Recognition of Offline Handwritten Mathematical Symbols Using Convolutional Neural Networks

Lanfang Dong(✉) and Hanchao Liu

School of Computer Science and Technology,
University of Science and Technology of China, Hefei, China
lfdong@ustc.edu.cn, lhanchao@mail.ustc.edu.cn

Abstract. This paper presents a method of Convolutional Neural Networks (CNN) to recognize offline handwritten mathematical symbols. In this paper, we propose a CNN model called HMS-VGGNet, in which the Batch Normalization and Global Average Pooling methods and only very small, specifically, 1×1 and 3×3 convolutional filters are applied. HMS-VGGNet uses only offline features of the symbols and has achieved the state-of-the-art accuracies in Competition on Recognition of Online Handwritten Mathematical Expressions (CROHME) 2014 test set and HASYv2 dataset. In CROHME 2016 test set, our result is only 0.39% less than the winner of CROHME 2016 who has used both online and offline features. The models proposed in this paper are accurate yet slim, which will be shown in our experiments.

Keywords: Batch normalization · Global average pooling
Very small convolutional filters · CROHME · HASYv2

1 Introduction

Handwritten mathematical symbols recognition is an essential component of handwritten mathematical expressions recognition which could convert the handwritten mathematical symbols images or traces to specific styles which could be shown and edited in computers, for example LATEX. This task, which has both offline and online model, is still a great challenge owning to its large scale classes, great differences in handwritten styles and very similar symbols. The input of online handwritten mathematical symbols recognition is the timing sampling point sequences gotten from pen-based or touch-based devices, such as smartphones and tablets, while in offline model the input is the images of symbols after written.

Currently, online handwritten mathematical symbols recognition has been studied widely and has achieved great performance. Competition on Recognition of Online Handwritten Mathematical Expressions (CROHME) has been held 5 times from 2011 to 2016 [1–3], attracting the researchers around the world. CROHME represents the highest performance of online handwritten mathematical expression recognition. Recognition of handwritten mathematical symbols has become an isolated task of the competition since CROHME 2014. Owning to the available tracing information, online

Y. Zhao et al. (Eds.): ICIG 2017, Part I, LNCS 10666, pp. 149–161, 2017.
https://doi.org/10.1007/978-3-319-71607-7_14

data can be converted to offline images. In recent years, researchers used the offline features extracted from the symbol images as an auxiliary to recognize the online mathematical symbols and got great achievements. Álvaro et al. [2, 4] combined 9 offline features including PRHLT and FKI features and 7 online features extracted from symbol images and the original online data separately. They used the Bidirectional Long Short Term Memory Recurrent Neural Networks (BLSTM) to classify the features and achieved 91.24% recognition rate in CROHME 2014. Dai et al. [5] used Convolutional Neural Network (CNN) and BLSTM to classify the symbol images and online data separately and combined the results and got 91.28% in CROHME 2014 test set. Davila et al. [6] also used a combination of online features such as normalized line length and covariance of point coordinates and offline features such as 2D fuzzy histograms of points and fuzzy histograms of orientations of the lines to recognize online symbols. MyScript [3], the winner of CROHME 2016 also extracted both online and offline features and processed with a combination of Deep MLP and Recurrent Neural Networks. MyScript achieved 92.81% in CROHME 2016 test set and this is the best result in that set as far as we know.

Nevertheless, researchers didn't give much attention on the recognition of offline handwritten mathematical symbols and little work was published. Since the datasets of offline handwritten mathematical symbols are rare, online data of CROHME were used by Ramadhan et al. [7] to generate symbol images for offline symbol recognition. Ranadhan et al. designed a CNN model that was trained using the images converted from CROHME 2014 training set and got 87.72% accuracy in CROHME 2014 test images drawn from online data [7]. However, due to the absence of the features of online data and the rough designed network architecture, the accuracy of [7] is obviously lower compared to the online handwritten mathematical symbols recognition results.

In recent years, convolutional neural network that was proposed by LeCun [8] for offline handwritten digits recognition has enjoyed a great success in lots of computer vision tasks, such as image recognition [9–12], object detection [13, 14] and semantic segmentation [14]. In this paper, we apply CNN to the recognition of offline handwritten mathematical symbols. And we design a deep and slim CNN architecture denoted as HMS-VGGNet. Previous research results have shown that the deeper the network is, the better results the network gets [9–12]. However, when the network goes deeper, it becomes harder to train and the model size usually grows larger. To overcome the difficulties of training and to keep the model size reasonable HMS-VGGNet which is elaborately designed for the recognition of offline handwritten mathematical symbols has applied Batch Normalization (BN) [15], Global Average Pooling (GAP) [16] and very small convolutional kernels. Considering the lack of offline data and for the convenience of comparing results, we use both the images drawn from CROHME dataset and the data of HASYv2 [17] to train and evaluate our models. As shown in our experiments, HMS-VGGNet raises the accuracy of offline handwritten mathematical symbols recognition significantly.

The rest of the paper is organized as follows. In Sect. 2, we give a brief introduction to BN, GAP and the benefits of 1×1 and 3×3 convolutional kernels. The details of the datasets used in our experiments are shown in Sect. 3, and our network configurations are present in Sect. 4. In Sect. 5, our training methods, experiments results and analyses are presented. Section 6 concludes the paper.

2 A Brief Introduction of BN, GAP and Very Small Convolutional Kernels

2.1 Batch Normalization

In the training process of CNN, it is especially hard when the network goes deeper by the fact that the inputs to each layer are affected by the parameters of all preceding layers [15]. Each layer in the network needs to adapt to the change of the inputs distribution, making the training process difficult and slow. Batch Normalization with benefits of accelerating training and achieving better performance is a solution of this problem by guaranteeing the inputs distribution of each layer stable.

In order to achieve the goal, BN takes two steps of input data processing. Firstly, BN normalizes the inputs distribution of each layer in every training step to make it with the mean of 0 and the variance of 1. For one dimension $x^{(k)}$ of the input x, BN normalizes the input by

$$\hat{x}^{(k)} = \frac{x^{(k)} - E\left[x^{(k)}\right]}{\sqrt{Var\left[x^{(k)}\right]}} \tag{1}$$

where $E\left[x^{(k)}\right]$ and $Var\left[x^{(k)}\right]$ are the mean and variance of $x^{(k)}$. However, this normalization step may destroy what the preceding layer can represent. To recover the features that should be learnt by the preceding layer, BN sets two parameters γ and β to learn in the second step. By the processing of

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)} \tag{2}$$

BN can finally make the inputs distribution of layers stable.

2.2 Global Average Pooling

Fully connected layers, following the convolutional or pooling layers, are common in classical CNN models such as LeNet-5 [8], AlexNet [9] and VGGNet [11]. However, fully connected layers are easy to overfit because of the huge number of parameters. In 2013, Lin et al. [16] proposed a new method called global average pooling to replace fully connected layers. In the layer of global average pooling, all the parameters in one feature map are averaged to generate the result, as illustrated in Fig. 1.

GAP layers have 3 benefits: (1) There are no extra parameters in GAP layers thus overfitting is avoided at GAP layers; (2) Since the output of GAP is the average of the whole feature map, GAP will be more robust to spatial translations; (3) Because of the huge number of parameters in fully connected layers which usually take over 50% in all the parameters of the whole network, replacing them by GAP layers can significantly reduce the size of the model, and this makes GAP very popular in model compression [18].

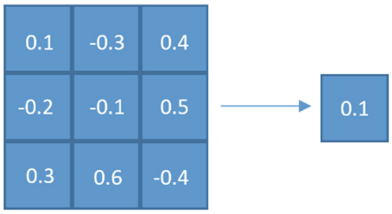

Fig. 1. Process of global average pooling

2.3 1×1 and 3×3 Convolutional Kernels

In recent years, 1×1 and 3×3 filters are widely used in new CNN models [10–12, 18, 19] for their benefits of reducing computations, pruning parameters and improving accuracies.

As a result of keeping the size of feature maps and reducing the number of feature maps with little effect on accuracies, 1×1 convolutional layers were used to reduce parameters and avoid computational blow up in [10, 19]. At the same time, 3×3 filters are the smallest filters that could capture the notion of left/right, up/down and center. Although the receptive fields of 3×3 filters are small, a few continuous 3×3 layers can get the same receptive field of bigger filters, for example, a stack of two 3×3 convolutional layers has an effective receptive field of 5×5, with the advantages of deeper layers and fewer parameters [11, 19].

3 Datasets

In our experiments we use the images converted from CROHME online data and the images of HASYv2 dataset to train and evaluate our models for the lack of offline data and the convenience of comparing results. CROHME is the most commonly used dataset when recognizing handwritten mathematical symbols and HASYv2 which has 151 k handwritten mathematical symbol images is the biggest public offline handwritten mathematical symbols dataset to our best knowledge.

3.1 CROHME Offline Data Generation

There are 101 different classes of mathematical symbol in CROHME 2016 dataset. The online data is given in Ink Markup Language (InkML) [20]. In the InkML file, a symbol S is consisted with a set of trances $\{T_1, T_2, \ldots, T_n\}$. Each trace $T_i(i = 1, \ldots, n)$ consists of a set of timing sampling points $\{p_{i1}, p_{i2}, \ldots, p_{im}\}$, and each point $p_{ij}(i = 1, \ldots, n; j = 1, \ldots, m)$ records its position. When generating symbol images from online data, we connect the points p_{ij} and p_{ij+1} from the same trace with a single line and finish the generation after all the traces from the same symbol are drawn. Due to the different data acquisition devices used in CROHME, the size of symbols differs a lot. In our generation approach, as shown in Algorithm 1, we normalize the symbols size.

Since the aspect ratio, which is an important feature of symbols, differs a lot from different mathematical symbols, the longer side of the images we get from Algorithm 1 is 70 pixels while the shorter is different from each other. We expand image *I* with white pixels to make its size to 70 × 70. Taking into account that 'COMMA', '.' and '\prime' are relatively small in real handwritten symbol images, the longer side are fixed to 16 pixels when drawing these symbol images. After generation we expand these images with white pixels to 70 × 70 pixels. At last we resize the images generated from online data to 48 × 48. The first row in Fig. 2 shows some samples of the generated images.

Algorithm 1. Symbol image generation from online data of CROHME

```
Input:  Online data of Symbol $S = \{T_1, T_2, \ldots, T_n\}$, where $T_i = \{p_{i1}, p_{i2}, \ldots, p_{im}\}, i = 1, \ldots, n$
Output: Symbol image $I$

1. Find the maximum and minimum horizontal and vertical axis of the points in $S$, presenting them by $x_{min}, y_{min}, x_{max}, y_{max}$;
2. For every point in $S$, convert the original position $p_{ij\_0}(x_{ij\_0}, y_{ij\_0})$ to a new position $p_{ij\_1}(x_{ij\_1}, y_{ij\_1})$, where $x_{ij\_1} = x_{ij\_0} - x_{min}$, $y_{ij\_1} = y_{ij\_0} - y_{min}$;
3. Compare $height = y_{max} - y_{min}$ and $width = x_{max} - x_{min}$ and name the longer one as $L$;
4. Scale every sampling point $p_{ij\_1}(x_{ij\_1}, y_{ij\_1})$ obtained in step 2 to get the new position $p_{ij2}(64 * x_{ij\_1}/L, 64 * y_{ij\_1}/L)$;
5. For every $T_i$ in $S$:
     Connect the adjacent points $p_{ij\_2}$ and $p_{ij+1\_2}$ with a line whose thickness is 2 pixels;
6. Extend 3 pixels around the image generated in step 5.
```

3.2 Data Enrichment

As a result of the expressive power of deep networks, overfitting is a common problem that is hard to deal. Researchers have proposed some methods to prevent overfitting such as Dropout [21] and Batch Normalization. However, the most effective way to prevent overfitting is enriching the training set to make the networks learn more universal features. In the training set of CROHME 2016 there are only 85802 symbols and there are 369 classes of 151 k training samples in HASYv2. In addition to the lack of training samples, the distributions of training set of CROHME and HASYv2 are also bias, for example the sample number of symbol '-' is 8390 and there are only 2 samples of '∃' in the training set of CROHME 2016. These drawbacks of the datasets will pull the accuracies down.

To avoid the drawbacks we use elastic distortion [22] to enrich our training set. There are two random matrices $\Delta x(x, y) = rand(-1, 1)$ and $\Delta y(x, y) = rand(-1, 1)$ representing the horizontal and vertical axis displacement of the pixel (x, y) in elastic distortion algorithm. The matrices are convolved with a Gaussian kernel, whose size is $n \times n$ and standard deviation is σ. All the pixels in the original image are moved following the convolution results Δconv_x and Δconv_y. After the movements we rotate the images by a random angle θ. In this paper, σ = 5, n = 11 and θ is in the range of $-25^\circ \sim 25^\circ$. Using elastic distortion, we have enriched the samples of each class to about 4000 and 1000 in CROHME and HASYv2 training sets. The second and third rows in Fig. 2 show several samples generated by elastic distortion.

As HASYv2 covers most of the CROHME symbol classes, we use the samples of HASYv2 whose class is also included in CROHME when conducting the experiments of CROHME. Since the size of images in HASYv2 is 32×32, the images from HASYv2 used in CROHME experiments are resized to 48×48. We use the symbols of CROHME 2013 test set as the validation set and the test set of CROHME 2014 and 2016 to evaluate our models in CROHME experiments. In HASYv2 experiments, we use cross validation as suggested in [17]. Table 1 shows the details of the datasets used in our experiments.

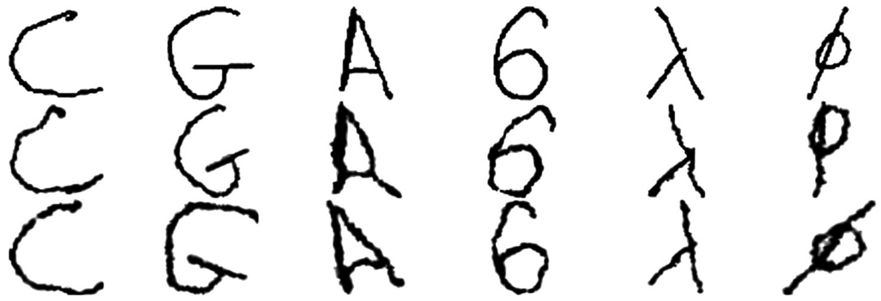

Fig. 2. Samples of the dataset used in our experiments. The first row shows the images drawn from online data, the second and third rows are the samples generated by elastic distortion. Samples of the second and third rows are generated by the images of the first row in the same column.

Table 1. Datasets used in our experiments

Experiments	Usage	Dataset	Image size	Dataset scale	
				Before distortion	After distortion
CROHME	Train	CROHME 2016 train + HASYv2 (part)	48 × 48	132120	403729
	Validation	CROHME 2013 test	48 × 48	6081	–
	Test	CROHME 2014 test	48 × 48	10061	–
		CROHME 2016 test	48 × 48	10019	–
HASYv2	Train	HASYv2 train	32 × 32	151406 ± 166	366566 ± 1356
	Test	HASYv2 test	32 × 32	16827 ± 166	–

4 Network Configurations

In order to make the effects of BN, GAP and small kernels clear, we have designed four networks with similar architecture and the details of these networks are shown in Table 2. Network C is the baseline of our contrast experiments. Network A uses fully connected layers while global average pooling layers are used in C. The only difference of B and C is that C uses Batch Normalization while B doesn't. Compared to C, D adds two extra 1 × 1 convolutional layers to reduce the dimension.

In Table 2, the convolutional layer parameters are denoted as "Conv-(filter size)-(number of filters)-(stride of filters)-(padding pixels)". All max-pooling layers in our network are performed over 2 × 2 pixel window, with stride 2. All convolutional/fully connected layers are equipped with the rectification non-linearity. And all convolutional layers are equipped with Batch Normalization before ReLU except those in network B. We omit the ReLU and BN for brevity in Table 2. The ratios of all the Dropout operations used in our networks are 0.5.

Table 2. HMS-VGGNet configurations (shown in columns). The detailed differences are shown in the contents of this section.

Input: 48 × 48 (CROHME)/32 × 32 (HASYv2) RGB images			
A	B	C	D
Conv-3-32-1-1 Conv-3-32-1-1			
MaxPool			
Conv-3-64-1-1 Conv-3-64-1-1			
MaxPool			
Conv-3-128-1-1 Conv-3-128-1-1			Conv-3-128-1-1 Conv-1-64-1-0 Conv-3-128-1-1
MaxPool			
Conv-3-256-1-1 Conv-3-256-1-1			Conv-3-256-1-1 Conv-1-128-1-0 Conv-3-256-1-1
MaxPool	MaxPool Dropout		
FC-512 Dropout	Conv-1-101-1-0 (CROHME)/ Conv-1-369-1-0 (HASYv2)		
FC-512 Dropout			
FC-101 (CROHME)/ FC-369 (HASYv2)	AveragePool		
Softmax			

The architecture of the networks, which is denoted as HMS-VGGNet, is inspired by VGGNet [11]. However, there are several improvements in our networks for the handwritten mathematical symbols recognition task compared with the original VGGNet. Firstly, the images of handwritten symbol images are much smaller and simpler than the natural images used in VGGNet, so we have pruned several layers and filters to fit our task. The second improvement is that Batch Normalization layers are added after all the convolutional layers of Net A, C and D to accelerate the training process and improve the accuracies. Thirdly, we use global average pooling layers to replace the fully connected layers in B, C and D and reduced the model size by a large margin. Besides, we also apply 1×1 filters which could reduce the model size further and effect the accuracy negligibly to Network D. All the conclusions above will be proven in the experiments of Sect. 5.

5 Experiments

5.1 Experiments in CROHME Dataset

Training Methods. Our experiments were conducted on Caffe framework [23] using a GTX 1060 GPU card. The training used stochastic gradient descent with 0.9 momentum. The initial learning rate was 0.01 and reduced to 10% every 40k iterations. The batch size was set to 40 and the training stopped after 202k iterations (around 20 epochs). Besides, we used the "xavier" algorithm to initialize the weights of all the convolutional layers in our networks.

Results and Analyses. In the CROHME experiments, we use the symbols of CROHME 2013 test set as the validation set. And we use the test sets of CROHME 2014 and 2016 to evaluate the models. Table 3 shows the results of the four networks in these datasets.

All the four networks have achieved great performance in the three datasets. The Top-1 recognition rates of Network C in the CROHME 2014 and CROHME 2016 test sets are about 0.5% to 1% higher than those of Network A and B, while the Top-3 and Top-5 accuracies are also have an improvement about 0.1%–0.5% compared with Network A and B. The gaps between the recognition results of C and D are rather small. C has a 0.39% and 0.15% higher Top-1 performance than D in CROHME 2014 and 2016. And the gaps of Top-3 and Top-5 recognition rates don't exceed 0.1%. These results give strong evidence that the usage of BN and GAP can get better accuracies.

Table 4 elaborates the parameter scales of the four models. Replacing the fully connected layers by global average pooling layers has a sharp decrease of model size. The number of parameters of C is only 44.92% of that of A. After applying the 1×1 convolutional layers, D has a further reduced model size than C and it doesn't have much effect on accuracies compared to C, as illustrated in Tables 3 and 4.

Combining isolated classifiers is an effective way to raise the accuracies which is also used in [4, 5, 9–12]. In order to increase recognition rates further, we have combined network C with D. The ensemble method is averaging the results of the two models. The results of our methods and existing systems in CROHME are shown in

Table 3. Accuracies of our models in CROHME datasets

Dataset	CROHME 2013 test			CROHME 2014 test			CROHME 2016 test		
Network	Top-N accuracies (%)			Top-N accuracies (%)			Top-N accuracies (%)		
	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5
A	**88.46**	99.14	99.62	90.93	98.20	99.04	91.63	98.66	99.41
B	87.80	**99.23**	99.65	90.85	98.32	99.05	91.32	98.71	99.44
C	88.11	99.18	99.62	**91.81**	98.67	**99.18**	**92.16**	99.08	99.51
D	88.39	99.19	**99.67**	91.42	**98.76**	99.12	92.01	**99.16**	**99.52**

Table 5. Our networks outperform all the other systems in CROHME 2014 test set with a 91.82% Top-1 recognition rate. In CROHME 2016 test set, our networks have achieved the second place with 0.39% less than MyScript, the winner of CROHME 2016, who has used both online and offline features. The accuracies of our networks have a significant increase compared with the existing methods that use offline features only in CROHME dataset as shown in Table 5.

Table 6 shows the average computational time of our four networks. Although network C and D spend more time than network A and B, our four networks are all quite fast in our CROHME and HASYv2 experiments.

Table 4. Parameter Scales of HMS-VGGNets. The model size is the size of caffemodel file generated by Caffe. Since the only difference of B and C is the usage of BN, the parameter scales of B and C are the same.

Network	Number of parameters	Model size
A	2.67 M	10.7 MB
B	1.20 M	4.8 MB
C	1.20 M	4.8 MB
D	**0.87 M**	**3.5 MB**

Although we have achieved rather good results in CROHME dataset, there are two questions shown in our experiments.

Question 1: Why the results of A and B in CROHME 2013 (validation set) are only slightly lower or even higher than those of C and D? There are some symbol classes difficult or even impossible to discriminate without context information due to the very confusable handwritten styles, such as ‘x-X-×’, ‘1-|’, ‘0-o-O’. We have analyzed the test sets of CROHME 2013, 2014 and 2016 and find 24 symbol classes that are difficult to classify. The percentage of these classes of CROHME 2013 test set is higher than those of CROHME 2014 and 2016 test sets, as shown in Table 7. This makes it harder to classify in CROHME 2013 test set, so the gaps of recognition rates of A, B, C and D are relatively small. This is also the reason why the Top-1 accuracy is significantly lower than Top-3 and Top-5 accuracies. Some misclassified symbols are illustrated in Fig. 3.

Table 5. Top-1 accuracies of our networks compared with other systems on CROHME. Top 3 accuracies in each dataset are bolded.

System	CROHME 2014 test top-1 accuracy	CROHME 2016 test top-1 accuracy	Features used
MyScript [2, 3]	91.04% [2]	**92.81% [3]**	Online + Offline
Alvaro [2, 4]	91.24% [2]	–	Online + Offline
Dai (1) [5]	89.39% [5]	–	Offline
Dai (2) [3, 5]	91.28% [5]	**92.27% [3]**	Online + Offline
Davila [2, 3, 6]	88.66% [2]	88.85% [3]	Online + Offline
Ramadhan [7]	87.72% [7]	–	Offline
Ours C	**91.81%**	92.16%	Offline
Ours D	**91.42%**	92.01%	Offline
Ours C + D	**91.82%**	**92.42%**	Offline

Table 6. Computational time of our networks in our experiments

Network	CROHME test	HASYv2 test
A	1.40 ms	1.31 ms
B	1.61 ms	1.52 ms
C	1.38 ms	1.25 ms
D	0.91 ms	0.81 ms
C + D	2.84 ms	2.59 ms

Question 2: Why the results of our methods are still less than that of MyScript? Since online data has the tracing information while offline data doesn't, online data has advantages when classifying symbols who have similar shapes and different writing processes such as '5' and 's'. Our networks only use offline features so it is hard for them to classify those symbols.

Table 7. Percentage of symbols hard to classify. These symbol classes are 'COMMA, (, 0, 1, 9, c, C, ., g, l,/, o, p, P, \prime, q, s, S, \times, v, V, |, x, X'

Datasets	Total	Symbols hard to classify	Percentage
CROHME 2013 test	6081	1923	31.62%
CROHME 2014 test	10061	2776	27.59%
CROHME 2016 test	10019	2762	27.57%

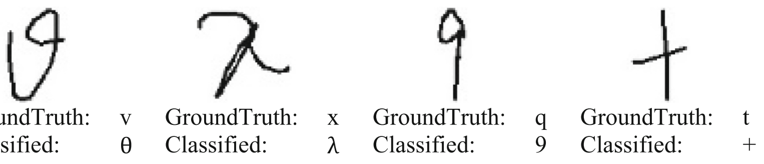

Fig. 3. Misclassified Samples of our networks

5.2 Experiments in HASYv2 Dataset

When conducting experiments in HASYv2 dataset, we have used cross validation to test our models as suggested by [17]. There are 10 folds in HASYv2, so we have evaluated 10 times using different folds in our experiments. The training method is almost the same as that in Sect. 5.1. We totally trained 185 k iterations (around 20 epochs) and divided the learning rate by 10 after every 35 k iterations.

Since the HASYv2 dataset is proposed lately, the other experiments on HASYv2 are still rare. We have compared our results with the model baselines in [17], as shown in Table 8. All the four networks proposed in this paper have higher accuracies than the baselines. Besides, the parameter scales of our networks are significantly smaller than TF-CNN which keeps the highest accuracy in the baselines due to the usage of small convolutional filters, GAP and well-designed architectures. Our models have achieved the state-of-the-art accuracy in HASYv2 dataset to our best knowledge.

Table 8. Accuracies of our networks compared with the model baseline of HASYv2

Classifiers	Top-N accuracies			Number of parameters
	Top-1	Top-3	Top-5	
TF-CNN [17]	81.0%	–	–	4.59 M
Random forest [17]	62.4%	–	–	–
MLP (1 layer) [17]	62.2%	–	–	–
Ours A	84.70%	97.15%	98.33%	2.15 M
Ours B	84.40%	97.14%	98.35%	1.27 M
Ours C	84.90%	97.25%	98.41%	1.27 M
Ours D	84.81%	97.26%	98.48%	0.94 M
Ours C + D	**85.05%**	**97.38%**	**98.52%**	2.20 M

There are 369 classes in HASYv2 dataset and it has more classes that are hard to discriminate than CROHME, such as $\mathcal{H}$, $\mathbb{H}$ and H; $\rightarrow$, $\mapsto$, $\rightharpoonup$, and $\hookrightarrow$. Besides, some symbols are even difficult to tell in printed form, such as \Sigma and \sum. These difficulties make great challenges for our task, so our four networks perform similarly and don't get better accuracies any more.

6 Conclusion

In this paper, we have elaborately designed a CNN architecture called HMS-VGGNet for offline handwritten mathematical symbols recognition. Experiments show that our models have achieved very competitive results in CROHME (91.82% and 92.42% Top-1 accuracy in CROHME 2014 and 2016 and around 99% Top-3 and Top-5 accuracies for both datasets) and HASYv2 (85.13% Top-1 accuracy, 97.38% Top-3 accuracy and 98.52% Top-5 accuracy) datasets using this slim and deep architecture. From our experiments results we also analyse the benefits of BN, GAP and very small filters. We will use these networks in our offline handwritten mathematical expression recognition system in the future. Since online data can generate offline images, our networks can also be used as an auxiliary method for online handwritten mathematical symbols recognition to improve accuracies further.

References

1. Mouchère, H., Zanibbi, R., Garain, U., et al.: Advancing the state of the art for handwritten math recognition: the CROHME competitions, 2011–2014. Int. J. Doc. Anal. Recogn. **19**(2), 173–189 (2016)
2. Mouchere, H., Viard-Gaudin, C., Zanibbi, R., et al.: ICFHR 2014 competition on recognition of on-line handwritten mathematical expressions (CROHME 2014). In: 14th International Conference on Frontiers in Handwriting Recognition, pp. 791–796. IEEE Press, Crete (2014)
3. Mouchère, H., Viard-Gaudin, C., Zanibbi, R., et al.: ICFHR 2016 CROHME: competition on recognition of online handwritten mathematical expressions. In: 15th International Conference on Frontiers in Handwriting Recognition, Shenzhen, pp. 607–612 (2016)
4. Álvaro, F., Sánchez, J.A., Benedí, J.M.: Offline features for classifying handwritten math symbols with recurrent neural networks. In: 2014 22nd International Conference on Pattern Recognition, pp. 2944–2949. IEEE Press, Stockholm (2014)
5. Dai, N.H., Le, A.D., Nakagawa, M.: Deep neural networks for recognizing online handwritten mathematical symbols. In: 2015 3rd IAPR Asian Conference on Pattern Recognition, pp. 121–125. IEEE Press, Kuala Lumpur (2015)
6. Davila, K., Ludi, S., Zanibbi, R.: Using off-line features and synthetic data for on-line handwritten math symbol recognition. In: 2014 14th International Conference on Frontiers in Handwriting Recognition, pp. 323–328. IEEE Press, Crete (2014)
7. Ramadhan, I., Purnama, B., Al, F.S.: Convolutional neural networks applied to handwritten mathematical symbols classification. In: 2016 4th International Conference on Information and Communication Technology, pp. 1–4. IEEE Press, Bandung (2016)
8. LeCun, Y., Bottou, L., Bengio, Y., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
9. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Proceedings of Advances in Neural Information Processing Systems, Lake Tahoe, pp. 1097–1105 (2012)
10. Szegedy, C., Liu, W., Jia, Y., et al.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, Boston, pp. 1–9 (2015)
11. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014). arXiv:1409.1556
12. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, pp. 770–778 (2016)
13. Girshick, R., Donahue, J., Darrell, T., et al.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, Columbus, pp. 580–587 (2014)
14. He, K., Gkioxari, G., Dollár, P., et al.: Mask R-CNN (2017). arXiv:1703.06870
15. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift (2015). arXiv:1502.03167
16. Lin, M., Chen, Q., Yan, S.: Network in network (2013). arXiv:1312.4400
17. Thoma, M.: The HASYv2 dataset (2017). arXiv:1701.08380

18. Iandola, F.N., Han, S., Moskewicz, M.W., et al.: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size (2016). arXiv:1602.07360
19. Szegedy, C., Vanhoucke, V., Ioffe, S., et al.: Rethinking the inception architecture for computer vision. In: IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, pp. 2818–2826 (2016)
20. Ink markup language. http://www.w3.org/TR/InkML/. Accessed 06 Apr 2017
21. Hinton, G.E., Srivastava, N., Krizhevsky, A., et al.: Improving neural networks by preventing co-adaptation of feature detectors (2012). arXiv:1207.0580
22. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: 2003 International Conference on Document Analysis and Recognition, Edinburgh, vol. 3, pp. 958–962 (2003)
23. Jia, Y., Shelhamer, E., Donahue, J., et al.: Caffe: convolutional architecture for fast feature embedding. In: ACM Proceedings of the 22nd International Conference on Multimedia, Orlando, pp. 675–678 (2014)