Recognizing Handwritten Characters with Local Descriptors and Bags of Visual Words

Presentation at EANN'2015, Island of Rhodes

O. Surinta, M.F. Karaaba, T.K. Mishra,
L.R.B. Schomaker, and M.A. Wiering

Institute of Artificial Intelligence and Cognitive Engineering
University of Groningen, The Netherlands

Overview

1. Introduction
2. Feature Extraction Methods
3. Handwritten Character Datasets and Pre-Processing
4. Experimental Results
5. Conclusion

Introduction

Introduction

- Obtaining high accuracies on handwritten character datasets can be difficult due to several factors such as
  - background noise
  - many different types of handwriting
  - an insufficient amount of training examples
- There are currently many character recognition systems which have been **tested on the MNIST dataset**.
- Compared to other handwritten datasets, MNIST is simpler as it contains much more training examples.
- It is not surprising that a lot of progress on the best test accuracy has been made.

Introduction MNIST / DBN

- Currently the best approaches for **MNIST** make use of **deep neural network architectures**.
- In *(Hinton et al., 2006)*, **the deep belief network (DBN)** has been investigated for MNIST.
- Three hidden layers are used where the sizes of each layer are 500, 500 and 2,000 hidden units.
- The recognition performance with this method is **98.65%**.

Introduction CNN

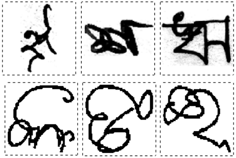
- In *(Cireşan et al., 2011)*, 35 **convolutional neural networks (CNN)** are trained and combined using a committee.
- This approach has obtained an accuracy of on average **99.77%**, which is *the best* performance on MNIST so far.
- This technique requires
  - a lot of training data
  - a huge amount of time for training for which the use of GPUs in mandatory

Contributions

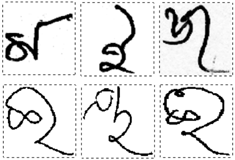
- To be able to deal with small datasets and create faster methods, we propose the use of feature descriptors for recognizing handwritten characters.
  - Histograms of oriented gradients (HOG)
  - Bags of visual words using pixel intensities (BOW)
  - Bags of visual words using HOG (HOG-BOW)
- These methods are compared on three handwritten character datasets including
  - Bangla (Bengali)
  - Odia (Oriya) and
  - MNIST

Contributions challenges in the dataset

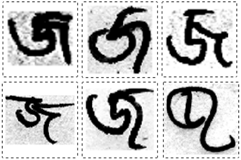
- There are some challenges in the Bangla and Odia handwritten character datasets such as
  - The writing styles (e.g., heavy cursively and arbitrary tail strokes)
  - Background noise
  - A lack of a large amount of handwritten character samples



cursive



longtail



noisy background

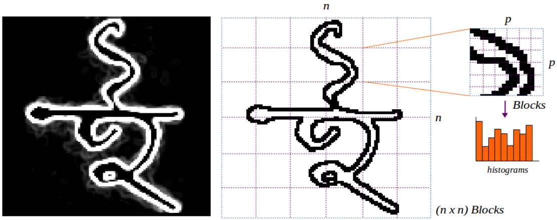
Contributions classifier

- We have evaluated the feature extraction techniques with three types of support vector machines (SVM) as a classifier.
  - A linear SVM
  - An SVM with a radial basis function (RBF) kernel and
  - A linear SVM with L2-norm regularization (L2-SVM)

Feature Extraction Methods

Histograms of Oriented Gradients (HOG)

○ The HOG descriptor was proposed in *(Dalal and Triggs, 2005)* for the purpose of human detection from images.



HOG descriptor

Compute the HOG descriptor

- The handwritten character image is divided into small regions (η), called 'blocks'.
- A simple kernel $[-1, 0, +1]$ is used as the gradient detector (*i.e.* Sobel or Prewitt operators).

$$G_x = f(x+1, y) - f(x-1, y)$$
$$G_y = f(x, y+1) - f(x, y-1)$$

where $f(x, y)$ is the intensity value at coordinate x, y.

Compute the HOG descriptor (Cont.)

- Compute the gradient magnitude M and the gradient Orientation θ.

$$M(x,y) = \sqrt{G_x^2 + G_y^2}$$
$$\theta(x,y) = \tan^{-1}\frac{G_y}{G_x}$$

- The image gradient orientations within each block are weighted into a specific orientation bin β of the histogram.
- The HOG descriptors from all blocks are combined and normalized by the L2-norm.

The HOG descriptor

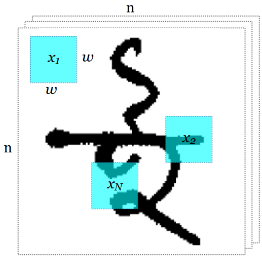
- The best η and β parameters we used are 6 and 9, respectively, which yields a **324-dimensional** ($6 \times 6 \times 9$) feature vector.

Bag of Visual Words with Pixel Intensities (BOW)

- The bag of visual words has been widely used in computer vision research.
- Local patches that contain local information of the image are extracted and used as a feature vector.
- A codebook is constructed by using an unsupervised clustering algorithm.
- In *(A. Coates et al, 2011)*, it was shown that the BOW method outperformed other feature learning methods such as RBMs and autoencoders.

BOW: Extracting patches from the training data

- The patches X are extracted randomly from the unlabeled training images, $X = \{x_1, x_2, ..., x_N\}$ where $x_k \in \mathbf{R}^p$ and N is the number of random patches.
- The size of each patch is defined as a square with $(p = w \times w)$ pixels.
- In our experiments we used $w = 15$, meaning 15×15 pixel windows are used.



Unlabeled training image

BOW: Construction of the codebook

- The codebook C is computed by using the K-means clustering method on pixel intensity information contained in each patch.
- Let $C = \{c_1, c_2, \ldots, c_K\}, c \in \mathbf{R}^p$ represent the codebook, where K is the number of centroids.
- In our experiments we used 400,000 randomly selected patches to compute the codebooks.

BOW: Feature extraction

○ To create the feature vectors for training and testing images, the soft-assignment coding scheme from *A. Coates et al (2011)* is used.

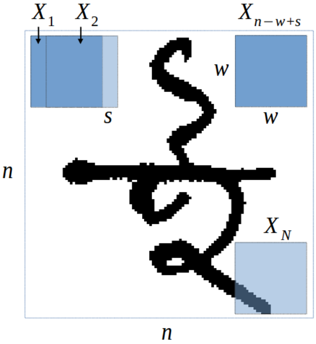
$$i_k(x) = max\left\{0, \mu(s) - s_k\right\}$$

where $s_k = \|x - c_k\|_2$ and $\mu(s)$ is the mean of the elements of s.

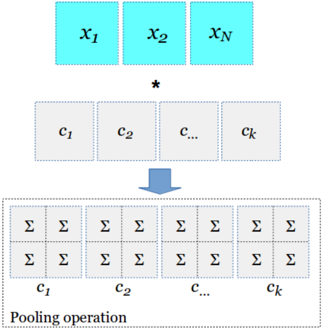
BOW: Feature extraction (Cont.)

- We use a sliding window on the train and test images to extract the patches. Because the stride is 1 pixel and the window size is 15×15 pixels, the method extracts **484 patches** from each image to compute the cluster activations.
- The image is split into **four quadrants** and the activities of each cluster for each patch in a quadrant are summed up.
- The feature vector size is $K \times 4$ and because we use $K = 600$ clusters, the feature vectors for the BOW method have **2,400 dimensions**.

BOW: Feature extraction Calculate the feature vector



Extract patches from input image



Calculate the feature from each centroid

Bag of Visual Words with HOG Features (HOG-BOW)

- HOG-BOW, feature vectors from patches are computed by using the state-of-the-art **HOG descriptor**.
- The advantages of the HOG descriptor are
  - capture the gradient structure of the local shape
  - provide more robust features
- In this experiment, the best HOG parameters used 36 rectangular blocks and 9 bins to compute feature vectors from each patch.
- The HOG-BOW used 4 quadrants and 600 centroids, yielding a 2,400 dimensional feature vector.

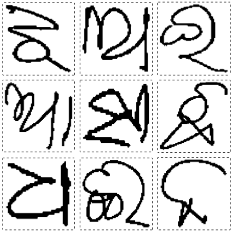
Handwritten Character Datasets and Pre-Processing

Handwritten Character Datasets

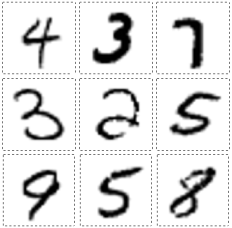
Table: Overview of the handwritten character datasets

Dataset	Color Format	No. of Writers	No. of Classes	Train	Test
Bangla character	Grayscale	Multi	45	4,627	900
Odia character	Binary	50	47	4,042	987
MNIST	Grayscale	250	10	60,000	10,000

Bangla handwritten characters



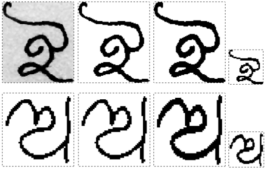
Odia handwritten characters



MNIST handwritten digits

Data Pre-Processing

- Image format of the handwritten dataset
  - The Bangla handwritten dataset contains different kinds of backgrounds and is stored in gray-scale format.
  - The Odia handwritten dataset is stored in binary image format.
  - The MNIST dataset is stored in gray-scale format.
- A few pre-processing steps are employed.
  - Background removal, Otsu's algorithm
  - Basic image morphological operations, dilation operation
  - Image normalization, 36 × 36 pixels with the aspect ratio preserved



pre-processing steps

Experimental Results

Results when combined with the linear SVM

Table: Results of training (10-fold cross validation with the standard deviation) and testing recognition performances (%) of the feature descriptors when combined with the linear SVM.

Algorithms	Bangla dataset		Odia dataset		MNIST dataset	
	10-cv	Test	10-cv	Test	10-cv	Test
PCA [1]	54.87 ± 0.20	53.67	56.57 ± 0.32	53.60	93.29 ± 0.02	92.69
DCT [2]	59.33 ± 0.32	52.33	60.77 ± 0.40	54.81	92.51 ± 0.06	91.32
IMG [3]	56.25 ± 0.22	54.33	56.12 ± 0.57	56.23	94.13 ± 0.05	94.58
BOW	77.96 ± 0.21	77.17	79.30 ± 0.34	78.01	98.71 ± 0.02	98.47
HOG	81.17 ± 0.30	80.11	79.86 ± 0.20	80.45	98.62 ± 0.01	99.11
HOG-BOW	**82.07 ± 0.24**	82.44	**81.74 ± 0.49**	82.43	**99.09 ± 0.03**	99.16

[1] PCA: Principal Component Analysis

[2] DCT: Discrete Cosine Transform

[3] IMG: Pixel-based Method

Results when combined with the SVM with the RBF kernel

Table: Results of training (10-fold cross validation with the standard deviation) and testing recognition performances (%) of the feature descriptors when combined with the SVM with the RBF kernel.

Algorithms	**Bangla dataset**		**Odia dataset**		**MNIST dataset**	
	10-cv	**Test**	**10-cv**	**Test**	**10-cv**	**Test**
IMG	63.25 ± 0.28	60.00	57.95 ± 0.42	60.28	96.95 ± 0.02	97.27
PCA	64.08 ± 0.30	61.11	60.57 ± 0.57	59.87	96.86 ± 0.02	96.64
DCT	70.18 ± 0.27	61.33	69.91 ± 0.34	63.63	98.18 ± 0.09	97.51
BOW	78.76 ± 0.38	77.17	81.29 ± 0.42	80.65	98.98 ± 0.01	98.97
HOG	83.11 ± 0.25	83.00	82.16 ± 0.27	83.38	99.13 ± 0.01	99.12
HOG-BOW	83.14 ± 0.18	83.33	**83.62 ± 0.17**	83.56	**99.30 ± 0.02**	99.35

Results when used with the L2-SVM

Table: Results of recognition performances (%) of the methods when used with the L2-SVM.

Algorithms	Feature dimensionality	Handwritten character dataset		
		Test Bangla	Test Odia	Test MNIST
DCT	60	51.67	56.94	90.84
PCA	80	50.33	53.90	91.02
IMG	1,296	31.33	42.65	91.53
HOG	324	74.89	74.27	98.53
BOW	2,400	86.56	84.60	99.10
HOG-BOW	2,400	87.22	85.61	99.43

Conclusion

Conclusion

- We have demonstrated the effectiveness of different feature extraction techniques of computer vision for handwritten character recognition.
- The **HOG-BOW** method combined with an **L2-SVM** outperforms all other methods.
- On the MNIST dataset, **HOG-BOW** combined with the **L2-SVM** obtains a recognition accuracy on the test set of **99.43%** which is a state-of-the-art performance.

Thank you

Thank you for your attention.