to misclassify objects based on their orientation or proportion [113]. Then, They applied the capsule network for the text recognition task. The motivation for this research is very similar to this thesis. But, we use spatial information as the new features to compensate for the max-pooling operation, which is the main difference between these two researches.

2.3 Literature Review for Offline Signature Verification

In the field of offline signature verification, it aims to verify whether a signature image is written by a genuine writer or a skilled forger [114, 115, 116]. Generally, signature verification systems are divided into two categories: online and offline [117, 118,

119, 120]. For online systems, the data is collected as a sequence which includes the positions of the pen, pressure coordinate sequence, pen elevation coordinate sequence, etc [121]. For offline systems, the data is collected from digital images. Since the dynamic information is not available in offline signature verification systems, the task becomes very challenging. In addition, the forgeries can be deliberately imitated by practiced persons, which also increases the difficulty for the verification systems.

2.3.1 Handcrafted Features for Offline Signature Verification

Traditional offline signature verification systems often use different handcrafted features to train the writer-dependent or writer-independent classifiers, such as geometrical features [122], Local Binary Patterns (LBP) features [123], Scale Invariant Feature Transform (SIFT) features [124], and Histogram of Oriented Gradients (HOG) features [125], etc. There are many types of researches that focus on designing robust features [126, 127, 128, 129, 130] to build signature verification systems.

In [128], Okawa proposed a feature extraction method based on a Fisher Vector (FV) with fused "KAZE" features from both foreground and background signature images. The "KAZE" features consider the structures between strokes and stroke contour information more effectively. In [127], Zois *et al.* proposed the post-oriented grid features which encode the geometric structure of the signatures by grid templates. However, using the handcrafted features is hard to discriminate the genuine signatures and the corresponding skilled forgeries and often need to set different parameters for specific tasks, which is hard to apply to other verification systems and large scale applications. To address this issue, designing an appropriate feature extractor to automatically learn the discriminative information between the genuine signatures and skilled forgeries is very important for offline signature verification systems.

2.3.2 Deep Learning-based Features for Offline Signature Verification

In recent years, the deep learning-based models have widely applied to online and offline signature verification systems [131, 132, 133, 134, 135], which demonstrates the powerful performance of the deep learning-based architectures for extracting the discriminative features for different signatures. In the field of offline signature verification, some deep learning-based features are proposed to capture the behaviors of different signatures and build the complete verification systems [131, 136, 137, 138, 34].

Due to the limitation of the handcrafted features, many deep learning-based feature extractors are proposed in recent years [131, 136, 137, 132]. In [137], Zhang *et al.* proposed an unsupervised feature for offline signature verification based on Deep Convolutional Generative Adversarial Networks (DCGANs), which has a robust generalization ability compared to handcrafted features. In [131] and [136], Hafemann *et al.* proposed a CNN based feature extraction approach, named "Signet" to obtain the discriminative features not only between the genuine signatures and skilled forgeries but also between the different users. However, the "Signet" cannot capture the micro differences between the genuine signatures and corresponding skilled forgeries and only trained on 531 different users cannot apply to large scale verification tasks.

2.3.3 Other Novel Offline Signature Verification Systems

In recent years, many novel offline signature verification systems are proposed, which promotes the development in this field [139, 140, 141, 142, 143]. In [139], Hafemann *et al.* characterized and evaluated adversarial examples for offline handwritten signature verification systems. This research based on two gradient-based attacks, the Fast Gradient Method (FGM) and the Carlini & Wagner attack (C&W), and two gradient-free attacks that can be used even if the features and/or classifiers are non-differentiable. This work evaluated the robustness and investigated the impact of

adversarial examples for genuine signatures. In [140], Gumusbas and Yildirim first aim to evaluate Capsule Network and compare the results with the CNN-based equivalent model under different input resolutions. They proved that the Capsule Network is much better than CNN-based architectures to learn representations to differentiate genuine signatures from skilled forgeries. Also, due to use the signatures with low resolutions could speed up the verification systems than use the signatures with high resolutions.

More recently, Hafemann proposed using meta-learning to solve the problems that a forger practice imitating the genuine signatures, and often can create forgeries visually close to the original signatures [141]. In particular, the meta-learner guides the adaptation (learning) of a classifier for each user, which is a very efficient operation that only requires genuine signatures. The meta-learning also learns what is common for the classification across different signers. Compared to the previous methods, the proposed method can train with a huge number of users to capture the micro differences or distortions between the genuine signatures and skilled forgeries, which is very useful for signature verification systems.

Chapter 3

Mining the Displacements of Max-pooling

In this chapter, we introduce how to extract the proposed displacement features from the max-pooling operation in detail. First, we present how to extract the pooling features and their corresponding position coordinates simultaneously from the max-pooling operation in a pre-trained CNN. Then, we introduce the displacement features based on the position information and how to transform the position information into the displacement features. Finally, we present how to address the problem if there is more than one maximal value in a pooling window.

3.1 Extracting Displacement Features from a Pooling Layer

In traditional CNNs, the max-pooling operation only obtains the maximum value from the pooling windows in the convolutional feature maps. Its result is a down-sampling of the convolutional feature maps which retain the maximum response. 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 from

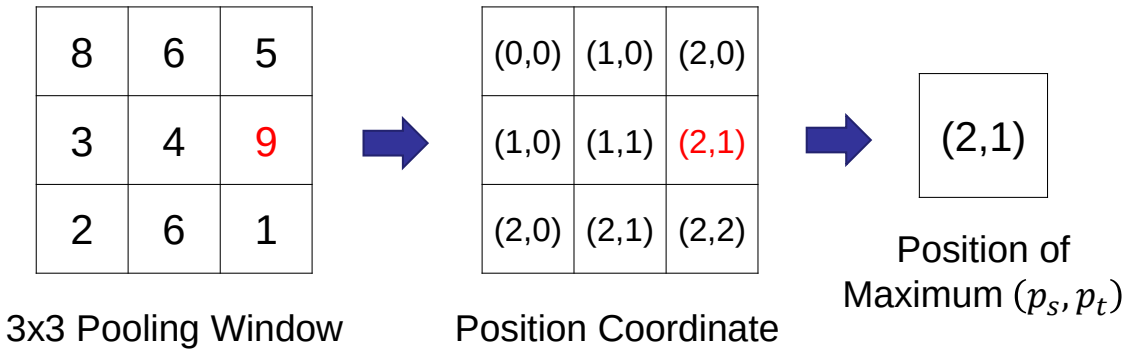

Figure 3-1: The position information in a 3×3 pooling window.

the original convolutional feature maps. In other words, we only know the specific maximal features in the feature maps, but we do not know where they came from.

In order to address this problem, we extract the pooling features and their corresponding position coordinates simultaneously. For any element in the pooling features, the corresponding position coordinate can be described as (p_s, p_t) as shown in Fig. 3-1,

$$p_s = \underset{s}{\operatorname{argmax}} \mathbf{F}^{n\times n}_{(s,t)}, \tag{3.1}$$

$$p_t = \underset{t}{\operatorname{argmax}} \mathbf{F}^{n\times n}_{(s,t)}. \tag{3.2}$$

Here, s and t are the horizontal and vertical coordinates, $\underset{s}{\operatorname{argmax}} \mathbf{F}^{n\times n}_{(s,t)}$ and $\underset{t}{\operatorname{argmax}} \mathbf{F}^{n\times n}_{(s,t)}$ represent the horizontal and vertical coordinates of the maximal value in a $n \times n$ pooling window $\mathbf{F}^{n\times n}$, respectively. The size of the position coordinates are same as the pooling features. If the pooling size is $n \times n$, p_s and $p_t \in \{0, 1, \cdots, n-1\}$.

The position information only represents the location of the maximum in pooling windows, it can not represent the displacements of maximum. To describe the displacement behavior of maximum, based on the position information, we transform it into the displacement features by the simple mathematical transformation. Since the pooling size can be set to even and odd, we calculate the displacement features in two different conditions.

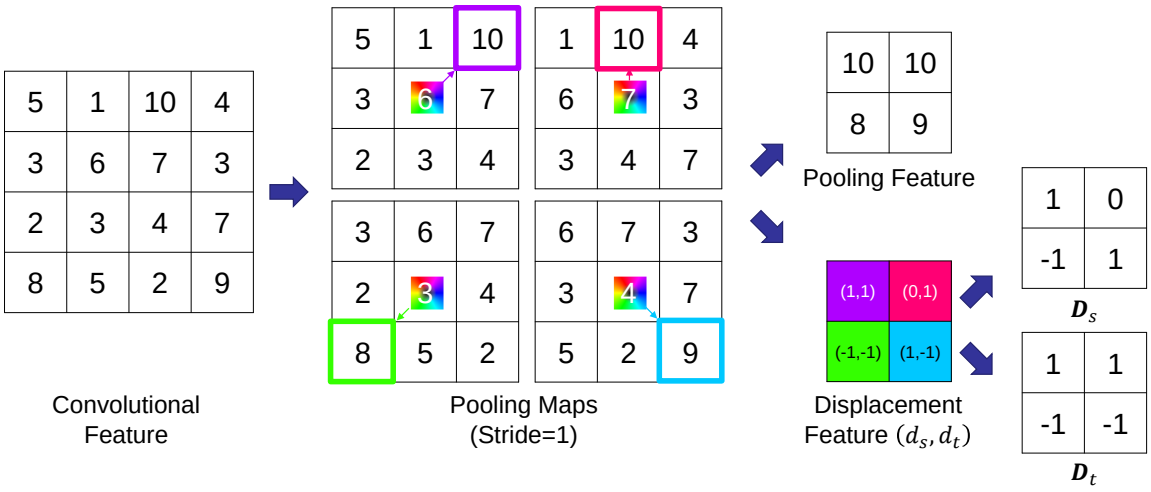

Figure 3-2: Extracting displacement features from a 3×3 pooling window.

3.1.1 The Case When n Is Odd

Fig. 3-2 shows a case of extracting the displacement features and pooling features when the size of the pooling window is odd. In the odd case, if the maximal value in the central row or column, the displacement would be zero in the horizontal or vertical directions. Except for this condition, all unites have displacements in both horizontal and vertical directions. Then, we transform the position coordinate (p_s, p_t) into a displacement feature (d_s, d_t),

$$d_s = p_t - \frac{n-1}{2}, \tag{3.3}$$

$$d_t = -p_s + \frac{n-1}{2}. \tag{3.4}$$

Finally, we describe the displacement features in two matrices, $\mathbf{D}_s$ and $\mathbf{D}_t$ which represent the displacements of the maximums in horizontal and vertical directions respectively. As shown in Fig. 3-2, the size of $\mathbf{D}_s$ or $\mathbf{D}_t$ is as same as the pooling features.

3.1.2 The Case When n Is Even

Fig. 1-7 shows a case of extracting the displacement features and pooling features when the size of the pooling window is even. In this case, all units have displacements in both horizontal and vertical directions. The displacement features (d_s, d_t) can be calculated as,

$$d_s = \begin{cases} p_t - \frac{n}{2}; & p_t < \frac{n}{2} \\ p_t + 1; & p_t \geq \frac{n}{2}, \end{cases} \tag{3.5}$$

$$d_t = \begin{cases} -p_s + \frac{n}{2}; & p_s < \frac{n}{2} \\ -p_s + \frac{n-2}{2}; & p_s \geq \frac{n}{2}. \end{cases} \tag{3.6}$$

Then, we also describe the displacement features in two matrices, $\mathbf{D}_s$ and $\mathbf{D}_t$. By using the proposed method, we can preserve the spatial information and the maximal response from a pooling window simultaneously. The advantage of this operation is that the pooling features absorb the geometric deformations and keep the most major features of the original convolutional features, and the displacement features keep the spatial information and capture the micro differences between different samples.

3.2 Multiple Maximal Values Condition

The previous section only discusses a single maximum in a pooling window. However, for some examples of extracting $\mathbf{D}_s$ and $\mathbf{D}_t$ from the displacement information, pooling windows may contain several maximums. Fig. 3-3 presents some cases with this problem. For an extreme example, if a pooling window contains a single background, all the units in the pooling window are the 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 a single value background.

To solve this problem, we first extract all the displacement features of the maximums in one pooling window. Then, we calculate the mean value of the displacement

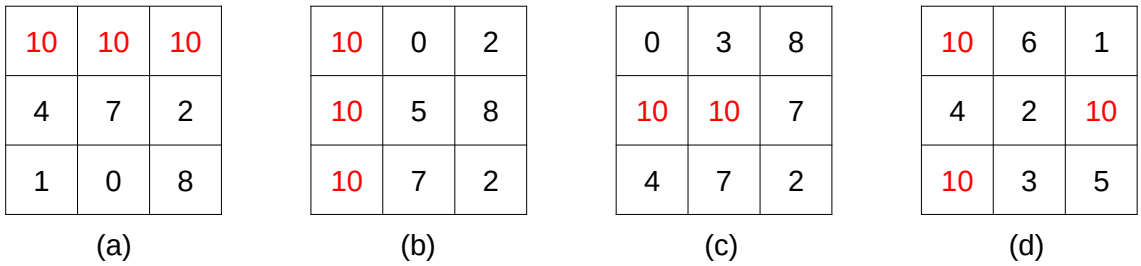

Figure 3-3: Different cases of multiple maximum. (a) the displacement features are (-1,1),(0,1),(1,1), so the mean displacement feature is (0,1). (b) Displacement features: (-1,1),(-1,0),(-1,-1). Mean displacement feature: (-1,0). (c) Displacement features: (-1,0),(0,0). Mean displacement feature: $(-\frac{1}{2},0)$.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

features as the final displacement features. The results would be displacement features with possibly non-integer values.

3.3 Summary

In this chapter, we introduce the proposed displacement features in detail. The displacement features are defined on the position coordinates of maximums in pooling windows, which could compensate for the spatial information lost in the traditional max-pooling operation of CNNs. To extract the displacement features, we first extract the maximum and its position in a pooling window at the same time. Then, we transform this position information into the displacement features to record the direction information of the maximum. To address the multiple maximal values in a pooling window, we calculate the mean value of the displacement features as the final displacement features. The next chapters will introduce how to apply the displacement features for some specific tasks to capture the micro differences between different samples.

Chapter 4

Experiment on Text Recognition

In this chapter, the displacement features are applied to capture the inter-class behaviors of the max-pooling operation in the text recognition tasks. As we discussed before, the inter-class micro differences are absorbed by the traditional max-pooling operation, which is very common in the text recognition tasks. However, the inter-class micro differences are very important and should not be ignored. The displacement features will capture them and thus we need to utilize them. The experiments designed in this chapter are to prove that the CNN-based architectures could capture these inter-class micro differences by the proposed displacement features.

To evaluate the effectiveness of the proposed displacement features, we design a series of experiments on several text recognition datasets. To utilize the displacement features, we introduce two different architectures to combine the displacement features with the pooling features to improve the performance of CNNs for the text recognition task. 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 based on PCA and combines the new cosine features with the pooling features in a similar way as the first architecture. Then, we use the MNIST dataset [1] for text

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

recognition and to discover the class-wise trends of the displacement features. Finally, we propose several frameworks that use the displacement features for text recognition on HASY [2] and Chars74K-font datasets [3].

4.1 Using the Displacement Features for Text Recognition

In this section, we introduce how to combine the displacement features with the pooling features to capture the inter-class micro differences between different samples and improve the performance of CNNs for the text recognition task. 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 as shown in Fig. 4-1. The second architecture transforms the displacement features into the new cosine features based on PCA and combines the new cosine features with the pooling features in a similar way with the first architecture as shown in Fig. 4-2.

4.1.1 Combining the Displacement Features with Pooling Features Based on CNNs

Since the size of the displacement features is the 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. To combine the displacement features in a CNN-based framework, we first 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

[2] http://zenodo.org/record/259444#.XB8r6PZuJPY
[3] http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/

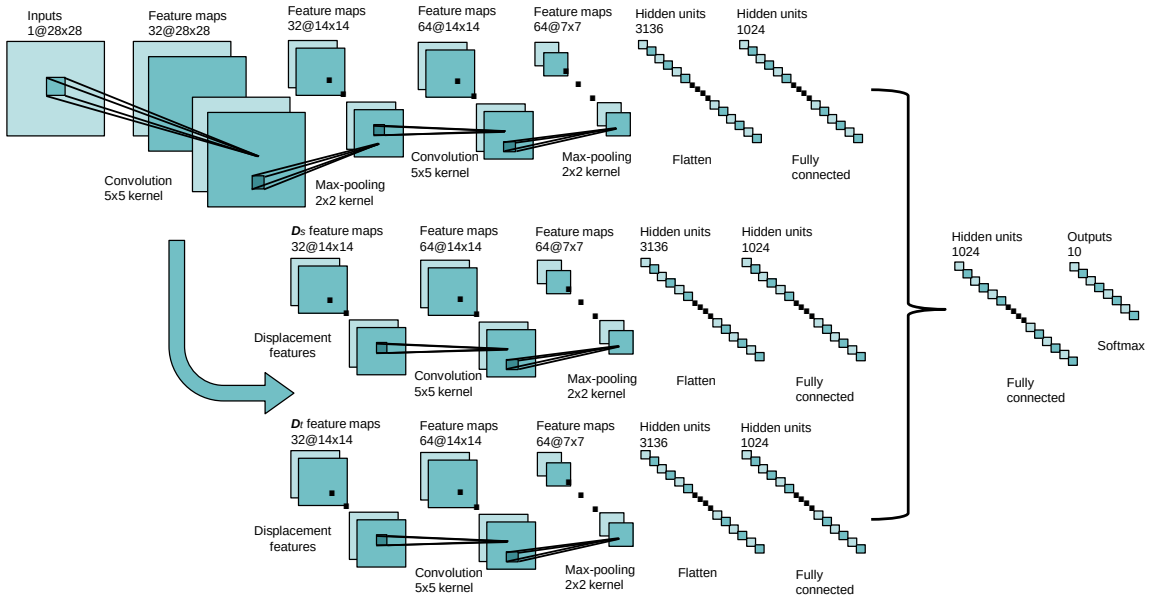

Figure 4-1: The architecture of the proposed method that combining the displacement features and pooling features in a CNN-based architecture. Here, $\mathbf{D}_s$ and $\mathbf{D}_t$ represent the horizontal and vertical directions of the displacement features.

final step is to fuse the pooling features and the new displacement features in a fully connected layer.

The structure of the network is illustrated in Fig. 4-1. This architecture contains two convolutional and pooling layers, three fully connected layers, and the final layer uses the softmax for classification. Here, we extract the displacement features from the first CNN (pre-trained). Then, we train the whole networks together and update the weights of whole architectures.

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

To further explore more information on 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

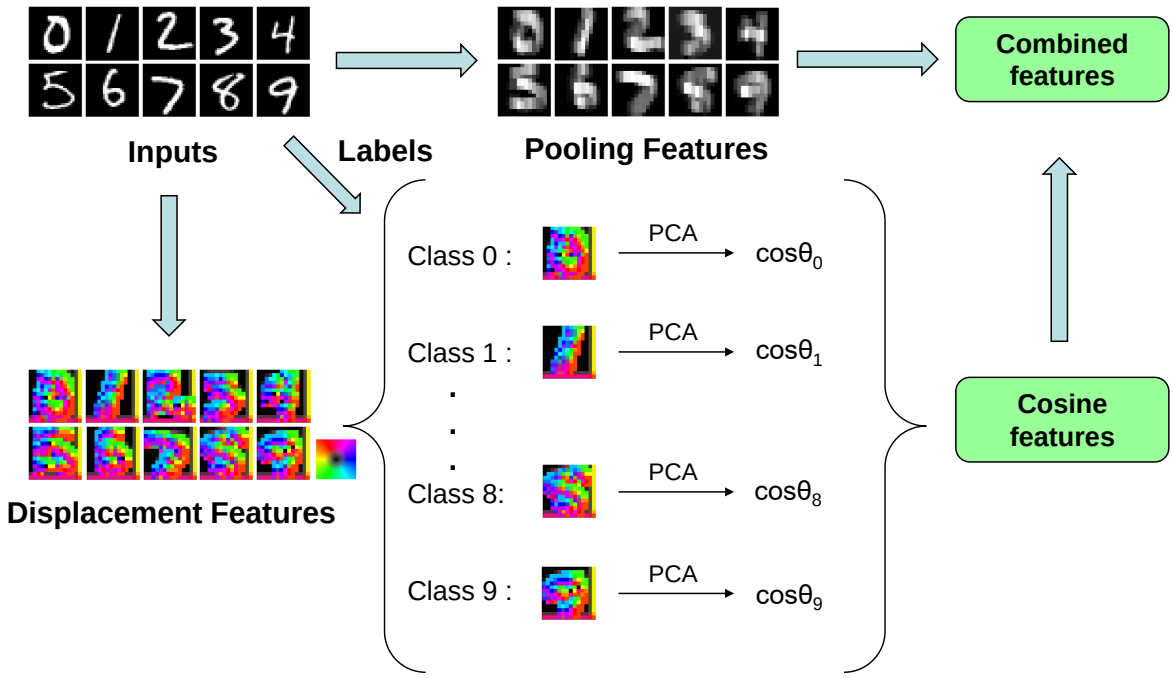

Figure 4-2: The training process of the cosine features based architecture.

possible. Due to the high dimensionality of the displacement features, PCA can reduce some redundancy and preserve the primary information of the displacement features. Besides, employing PCA can catch only the trends of the intra-class micro-differences captured by the displacement features, which is similar to have a penalty for the pooling features.

In order to apply PCA to the displacement features, we design the architecture as in Fig. 4-2. For the all 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,

$$\cos\theta = \frac{\mathbf{d} \cdot \sum_{i=1}^{k} \lambda_i \boldsymbol{\nu_i}}{|\mathbf{d}||\sum_{i=1}^{k} \lambda_i|}, \tag{4.1}$$

where $\mathbf{d}$ is the vector that concatenates the displacement features both in horizontal ($\mathbf{D}_s$) and vertical ($\mathbf{D}_t$) directions, λ_i is the i-th eigenvalue of $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_i, \ldots, \lambda_k)$, k is the number of largest eigenvalues that we want to use in PCA, and $\boldsymbol{\nu}_i$ is the i-th

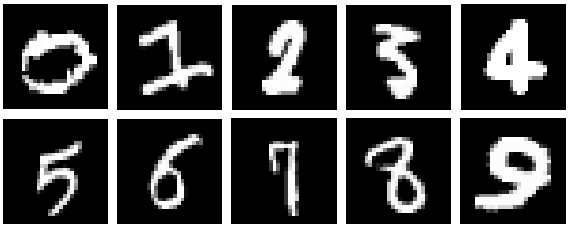

Figure 4-3: The Samples from MNIST dataset.

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 the same as one sample in training data, it should have a very similar $\cos\theta$. Combining (concatenating) the cosine features with pooling features in a fully connected layer is similar to have a penalty for the pooling features.

4.2 Classification on MNIST Dataset

To evaluate the proposed displacement features and cosine features, we design a series of experiments on the MNIST dataset. MNIST is an isolated handwritten digit dataset that 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 a size of 28×28 pixels. Fig. 4-3 shows some samples from the MNIST dataset.

In the first architecture, we used 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 belong to $[-1, 1]$. We used cross entropy as the loss function, and minimize it by Adam [144] with a 1×10^{-4} learning rate, the batch size, and the number of iterations are set to 50 and 20,000, respectively.

The second architecture is based on the first model. We computed the cosine features from the displacement features to further penalize the pooling features. We just used 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 is redundancy or noise in most cases. Then, we combined the cosine features with the pooling features in the fully connected layer.

The experimental results are shown in Table 4.1. From Table 4.1, we can see that with the size of the pooling windows increasing, the performance of the displacement features increases too. It is easy to understand because the larger pooling windows will provide more displacement information for the learning procedure. Using only the displacement features for classification can also obtain 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 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. 4-4 presents some improved samples (misclassifications by the only pooling features trial but correct classifications 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 downsample those features with max-pooling so much that the discriminating information is lost. Therefore, introducing the cosine features acts as a penalty for the final decision and introduces the position information of the local features. However, introducing the cosine features also brings the problem that the

Table 4.1: Classification accuracy on the MNIST dataset. Here, '**Pool**' represents the pooling features, '$\mathbf{D}_{s_1}$' and '$\mathbf{D}_{t_1}$' represent the displacement features in the horizontal and vertical directions from the first layer. '$\mathbf{D}_1$' and '$\mathbf{D}_2$' represent the displacement features from the first and second layers. '$\mathbf{Cos}_1$' and '$\mathbf{Cos}_2$' represent the cosine features from the first and second layers.

Pooling Window	$\mathbf{2 \times 2}$	$\mathbf{3 \times 3}$	$\mathbf{4 \times 4}$	$\mathbf{5 \times 5}$	$\mathbf{6 \times 6}$
Pool	0.9932	0.9928	0.9929	0.9912	0.9911
$\mathbf{D}_{s_1}$	0.9785	0.9843	0.9821	0.9895	0.9872
$\mathbf{D}_{t_1}$	0.9804	0.9835	0.9818	0.9889	0.9896
$\mathbf{D}_{s_2}$	0.9758	0.9768	0.9813	0.9823	0.9822
$\mathbf{D}_{t_2}$	0.9725	0.9753	0.9802	0.9811	0.9815
$\mathbf{Pool} + \mathbf{D}_1$	0.9932	0.9940	0.9934	0.9923	0.9920
$\mathbf{Pool} + \mathbf{D}_2$	0.9922	0.9920	0.9920	0.9902	0.9911
$\mathbf{Pool} + \mathbf{Cos}_1$	**0.9940**	**0.9943**	**0.9934**	**0.9930**	**0.9928**
$\mathbf{Pool} + \mathbf{Cos}_2$	0.9928	0.9930	0.9929	0.9908	0.9911
$\mathbf{Pool} + \mathbf{Cos}_1 + \mathbf{Cos}_2$	**0.9940**	0.9938	0.9932	**0.9930**	0.9925

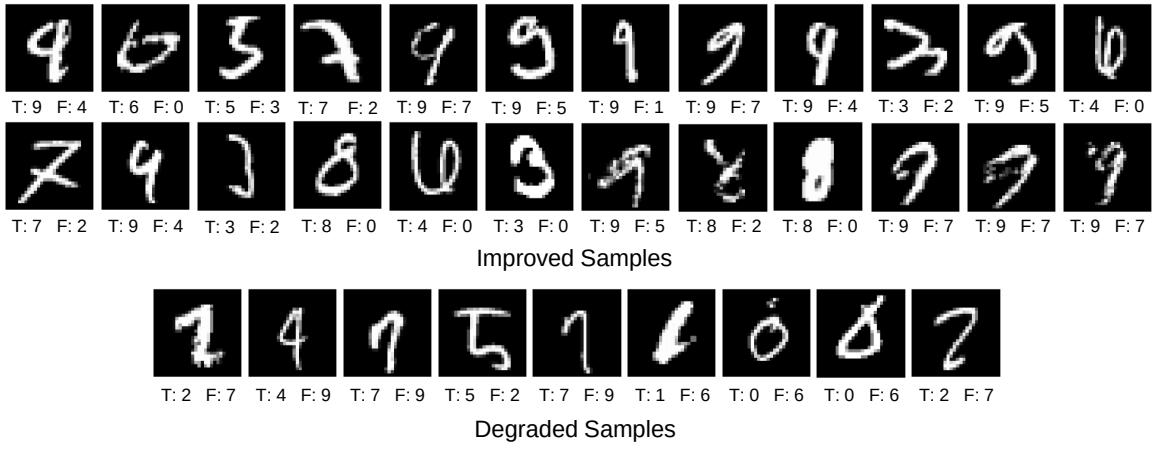

Figure 4-4: Example of improved and degraded samples by using the pooling+cosine features with 3×3 pooling size. Here, 'T' represents the true class and 'F' represents the false class by prediction.

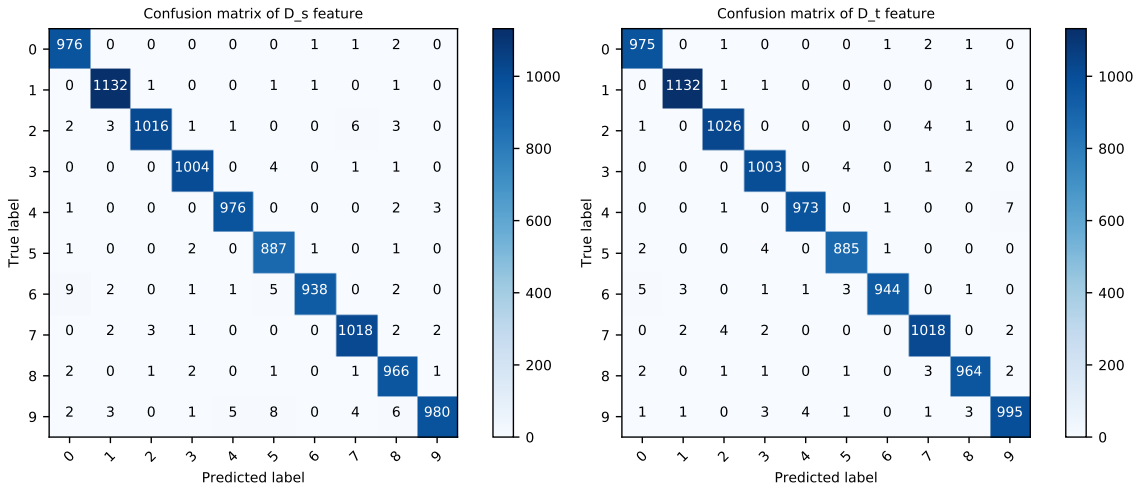

Figure 4-5: Confusion matrices by using the '$\mathbf{D}_s$' and '$\mathbf{D}_t$' features (Displacement features in horizontal and vertical directions).

network will more sensitive to the local features.

The confusion matrices are shown in Fig. 4-5 and Fig. 4-6. We can see that only using the displacement features for classification also can obtain relatively high accuracies. In classes '2' and '9', the vertical displacement features are better than the pooling features for classification. Besides, combining the displacement features with the pooling features improved the recognition accuracies on classes '3', '4', '5', '7', '8', and '9', and just degraded a little on classes '2', and '6'. For the class '9', the proposed method is improved a lot when some test samples are misclassified to classes '4', '5', and '7'.

4.3 Discovering Class-wise Trends of the Displacement Features

In this section, we discovered the class-wise trends of the displacement features in different ways. First, we visualized 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 summarized the displacement features on all feature maps in

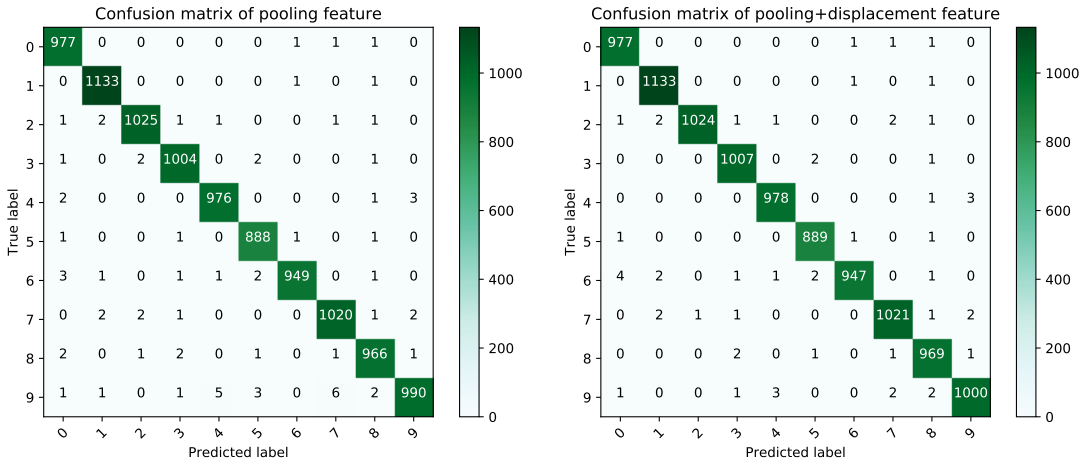

Figure 4-6: Confusion matrices by only using the pooling features and combining the displacement features with the pooling features.

each class and illustrate the cumulative histograms to understand the distribution of the displacement features. Finally, we introduced 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.

4.3.1 Visualization of the Displacement Features

To observe the behaviors and analyze the class-wise trends of the displacement features, we visualized 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. 4-7 and Fig. 4-8. We can see that each corresponding displacement feature records the direction information that describes the displacement of the maximums. The displacement features in the inter-class samples have large dissimilarities. In addition, we can see that the intra-class samples often have similar behaviors. For instance, in the first filter of all class '3' samples, the top left corners are in 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 high accuracies for classification tasks is due to the large differences in the class-wise behaviors.

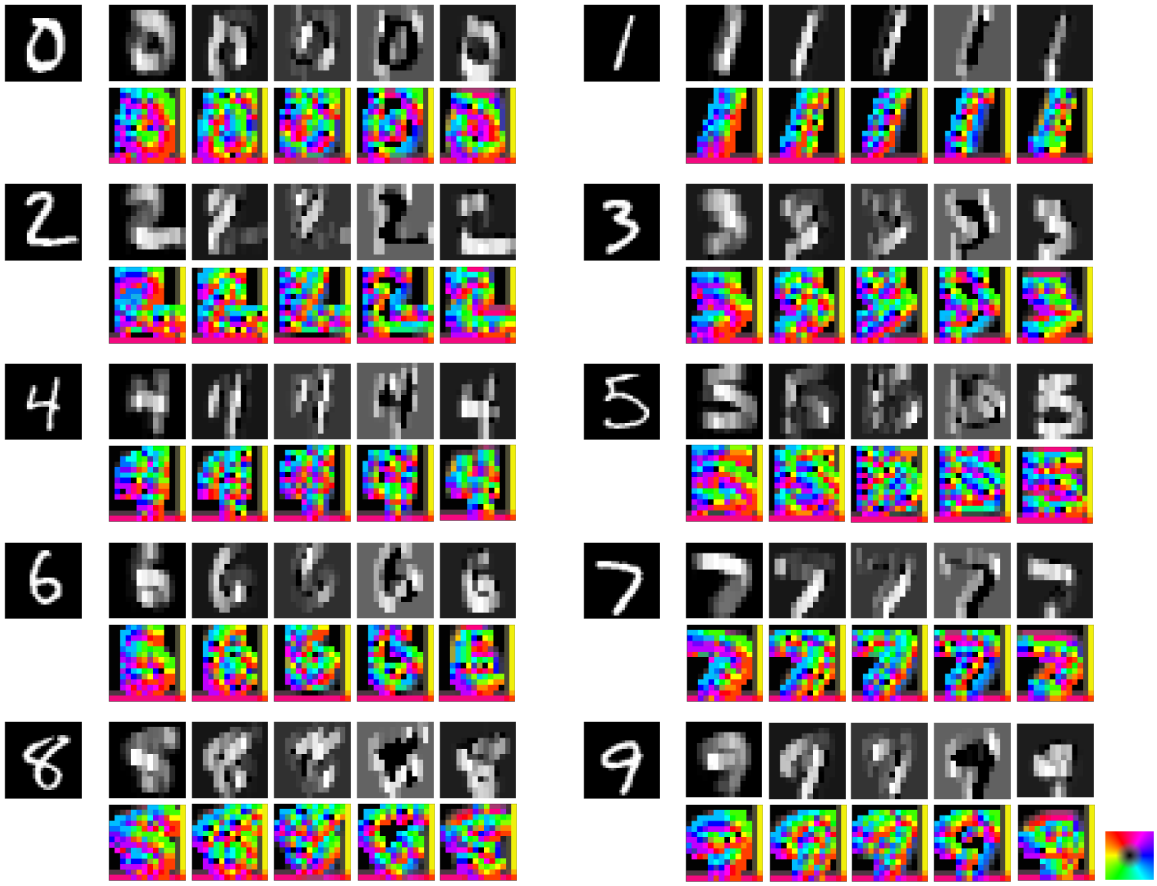

Figure 4-7: Visualization of the displacement features on the different samples that are in the different classes on MNIST dataset. Here, the pooling size is 3×3. Each column represents one convolutional filter.

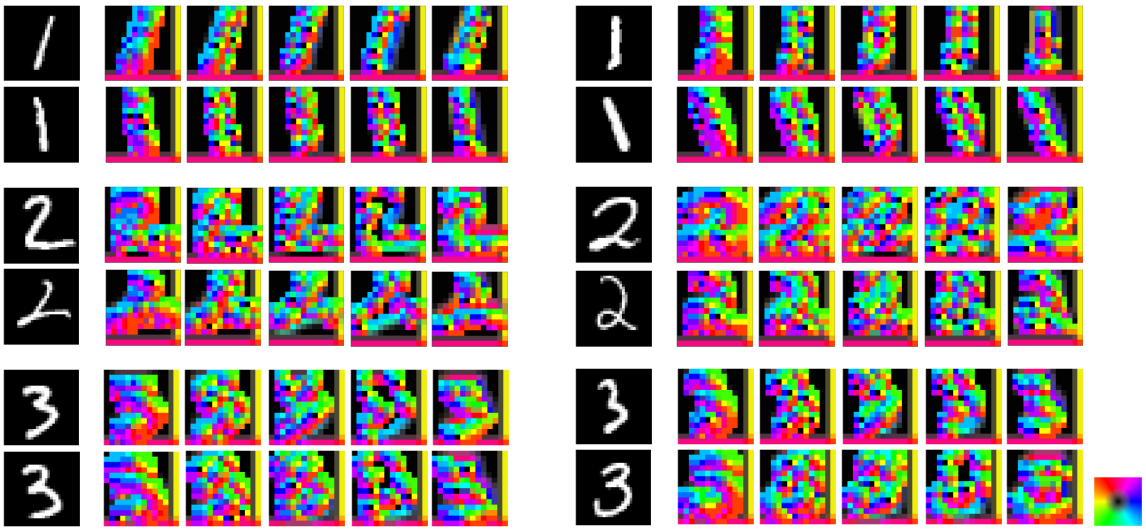

Figure 4-8: Visualization of the displacement features on the different samples that are in the same class on MNIST dataset. Here, the pooling size is 3×3. Each column represents one convolutional filter.

4.3.2 The Distribution of the Displacement Features

To further observe the behaviors of the displacement features on inter-class samples, we illustrated the cumulative histograms of the displacement features, which accounts for the coordinate points of the displacement features of different samples on all feature maps. Here, the pooling size of the first pooling layer is 3×3 and 5×5 with stride 2.

The distribution of the displacement features in the first pooling layer is shown in Fig. 4-9 and Fig. 4-10. In Fig. 4-10, due to the pooling size being 3×3, the displacement features $\mathbf{D}_s$ and $\mathbf{D}_t$ belong to $[-1, 1]$. We can see that the intra-class samples have similar distributions and inter-class samples 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 the vertical direction (the shape of class '1' goes from top to bottom). For the class '2', the distribution is more scattered than the classes '1', and '3'. For the class '3', there are more points around (0,0) than classes '1', and '2' and the distribution is more

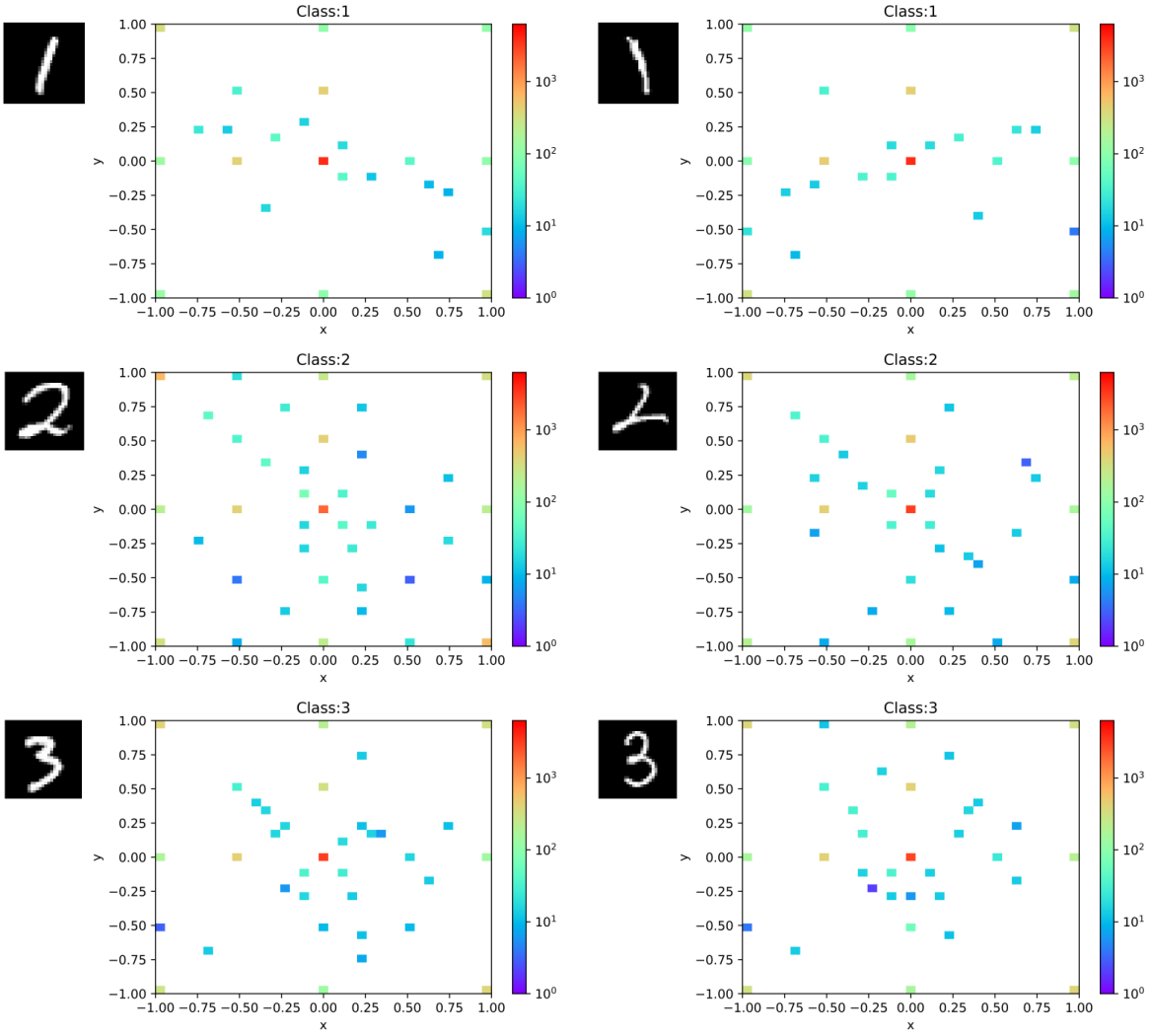

Figure 4-9: The distribution of displacement features from class '1', '2', and '3' (each row represents 2 different samples in the same class). Here, the pooling size is 3×3, the color histograms represent the number of the displacement features on corresponding coordinate points.

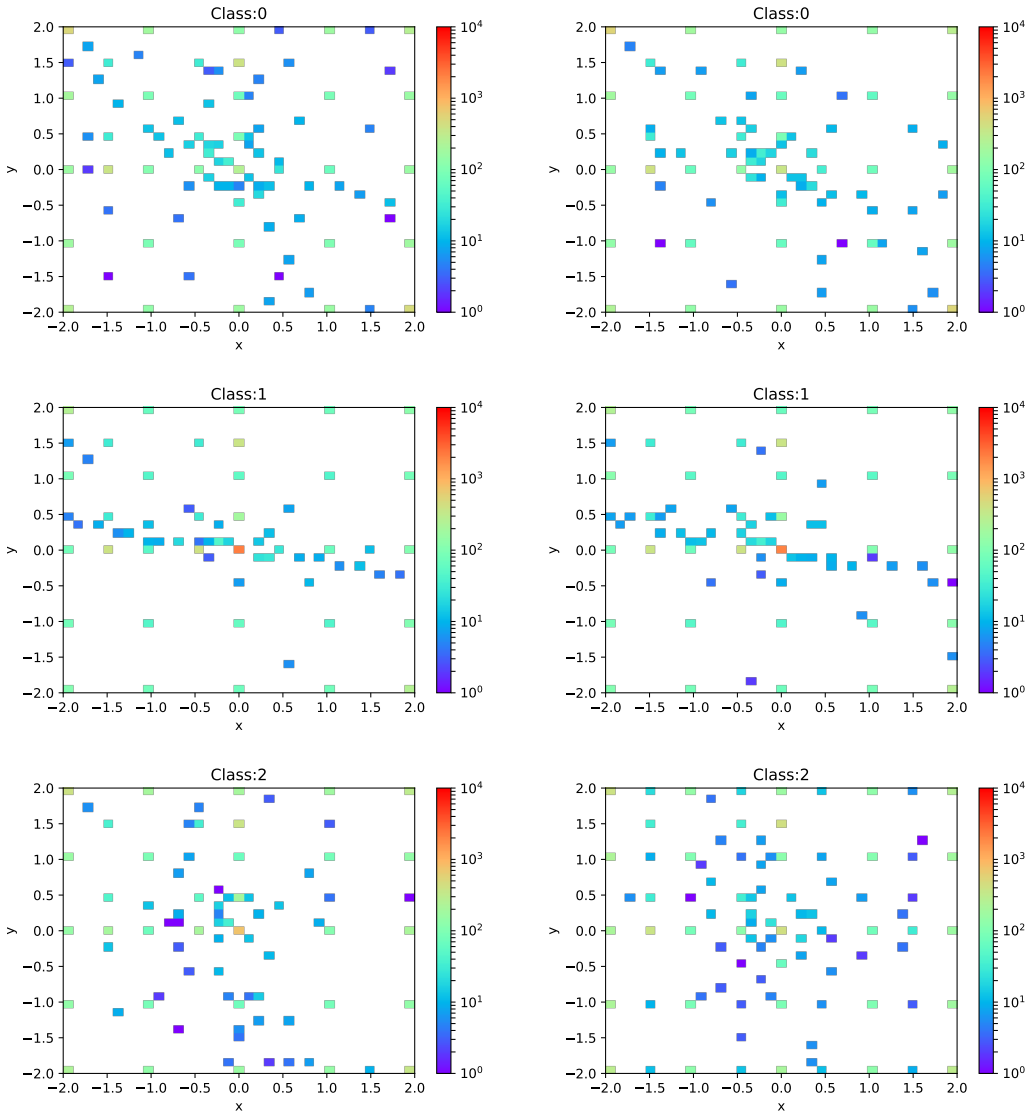

Figure 4-10: The distribution of displacement features from class '0', '1', and '2' (each row represents each class). Here, the pooling size is 5×5, the color histograms represent the number of the displacement features on corresponding coordinate points.

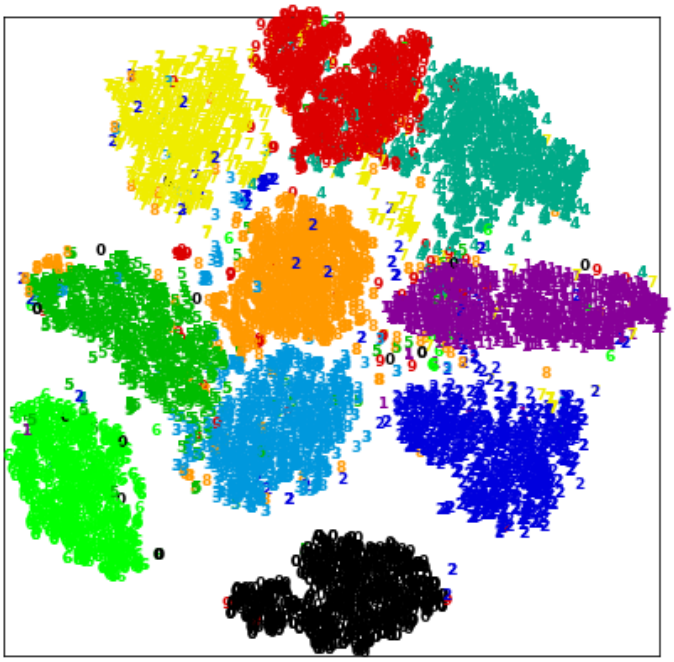

Figure 4-11: Visualization of the displacement features in horizontal direction ($\mathbf{D}_s$) by t-SNE.

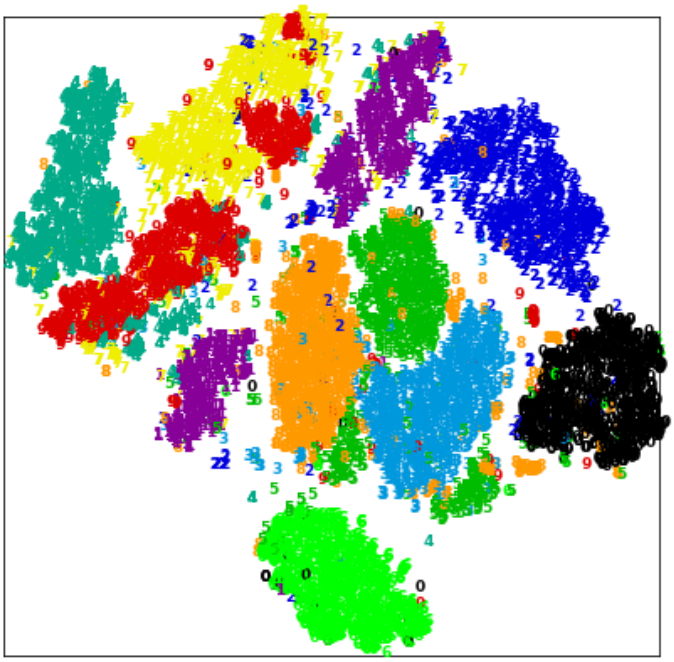

Figure 4-12: Visualization of the displacement features in vertical direction ($\mathbf{D}_t$) by t-SNE.

compact. In addition, most values in all of the samples are (0,0) because the most features in original images are background.

In Fig. 4-10, due to the pooling size being 5×5, the displacement features belong to $[-2, 2]$. We can see that the intra-class samples have similar distributions. For the class '0', there are some values around the central point (0,0) and their quantities are about 100. For the class '1', there is a relatively horizontal line on all samples. It is very similar to class '1' because the displacement information is rare in the vertical direction. For the class '2', the distribution is more scattered than the classes '0' and '1'. In addition, most values in all samples are(0,0), that is because the most features in original images are backgrounds.

In Figs. 4-11 and 4-12, we also visualized the displacement features of the test dataset on 2D space by t-SNE [145]. We can see that there is a very clear class-wise trend of the displacement features. The distance between the intra-class samples is smaller than the inter-class samples. In Fig. 4-11, the classes '1', '4', '5', and '9' have more clusters than other classes. But, this phenomenon does not appear in Fig. 4-12. Therefore, it may mean that the vertical displacement features are better than the horizontal displacement features for classification tasks.

4.3.3 Class-wise Similarity of the Displacement Features in the PCA Subspaces

We also measured the displacement features in PCA subspaces to compare the similarities between different categories for observing the class-wise trends of the displacement features. First, we adopted ten different PCA models on the samples, one for each class. Then, we preserved 10% of eigenvalues for each PCA model to build up the PCA subspaces. Based on these subspaces, we calculated the class-wise similarity matrix for each feature map. Then, the similarity can be defined by using the

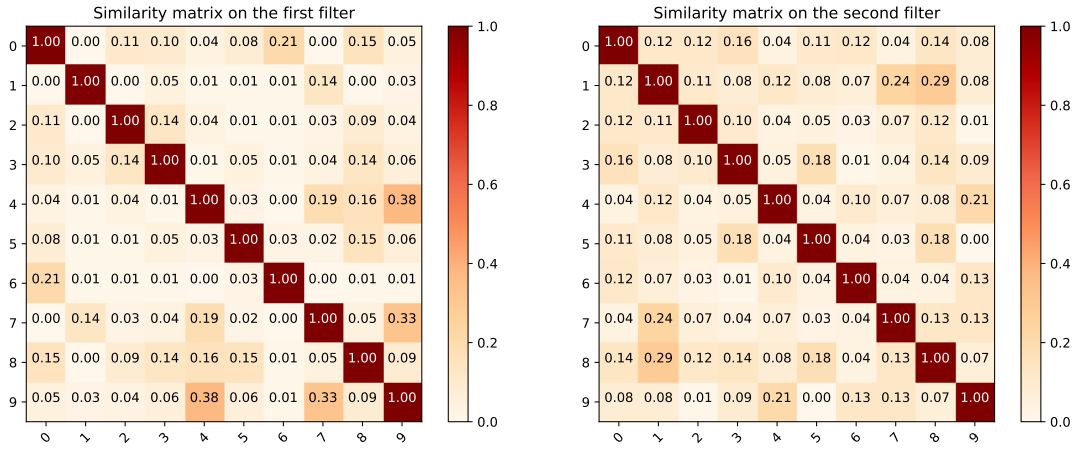

Figure 4-13: Similarity matrix on the first and second filters in PCA subspaces.

canonical angles,

$$\cos\delta_i = \sup_{\substack{\boldsymbol{\alpha}_i \perp \boldsymbol{\alpha}_j, \boldsymbol{\beta}_i \perp \boldsymbol{\beta}_j \\ 1 \le i,j \le p}} \frac{\boldsymbol{\alpha}_i^T \boldsymbol{\beta}_i}{\|\boldsymbol{\alpha}_i\| \|\boldsymbol{\beta}_i\|}. \tag{4.2}$$

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 $\mathbf{P} = p \le$ dim $\mathbf{Q} = q$. Then the similarity can be defined as,

$$S = \frac{1}{p} \sum_{i=1}^{p} \cos^2 \delta_i. \tag{4.3}$$

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 [146].

The similarity matrix in PCA subspaces is shown in Fig. 4-13. 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 similarities of classes '4', and '9', '7', and '9' are larger than others. It means that it is easier to misclassify classes '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.

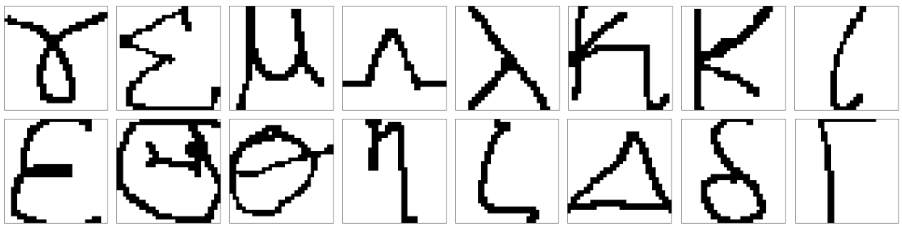

Figure 4-14: The Samples from HASY dataset.

4.4 Classification on HASY Dataset

The HASY dataset is a dataset of handwritten mathematical symbols. It contains 168,233 instances of 369 classes with size 32×32. Fig. 4-14 shows some samples from the HASY dataset. We can see that the HASY is a more challenging dataset than the MNIST dataset because the number of classes in the HASY dataset is larger than MNIST dataset [1].

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. It means that the inter-class micro differences are easily eliminated in this dataset by using the traditional max-pooling operation.

We deployed our experiments on 10 pre-defined folds for 10-fold cross-validation. In the experiment, we used the PCA-based cosine features that are transformed from the displacement features on 3-layer (CNN-3) and 4-layer (CNN-4) CNNs and the CNN-4a architecture [1]. We also compared the proposed method with Random Forest [147, 148], MLP [149, 148], and LDA [150, 148]. For all CNN based architectures, Adam optimizer are used for training procedure. The hyper-parameters are the same as in [1]. The descriptions of the compared models are listed as follows.

- **Random Forest** [147, 1]: Random forests use a large number of decision trees in an ensemble classifier.

Table 4.2: Classification results on HASY Dataset.

Models	Accuracy
Random Forest [147, 1]	0.624
MLP [149, 1]	0.622
LDA [150, 1]	0.468
CNN-3 [1]	0.784
CNN-4 [1]	0.805
CNN-4a [1]	0.810
CNN-3+ours	0.788±0.005
CNN-4+ours	0.814±0.008
CNN-4a+ours	**0.823±0.002**

- **MLP** [149, 1]: The multilayer perceptron (MLP) is a fully-connected feed forward neural network.

- **LDA** [150, 1]: 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** [1]: 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 with 369 nodes.

- **CNN-4** [1]: 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** [1]: 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.

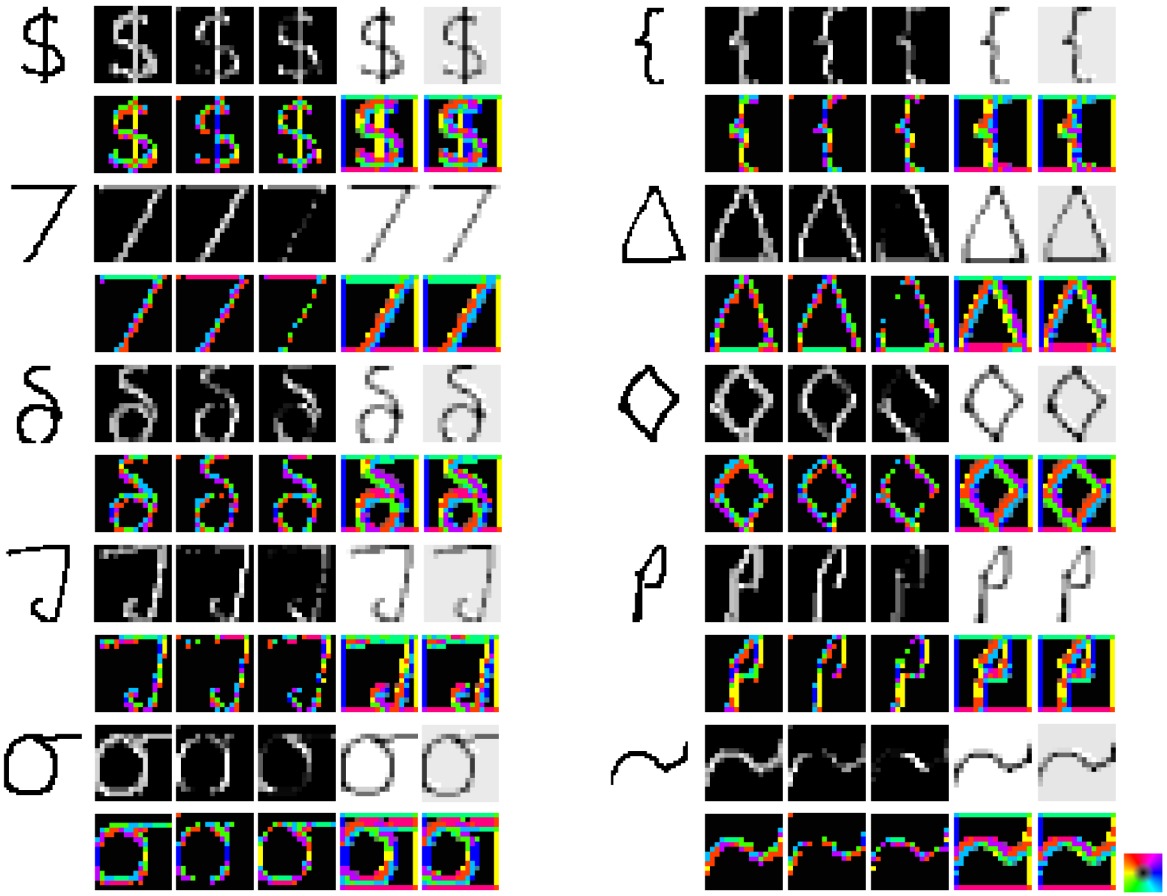

Figure 4-15: Visualization of the displacement features on the different samples that are in the different classes on HASY dataset. Here, the pooling size is 3×3. Each column represents one convolutional filter. The ground-truths from top to bottom are '$", '{', '7', '\Delta ($\Delta$)', '\delta ($\delta$)', '\diomond ($\diamond$)', 'J', 'p', '\sigma ($\sigma$)', '\sim ($\sim$)'.

Table 4.2 shows the 10-fold cross-validation results for three architectures. We conducted the t-test with confidence value 0.05 on the results. From Table 4.2, 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. 4-15 shows the visualization of the displacement features on different inter-class samples on the HASY dataset. We can see that the different filters can capture

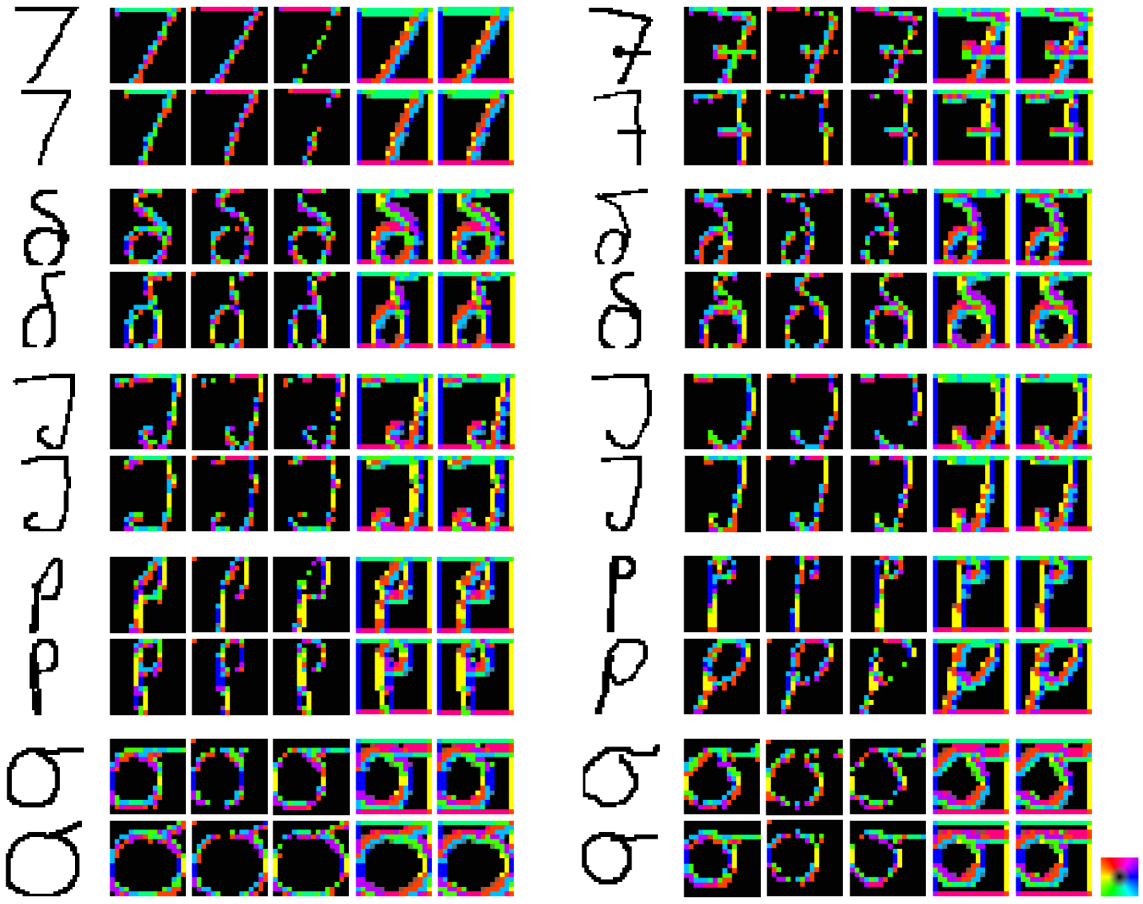

Figure 4-16: Visualization of the displacement features on the different samples that are in the same classes on HASY dataset. Here, the pooling size is 3×3. Each column represents one convolutional filter. The ground-truths from top to bottom are '7', '\delta (δ)', 'J', 'p', '\sigma (σ)', '\sim ($\sim$)'.

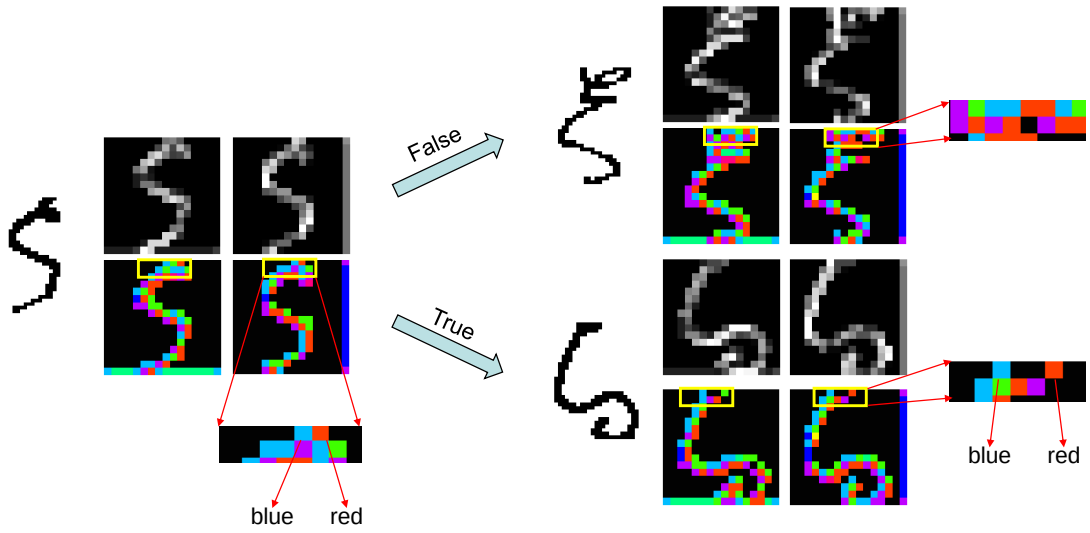

Figure 4-17: One improved sample by using our proposed method. The left is a test sample that is improved by our method, the upper right is a sample whose label is same as the misclassified label, the lower right is another sample whose label is same as test sample.

the different local features in an image, which is shown in the pooling features. For example, the second filter may focus on the vertical direction in the images. The third filter may focus on the 45° of direction. The finally two filters may focus on the information from the background. The behaviors of the displacement features from different classes are also very different. For example, in the first filter, the top part in class '7' is green from the left to right. However, the top parts in other classes are hugely different from class '7'. In the third filter, the bottom part in class '\Delta (Δ)' is green. However, the bottom part in class '\sigma (σ)' is red. It means that the displacement features can also capture the different behaviors between the inter-class samples.

Fig. 4-16 shows the visualization of the displacement features on the different intra-class samples on the HASY dataset. We can see that the displacement features present similar behaviors in the same class. For example, in the first filter, the top part in class 'J' is in green and blue directions. In the second filter, the top part in class 'J' is in green and purple, and red directions. In addition, we can see that

the displacement features can also capture some micro differences from some samples that are "strange" compared with other samples. For example, a horizontal stroke exists in some samples in class '7'. In the final two filters, the displacement features may capture the red and green directions from the left part, and the purple and blue directions from the right part. However, in the polling features, these micro behaviors may be absorbed by using the traditional max-pooling operations.

Fig. 4-17 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). The reason is that the pooling features of the \xi (ξ) and \xi (ξ) is very similar. For our proposed displacement features, there is clear discrimination between these two samples if they are from different classes. For example, the top of the image is in the blue and red directions in two \zeta (ζ) samples, and the left edge is in the blue direction. These good characteristics help for improving the performance of the max-pooling operation.

4.5 Classification on Chars74k-font Dataset

The Chars74K dataset is composed of several sub-datasets. We used the sub-dataset which includes characters from computer fonts with different variations (combinations of italic, bold, and normal). We referred to this sub-dataset as Chars74K-font and it contains 62,992 samples of 62 classes with the size 128×128.

Fig. 4-18 shows some samples from Chars74k-font datasets. It includes 62 categories from the numbers '0' to '9', capital letters 'A' to 'Z', and lowercase letters 'a' to 'z' which are synthesized by different computer fonts. Some categories such as number '0' and letter 'O' are very similar except for several micro differences. After the max-pooling operation, these inter-class micro differences might be eliminated. Therefore, the Chars74K-font dataset is also a more challenging dataset than the

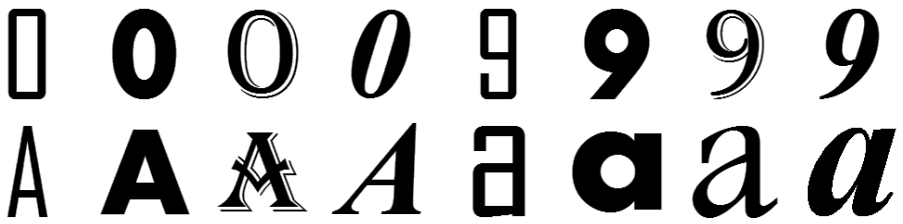

Figure 4-18: The Samples from Chars74k-font dataset.

MNIST dataset.

In the experiment, we also used the PCA-based cosine features that are transformed from the displacement features on there architectures, Lenet [151], Lenet-5 [151], and SPnet [152]. For SPnet, we used 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 used 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 4.3. We also compared the proposed method with many state-of-the-art methods in our experiments. The methods are listed as follows.

- **Global DCT** [148]: 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** [148]: This method uses Angular Radial Transform (ART) descriptor to obtain the feature vectors. Then, it uses a nearest neighbor classifier for classification.

- **Shape Context** [91, 153]: This method uses Sobel edge detector to extract the features. Then, it uses a nearest neighbor classifier.

- **SIFT** [154, 153]: This method used Harris Hessian-Laplace detector to extract the Scale Invariant Feature Transform (SIFT). Then it uses the nearest neighbor

classifier for classification.

- **Block DCT** [148]: This method is based on Global DCT. It performs 8×8 block-based DCT and retains 15% coefficients for every block.
- **Neumann et.al** [155]: This method uses directional feature vectors for training multi-class support vector machines (SVMs) classifier with Radial Basis Function (RBF) kernel.
- **Geometric Blur** [156, 153]: 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** [157]: 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** [151]: Lenet contains 2 convolutional layers, 2 pooling layers and 3 fully connected layers.
- **Lenet-5** [151]: Lenet-5 contains 3 convolutional layers, 2 pooling layers and 3 fully connected layers.
- **SPnet** [152]: SPnet contains 3 convolutional layers, 3 pooling layers and 3 fully connected layers.

We conducted the t-test with confidence value 0.05 on the results in Table 4.3. We can see that the proposed methods (combining the cosine features with pooling features) are significantly better and obtain 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.

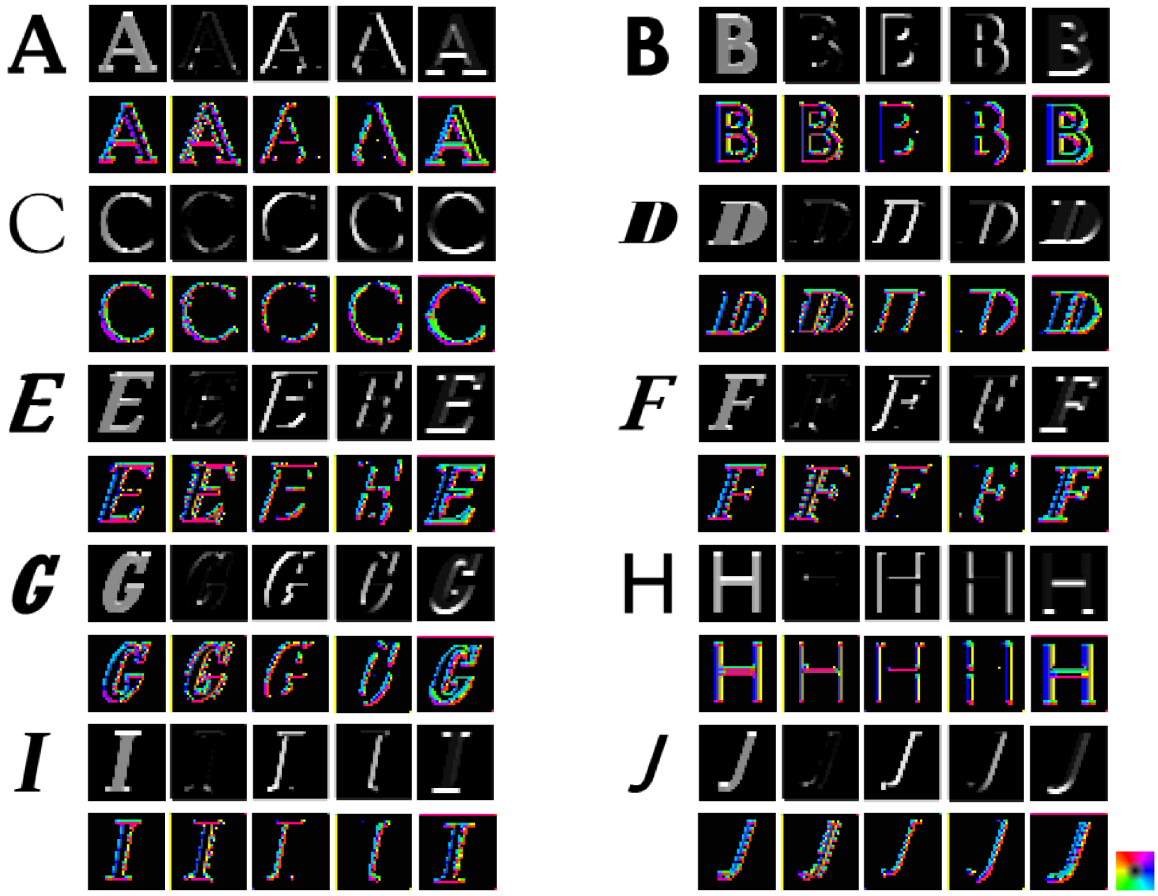

Figure 4-19: Visualization of the displacement features on the different samples that are in the different classes on Chars74K-font dataset. Here, the pooling size is 3×3. Each column represents one convolutional filter. The ground-truths from top to bottom are 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'.

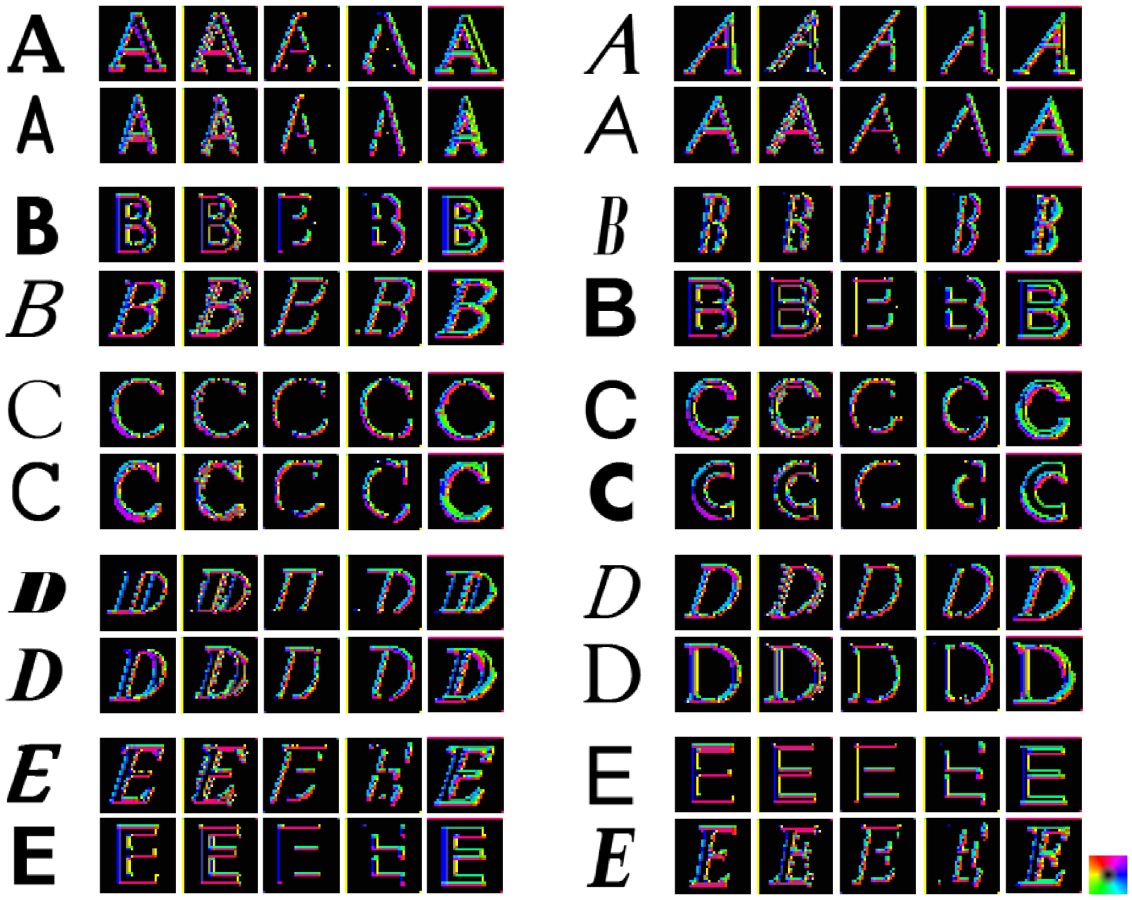

Figure 4-20: Visualization of the displacement features on the different samples that are in the same classes on Chars74K-font dataset. Here, the pooling size is 3×3. Each column represents one convolutional filter. The ground-truths from top to bottom are 'A', 'B', 'C', 'D', 'E'.

Table 4.3: Classification results on Chars74K-font dataset.

Method	Accuracy
Global DCT [148]	0.6630
ART [148]	0.4920
Shape Context [91, 153]	0.6483
SIFT [154, 153]	0.4694
Block DCT [148]	0.6640
Neumann et.al [155]	0.7160
Geometric Blur [156, 153]	0.6971
I2CDML [157]	0.7300
Lenet [151]	0.8839
Lenet-5 [151]	0.8553
SPnet [152]	0.9056
Lenet+ours	0.8896±0.002
Lenet-5+ours	0.8593±0.004
SPnet+ours	**0.9145±0.005**

Fig. 4-19 shows the visualization of the displacement features on the different inter-class samples on the Chars74K-font dataset. We can obtain a similar conclusion as before. The pooling features capture the local information in each filter in an image. For example, the first filter extracts more features from the top part in different samples than other parts. The third and fourth filters extract information from the left and right boundaries of the characters. The final filter extracts the horizontal information in characters.

For the displacement features, the difference from different samples is relatively huge. For example, in the first filter, the displacement features capture the red direction in class 'A', but the purple direction in class 'C'. In the final filter, the displacement features capture the green direction in class 'A', but the yellow direction in class 'H'. We note that, in the final filter, the pooling features are very similar between the classes 'A' and 'H', but the displacement features are relatively different in many parts. It means that the displacement features can help the traditional pooling features to discriminate the different samples by capturing these micro differences.

Fig. 4-20 shows the visualization of the displacement features on the different intra-class samples on the Chars74K-font dataset. We can see that the displacement features present similar behaviors in the same class and different behaviors from the different classes. For example, In the first filter, the displacement features can capture the red direction in the middle stroke of class 'A' and the green direction in the top stroke of class 'E'. In the final filter, the displacement features can capture the dark blue direction in the left stroke of class 'B' and the purple direction in the left bottom corner of class 'C'. The displacement features capture similar behaviors in the same class, which is very useful for recognition tasks.

4.6 Summary

In this chapter, we present the experiment results and analysis of the text recognition tasks. First, we show the classification results on the MNIST dataset and discover the class-wise trends of the displacement features in different ways. Then, we present the classification result on HASY and Chars74K-font datasets. After analysis and discussion, we can see that the proposed method can capture some inter-class micro differences from the max-pooling operation in CNN based architectures. The displacement features may present different micro differences between similar classes. We prove that combining the displacement features with the traditional pooling features could capture the inter-class micro differences and improve the CNN based architectures for some specific text recognition tasks. The pooling features can be described as the main differences between different samples, and the displacement features present the micro differences between the inter-class samples.

Chapter 5

Experiment on Offline Signature Verification

In this chapter, we introduce how to apply the displacement features for offline signature verification tasks. Since the forged signatures are obtained by practiced imitators, the differences between the genuine signatures and skilled forgeries are very tiny. Therefore, capturing these "intra-class" micro differences is crucial for verification systems. To address this issue, the displacement features are extracted from a pre-trained CNN to represent the micro differences between the genuine signatures and their corresponding skilled forgeries.

In order to build the complete verification system, first, we train a CNN between the genuine signatures and skilled forgeries on a large scale dataset. Next, we extract the displacement features from the first convolutional feature maps and fuse the displacement features with pooling features to capture the micro differences between the genuine signatures and their corresponding skilled forgeries for the feature extraction procedure. Here, the PCA-based cosine features are not applied in this task because the number of users (categories) is huge and the users in test set are not include in train set.

Since this research is mainly to learn the discriminative features between the

genuine signatures and skilled forgeries, we could apply writer-dependent or writer-independent classifiers to build a complete verification system. Normally, the SVMs can be used as the writer-dependent classifiers in verification systems. Therefore, we train the SVMs as the writer-dependent classifiers for each user to build a complete verification system. Finally, we present the experiment results and discussions on GPDS-150, GPDS-300, GPDS-1000, GPDS-2000, and GPDS-5000 datasets which are the subsets of GPDS-10000 dataset [1].

5.1 Capturing Micro Differences by Displacement Features for Offline Signature Verification

In this section, we introduce the proposed method to fuse the displacement features and pooling features as final discriminative features to capture the micro differences between the genuine signatures and their corresponding skilled forgeries for offline signature verification systems. First, we introduce a normal CNN-based architecture trained between the genuine signatures and skilled forgeries. This architecture just divides all the signatures into two classes, genuine and forged signatures. Then, based on the pre-trained CNN, we introduce how to extract the displacement features and fuse it with the pooling features in a combined architecture to further capture the micro differences between the genuine signatures and skilled forgeries. Finally, we introduce how to train the SVMs as the writer-dependent classifiers based on the fused features to build the complete verification system.

[1] http://www.gpds.ulpgc.es/

5.1.1 Training a CNN Between the Genuine Signatures and Skilled Forgeries

To distinguish genuine signatures from skilled forgeries in the feature extraction phase, we design a CNN-based architecture with 3 convolutional and pooling layers, 2 fully-connected layers, and a softmax layer. In the convolutional layers, the convolutional kernel size is 3×3 with stride 1, and the number of the filters is 32, 64, and 128, respectively. The pooling size is 2×2 with stride 2. In the fully-connected layers, the first fully-connected layer has 2048 hidden nodes and reduces to 1024 in the second fully-connected layer. Rectified Linear Unit (ReLU) is used as the activation function for the network. The final softmax layer contains 2 nodes, which is designed to judge whether the input is a genuine signature or skilled forgery. The cross-entropy is used as the loss function to train the network.

5.1.2 Extracting and Fusing the Displacement Features with Pooling Features

To further capture the micro differences between the genuine signatures and skilled forgeries, we extract the displacement features [26, 27] from the first pooling layer in previous pre-trained CNN and fuse it with the pooling features. Fig. 5-1 shows the procedure of the feature extraction. Here, the pooling size is 2×2 with stride 2, the value of the displacement features both in horizontal and vertical directions belongs to $[-1, 1]$. The pooling features represent the general information of the signatures from different channels. The displacement features describe the location information of maximums in the max-pooling operation, which might capture some micro differences of forgery signatures when some skilled writers imitated the genuine signatures. The architecture that we used for fusing the pooling features and displacement features is shown in Fig. 5-2, which is the same with the architectures in text recognition tasks. We can see that the architecture for processing the displacement features is

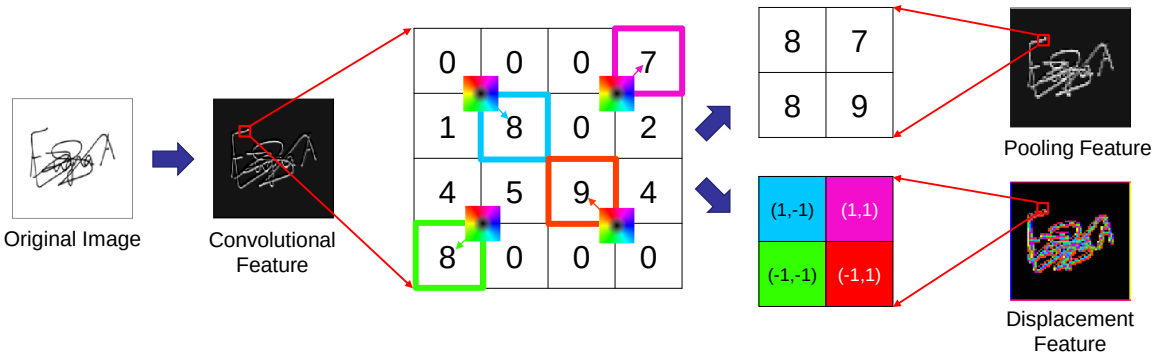

Figure 5-1: Extracting the pooling features and the displacement features simultaneously. Here, the pooling size is 2×2 with stride 2. The displacement features capture micro differences within the pooling window. The displacement vector (-1,1) means that the vertical displacement from the center to the maximum value is -1 and the horizontal displacement is 1.

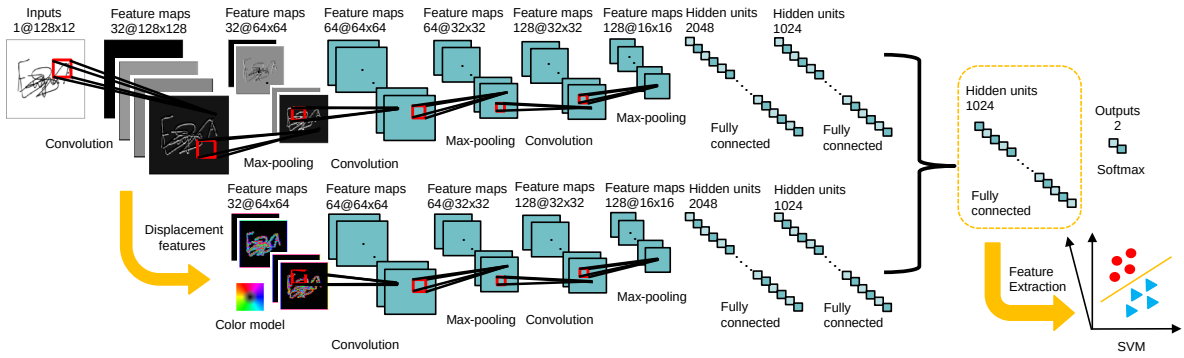

Figure 5-2: The procedure of feature extraction and verification.

the same as the pre-trained CNN without the first convolutional and pooling layers. Here, we divide the displacement into the horizontal and vertical directions and apply the same architecture. We then fuse the pooling features and displacement features in the last fully-connected layer to retrain the networks. Finally, we take this trained architecture as a feature extractor for all signatures. The next step is training the writer-dependent classifiers to build a complete verification system.

5.2 Training the Writer-dependent Classifiers

After the CNN training procedure, we extract the fused features of each user from the CNN-based feature extractor and train the SVM as the writer-dependent classifiers for building the complete verification systems. It should be noted that this research is mainly to focus on the feature extraction procedure and capturing the micro differences between the genuine signatures and corresponding skilled forgeries. Training the SVM as the writer-dependent classifiers is just one kind of choice in this procedure. We follow the settings proposed in [133] as a baseline to compare with other state-of-the-art systems.

For each user (not included in CNN training procedure), we use the genuine signatures as the positive samples and genuine signatures from other users as the negative samples to build the training set. Then, we choose the linear SVM as the writer-dependent classifier for each user to build the verification system. This procedure is shown in Fig. 5-2.

To overcome the imbalanced problem that the negative samples (genuine signatures from the target user) are much more than the positive samples (genuine signatures form other users), we use different weights for the positive and negative classes [131]. Then, the SVM objective function becomes,

$$\begin{aligned} \min \ & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C_{pos} \sum_{\substack{i=1 \\ y_i=+1}}^{M} \xi_i + C_{neg} \sum_{\substack{i=1 \\ y_i=-1}}^{N} \xi_i \\ s.t. \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \ \text{ and } \ \xi_i \geq 0, \end{aligned} \tag{5.1}$$

where $\mathbf{x}_i$ is a training sample with target value y_i, ξ_i is a slack variable, M and N are the numbers of the positive and negative samples, C_{pos} and C_{neg} are the weights for the positive and negative classes,

$$C_{pos} = \frac{N}{M} C_{neg} \tag{5.2}$$

For the testing procedure, we use the remaining genuine signatures of the target user and the skilled forgeries to build the test set and evaluate the performance of the complete verification system.

5.3 Experimental Protocol

We conducted the experiments on the GPDS-10000 dataset to evaluate the proposed method. The GPDS-10000 is a large scale dataset that contains 24 genuine signatures and 30 skilled forgeries for each user. The number of users is 10,000, so the GPDS-10000 dataset contains 240,000 genuine signatures and 300,000 skilled forgeries and it is very suitable for deep learning-based methods. In our experiment, we divided the GPDS-10000 dataset into two parts, the first 5,000 users, and the final 5,000 users.

For the procedure of training CNN, we used the signatures from the final 5,000 users. To normalize the input images, we first resized all images to 128×128. Then, we use 90% data for training and 10% data for validation to train the network. We used Adam as an optimizer to minimize the loss function with mini-batch size 32. The model is trained with 40 epochs. The initial learning rate is set to 1e-4 and reduced by a factor of 0.95 after each epoch. After the CNN training process, we took the trained model as a feature extractor to extract the fused features.

To build the signature verification systems, we trained the linear SVMs as the writer-dependent classifiers for each user. Compared to other state-of-the-art methods, we used 5 sub-datasets, GPDS-150, GPDS-300, GPDS-1000, GPDS-2000, GPDS-5000 (the first 100, 150, 1,000, 2,000, 5,000 users of GPDS-10000 dataset) for final evaluation. For a specific user, we randomly selected 5 genuine signatures as the positive samples and 5 genuine signatures from each of the final 5,000 users as the negative samples as the training set. For the training process, the weights C_{neg} are found by grid search with 5-fold cross-validation, and the C_{pos} is calculated by Equation 5.2.

For the evaluation of the test set, the remaining genuine signatures from the target

user are used for calculating the False Rejection Rate (FRR). The False Acceptance Rate for the skilled forgeries ($FAR_{skilled}$) experiment has been obtained with forgery samples of the target user. The False Acceptance Rate for the random impostor (FAR_{random}) experiment has been obtained with the genuine signatures from all the remaining users. The Equal Error Rate for skilled forgeries experiment ($EER_{skilled}$) is calculated by $FAR_{skilled} = FRR$, and the EER for the random impostor experiment (EER_{random}) is calculated by $FAR_{random} = FRR$.

5.4 Experimental Results and Discussion

To evaluate the performance of the proposed method, we tested the proposed method on GPDS-150, GPDS-300, GPDS-1000, GPDS-2000, and GPDS-5000 datasets and compared it with two baseline models [122, 123] and the traditional CNN based features with SVMs. In [122], the authors applied a Hidden Markov Model (HMM) on the geometrical features (GF) for verification systems. In [123], the authors extracted the LBP features to train the SVMs as the writer-dependent classifiers. For the traditional CNN model, it just extracts the features from the last fully-connected layer. Then, using the linear SVMs for each user to build the verification system. The experimental results are the averages of all users with 10 trials.

Table 5.1 shows the results in skilled forgeries experiment. This experiment is to verify whether the query samples are genuine signatures or skilled forgeries. We can see that only using the SVMs with features extracted from a traditional CNN can achieve the desired results. The proposed method obtains better results than the other two baseline models on all datasets. When the number of the randomly selected samples increased to 10, the EERs are smaller than before, which means that using more samples to train the SVMs can improve the performance of the verification systems. In addition, as the datasets become larger (the number of users is increasing), the EER becomes higher when using the traditional models. It means

Table 5.1: The skilled forgeries experiment ($EER_{skilled}$ in %). The 5 samples and 10 samples represent the randomly selecting 5 or 10 samples of each user for training the SVMs.

Dataset	HMM+GF [122] (5 samples)	SVM+LBP [123] (5 samples)	Traditional CNN (5 samples)	Proposed (5 samples)	Proposed (10 samples)
GPDS-150	11.48	16.45	8.34±0.52	**7.45±0.41**	**6.32±0.35**
GPDS-300	12.11	16.50	8.41±0.61	**7.48±0.52**	**6.25±0.41**
GPDS-1000	11.07	17.01	8.31±0.47	**7.12±0.45**	**6.43±0.38**
GPDS-2000	11.34	16.63	8.20±0.42	**7.23±0.53**	**5.92±0.41**
GPDS-5000	11.10	16.93	8.08±0.53	**7.15±0.48**	**6.11±0.47**

Table 5.2: The random impostor experiment (EER_{random} in %), where micro differences are not important.

Dataset	HMM+GF [122] (5 samples)	SVM+LBP [123] (5 samples)	Traditional CNN (5 samples)	Proposed (5 samples)	Proposed (10 samples)
GPDS-150	4.17	**1.31**	4.89±0.48	4.64±0.38	**3.08±0.33**
GPDS-300	4.32	**1.45**	4.77±0.52	4.72±0.54	**3.23±0.42**
GPDS-1000	4.37	**1.63**	4.82±0.58	4.91±0.43	**3.17±0.35**
GPDS-2000	4.44	**1.73**	4.94±0.42	4.95±0.55	**3.22±0.41**
GPDS-5000	4.53	**1.63**	4.56±0.43	4.58±0.41	**2.84±0.43**

that the traditional models are not general for all the users. The proposed method is more stable than the traditional models when the dataset size becomes huger.

Table 5.2 shows the results in random impostors experiment. This experiment is mainly to discriminate different users. Since the purpose of the feature extractor is to capture different behaviors between the genuine signatures and skilled forgeries, the proposed method does not obtain the best results on this experiment. However, we achieve a competitive performance compared to the baseline models. Even if the model in [123] can classify the different users well, the ability to distinguish the genuine signatures and skilled forgeries of this model is far worse than the proposed method.

Then, to further improve the performance of the verification system, we applied the SVMs with Radial Basis Function (RBF kernel) as the writer-dependent classifiers to build the verification system. The RBF kernel can achieve nonlinear maps that change each linearly inseparable problem into a separable one. The parameter, weight C_{neg} is found by grid search on the validation set, and the C_{pos} is calculated by

Table 5.3: The skilled forgeries experiment ($EER_{skilled}$ in %). The 5 samples and 10 samples represent the randomly selecting 5 or 10 samples of each user for training the RBF kernel SVMs.

Dataset	Traditional CNN (5 samples)	Traditional CNN (10 samples)	Proposed (5 samples)	Proposed (10 samples)
GPDS-150	8.24±0.48	7.48±0.35	**7.31±0.35**	**6.24±0.42**
GPDS-300	8.28±0.42	7.35±0.26	**7.21±0.52**	**6.11±0.33**
GPDS-1000	8.19±0.38	7.07±0.41	**6.95±0.31**	**6.08±0.29**
GPDS-2000	8.17±0.35	7.22±0.48	**7.11±0.24**	**5.84±0.33**
GPDS-5000	7.98±0.42	7.08±0.39	**7.18±0.35**	**6.05±0.28**

Table 5.4: The random impostor experiment (EER_{random} in %). Here, RBF kernel SVMs are used as the writer-dependent classifiers.

Dataset	Traditional CNN (5 samples)	Traditional CNN (10 samples)	Proposed (5 samples)	Proposed (10 samples)
GPDS-150	4.58±0.39	2.99±0.31	4.59±0.41	**2.95±0.42**
GPDS-300	4.56±0.43	**3.11±0.38**	4.60±0.38	3.18±0.37
GPDS-1000	4.64±0.51	3.17±0.39	4.68±0.45	**3.11±0.36**
GPDS-2000	4.68±0.35	**3.08±0.42**	4.78±0.47	3.17±0.48
GPDS-5000	4.51±0.38	2.79±0.41	4.49±0.42	**2.74±0.46**

Eq. (5.2). The parameter, γ is set to default. Table 5.3 and Table 5.4 show the results on skilled forgeries and random impostors respectively. We can see that, if we use the SVMs with RBF kernel function, the results are a little better than SVMs with a linear kernel. We can also obtain the same conclusion as before, the proposed method is stable when the dataset size becomes larger.

In addition, we also compare our proposed system with many state-of-the-art verification systems. The description of the compared systems are listed as follow.

- **[158]**: This system applies a convolutional Siamese neural network for offline signature verification task [159]. It proposes a hybrid similarity layer that combines the property of Euclidean and Cosine distances. Unlike the other methods which consider feature extraction and metric learning as two independent stages, this system adopts a deep leaning-based framework which combines

the two stages together and can be trained end-to-end.

- **[160]**: This system consists of two steps: feature extraction and verification. In the first step of the feature extraction phase, a Residua CNN [19] is trained on a source task in the handwriting domain. Next, by using a transfer learning approach, the model will be transferred into the signature domain. Finally, for the verification phase, SVMs are trained as writer-dependent classifiers for building a complete verification system.

- **[161]**: This system uses a novel writer-dependent Recurrent Binary Pattern (RBP) network [162], and a novel signer identification CNN building verification system. For the feature extraction phase, signature representations that are extracted by CNN. Then, the RBP network learns the sequential relation between binary pattern histograms over image windows, and a novel histogram selection method is introduced to remove the stop-word codes.

- **[163]**: This system proposes the Multi-Loss Snapshot Ensemble (MLSE) of CNNs for the feature extraction phase. In this phase, cross entropy, Cauchy-Schwarz divergence [164], and hinge loss [165] are combined into a dynamic multi-loss function to train the CNNs. Then, an ensemble of SVMs is trained on these features, and their decisions are finally combined according to the selection of most generalizable SVM for each user to build the verification system.

- **[133]**: This system applies AlexNet [166] and VGG [16] For feature extraction step. In this step, it trains the AlexNet and VGG between different users. Then, it trains SVMs as writer-dependent classifiers for building a complete verification system. it also investigates the impact of the depth (number of layers) of the CNNs, and the size of the embedding layer on learning good representations for signatures.

- **[167]**: This system proposes a combined system that integrates keypoint graphs

and inkball models as two complementary handwriting models. First, it calculates the distance and dissimilarity by using keypoint graphs [168] and inkball models [169]. Then, it proposes a Multiple Classifier System (MCS) using a linear combination of the two dissimilarity measures as the combined dissimilarity score for building the verification system.

- **[170]**: This system proposes an offline signature verification system by using a deep residual neural network and graph edit distance. First, it calculates the distance by using metric learning on a deep CNN [19] with the triplet loss function [171] and keypoint graphs models [172, 168]. Then it uses MCS as a linear combination of the graph-based dissimilarity and the neural network-based dissimilarity for building the verification system.

- **[132]**: This system uses Deep Multitask Metric Learning (DMML) [173] to train a distance metric for each class together with other classes simultaneously. For feature learning step, pairs of the signatures are fed in two networks for training. Then, the distance between the two representations is calculated for building the verification system.

- **[174]**: This system proposes a new descriptor founded on a quad-tree structure of the Histogram Of Templates (HOT) [175] for the feature extraction step. For verification step, it proposes a robust implementation of the Artificial Immune Recognition System (AIRS) [176] which develops new memory cells that are subsequently used to recognize data through a k Nearest Neighbor (kNN) classification. Then, the kNN classification is substituted by a Support Vector (SV) decision, yielding the AIRSV classifier.

- **[137]**: This system uses Deep Convolutional Generative Adversarial Networks (DCGANs) [177] to learn features From different signatures. After the feature extraction step, a hybrid writer-independent and writer-dependent classifier [178] is used, which is an approach compromise two different kinds of

Table 5.5: Comparison with state-of-the-art systems on the GPDS-10000 dataset ($EER_{skilled}$ in %). '#Refs' represents the number of samples for each user to train the writer-dependent classifiers.

Systems	#Refs	Number of user						
		75	150	300	1000	2000	4000	5000
[158]	pairs	-	-	-	-	10.37	-	-
[160]	5	-	-	-	-	-	7.99	-
[160]	7	-	-	-	-	-	7.34	-
[160]	10	-	-	-	-	-	6.81	-
[161]	5	-	-	22.13(0.42)	-	-	-	-
[161]	12	-	-	14.93(0.18)	-	-	-	-
[163]	10	-	-	-	-	-	6.13(0.29)	-
[133]	10	-	-	-	-	-	8.70(0.35)	-
[167]	10	6.84	-	-	-	-	-	-
[170]	10	7.24	-	-	-	-	-	-
[132]	10	12.83	12.67	-	12.43	12.80	13.3	-
[174]	10	-	-	-	-	-	18.32	-
[137]	14	-	-	-	-	-	14.79	-
[179]	10	6.62	-	-	-	-	-	-
[122]	5	-	11.48	12.11	11.07	11.34	-	11.10
[123]	5	-	16.45	16.50	17.01	16.63	-	16.93
Ours	5	-	7.31(0.35)	7.21(0.52)	6.95(0.31)	7.11(0.24)	7.22(0.41)	7.18(0.35)
Ours	10	-	**6.24(0.42)**	**6.11(0.33)**	**6.08(0.29)**	**5.84(0.33)**	**5.92(0.38)**	**6.05(0.28)**

classifiers for building verification systems.

- **[179]**: This system uses two recent graph-based approaches to offline signature verification: keypoint graphs with approximated graph edit distance [180] and inkball models [181]. Then, it calculates the dissimilarities between the signature pairs of these two models. Finally, it uses a MCS to combine these two dissimilarities as the final score for building verification system.

- **[122]**: This systems proposes new geometrical features which can be extracted from inside a personal device such as a smart card. For verification system, the Hidden Markov Models (HMMs) are trained as writer-dependent classifiers for each user to build the complete verification system.

- **[123]**: This systems uses the Local Binary Patterns (LBP) as the feature extractor. Then, it trains SVMs as writer-dependent classifiers for each user to build the verification systems.

Table 5.5 shows the results of comparison with state-of-the-art systems on the GPDS-10000 dataset. Here, we randomly select 5 and 10 genuine signatures for each

user to train the RBF kernel SVMs as the writer-dependent classifiers. We can see that the proposed method obtains the best performance (the lowest EER) on all subset of the GPDS-10000 dataset. We note that only using 5 genuine signatures for each user already achieved good results compared with other state-of-the-art systems. When the number of users is 4,000, the EERs in [160] and [163] are 6.81% and 6.13% which are lower than the 7.22% in our proposed method. However, they use 10 genuine signatures for each user for reference, our method only uses 5 genuine signatures for reference. When we use 10 genuine signatures for each user, the best EER is obtained than other systems. This experiment also demonstrates the proposed method is robust and competitive compared with other state-of-the-art systems.

Fig. 5-3 shows the visualization of the pooling features and displacement features of some samples on the GPDS-10000 dataset. Here, the left part is the samples of genuine signatures and the right part is the samples of skilled forgeries. In the same channel, the pooling features between the genuine signatures and corresponding skilled forgeries are very similar. But, the displacement features describe the location information of maximums in the max-pooling operation, which might capture some micro differences of forgery signatures when some skilled writers imitated the genuine signatures.

Fig. 5-4 presents two improved examples by using the proposed method. We visualize the displacement features to observe the micro differences between the genuine signatures and skilled forgeries. If we use traditional CNN based features, they are classified to the same class (positive class). But using the fused features by the proposed method, they can be classified correctly.

From Fig. 5-4, we can see that the genuine signatures have similar behaviors on the displacement features and the skilled forgeries are different from the genuine samples in some places. For the first example, in the bottom left corner of the first filter, the genuine samples have some features in yellow and purple directions, but it is rare in the skilled forgery samples. And in the second filter, the displacement features can

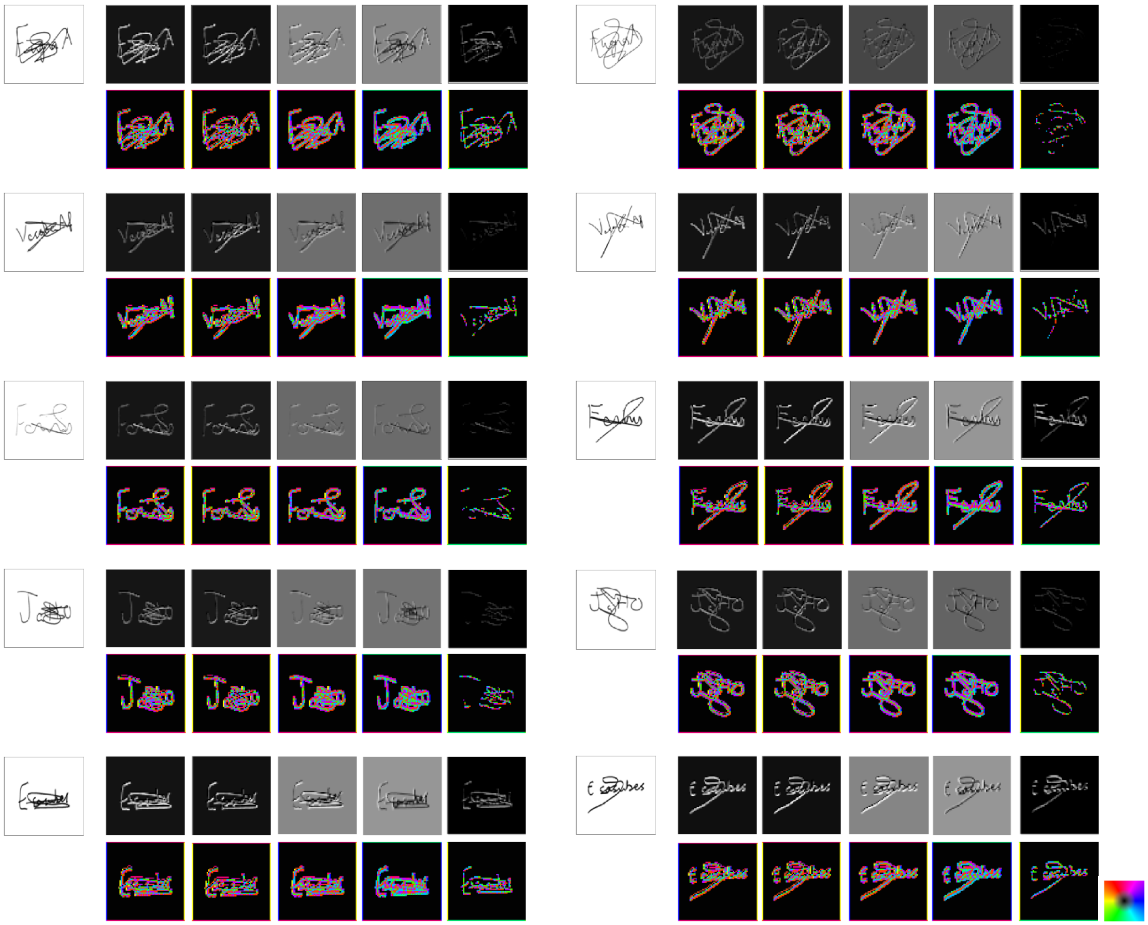

Figure 5-3: Visualization of the pooling features and the displacement features of some samples on GPDS-10000 dataset. The samples on the left are genuine samples and the samples on the right are skilled forgeries. For each sub-figure, the upper left image is the original signature, the first row shows the corresponding pooling features, and the second row shows the displacement features. Each column represents one convolutional filter.

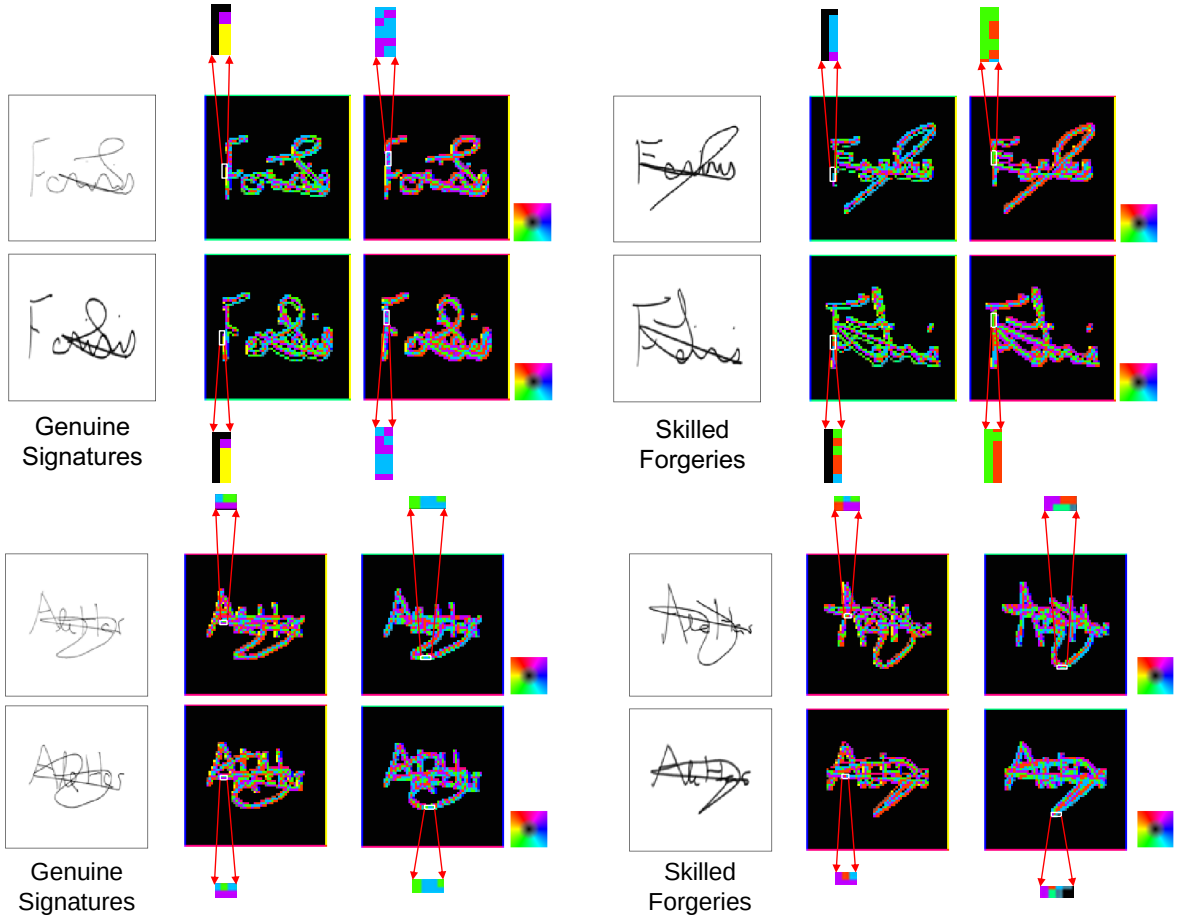

Figure 5-4: Examples improvement of the displacement features by capturing the micro differences between the genuine signatures and skilled forgeries. Here, all signatures are from the same user. The first column is the original signature images, the second and the third columns are the displacement features extracted from two filters.

capture some blue and purple directions in the genuine signatures, but in the skilled forgeries, the corresponding position is green and red. For the second example, in the first filter, under the part of 'A', the displacement features can capture some blue, green, and purple directions in the genuine signatures, but in the skilled forgeries, the corresponding position is purple and red. In the second filter, at the bottom part, the displacement features can capture some blue and green directions in the genuine signatures, but in the skilled forgeries, the corresponding position is purple, green, and red. It means that the proposed method can capture the micro differences or distortions between the genuine signatures and their corresponding skilled forgeries from different convolutional filters.

5.5 Summary

In this chapter, we present how to apply the proposed displacement features for offline signature verification task. First, we train a CNN between the genuine signatures and skilled forgeries. Then, based on the trained CNN, we extract the displacement features from the first convolutional layer. The displacement features capture the intra-class micro differences between genuine signatures and their corresponding skilled forgeries. We prove that combining the displacement features with the traditional pooling features could dramatically improve the CNN based architectures for offline signature verification tasks, especially in discriminating the genuine signatures and their corresponding skilled forgeries. We achieve state-of-the-art results on GPDS-1000 datasets and demonstrate the excellent performance of the proposed verification system by applying the displacement features for the feature extraction procedure.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we introduce the negative affects of the traditional max-pooling operation that absorbs the inter-class or intra-class micro differences between different samples. The elimination of inter-class or intra-class micro differences are very common for recognition and identification tasks respectively. The motivation of this thesis is to detect and penalize the cases when the max-pooling operation is going to remove the important inter-class or inter-individual micro differences.

To address this issue, a new feature named a displacement feature is extracted from the max-pooling operation. The displacement features record the location information of the maximum values in pooling windows. The displacement features could discriminate unnecessary absorptions from necessary absorptions in max-pooling operation and capture inter-class or intra-class micro differences between different samples. To improve the performance of the traditional CNN based architectures, we fused the displacement features with the pooling features for text recognition and offline signature verification tasks.

For the text recognition tasks, the proposed displacement features aim to capture inter-class micro differences between similar classes. First, the displacement features

and pooling features are extracted from the first convolutional layer in a pre-trained CNN. Then, we combined the displacement features with pooling features for training another CNN. Next, cosine features based on the displacement features are proposed to further improve the performance for different text recognition tasks.

We designed a series of experiments on three text recognition tasks, MNIST, HASY, and Chars74K-font datasets, and compared the proposed method with traditional CNN based models and state-of-the-art models. Extensive experimental results demonstrate that the displacement features can improve the CNN based architectures and achieve state-of-the-art results. In addition, we also conducted many meaningful analyses on the MNIST dataset. It included the visualization and distribution of the displacement features and the class-wise similarity of the displacement features in the PCA subspaces. We found that the displacement features can find some unnecessary absorptions in max-pooling operation, which is very useful for some specific text recognition tasks.

For the offline signature verification tasks, the proposed displacement features aim to capture intra-class micro differences between the genuine signatures and their corresponding skilled forgeries. First, a CNN is trained between the genuine signatures and skilled forgeries from a large scale dataset to capture the general behaviors of all signatures. Then, based on this pre-trained CNN, we extracted the displacement features from the first convolutional layer and fused the displacement features with pooling features to capture the micro differences between the genuine signatures and their corresponding skilled forgeries. Next, we took this procedure as a feature extraction procedure and extracted the features from the fully connected layers. To build the complete verification system, we fed the extracted features to SVMs as the writer-dependent classifiers for each user.

The extensive experiments and analyses are conducted on GPDS-150, GPDS-300, GPDS-1000, GPDS-2000, and GPDS-5000 datasets. The results showed that the proposed method can outperform the baseline and state-of-the-art offline signature

verification systems. We also proved that the CNNs have the potential to capture the micro differences in max-pooling layers.

6.2 Future Work

For future work, it is expected that the displacement features can greatly improve the performance of different architectures and applications. We list several potential research directions to further explore the displacement features as follows.

- Attention models can be designed by using the displacement features to control the convolutional features automatically. In recent research, attention models are applied to different architectures to learn "where" and "what" is important in feature maps for the target tasks. Our goal is to fuse the convolutional features and displacement features by designing novel attention models and increase representation power by using an attention mechanism.

- The displacement features extracted from deeper max-pooling layers should be considered in future. It is well known that the features from deep convolutional layers are more abstract than previous layers. It is better to study the behaviors of displacement features in deeper layers and compare the displacement features from different layers.

- The negative effects of the displacement features should be considered and evaluated in future. In this thesis, it was proved that the displacement features can capture the inter-class or intra-class micro differences between different samples. However, if the original samples undergo random or some adversarial noises, the displacement features should change drastically because they are sensitive to the change of the input values. Designing a novel approach to address this problem is also interesting and promising.

- For the offline signature verification tasks, the convolutional features can be

applied to capture the main differences between different users, and the displacement features can be applied to capture the micro differences between the genuine signatures and skilled forgeries respectively. Then, combining the two architectures as the feature extraction procedure for verification systems. Besides, the number of users for training the feature extractor can be evaluated in future.

Bibliography

[1] M. Thoma, "The HASYv2 dataset," *arXiv preprint arXiv:1701.08380*, 2017.

[2] H. Xie, D. Yang, N. Sun, Z. Chen, and Y. Zhang, "Automated Pulmonary Nodule Detection in CT Images Using Deep Convolutional Neural Networks," *Pattern Recognition*, vol. 85, pp. 109–119, 2019.

[3] D. Lu, M. Heisler, S. Lee, G. W. Ding, E. Navajas, M. V. Sarunic, and M. F. Beg, "Deep-learning Based Multiclass Retinal Fluid Segmentation and Detection in Optical Coherence Tomography Images Using a Fully Convolutional Neural Network," *Medical Image Analysis*, vol. 54, pp. 100–110, 2019.

[4] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, "Espnetv2: A Lightweight, Power Efficient, and General Purpose Convolutional Neural Network," in *Proceedings of the CVPR*, 2019, pp. 9190–9200.

[5] Z. Gong, P. Zhong, Y. Yu, W. Hu, and S. Li, "A CNN with Multiscale Convolution and Diversified Metric for Hyperspectral Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 6, pp. 3599–3618, 2019.

[6] L. Fang, C. Wang, S. Li, H. Rabbani, X. Chen, and Z. Liu, "Attention to Lesion: Lesion-aware Convolutional Neural Network for Retinal Optical Coherence Tomography Image Classification," *IEEE Transactions on Medical Imaging*, vol. 38, no. 8, pp. 1959–1970, 2019.

[7] M. Giménez, J. Palanca, and V. Botti, "Semantic-based Padding in Convolutional Neural Networks for Improving the Performance in Natural Language Processing. A Case of Study in Sentiment Analysis," *Neurocomputing*, vol. 378, pp. 315–323, 2020.

[8] P. Li and K. Mao, "Knowledge-oriented Convolutional Neural Network for Causal Relation Extraction from Natural Language Texts," *Expert Systems with Applications*, vol. 115, pp. 512–523, 2019.

[9] Y. Zhang, Z. Zhang, D. Miao, and J. Wang, "Three-way Enhanced Convolutional Neural Networks for Sentence-level Sentiment Classification," *Information Sciences*, vol. 477, pp. 55–64, 2019.

[10] J. Guo, H. He, T. He, L. Lausen, M. Li, H. Lin, X. Shi, C. Wang, J. Xie, S. Zha *et al.*, "Gluoncv and Gluonnlp: Deep Learning in Computer Vision and Natural Language Processing," *Journal of Machine Learning Research*, vol. 21, no. 23, pp. 1–7, 2020.

[11] P. Maergner, V. Pondenkandath, M. Alberti, M. Liwicki, K. Riesen, R. Ingold, and A. Fischer, "Combining Graph Edit Distance and Triplet Networks for Offline Signature Verification," *Pattern Recognition Letters*, vol. 125, pp. 527–533, 2019.

[12] M. Liao, J. Zhang, Z. Wan, F. Xie, J. Liang, P. Lyu, C. Yao, and X. Bai, "Scene Text Recognition from Two-dimensional Perspective," in *Proceedings of the AAAI*, vol. 33, 2019, pp. 8714–8721.

[13] Z.-R. Wang, J. Du, and J.-M. Wang, "Writer-aware CNN for Parsimonious HMM-based Offline Handwritten Chinese Text Recognition," *Pattern Recognition*, vol. 100, p. 107102, 2020.

[14] Y. Zheng, Y. Cai, G. Zhong, Y. Chherawala, Y. Shi, and J. Dong, "Stretching Deep Architectures for Text Recognition," in *Proceedings of the ICDAR*, 2015, pp. 236–240.

[15] F. Wang, L. Zhao, X. Li, X. Wang, and D. Tao, "Geometry-aware Scene Text Detection with Instance Transformation Network," in *Proceedings of the CVPR*, 2018, pp. 1381–1389.

[16] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the NIPS*, 2012, pp. 1097–1105.

[17] 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.

[18] R. Girshick, "Fast R-CNN," in *Proceedings of the ICCV*, 2015, pp. 1440–1448.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the CVPR*, 2016, pp. 770–778.

[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017.

[21] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *Proceedings of the CVPR*, 2018, pp. 6848–6856.

[22] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet V2: Practical Guidelines for Efficient CNN Architecture Design," in *Proceedings of the ECCV*, 2018, pp. 116–131.

[23] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A Theoretical Analysis of Feature Pooling in Visual Recognition," in *Proceedings of the ICML*, 2010, pp. 111–118.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–444", 2015.

[25] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Patial Transformer Networks," in *Proceedings of the NIPS*, 2015, pp. 2017–2025.

[26] Y. Zheng, B. K. Iwana, and S. Uchida, "Discovering Class-Wise Trends of Max-Pooling in Subspace," in *Proceedings of the ICFHR*, 2018, pp. 98–103.

[27] ——, "Mining the Displacement of Max-pooling for Text Recognition," *Pattern Recognition*, vol. 93, pp. 558–569, 2019.

[28] M. A. Ferrer, M. Diaz, C. Carmona-Duarte, and A. Morales, "A Behavioral Handwriting Model for Static and Dynamic Signature Synthesis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1041–1053, 2017.

[29] Y.-C. Wu, F. Yin, and C.-L. Liu, "Improving Handwritten Chinese Text Recognition Using Neural Network Language Models and Convolutional Neural Network Shape Models," *Pattern Recognition*, vol. 65, pp. 251–264, 2017.

[30] Z.-R. Wang, J. Du, W.-C. Wang, J.-F. Zhai, and J.-S. Hu, "A Comprehensive Study of Hybrid Neural Network Hidden Markov Model for Offline Handwritten Chinese Text Recognition," *International Journal on Document Analysis and Recognition*, vol. 21, no. 4, pp. 241–251, 2018.

[31] H. Xie, S. Fang, Z.-J. Zha, Y. Yang, Y. Li, and Y. Zhang, "Convolutional Attention Networks for Scene Text Recognition," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 15, no. 1s, pp. 1–17, 2019.

[32] X. Liu, G. Meng, and C. Pan, "Scene Text Detection and Recognition with Advances in Deep Learning: a Survey," *International Journal on Document Analysis and Recognition*, vol. 22, no. 2, pp. 143–162, 2019.

[33] F. Zhan and S. Lu, "Esir: End-to-end Scene Text Recognition via Iterative Image Rectification," in *Proceedings of the CVPR*, 2019, pp. 2059–2068.

[34] S. Shariatmadari, S. Emadi, and Y. Akbari, "Patch-based Offline Signature Verification Using One-class Hierarchical Deep Learning," *International Journal on Document Analysis and Recognition*, vol. 22, no. 4, pp. 375–385, 2019.

[35] P. Wei, H. Li, and P. Hu, "Inverse Discriminative Networks for Handwritten Signature Verification," in *Proceedings of the CVPR*, 2019, pp. 5764–5772.

[36] C. Li, F. Lin, Z. Wang, G. Yu, L. Yuan, and H. Wang, "DeepHSV: User-Independent Offline Signature Verification Using Two-Channel CNN," in *Proceedings of the ICDAR*, 2019, pp. 166–171.

[37] M. Berkay Yilmaz and K. Ozturk, "Hybrid User-independent and User-dependent Offline Signature Verification with a Two-channel CNN," in *Proceedings of the CVPR Workshops*, 2018, pp. 526–534.

[38] C. Adak, S. Marinai, B. B. Chaudhuri, and M. Blumenstein, "Offline Bengali Writer Verification by PDF-CNN and Siamese Net," in *Proceedings of the DAS*, 2018, pp. 381–386.

[39] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale Orderless Pooling of Deep Convolutional Activation Features," in *Proceedings of the ECCV*, 2014, pp. 392–407.

[40] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, "Compact Bilinear Pooling," in *Proceedings of CVPR*, 2016, pp. 317–326.

[41] S. S. Husain and M. Bober, "REMAP: Multi-layer Entropy-guided Pooling of Dense CNN Features for Image Retrieval," *IEEE Transactions on Image Processing*, vol. 28, no. 10, pp. 5201–5213, 2019.

[42] B. Graham, "Fractional Max-pooling," *arXiv preprint arXiv:1412.6071*, 2014.

[43] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed Pooling for Convolutional Neural Networks," in *Proceedings of RSKD*, 2014, pp. 364–375.

[44] Z. Wei, J. Zhang, L. Liu, F. Zhu, F. Shen, Y. Zhou, S. Liu, Y. Sun, and L. Shao, "Building Detail-Sensitive Semantic Segmentation Networks With Polynomial Pooling," in *Proceedings of CVPR*, 2019, pp. 7115–7123.

[45] G. Tong, Y. Li, W. Zhang, D. Chen, Z. Zhang, J. Yang, and J. Zhang, "Point Set Multi-level Aggregation Feature Extraction Based on Multi-scale Max Pooling and LDA for Point Cloud Classification," *Remote Sensing*, vol. 11, no. 23, p. 2846, 2019.

[46] K. Yue, F. Xu, and J. Yu, "Shallow and Wide Fractional Max-pooling Network for Image Classification," *Neural Computing and Applications*, vol. 31, no. 2, pp. 409–419, 2019.

[47] S. Zhai, H. Wu, A. Kumar, Y. Cheng, Y. Lu, Z. Zhang, and R. Feris, "S3pool: Pooling with Stochastic Spatial Sampling," in *Proceedings of CVPR*, 2017, pp. 4970–4978.

[48] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving Neural Networks by Preventing Co-adaptation of Feature Detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[49] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of Neural Networks Using Dropconnect," in *Proceedings of the ICML*, 2013, pp. 1058–1066.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[51] F. Saeedan, N. Weber, M. Goesele, and S. Roth, "Detail-preserving Pooling in Deep Networks," in *Proceedings of the CVPR*, 2018, pp. 9108–9116.

[52] T. Kobayashi, "Global Feature Guided Local Pooling," in *Proceedings of the ICCV*, 2019, pp. 3365–3374.

[53] Z. Gao, L. Wang, and G. Wu, "LIP: Local Importance-based Pooling," in *Proceedings of the ICCV*, 2019, pp. 3355–3364.

[54] S. Wang, Y. Guan, and L. Shao, "Multi-Granularity Canonical Appearance Pooling for Remote Sensing Scene Classification," *IEEE Transactions on Image Processing*, vol. 29, pp. 5396–5407, 2020.

[55] C.-Y. Lee, A. Bhardwaj, W. Di, V. Jagadeesh, and R. Piramuthu, "Region-based Discriminative Feature Pooling for Scene Text Recognition," in *Proceedings of the CVPR*, 2014, pp. 4050–4057.

[56] S. Lai, L. Jin, and W. Yang, "Toward High-performance Online HCCR: A CNN Approach with DropDistortion, Path Signature and Spatial Stochastic Max-pooling," *Pattern Recognition Letters*, vol. 89, pp. 60–66, 2017.

[57] D. Nguyen, S. Lu, S. Tian, N. Ouarti, and M. Mokhtari, "A Pooling Based Scene Text Proposal Technique for Scene Text Reading in the Wild," *Pattern Recognition*, vol. 87, pp. 118–129, 2019.

[58] Z. Zhong, L. Jin, and Z. Feng, "Multi-font Printed Chinese Character Recognition Using Multi-pooling Convolutional Neural Network," in *Proceedings of the ICDAR)*, 2015, pp. 96–100.

[59] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, "Semantic Segmentation with Second-order Pooling," in *Proceedings of the ECCV*, 2012, pp. 430–443.

[60] S. R. Bulo, G. Neuhold, and P. Kontschieder, "Loss Max-pooling for Semantic Image Segmentation," in *Proceedings of the CVPR*, 2017, pp. 7082–7091.

[61] Y. He, W.-C. Chiu, M. Keuper, and M. Fritz, "Std2p: RGBD Semantic Segmentation using Spatio-temporal Data-driven Pooling," in *Proceedings of the CVPR*, 2017, pp. 4837–4846.

[62] F. Zhou, Y. Hu, and X. Shen, "Scale-aware Spatial Pyramid Pooling with Both Encoder-mask and Scale-attention for Semantic Segmentation," *Neurocomputing*, vol. 383, pp. 174–182, 2020.

[63] R. Girdhar and D. Ramanan, "Attentional Pooling for Action Recognition," in *Proceedings of the NIPS*, 2017, pp. 34–45.

[64] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical Graph Representation Learning with Differentiable Pooling," in *Proceedings of the NeurIPS*, 2018, pp. 4800–4810.

[65] Y. Zhang, S. Tang, K. Muandet, C. Jarvers, and H. Neumann, "Local Temporal Bilinear Pooling for Fine-grained Action Parsing," in *Proceedings of the CVPR*, 2019, pp. 12 005–12 015.

[66] J.-J. Liu, Q. Hou, M.-M. Cheng, J. Feng, and J. Jiang, "A Simple Pooling-based Design for Real-time Salient Object Detection," in *Proceedings of the CVPR*, 2019, pp. 3917–3926.

[67] Z. Huang, J. Wang, X. Fu, T. Yu, Y. Guo, and R. Wang, "DC-SPP-YOLO: Dense Connection and Spatial Pyramid Pooling Based YOLO for Object Detection," *Information Sciences*, vol. 522, pp. 241–258, 2020.

[68] R. Qian, Y. Yue, F. Coenen, and B. Zhang, "Traffic Sign Recognition with Convolutional Neural Network Based on Max Pooling Positions," in *Proceedings of the ICNC-FSKD*, 2016, pp. 578–582.

[69] J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun, "Stacked What-Where Auto-encoders," in *Proceedings of the ICLR Workshop*, 2015.

[70] K. Sun, L. Li, L. Li, N. He, and J. Zhu, "Spatial Attentional Bilinear 3D Convolutional Network for Video-Based Autism Spectrum Disorder Detection," in *Proceedings of the ICASSP*, 2020, pp. 3387–3391.

[71] F. Guo, R. He, and J. Dang, "Implicit Discourse Relation Recognition via a BiLSTM-CNN Architecture with Dynamic Chunk-based Max Pooling," *IEEE Access*, vol. 7, pp. 169 281–169 292, 2019.

[72] Z. Ma, D. Chang, J. Xie, Y. Ding, S. Wen, X. Li, Z. Si, and J. Guo, "Fine-grained Vehicle Classification with Channel Max Pooling Modified CNNs," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3224–3233, 2019.

[73] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski, "An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution," *arXiv preprint arXiv:1807.03247, 2018*, 2018.

[74] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable Convolutional Networks," in *Proceedings of the ICCV*, 2017, pp. 764–773.

[75] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," in *Proceedings of the NIPS*, 2016, pp. 379–387.

[76] P. M. Manwatkar and K. R. Singh, "A Technical Review on Text Recognition from Images," in *Proceedings of the ISCO*, 2015, pp. 1–5.

[77] M. Pejić Bach, Ž. Krstić, S. Seljan, and L. Turulja, "Text Mining for Big Data Analysis in Financial Sector: A Literature Review," *Sustainability*, vol. 11, no. 5, p. 1277, 2019.

[78] C. Lewis and S. Young, "Fad or Future? Automated Analysis of Financial Text and Its Implications for Corporate Reporting," *Accounting and Business Research*, vol. 49, no. 5, pp. 587–615, 2019.

[79] M. Jha, M. Kabra, S. Jobanputra, and R. Sawant, "Automation of Cheque Transaction Using Deep Learning and Optical Character Recognition," in *Proceedings of the ICSSIT*, 2019, pp. 309–312.

[80] M. Lei, Y. Zhou, L. Zhou, J. Zheng, M. Li, and L. Zou, "Noise-robust Wagon Text Extraction Based on Defect-restore Generative Adversarial Network," *IEEE Access*, vol. 7, pp. 168 236–168 246, 2019.

[81] F. Ali, S. El-Sappagh, and D. Kwak, "Fuzzy Ontology and LSTM-based Text Mining: a Transportation Network Monitoring System for Assisting Travel," *Sensors*, vol. 19, no. 2, p. 234, 2019.

[82] F. Ali, D. Kwak, P. Khan, S. El-Sappagh, A. Ali, S. Ullah, K. H. Kim, and K.-S. Kwak, "Transportation sentiment analysis using word embedding and ontology-based topic modeling," *Knowledge-Based Systems*, vol. 174, pp. 27–42, 2019.

[83] S. Zhao, Z. Cai, H. Chen, Y. Wang, F. Liu, and A. Liu, "Adversarial Training Based Lattice LSTM for Chinese Clinical Named Entity Recognition," *Journal of Biomedical Informatics*, vol. 99, p. 103290, 2019.

[84] B. Tang, X. Wang, J. Yan, and Q. Chen, "Entity Recognition in Chinese Clinical Text Using Attention-based CNN-LSTM-CRF," *BMC Medical Informatics and Decision Making*, vol. 19, no. 3, p. 74, 2019.

[85] N. Vaci, Q. Liu, A. Kormilitzin, F. De Crescenzo, A. Kurtulmus, J. Harvey, B. O'Dell, S. Innocent, A. Tomlinson, A. Cipriani *et al.*, "Natural Language Processing for Structuring Clinical Text Data on Depression Using UK-CRIS," *Evidence-Based Mental Health*, vol. 23, no. 1, pp. 21–26, 2020.

[86] D. G. Elliman and I. T. Lancaster, "A Review of Segmentation and Contextual Analysis Techniques for Text Recognition," *Pattern Recognition*, vol. 23, no. 3-4, pp. 337–346, 1990.

[87] P. K. Charles, V. Harish, M. Swathi, and C. Deepthi, "A Review on the Various Techniques Used for Pptical Character Recognition," *International Journal of Engineering Research and Applications*, vol. 2, no. 1, pp. 659–662, 2012.

[88] D. Ghosh, T. Dube, and A. Shivaprasad, "Script Recognitionąła Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2142–2161, 2010.

[89] K. Santosh, "Character Recognition Based on DTW-Radon," in *Proceedings of the ICDAR*, 2011, pp. 264–268.

[90] K. Santosh, B. Lamiroy, and L. Wendling, "DTW-Radon-based Shape Descriptor for Pattern Recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 27, no. 03, p. 1350008, 2013.

[91] S. Belongie, J. Malik, and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[92] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality Reduction: a Comparative," *Journal of Machine Learning Research*, vol. 10, no. 66-71, p. 13, 2009.

[93] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[94] A. Poznanski and L. Wolf, "CNN-N-Gram for Handwriting Word Recognition," in *Proceedings of the CVPR*, 2016, pp. 2305–2314.

[95] X. Xiao, Y. Yang, T. Ahmad, L. Jin, and 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.

[96] X. Xiao, L. Jin, Y. Yang, W. Yang, J. Sun, and T. Chang, "Building Fast and Compact Convolutional Neural Networks for Offline Handwritten Chinese Character Recognition," *Pattern Recognition*, vol. 72, pp. 72–81, 2017.

[97] J. Gan, W. Wang, and K. Lu, "A New Perspective: Recognizing Online Handwritten Chinese Characters via 1-dimensional CNN," *Information Sciences*, vol. 478, pp. 375–390, 2019.

[98] N. Majid and E. H. B. Smith, "Segmentation-free Bangla Offline Handwriting Recognition Using Sequential Detection of Characters and Diacritics with a Faster R-CNN," in *Proceedings of the ICDAR*, 2019, pp. 228–233.

[99] M. Bušta, L. Neumann, and J. Matas, "Deep Textspotter: An End-to-end Trainable Scene Text Localization and Recognition Framework," in *Proceedings of the ICCV*, 2017, pp. 2223–2231.

[100] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue, "Arbitrary-Oriented Scene Text Detection via Rotation Rroposals," *IEEE Transactions on Multimedia*, vol. 20, pp. 3111–3122, 2018.

[101] X. Rong, C. Yi, and Y. Tian, "Unambiguous Text Localization and Retrieval for Cluttered Scenes," in *Proceedings of the CVPR*, 2017, pp. 3279–3287.

[102] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "EAST: An Efficient and Accurate Scene Text Detector," in *Proceedings of the CVPR*, 2017, pp. 2642–2651.

[103] Z. Zhong, L. Sun, and Q. Huo, "Improved Localization Accuracy by LocNet for Faster R-CNN Based Text Detection in Natural Scene Images," *Pattern Recognition*, vol. 96, p. 106986, 2019.

[104] L. Gomez, A. Nicolaou, and D. Karatzas, "Improving Patch-based Scene Text Script Identification with Ensembles of Conjoined Networks," *Pattern Recognition*, vol. 67, pp. 85–96, 2017.

[105] A. K. Bhunia, A. Konwer, A. K. Bhunia, A. Bhowmick, P. P. Roy, and U. Pal, "Script Identification in Natural Scene Image and Video Frames Using an Attention Based Convolutional-LSTM Network," *Pattern Recognition*, vol. 85, pp. 172–184, 2019.

[106] B. Shi, X. Bai, and C. Yao, "Script Identification in the Wild via Discriminative Convolutional Neural Network," *Pattern Recognition*, vol. 52, pp. 448–458, 2016.

[107] S. Ukil, S. Ghosh, S. M. Obaidullah, K. Santosh, K. Roy, and N. Das, "Improved Word-level Handwritten Indic Script Identification by Integrating Small Convolutional Neural Networks," *Neural Computing and Applications*, pp. 1–16, 2019.

[108] A. K. Bhunia, S. Mukherjee, A. Sain, A. K. Bhunia, P. P. Roy, and U. Pal, "Indic Handwritten Script Identification Using Offline-online Multi-modal Deep Network," *Information Fusion*, vol. 57, pp. 1–14, 2020.

[109] Z. Liu, Y. Li, F. Ren, W. L. Goh, and H. Yu, "SqueezedText: A Real-Time Scene Text Recognition by Binary Convolutional Encoder-Decoder Network," in *Proceedings of the AAAI*, 2018, pp. 7194–7201.

[110] H. Gao, Y. Chen, and S. Ji, "Learning Graph Pooling and Hybrid Convolutional Operations for Text Representations," in *Proceedings of the WWW*, 2019, pp. 2743–2749.

[111] Z. Zhang, H. Wang, S. Liu, and B. Xiao, "Deep Contextual Stroke Pooling for Scene Character Recognition," *IEEE Access*, vol. 6, pp. 16 454–16 463, 2018.

[112] S. Gao, C. Wang, B. Xiao, C. Shi, W. Zhou, and Z. Zhang, "Learning Co-occurrence Strokes for Scene Character Recognition Based on Spatiality Embedded Dictionary," in *Proceedings of the ICIP*, 2014, pp. 5956–5960.

[113] J. Kim, S. Jang, E. Park, and S. Choi, "Text Classification Using Capsules," *Neurocomputing*, vol. 376, pp. 214–221, 2020.

[114] M. I. Malik, S. Ahmed, A. Marcelli, U. Pal, M. Blumenstein, L. Alewijns, and M. Liwicki, "ICDAR2015 Competition on Signature Verification and Writer Identification for On-and Off-line Skilled Forgeries (SigWIcomp2015)," in *Proceedings of the ICDAR*, 2015, pp. 1186–1190.

[115] M. Liwicki, M. I. Malik, C. E. Van Den Heuvel, X. Chen, C. Berger, R. Stoel, M. Blumenstein, and B. Found, "Signature Verification Competition for Online and Offline Skilled Forgeries (Sigcomp2011)," in *Proceedings of the ICDAR*, 2011, pp. 1480–1484.

[116] M. I. Malik, M. Liwicki, L. Alewijnse, W. Ohyama, M. Blumenstein, and B. Found, "ICDAR 2013 Competitions on Signature Verification and Writer Identification for On-and Offline Skilled Forgeries (SigWiComp 2013)," in *Proceedings of the ICDAR*, 2013, pp. 1477–1483.

[117] K. Riesen and R. Schmidt, "Online Signature Verification Based on String Edit Distance," *International Journal on Document Analysis and Recognition*, vol. 22, no. 1, pp. 41–54, 2019.

[118] Y. Ren, C. Wang, Y. Chen, M. C. Chuah, and J. Yang, "Signature Verification Using Critical Segments for Securing Mobile Transactions," *IEEE Transactions on Mobile Computing*, vol. 19, no. 3, pp. 724–739, 2019.

[119] E. N. Zois, D. Tsourounis, I. Theodorakopoulos, A. L. Kesidis, and G. Economou, "A Comprehensive Study of Sparse Representation Techniques for Offline Signature Verification," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 1, no. 1, pp. 68–81, 2019.

[120] M. Diaz, M. A. Ferrer, D. Impedovo, M. I. Malik, G. Pirlo, and R. Plamondon, "A Perspective Analysis of Handwritten Signature Technology," *ACM Computing Surveys*, vol. 51, no. 6, pp. 1–39, 2019.

[121] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Offline Handwritten Signature Verification-Literature Review," in *Proceedings of the IPTA*, 2017, pp. 1–8.

[122] M. A. Ferrer, J. B. Alonso, and C. M. Travieso, "Offline Geometric Parameters for Automatic Signature Verification Using Fixed-point Arithmetic," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 993–997, 2005.

[123] M. A. Ferrer, J. F. Vargas, A. Morales, and A. Ordonez, "Robustness of Offline Signature Verification Based on Gray Level Features," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 966–977, 2012.

[124] J. Ruiz-del Solar, C. Devia, P. Loncomilla, and F. Concha, "Offline Signature Verification Using Local Interest Points and Descriptors," in *Proceedings of the CIAPR*, 2008, pp. 22–29.

[125] M. B. Yilmaz, B. Yanikoglu, C. Tirkaz, and A. Kholmatov, "Offline Signature Verification Using Classifier Combination of HOG and LBP Features," in *Proceedings of the IJCB*, 2011, pp. 1–7.

[126] J. Hu and Y. Chen, "Offline Signature Verification Using Real Adaboost Classifier Combination of Pseudo-dynamic Features," in *Proceedings of the ICDAR*, 2013, pp. 1345–1349.

[127] E. N. Zois, L. Alewijnse, and G. Economou, "Offline Signature Verification and Quality Characterization Using Poset-oriented Grid Features," *Pattern Recognition*, vol. 54, pp. 162–177, 2016.

[128] M. Okawa, "Synergy of Foreground-background Images for Feature Extraction: Offline Signature Verification Using Fisher Vector With Fused KAZE Features," *Pattern Recognition*, vol. 79, pp. 480–489, 2018.

[129] E. N. Zois, A. Alexandridis, and G. Economou, "Writer Independent Offline Signature Verification Based on Asymmetric Pixel Relations and Unrelated Training-testing Datasets," *Expert Systems with Applications*, vol. 125, pp. 14–32, 2019.

[130] A. K. Bhunia, A. Alaei, and P. P. Roy, "Signature Verification Approach Using Fusion of Hybrid Texture Features," *Neural Computing and Applications*, vol. 31, no. 12, pp. 8737–8748, 2019.

[131] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Learning Features for Offline Handwritten Signature Verification Using Deep Convolutional Neural Networks," *Pattern Recognition*, vol. 70, pp. 163–176, 2017.

[132] A. Soleimani, B. N. Araabi, and K. Fouladi, "Deep Multitask Metric Learning for Offline Signature Verification," *Pattern Recognition Letters*, vol. 80, pp. 84–90, 2016.

[133] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Analyzing Features Learned for Offline Signature Verification Using Deep CNNs," in *Proceedings of the ICPR*, 2016, pp. 2989–2994.

[134] E. N. Zois, M. Papagiannopoulou, D. Tsourounis, and G. Economou, "Hierarchical Dictionary Learning and Sparse Coding for Static Signature Verification," in *Proceedings of the CVPR Workshops*, 2018, pp. 432–442.

[135] K. Ahrabian and B. Babaali, "Usage of Autoencoders and Siamese Networks for Online Handwritten Signature Verification," *Neural Computing and Applications*, vol. 31, no. 12, pp. 9321–9334, 2017.

[136] L. G. Hafemann, L. S. Oliveira, and R. Sabourin, "Fixed-sized Representation Learning from Offline Handwritten Signatures of Different Sizes," *International Journal on Document Analysis and Recognition*, vol. 21, no. 3, pp. 219–232, 2018.

[137] Z. Zhang, X. Liu, and Y. Cui, "Multi-phase Offline Signature Verification System Using Deep Convolutional Generative Adversarial Networks," in *Proceedings of the ISCID*, vol. 2, 2016, pp. 103–107.

[138] S. Lai and L. Jin, "Learning Discriminative Feature Hierarchies for Off-Line Signature Verification," in *Proceedings of the ICFHR*, 2018, pp. 175–180.

[139] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Characterizing and Evaluating Adversarial Examples for Offline Handwritten Signature Verification," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 8, pp. 2153–2166, 2019.

[140] D. Gumusbas and T. Yildirim, "Offline Signature Identification and Verification Using Capsule Network," in *Proceedings of the INISTA*, 2019, pp. 1–5.

[141] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Meta-learning for Fast Classifier Adaptation to New Users of Signature Verification Systems," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1735–1745.

[142] C. Li, X. Zhang, F. Lin, Z. Wang, L. Jun'E, R. Zhang, and H. Wang, "A Stroke-Based RNN for Writer-Independent Online Signature Verification," in *Proceedings of the ICDAR*, 2019, pp. 526–532.

[143] Z. Yan, Z. Yuchen, O. Wataru, S. Daiki, and U. Seiichi, "RankSVM for Offline Signature Verification," *Proceedings of the ICDAR*, pp. 928–933, 2019.

[144] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[145] L. v. d. Maaten and G. Hinton, "Visualizing Data Using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2625, 2008.

[146] Y. Igarashi and K. Fukui, "3D Object Recognition Based on Canonical Angles between Shape Subspaces," in *Proceedings of the ACCV*, 2010, pp. 580–591.

[147] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[148] D. Kumar and A. Ramakrishnan, "Recognition of Kannada Characters Extracted from Scene Images," in *Proceedings of the DAR*, 2012, pp. 15–21.

[149] H. Bourlard and C. J. Wellekens, "Links between Markov Models and Multilayer Perceptrons," in *Proceedings of the NIPS*, 1989, pp. 502–510.

[150] 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*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[151] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[152] S. B. Driss, M. Soua, R. Kachouri, and 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.

[153] 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.

[154] D. G. Lowe, "Object Recognition from Local Scale-invariant Features," in *Proceedings of the ICCV*, vol. 2, 1999, pp. 1150–1157.

[155] L. Neumann and J. Matas, "A Method for Text Localization and Recognition in Real-world Images," in *Proceddings of the ACCV*, 2010, pp. 770–783.

[156] A. C. Berg, T. L. Berg, and J. Malik, "Shape Matching and Object Recognition Using Low Distortion Correspondences," in *Proceedings of the CVPR*, vol. 1, 2005, pp. 26–33.

[157] M. Ali and H. Foroosh, "Character Recognition in Natural Scene Images Using Rank-1 Tensor Decomposition," in *Proceedings of the ICIP*, 2016, pp. 2891–2895.

[158] Z.-J. Xing, F. Yin, Y.-C. Wu, and C.-L. Liu, "Offline Signature Verification Using Convolution Siamese Network," in *Proceedings of the ICGIP*, vol. 10615, 2018, p. 106151I.

[159] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Deep Metric Learning for Person Re-identification," in *Proceedings of the ICPR*, 2014, pp. 34–39.

[160] O. Mersa, F. Etaati, S. Masoudnia, and B. N. Araabi, "Learning Representations from Persian Handwriting for Offline Signature Verification, a Deep Transfer Learning Approach," *arXiv preprint arXiv:1903.06249*, 2019.

[161] M. B. Yılmaz and K. Öztürk, "Recurrent Binary Patterns and CNNs for Offline Signature Verification," in *Proceedings of the FTC*, 2019, pp. 417–434.

[162] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," in *Proceedings of the NIPS*, 2009, pp. 545–552.

[163] S. Masoudnia, O. Mersa, B. N. Araabi, A.-H. Vahabie, M. A. Sadeghi, and M. N. Ahmadabadi, "Multi-Representational Learning for Offline Signature Verification using Multi-Loss Snapshot Ensemble of CNNs," *Expert Systems with Applications*, vol. 133, pp. 317–330, 2019.

[164] K. Janocha and W. M. Czarnecki, "On Loss Functions for Deep Neural Networks in Classification," *arXiv preprint arXiv:1702.05659*, 2017.

[165] Y. Tang, "Deep Learning Using Linear Support Vector Machines," *arXiv preprint arXiv:1306.0239*, 2013.

[166] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-scale Image Recognition," in *Proceedings of the ICLR*, 2015.

[167] P. Maergner, N. Howe, K. Riesen, R. Ingold, and A. Fischer, "Offline Signature Verification via Structural Methods: Graph Edit Distance and Inkball Models," in *Proceedings of the ICFHR*, 2018, pp. 163–168.

[168] P. Maergner, K. Riesen, R. Ingold, and A. Fischer, "A Structural Approach to Offline Signature Verification Using Graph Edit Distance," in *Proceedings of the ICDAR*, vol. 1, 2017, pp. 1216–1222.

[169] N. R. Howe, "Part-structured Inkball Models for One-shot Handwritten Word Spotting," in *Proceedings of the ICDAR*, 2013, pp. 582–586.

[170] P. Maergner, V. Pondenkandath, M. Alberti, M. Liwicki, K. Riesen, R. Ingold, and A. Fischer, "Offline Signature Verification by Combining Graph Edit Distance and Triplet Networks," in *Proceedings of the S+SSPR*, 2018, pp. 470–480.

[171] E. Hoffer and N. Ailon, "Deep Metric Learning Using Triplet Network," in *Proceedings of the SIMBAD*, 2015, pp. 84–92.

[172] K. Riesen and H. Bunke, "Approximate Graph Edit Distance Computation by Means of Bipartite Graph Matching," *Image and Vision Computing*, vol. 27, no. 7, pp. 950–959, 2009.

[173] J. Hu, J. Lu, and Y.-P. Tan, "Discriminative Deep Metric Learning for Face Verification in the Wild," in *Proceedings of the CVPR*, 2014, pp. 1875–1882.

[174] Y. Serdouk, H. Nemmour, and Y. Chibani, "Handwritten Signature Verification Using the Quad-tree Histogram of Templates and a Support Vector-based Artificial Immune Classification," *Image and Vision Computing*, vol. 66, pp. 26–35, 2017.

[175] S. Tang and S. Goto, "Histogram of Template for Human Detection," in *Proceedings of the ICASSP*, 2010, pp. 2186–2189.

[176] A. B. Watkins, "AIRS: A Resource Limited Artificial Immune Classifier," Ph.D. dissertation, Mississippi State University Mississippi, 2001.

[177] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Proceedings of the NIPS*, 2014, pp. 2672–2680.

[178] G. S. Eskander, R. Sabourin, and E. Granger, "Hybrid Writer-independent-writer-dependent Offline Signature Verification System," *IET Biometrics*, vol. 2, no. 4, pp. 169–181, 2013.

[179] P. Maergner, N. R. Howe, K. Riesen, R. Ingold, and A. Fischer, "Graph-Based Offline Signature Verification," *arXiv preprint arXiv:1906.10401*, 2019.

[180] P. Maergner, K. Riesen, R. Ingold, and A. Fischer, "Offline Signature Verification Based on Bipartite Approximation of Graph Edit Distance," in *Proceedings of the IGSC*, 2017, pp. 1–6.

[181] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial Structures for Object Recognition," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 55–79, 2005.