CNN-based Hindi Numeral String Recognition for Indian Postal Automation

Hongjian Zhan†, Shujing Lyu†, Umapada Pal‡, Yue Lu†
†*Shanghai Key Laboratory of Multidimensional Information Processing*
Department of Computer Science and Technology
East China Normal University, Shanghai 200062, China
Email: ylu@cs.ecnu.edu.cn
‡*CVPR Unit, Indian Statistical Institute, Kolkata, India*

***Abstract*—Digits/numerals in the Indian pin-code of handwritten postal documents may touch each other and hence digit string recognition is a very challenging task. In this paper, we propose a digit string recognition system for Indian postal documents written in Hindi. Unlike normal text string, in a string of digits there is no contextual information among the digits as a digit may be followed by an arbitrary digit in a string of digits. Because of this, here we propose a new architecture which is based on CNN (Convolutional Neural Network) and CTC (Connectionist Temporal Classification), without using RNN for Hindi numeral string recognition. Also to connect CNN with CTC, we transform the outputs of CNN to a two-dimension vector to meet the feeding requirement of CTC. Furthermore, we utilize dense blocks to build CNN part to extract efficient image features. Comparative studies with the state-of-the-art methods show that the proposed method outperforms the other existing methods on Hindi numeral string recognition.**

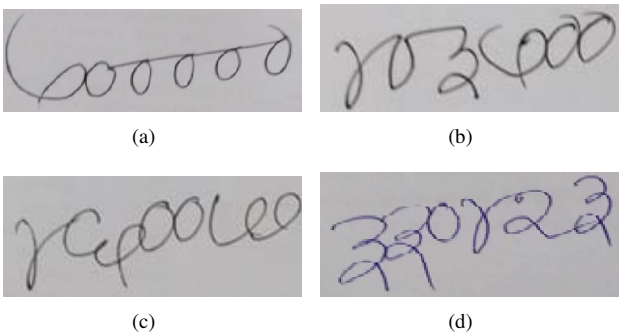
***Keywords*-Hindi Numeral String; Convolutional Neural Network; Connectionist Temporal Classification; Postal Automation**

I. Introduction

Recognition of handwritten numeral string has been a popular research area for many years because of its various application potentials. Some of its potential application areas are postal automation, bank cheque processing, etc. There are many pieces of works on non-Indian character recognition [4, 5]. Although India is a multi-lingual and multi-script country (in India there are about 25 official languages and 11 different scripts are used to write these languages) but not much work is done towards the recognition of Indian handwritten numerals [2, 3, 12, 13, 17].

Indian pin-code (postal code) is a six-digit number [8, 9] and system development towards Indian postal automation is more difficult and challenging because of the unconstrained structure of the postal documents. In this paper we propose a new architecture which is based on CNN and CTC, but without using RNN for pin-code string recognition written in Hindi.

There are two approaches for handwriting numeral string recognition. One is segmentation based [6] and other is segmentation free [2]. In Indian postal documents, digits in a pin-code may touch in different manners like top touching, middle-touching and bottom touching. Such touching may be categorized as single point touching, multiple point touching, ligature touching etc. Moreover, two, three, four, five digit touching strings are also available in Indian pin-code. See Figure 1 where touching strings of different digits are shown. It is very difficult for accurate segmentation of individual digits from such touching string and hence to avoid such segmentation of touching string into individual digits, in this paper, we propose a new segmentation free architecture based on CNN and CTC.



(a) (b) (c) (d)

Figure 1. Examples of some Hindi touching digit strings.

To get the idea about the earlier pieces of work on Hindi numeral recognition, a brief discussion of those works are reported here. Pal et al. [13] proposed a Modified Quadratic Discriminant Function (MQDF) based methods for the recognition of the Hindi numerals. In the proposed scheme, at first, the bounding box of a numeral is segmented into blocks and directional features are computed in each of these blocks. Next, these blocks are down sampled by a Gaussian filter. Finally, the features obtained from the down sampled blocks are fed to the classifier for recognition. In 2019 Pal et al. [2] proposed a Hindi digit-string recognition system where, at first, binarization of the pin-code is done and it is pre-segmented into possible primitive components (individual digits or its parts). Each primitive ideally consists of a single digit or a sub-image of a single digit. Pre-segmented components of a pin-code are then merged into possible digits to get the best possible pin-code. In order to merge these primitive components into digits and to find optimum

978-1-7281-5054-3/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDARW.2019.40085

segmentation, dynamic programming (DP) is applied using total likelihood of digits as the objective function. To compute the likelihood of a digit, Modified Quadratic Discriminant Function (MQDF) [1] based on directional feature was applied. Bhattacharyya et al. in [11] proposed a two-stage classification system for recognition of handwritten Hindi numerals. Here a shape feature vector computed from certain directional-view-based strokes of an input character image, has been used by both the HMM and ANN classifiers for recognition. The two sets of posterior probabilities obtained from the outputs of the above two classifiers are combined by using another ANN classifier. Finally, the numeral image is classified according to the maximum score provided by the ANN of the second stage. Among other pieces of work in Hindi, Ramakrishnan et al. [14] used independent component analysis technique for feature extraction from numeral images. Bajaj et al [15] considered a strategy combining decisions of multiple classifiers. In an attempt to develop a bilingual handwritten numeral recognition system, Lehal and Bhatt [16] used a set of global and local features derived from the right and left projection profiles of the numeral images for recognition of Hindi handwritten numerals.

In 2007, Hanmandlu and Murthy [10] proposed a fuzzy model based Hindi numeral recognition. Here each feature yields a fuzzy set and the database is divided into training and testing samples. The scanned numeral is partitioned into 24 boxes from which 24 features are extracted. Thus, 24 fuzzy sets are obtained. Each fuzzy set is represented by the exponential membership function whose parameters are the mean and variance, and the means and variances of all training samples form the knowledge base. When an unknown numeral comes for recognition, its features are found and then these are fitted to the membership functions of the known (i.e. reference) numerals using the optimized parameters. Whichever numeral yields the minimum objective function is the identity of the unknown numeral.

From the above discussion, it can be seen that although there are some pieces of work on isolated Hindi numeral recognition but there is only one work [2] on digit string recognition of Hindi. Also reported string recognition accuracy is not very high (93.32%). Thus, there is a need to work on Hindi digit string recognition and hence, in this paper, we propose a new architecture with only convolutional neural network and CTC and apply it to handwritten digit string recognition. This network is composed of several dense blocks to extract feature, dimension transposition layers to conduct dimension adjustment and a CTC output layer to produce the recognition results. There are several contributions of this work. Firstly, to avoid any segmentation, we apply a new architecture which only based on CNNs for Hindi humeral string recognition. Secondly, we achieve 3.41% better results than the state-of-the-art method.

II. The Characteristics of Hindi Numerals

Devnagari is the most popular script in India and the most popular Indian language Hindi is written in this script. Hindi is the national language of India and the third most popular language in the world [7]. Thus, the work on Hindi is very useful to the country. Writing style in Devnagari script is from left to right. There are 10 numerals in Hindi and characteristics and shape variety of numerals make the recognition of Hindi numerals more complicated than in other languages. Different shapes of Hindi handwritten digits are shown in Figure 2.

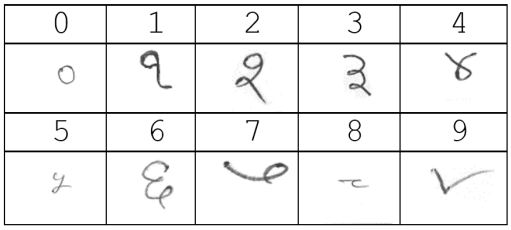


Figure 2. Examples of handwritten Hindi numerals [Hindi handwritten numerals are shown under respective Arabic (English) printed numerals].

III. Proposed method

The proposed architecture contains three components, feature extraction, feature dimension transformation and output layer, from left to right.

Motivated by the high performance of DenseNet [18], we use dense blocks to conduct the features extraction. On the top of this network a CTC output layer is adopted to calculate the loss at the training procedure and output the prediction results in testing phase. Due to the output of dense blocks is not suitable for feeding to CTC directly. So we transform the dimensions of the output before feeding into the CTC.

A. Feature Extraction

Dense Block: The main component of feature extraction is a stack of dense block. A dense block is a group of layers connected to all their previous layers. It can be treated as an enhanced version of residual convolutional layers [19].

Unlike the residual convolutional layers, dense block uses a group of layers to replace the ordinary convolutional layer. It consists of a group of layers, the batch normalization layer [20], ReLU layer [21] and a convolutional layer. The kernel size of the convolutional layer is $3 * 3$, which can maintain the size of feature map in the whole dense block. After the convolutional layer, we apply a dropout layer [22] with rate 0.2.

Transition Layer: However down-sampling is essential to CNNs. Since the dense block don't change the feature map size, a transition layers is applied between two dense blocks. A typical transition layer consists of batch normalization layer, a $1 * 1$ convolutional layer and an average pooling layer with kernel $2 * 2$.

B. Feature Dimension Transformation

The output dimension of convolutional layer cannot meet the requirements of CTC directly. So we conduct the dimension transformation. The output of dense block is always a 3-D tensor (4-D if we consider the $batchsize$ dimension), i.e., the $number$, $height$ and $width$ of feature maps. But CTC requires a 2-D tensor as its input. First we flatten the 3-D feature tensor to 2-D by expanding on the $number$ dimension, then we apply a column-wise fully connected layer to reduce the $height$ dimension to the assigned value. With these actions, we can generate the suitable input for the following CTC output layer. The details are shown in Figure 3.

C. Output Layer

The connectionist temporal classification output layer has two function. In the training phase, it calculates the loss and update the network parameters, and in the testing phase, it outputs the prediction results.

In this paper, we consider the digit string recognition task, so the labels of the string images are drawn from a set A with the ten digits. After adding an extra label named *blank*, we get a new set $A^{'} = A \cup \{blank\}$, which is used in practice. After feature dimension transformation, we get a sequence $y = y_1, ..., y_T$, where T is the sequence length. The corresponding label donates as I over A. We define a many-to-one function $\mathscr{F} : {A^{'}}^T \mapsto A^{\leq T}$ to resume the repeated labels and blanks. For example $\mathscr{F}(11-8-333-00--9-----) = 13809$. Then, a conditional probability is defined as the sum of probabilities of all π which are mapped by $\mathscr{F}$ onto I:

$$p(I|y) = \sum_{\pi \in \mathscr{F}^{-1}(I)} p(\pi|y) \tag{1}$$

where the conditional probability of π is defined as:

$$p(\pi|y) = \prod_{t=1}^{T} y_{\pi_t}^t \tag{2}$$

$y_{\pi_t}^t$ is the probability of having label π_t at timestep t. Direct computation on Eq.1 is not feasible. In practice, Eq.1 is usually calculated using the forward-backward algorithm.

Donate the training dataset by $S = (\boldsymbol{X}, \boldsymbol{I})$, where X is the training image and I is the ground truth label sequence. The CTC object function $\mathscr{O}(S)$ is defined as the negative log probability of ground truth all the training examples in training set S,

$$\mathscr{O}(S) = -\sum_{(x,i) \in S} \log p(I|y) \tag{3}$$

where y is the sequence produced by the recurrent layers from x.

In the testing phase, we apply Greedy Algorithm to get the prediction results.

IV. Datasets and pre-processing

For the experiment of the pin-code recognition scheme proposed in this paper, we have used a total of 5414 Hindi handwritten pin-code string samples. Number of total pin-code class was 300 and minimum (maximum) number of samples in a class was 6 (40). These pin-code samples are collected from handwritten address block of Indian postal documents as well as from some individuals using some specially designed forms. We have used 5-fold cross validation scheme for recognition result computation. Here database is divided into 5 subsets and testing is done on each subset using other four subsets for learning. The recognition rates for all the test subsets are averaged to calculate recognition accuracy. As we did not have enough touching string samples for learning our network, we used a separate dataset of more than 3000 isolated numeral samples for training purpose.

The fully connected layers in our model requires fixed input dimension, but image sizes are different in our dataset. In order to address this problem, two approaches are usually applied. One is to extend the dimensions of feature vectors of a fixed size before feeding into fully connected layer, and the other one is to resize the input images before feeding into the network. In practice we resize the input images into $50 * 150$.

V. Experimental Results and discussion

In order to investigate the performance of the proposed method, we conduct a series of experiments on our Hindi digit-string datasets and details of them are as follows.

A. Evaluation metric

We use two metrics to evaluate our model, one is calculated on string level (hard metric) and the other is on individual digit level (soft metric).

We apply Levenshtein Distance to define these metrics, which can be calculated by:

$$LD(S_T, S_R) = D_e + S_e + I_e \tag{4}$$

where D_e is the deletion errors , S_e is the substitution errors and I_e is the insertion errors, while S_T, S_R are the input string and its corresponding network prediction, respectively.

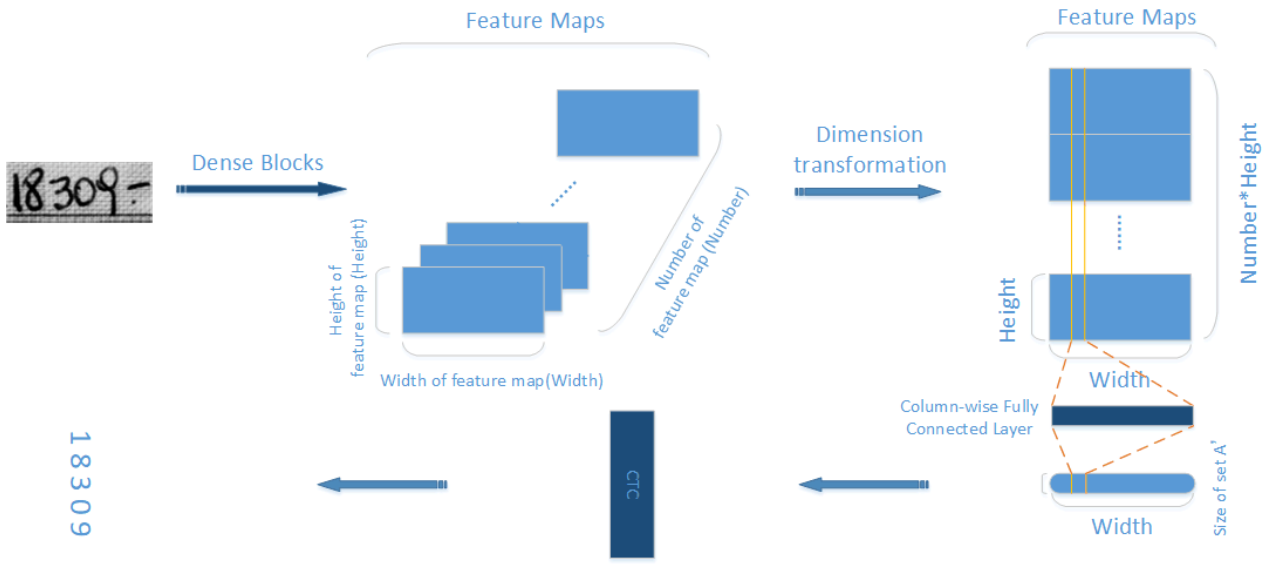


Figure 3. Illustration of Feature Dimension Transformation.

So the hard metric can be computed by:

$$AR_H = N_A/N \tag{5}$$

Where N is the total number of testing images and N_A is the number of images that meet $LD(S_T, S_R) = 0$.

For the soft metric, the accuracy rate AR_S is defined as:

$$AR_S = N_{RC}/N_C \tag{6}$$

Where N_C is the total number of digits in testing images and N_{RC} can be counted as:

$$N_{RC} = N_C - D_e - S_e - I_e. \tag{7}$$

B. Experimental details

The network is trained with ADADELTA, setting the parameter delta to 10^{-6}. On all datasets we train the network using batch size 128 for $300,000$ iterations.

The hyper-parameters of three dense blocks are the same. The growth rate is 8 and the number of convolution layer is 16. We set the initial channel to 128. The number of neuron in the first fully connected layer is 100 and 11 for the second fully connected layer.

Our experiments are performed on a DELL workstation. The CPU is Intel Xeon E5-1650 with 3.5GHz and the GPU is NVIDIA TITAN X. The software is the Caffe framework with cuDNN V5 accelerated. The operator system is Ubuntu 14.04 LTS system. The implementation of dense block in Caffe [23] can be found in this website[1].

[1]https://github.com/Tongcheng/caffe/

Table I
DISTRIBUTION OF DIFFERENT DIGIT ERRORS.

Error in number of digits	Recognition error
1	3.04%
2	0.08%
3 or more digits errors	0.15%

C. Experimental results

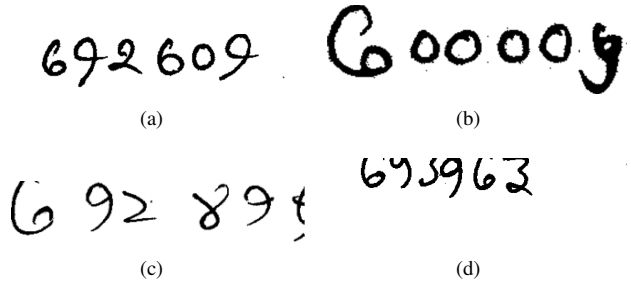
As mentioned earlier, we have used 5-fold cross validation scheme for recognition result computation. Here database is divided into 5 subsets and testing is done on each subset using other four subsets for learning. The recognition rates for all the test subsets are averaged to calculate recognition accuracy. The overall accuracy rate of our proposed method under hard metric (i.e. for correct full digit string recognition rate) is 96.73% whereas it is 99.46% under the soft metric (i.e. individual digit recognition rate).

D. Error analysis

We also classified different pin-code recognition errors in term of the errors in number of digits. We noted that 3.04% of the pin-code errors are one-digit errors. In 0.08% cases the pin-code errors are two-digit errors. Distribution of pin-code errors in terms of the error in number of digits is shown in Table I. To get the idea about the erroneous samples, some examples of erroneous pin-code are shown in Figure 4. From the figure it can be seen that most of the errors are due to shape similarity of the digits. For example, in Fig. 4(a) digit seven is recognized as zero and this is due shape similarity of these two samples for this writer.

Table II
COMPARATIVE RESULTS ON HINDI ISOLATED DIGITS AS WELL AS ON DIGIT STRING. (HERE NA MEANS NOT APPLICABLE).

Approach	Data Size	Accuracy	
		In digit level	In digit-string level
Bhattacharya et al. [11]	22547 isolated characters	92.83%	NA
Hanmandlu and Murthy [10]	Not known	95.00%	NA
Pal et al. [13]	22547 isolated characters	99.51%	NA
Pal et al. [2]	5414 string digits	98.41%	93.32%
Proposed approach	5414 string digits	99.46%	96.73%



(a) (b) (c) (d)

Figure 4. Examples of some miss-recognized pin- codes (a) Pin-code 712701 is recognized as 712001; (b) Pin-code 700005 is recognized as 700007; (c) Pin-code 712419 is recognized as 712411; (d) Pin-code 713173 is recognized as 712173.

E. Speed

We test on a GPU (Version: NVIDIA TITAN X) machine and the average the speed is $60ms$ for processing one image.

F. Comparative results

To the best of our knowledge there is only one work [2] on Hindi digit-string recognition and an accuracy of 93.32% is reported in that work. Whereas using the same dataset we obtained 96.73% accuracy from our proposed method. So the proposed method gives 3.41% better results than the state-of-the-art method.

There exists many pieces of work on isolated numeral recognition and to get an idea of comparative results of isolated Hindi digit recognition here we compare some results as shown in Table II.

VI. CONCLUSION

In this paper we proposed a system for Hindi handwritten 6-digit full pin-code string recognition. Unlike text words, as in string digits there is no contextual information among the digits, here we propose a new architecture which is based on CNN and CTC without using RNN for Hindi numeral string recognition. To our knowledge, the proposed method gives the best results compare to other state-of-the-art methods. At present there is no rejection criteria and it future we plan to incorporate it.

VII. REFERENCES

[1] F. Kimura, K. Takashina, S. Tsuruoka and Y. Miyake, "Modified quadratic discriminant function and the application to Chinese character recognition", IEEE Trans. on Pattern Analysis Machine Intelligence, Vol. 9, pp 149-153, 1987.

[2] U. Pal, R. K. Roy, K. Roy and F. Kimura, "Indian Multi-Script Full Pin-code String Recognition for Postal Automation", In Proc. 10th International Conference on Document Analysis and Recognition (ICDAR), pp.456-460, 2009.

[3] Y. Wen, Y. Lu and P. Shi, "Handwritten Bangla digit recognition system and its application to postal automation", Pattern Recognition, vol.40, pp.99-107, 2007.

[4] L. Liu and M. Koga and H, Fujisawa, "Lexicon driven segmentation and recognition of handwritten character strings for Japanese address reading", IEEE Trans on PAMI, vol. 24, pp. 1425-1437, 2002.

[5] R. Plamondon and S. N. Srihari, "On-Line and off-line handwritten recognition: A comprehensive survey", IEEE Trans on PAMI, Vol.22, pp.62-84, 2000.

[6] U. Pal, A. Belaid and C. Choisy "Touching digit segmentation using water reservoir concept", Pattern Recogniton Letter, vol. 24, pp. 261-272, 2003.

[7] U. Pal, K. Roy and F. Kimura, "A Lexicon driven method for unconstrained Bangla handwritten word recognition", In Proc. 10th IWFHR, pp. 601-606, 2006.

[8] K. Roy, S. Vajda, U. Pal, B. B. Chaudhuri, and A. Belaid, "A system for Indian postal automation", In Proc. 8th ICDAR, pp.1060-1064, 2005.

[9] S. Vajda, K. Roy, U. PaL, et al. Automation of Indian postal documents written in Bangla and English[J]. International Journal of Pattern Recognition and Artificial Intelligence, vol.23, pp. 1599-1632, 2009.

[10] M. Hanmandlu and O. V. Ramana Murthy, "Fuzzy model based recognition of handwritten numerals", Pattern Recognition, vol.40, pp. 1840-1854, 2007.

[11] U. Bhattacharya et al., "Neural combination of ANN and HMM for handwritten Devnagari numeral recognition", In Proc. 10th IWFHR, pp.613-618, 2006.

[12] K. Roy, A. Banerjee, U. Pal. A system for word-wise handwritten script identification for Indian postal

automation[C]//Proceedings of the IEEE First India Annual Conference, pp. 266-271, 2004.

[13] U. Pal, T. Wakabayashi, N. Sharma and F. Kimura, "Handwritten Numeral Recognition of Six Popular Indian Scripts", In Proceedings 9th International Conference on Document Analysis and Recognition. pp. 749-753, Curitiba, Brazil, September 24-26, 2007.

[14] K.R. Ramakrishnan, S.H. Srinivasan and S. Bhagavathy, "The independent components of characters are Strokes", Proc. of the 5th ICDAR, 1999, pp. 414-417.

[15] R. Bajaj, L. Dey and S. Chaudhuri, "Devanagari numeral recognition by combining decision of multiple connectionist classifiers". Sadhana, Vol. 27, Part 1, 2002, pp. 59-72.

[16] G.S. Lehal and Nivedan Bhatt, "A recognition system for Devnagri and English handwritten numerals", Advances in Multimodal Interfaces-ICMI 2001, Vol. 1948, 2000, pp. 442-449.

[17] Jayadevan R, Satish R Kolhe, Pradeep M Patil, and Umapada Pal, "Offline Recognition of Devanagari Script: A Survey" IEEE Transactions on Systems, Man, and Cybernetics: Part C: Applications & Reviews, Vol.41. No.6, 2011, pp.782-796.

[18] G. Huang, Z. Liu, K. Q. Weinberger, L. van der Maaten, "Densely connected convolutional networks", arXiv preprint arXiv:1608.06993, 2016.

[19] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition", in: Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770-778.

[20] S. Ioffe, C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in: Proceedings of International Conference on Machine Learning, 2015, pp. 448-456.

[21] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, 2011, pp. 315-323.

[22] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arXiv preprint arXiv:1207.0580.

[23] Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor, Caffe: Convolutional Architecture for Fast Feature Embedding, arXiv preprint arXiv:1408.5093, 2014.