CNN Based Handwritten Devanagri Digits Recognition

Gaurav Kumar
Bangaluru, India 560095
Email: gaurav.sachin.007@gmail.com

Sachin Kumar
MIT Pune, Pune, India
Email: sachin.pitbull.12@gmail.com

***Abstract*—Handwritten Digit Recognition has huge demand in commercial, administrative and academic domains. In recent years lot of good work has been done to improve accuracy of Handwritten Digit Recognition System. But accuracy of such systems depend on large datasets. Deep Convolutional Neural Network have shown superior results to traditional shallow networks in many recognition task. In this paper, a convolunational nural network(CNN) based devanagri digit recognition system is highlighted. The dataset contains 21969 hand written 28x28 size images. The result of proposed system showed 99.07% accuracy on our dataset.**

***Keywords*—Devanagri Digits, CNN, SVM, KNN, CUDA, GPU, Tensorflow.**

I. Introduction

The optical character recognition (OCR) of digits on scanned images hold widespread commercial and pedagogical importance in fields such as automatic recognition, check reading, data collection from forms, and textbook digitization. Being still and active area of research, OCR recognition is being acquainted to different languages around the world.

The Devanagari script is used for over 120 languages, making it one of the most used and adopted writing systems in the world. Among the languages using it are Sanskrit, Hindi, Marathi, Gujrati, Nepali, Bhojpuri, Maithili, Rajasthani, Chhattisgarhi. Hindi is the fourth most-spoken language in the world, after Mandarin, Spanish and English. There is not much substantial work done in Devanagri HWCR. Hence, it is an area of ongoing research.

Devanagari script has forty-seven primary characters, of which fourteen are vowels and thirty-three are consonants.The script has no distinction similar to the capital and small letters of the Latin alphabet. Generally the orthography of the script reflects the pronunciation of the language.

Fig. 1: Devanagry Digits

Among the deep learning models, the convolutional neural networks (CNN) is the most popular one especially for image recognition. This is because CNN is very suitable to represent the image structure. First, the pixels of the image are strongly related to their neighbor pixels but has little correlation with the far away pixels. As shown in Fig. 2, this property is very similar to the local connectivity strategy of CNN, in which the neuron only connected to a local set of the neurons (local receptive field). Second, in CNN, the weight sharing strategy ensures that different part of the image can share similar properties, such as texture and brightness. In conclusion, because of the above features of CNN, it becomes one of the best choice for image recognition tasks.

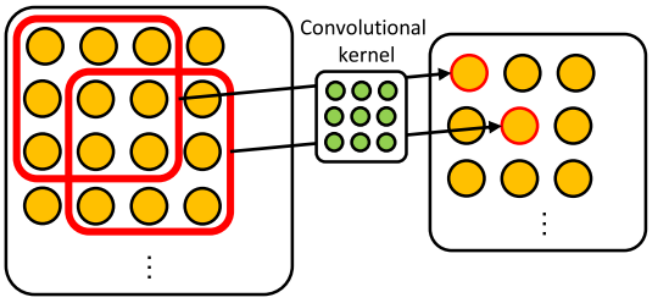

Fig. 2: Convolutional process of CNN

The handwritten character recognition is very difficult task since the character usually has various appearances ac-cording to different writer, writing style and noise. Many researchers have made great efforts to improve the recognition, such as better feature design and classifier optimization. However, as mentioned above, compared with CNN, those improvements of traditional methods are very limited. Simply by using CNN, a promising recognition rate can be obtained. Nevertheless, in order to pursue better performance of CNN, there are other issues which need to be addressed. In this paper, we proposed a CNNbased framework for handwritten character recognition and which achieved better performance compared with other CNN-based recognition methods.

II. DataSet

Dataset contains 21969 hand written images collected from different users. Each Digit image is 28 x 28 pixels. Each image is a gray-scale image having background value as 0.

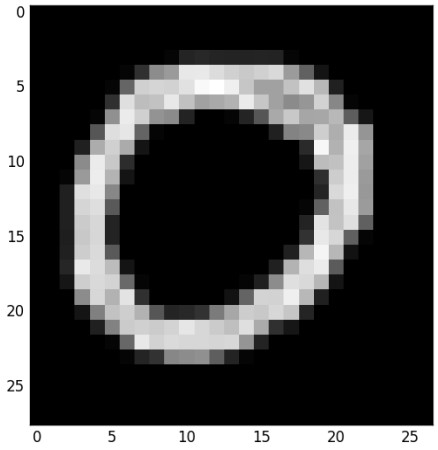

Fig. 3: 0 in devanagri

III. Digit Recognition

A. Convolutional Neural Networks

Convolutional Neural Network (CNN or ConvNet) is a biologically-inspired trainable machine leaning architecture that can learn from experiences like standard multilayer neural networks. ConvNets consist of multiple layers of overlapped tiling collections of small neurons to achieve better representation of the original image. ConvNets are widely used for image and video recognition. There are three main types of layers used to build a ConvNet architecture.

1) Convolution Layer: The convolution layer is the core building block of a convolutional neural network. It convolves the input image with a set of learnable filters or weights, each producing one feature map in the output image.

2) Pooling Layer: The pooling layer is used to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. The pooling layer takes small rectangular blocks from the convolution layer and subsamples it to produce a single output from that block. There are several ways to do this pooling, such as taking the average or the maximum, or a learned linear combination of the neurons in the block.

3) Fully-Connected Layer: The fully-connected layer is used for the high-level reasoning in the neural network. It takes all neurons in the previous layer and connects it to every single neuron it has. Their activations can be computed with a matrix multiplication followed by a bias offset as a standard neural networks.

B. The Architecture

A simple convolutional neural network similar to the one used in our recognition system is shown in Fig. 4. The input layer consists of the raw pixel values from the 32x32 grayscale image and has no trainable parameters. The first convolution layer has 32 feature maps with 784 units/neurons each(32 x 32). Each feature map is shown in figure as 2D planes and they have different set of weights. All the units in a feature map share the same set of weights and so they are activated by the same features at different locations. This weight sharing not only provides invariance to local shift in feature position but also reduces the true number of trainable parameters at each layer. Each unit in a layer receives its input from a small neighborhood at same position of previous layer. So the number of trainable weights associated with each unit in a convolutional layer depends on the chosen size of the neighborhood of previous layer mapped to that unit. Since all the units are activated only from the input taken from a local neighborhood they detect local features such as corners, edges, end-points. This concept of local receptive field is inspired from study of the locally-sensitive, orientation selective neurons in the cats visual system. For a 5x5 kernel as shown in Fig. 5 the number of input weights for each unit is 25. In addition the units also have a trainable bias. The total number of units in a layer depends upon the size of kernel in the previous layer and overlap between the kernels.

The convolutional layer is followed by a subsampling/pooling layer. Sub sampling layer reduces the resolution of the feature map from convolution layer by averaging the features in the neighborhood or pooling for a maximum value. Because the exact position of features vary for different images of the same character, it is more desirable that the system does not learn the absolute position of feature but instead learn the relative position among the features. The pooling layer helps achieve this goal and makes the classifier more immune to shift and distortion. It aggregates the information within a set of small local regions and produces a pooled feature map as output. The number of units in a pooling layer thus depends upon the local region of the previous convolution layer feeding input to the units in pooling layer. So for a non overlapping scheme and a 2x2 region from previous layer connected to units in pooling layer the dimension of feature maps reduce to half of the convolution layer. The max pooling method checks for the maximum value on its local receptive field, multiplies it by a trainable coefficient, adds a trainable bias and generates output.

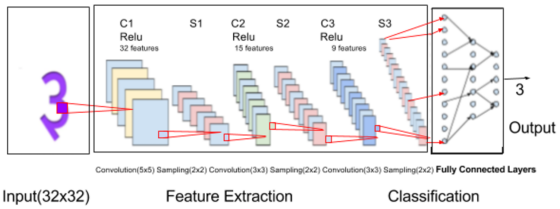

Fig. 4: CNN Architecture

The second convolution layer follows this sub-sampling layer. Each feature map in C2 layer is generated by taking input from S1. The units in C2 get their input from the 3x3 neighborhood at identical position of some layers in S1 and not all. The reason for not connecting C2 feature maps to all feature maps of S1 layer is to reduce the total number of trainable parameters and, this also introduces randomness in providing input to different feature maps with the assumption that this will help them to learn complementary features with

one another. Similarly, the third convolution layer follows this sub-sampling layer. Each feature map in C3 layer is generated by taking input from S2. The units in C3 get their input from the 3x3 neighborhood at identical position of some layers in S2 and not all. The output of this convolution layer is subsampled, convolved and forwarded to fully connected layer. Fully connected layer consists of 3 layes having 128, 50 and 30 neurons respectivly. From this point we obtain a 1D feature vector. The fully connected layers model the input by applying non-linearity like in a traditional feed-forward network. The type of nonlinearity used is ReLU-non linearity.

$f(x) = \max(0, x)$

The reason for using it instead of the widely popular nonlinear functions like

$f(x) = \tanh(x)$

is because training with gradient-descent is comparatively much faster for ReLU than the other non-linearities. The depth of the network and the size of different layers to be used depends greatly on the dataset and the problem domain. Furthermore, the number of feature maps in a layer, the size of the kernel on each layer and the choice of non-overlapping or overlapping kernel and the extent of overlap also produces different results. So, in our case we tested different architectures by varying these parameters and presented results of the architecture producing the highest accuracy on the test data set. The result of the tests are summarized on the Experimental setting and results section.

C. Over-fitting in the Deep Network

The large and deep architecture of CNN with large bank of trainable parameter makes it susceptible to overfitting. While training the deep networks, it is very difficult to find optimal hyper parameters of the functions that share the parameters. These networks being large, require large amount of training data. Below given are some approaches we used to prevent our model from overfitting.

1) Dataset augmentation: For the better recognition models, we require to have more training samples while training the system. This can be achieved by augmenting available dataset by mirroring, rotation, jittering, noise injection and random crops.

2) Dropout: Dropout simply refers to dropping out units; units representing both hidden and visible in the deep network. We temporarily remove the random units from the network along with all its inlet and outlet connections. For each training iterations, there will be new lighter network that remains after dropping the random units from the common denser architecture which will be sampled and trained. Each unit is retained with the fixed probability of p independent of other units and we set 0.2 for p, the number being optimal choice for most of the cases.

IV. RESULTS AND OBSERVATIONS

We tested the dataset on different existing classification algorithms like SVM, KNN, Gradint Descent. The accuracy result is shown in table 1.

Classifier	Accuracy(%)
SVM	81.29
Gradint Descent	86.82
KNN	97.10
CNN	99.07

TABLE I: Digit Recognition

In case of CNN we tested the dataset with different architectures by varying depth, width and number of parameters of network. Final model consists of 3 convolation, 3 pooling and fully connected layer. First convolution layer C1 uses 5x5 size filter. Second and Third convolution layer C2, C3 uses 3x3 size filter. Pooling layers uses 2x2 size filter with relu as activation function. Fully connected layer has one input layer with 128 nodes, two hidden layers with 50 and 30 nodes and output layer with 10 nodes. It uses softmax activation function in the output layer. we used dropping of 50 % in this model.

After training above model with our dataset having 21969 images we got 99.07 % accuracy rate.

V. ACKNOWLEDGEMENT

We would like to thank Amitabh Mukharji, CSE Department, IIT Kanpur and his team in putting hard effort to collect hand written digits in devanagri script and making it public.

VI. CONCLUSION

We took 21969 hand written images of digits of devanagri script. We did the relative accuracy comparasion of existing classification models like SVM, KNN. We built a CNN based learning model and tried to determine the impact of convolution, dropout and fully connected layer on accuracy. The experimental results suggested that Deep CNNs with added Dropout layer and convolutional features can result in very high test accuracy.

REFERENCES

[1] Writer-adaptation for on-line handwritten character recognition N. Matic; I. Guyon; J. Denker; V. Vapnik

[2] Handwritten Character Recognition by Alternately Trained Relaxation Convolutional Neural Network Chunpeng Wu; Wei Fan; Yuan He; Jun Sun; Satoshi Naoi

[3] Recognition of Handwritten Kannada Numerals N. Sharma; U. Pal; F. Kimura

[4] Handwritten/Printed Receipt Classification Using Attention-Based Convolutional Neural Network Fan Yang; Lianwen Jin; Weixin Yang; Ziyong Feng; Shuye Zhang

[5] MapReduce-based deep learning with handwritten digit recognition case study Nada Basit; Yutong Zhang; Hao Wu; Haoran Liu; Jieming Bin; Yijun He; Abdeltawab M. Hendawi

[6] Towards optimal convolutional neural network parameters for bengali handwritten numerals recognition Al Mehdi Saadat Chowdhury; M. Shahidur Rahman

[7] Handwritten Music Symbol Classification Using Deep Convolutional Neural Networks Sangkuk Lee; Sung Joon Son; Jiyong Oh; Nojun Kwak

[8] The recognition of car license plate for automatic parking system T. Sirithinaphong; K. Chamnongthai

[9] Recognition of isolated handwritten Persian/Arabic characters and numerals using support vector machines A. Mowlaei; K. Faez