Capsule-Based Persian/Arabic Robust Handwritten Digit Recognition Using EM Routing

Ali Ghofrani
Faculty of Media Technology and Engineering
IRIB University
Tehran, Iran
alighofrani@iribu.ac.ir

Rahil Mahdian Toroghi
Faculty of Media Technology and Engineering
IRIB University
Tehran, Iran
mahdian.t.r@gmail.com

***Abstract*—In this paper, the problem of handwritten digit recognition has been addressed. However, the underlying language is Persian/Arabic, and the system with which this task is a capsule network (CapsNet) which has recently emerged as a more advanced architecture than its ancestor, namely CNN (Convolutional Neural Network). The training of the architecture is performed using Hoda dataset, which has been provided for Persian/Arabic handwritten digits. The output of the system, clearly outperforms the results achieved by its ancestors, as well as other previously presented recognition algorithms.**

***Index Terms*—Handwritten digit recognition, CAPSNET, Deep learning, Hoda dataset**

I. Introduction

Handwritten digit recognition is among the very first applications of artificial intelligence in our life, which aimed at automatically reading the zip-codes being written on letters, packages or similar applications. As a branch of image processing, it gained the interest of researchers and many heuristic works were proposed in this regard, [1]–[6].

MNIST is the most widely used dataset as a benchmark for handwritten digit recognition tasks. During 70s and 80s, there were lots of MLP-based neural network classifiers among the very first ones being tested on MNIST data. While this dataset is created for English language, and doing research tasks on such a subject is becoming arguable today, there are only few attempts on Persian or Arabic handwritten digits which are quite different and somehow more difficult to be recognized, in the sense that some of the digits could be written in many different ways [7].

In this paper, we concentrate on Persian/Arabic-based digital numbers which are the same, however quite different from the English digits gathered in MNIST dataset. Hence, here we use **Hoda** dataset [7], which is recorded for Persian/Arabic language and alongside a new method being proposed in this work the results are compared with the state-of-the-art [8]–[11], even though we strongly believe that this technique could be equally applied to the digits being written in English or other languages, if previously trained.

The novelty of this paper comes in utilizing a capsule network for training and testing the handwritten digit recognition in Persian/Arabic language for the first time.

The outline of this paper, occur as the following: First, the structure of a typical capsule network for the considered purpose is explained. Then, the structure of the system being designed for our digit recognition is mentioned. Section IV, brings about the experimental setup and parameter tunings along with all the requirements for the experiments to be reimplemented. In the sequel, the paper is terminated by the conclusion, and the cited references.

II. Capsule Networks (CAPSNET)

While Convolutional Neural Networks (CNNs) happened to be very successful in computer vision applications, they are still weak in understanding the rotation or change of proportion of the input images. CapsNet overcomes these shortcomings and provide local translation invariance (via max-pooling, typically) by addressing various kinds of visual stimulus and encoding things such as position, orientation, deformation, and so on, [12]–[15].The complete explanation of capsule networks with Expectation-Maximization (EM) routing mechanism for training of the parameters, are rephrased from [16]–[18].

A. How Capsules Work?

A capsule network consists of several layers of capsules. Between two adjacent layers of capsules L and $L+1$, there is a learned transformation matrix $\boldsymbol{W}_{ij}$ for every pair of capsules, i and j. The vote from capsule i to capsule j is computed with a simple matrix multiplication of pose and transformation matrices:

$$\boldsymbol{V}_{ij} = M_i \boldsymbol{W}_{ij} \tag{1}$$

The pose matrix M, represents the relationship between an object or object-part and the pose. In other words, the pose matrix represents the viewpoint change of an object through rotation and transformation. A capsule in one layer votes for the pose matrix of many different capsules in the layer above, by multiplying its own pose matrix by the transformation matrix $\boldsymbol{W}$, that could learn to represent the part-whole relationship.

The activations a_i and votes $\boldsymbol{V}_{ij}$ for all capsules in layers i and j, are used in the routing algorithm to compute the activations a_j and poses $\boldsymbol{M}_j$ in layer j, therefore the routing algorithm decides where the output of the capsule should go. The *dynamic routing by agreement* algorithm uses a routing coefficient that is dependent on the input and gets computed

978-1-7281-1621-1/19/$31.00 ©2019 IEEE

dynamically. Thereby capsule i gets assigned to a capsule j that has the best fitting cluster of votes for its pose.

When given the activations and poses of all capsules in layer L the routing algorithm decides which capsules to activate in layer $L+1$ and how to assign every capsule in layer L to one capsule in layer $L+1$.

In EM routing, the pose matrix of the parent capsule is modeled using a Gaussian distribution. Assume that the pose matrix is a 4×4 matrix, i.e. 16 components. The modeled pose matrix would be a Gaussian containing 16 μ's and 16 σ's, in which each μ represents a component in the pose matrix. Next, we apply the Gaussian pdf to compute the $\boldsymbol{V}_{ij}^h$ probability (the h^{th} dimension or component of the vectorized vote) belonging to the capsule j's Gaussian model, as

$$p_{i|j}^h = \frac{1}{\sqrt{2\pi(\sigma_j^h)^2}} \exp\Big(- \frac{(\boldsymbol{V}_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2} \Big) \tag{2}$$

where μ_j, and σ_j^2 are the mean and variance of the fitted Gaussian, respectively. We take natural log from (2), and the cost of activating the capsule j (parent capsule) by the capsule i, would be

$$\text{cost}_{ij}^h = -\ln\big(p_{i|j}^h\big) \tag{3}$$

Since capsules of a lower layer i are not equally linked with capsule j, and a capsule with a low activation has no big impact on the routing procedure, we allocate the runtime assignment probabilities r_{ij} to the cost, which is called *routing coefficient* and is defined between any capsule i and the higher layer capsule j. The cost from all lower layer capsules would be then [16],

$$\text{cost}_j^h = \sum_i r_{ij} \text{cost}_{ij}^h = \overbrace{\cdots}^{Math} = \big(\ln(\sigma_j^h + k)\big) \sum_i r_{ij} \tag{4}$$

which k is a constant, and $\sum_i r_{ij}$ also denotes the amount of data being assigned to j. To show, how informations flows through the network, we can compute the following,

$$a_i = \sum_j r_{ij} \tag{5}$$

Then, we compute the logistic function of the cost, as follows in order to determine the likelihood of capsule j to be activated or not,

$$\begin{aligned} a_j &= \text{sigmoid}\big(\lambda(b_j - \sum_h \text{cost}_j^h)\big) \qquad (6) \\ &= \text{sigmoid}\Big(\lambda(\beta_a - \beta_u \sum_i r_{ij} - \text{cost}_j)\Big) \\ \text{Where:}\quad \text{cost}_j &= \sum_i r_{ij} \text{cost}_{j|i} \\ &= \sum_h \big(\ln(\sigma_j^h) + \frac{1 + \ln 2\pi}{2}\big) \sum_i r_{ij} \end{aligned}$$

where β_a denotes the trained parameters for all capsules, and β_u denotes the trained parameters for one capsule. "$-b_j$" (as explained in the original paper [16]) is the cost of describing the mean and variance of capsule j. In other words, if the benefit rank b_j of representing datapoints by the parent capsule j exceeds the cost which is caused by the discrepancy in their votes, we activate the output capsules. b_j is approximated using a cost function and a backpropagation process, rather than being analytically computed. In the above equations, r_{ij}, μ, σ, and a_j are trained iteratively using EM-routing. λ, which controls the steepness of the sigmoid function, initializes to 1 at the beginning of the iterations and then is incremented, after each routing iteration. Capsule j is activated when σ_j is small enough, and conversely the $\sum_i r_{ij}$ is sufficiently big. What EM routing performs using an EM clustering technique, is to group capsules in order to form a part-whole relationship. The individual capsules in the lower layer vote (i.e., make predictions) on the pose matrices of their possible parent capsules. Each vote is a predicted value for a parent-capsule's pose matrix. If the capsules which are associated to the parts, all vote a similar pose matrix value, we cluster them together to form a parent capsule (e.g., in face detection application, if the nose, mouth and eyes capsules all vote a similar pose matrix value, we cluster them together to form a parent capsule: the face capsule). The entire algorithm as in [16] is depicted, as

Procedure 1 Routing algorithm returns **activation** and **pose** of the capsules in layer $L+1$ given the **activations** and **votes** of capsules in layer L. V_{ij}^h is the h^{th} dimension of the vote from capsule i with activation a_i in layer L to capsule j in layer $L+1$. β_a, β_u are learned discriminatively and the inverse temperature λ increases at each iteration with a fixed schedule.

procedure EM ROUTING($\boldsymbol{a}$, V)
  $\forall i \in \Omega_L, j \in \Omega_{L+1}$: $R_{ij} \leftarrow 1/|\Omega_{L+1}|$
  for t iterations **do**
    $\forall j \in \Omega_{L+1}$: M-STEP($\boldsymbol{a}$, R, V, j)
    $\forall i \in \Omega_L$: E-STEP(μ, σ, $\boldsymbol{a}$, V, i)
  return $\boldsymbol{a}$, M

procedure M-STEP($\boldsymbol{a}$, R, V, j) ▷ for one higher-level capsule, j
  $\forall i \in \Omega_L$: $R_{ij} \leftarrow R_{ij} * \boldsymbol{a}_i$
  $\forall h$: $\mu_j^h \leftarrow \frac{\sum_i R_{ij} V_{ij}^h}{\sum_i R_{ij}}$
  $\forall h$: $(\sigma_j^h)^2 \leftarrow \frac{\sum_i R_{ij} (V_{ij}^h - \mu_j^h)^2}{\sum_i R_{ij}}$
  $cost^h \leftarrow \big(\beta_u + log(\sigma_j^h)\big) \sum_i R_{ij}$
  $a_j \leftarrow logistic(\lambda(\beta_a - \sum_h cost^h))$

procedure E-STEP(μ, σ, $\boldsymbol{a}$, V, i) ▷ for one lower-level capsule, i
  $\forall j \in \Omega_{L+1}$: $\boldsymbol{p}_j \leftarrow \frac{1}{\sqrt{\prod_h^H 2\pi(\sigma_j^h)^2}} \exp\Big(-\sum_h^H \frac{(V_{ij}^h - \boldsymbol{\mu}_j^h)^2}{2(\sigma_j^h)^2}\Big)$
  $\forall j \in \Omega_{L+1}$: $\boldsymbol{R}_{ij} \leftarrow \frac{\boldsymbol{a}_j p_j}{\sum_{k \in \Omega_{L+1}} \boldsymbol{a}_k p_k}$

During the E-step, the assignment probability r_{ij} of each datapoint to a parent capsule is computed. The M-step calculates the values of the Gaussian models based on the determined r_{ij}. In E-step, we re-calculate the r_{ij} probability based on the new μ, σ and a_j. If the vote is closer to the μ of the updated Gaussian model, the assignment would be increased.

In M-step, we use the activation a_i, the current r_{ij} and votes $\boldsymbol{V}$ in order to calculate μ and σ from the children capsules. In M-step we also re-calculate the cost and the activation a_j for the parent capsules.

Finally, the "*spread loss*", L, is used to maximize the gap between the activation of the target class (a_t) and the activation of the other classes, considering a margin, m, so that if the squared distance between them is smaller than m, the loss of

this pair is set to zero:

$$L_i = \max\big(0, m - (a_t - a_i)^2\big), \qquad L = \sum_{i \neq t} L_i \tag{7}$$

III. Experimental Study

The data in our study is provided from **Hoda** dataset which contains handwritten Persian/Arabic images, which contains 60000 training-data, as well as 20000 test-data samples, in grayscale with 32×32 bits resolution. For cross-validation, 5000 data samples is considered for each epoch.

The hardware used for the experiments, was a laptop with CPU 4700MQ, Core i7- 2.4GHz, with 8 GB RAM. All programs are implemented on Tensorflow ver 1.11.0, and Keras ver 2.1.5 software platforms. The training time cost, using the above-mentioned hardware, for every epoch was taken about $75 \sim 80$ minutes.

The high-level architecture of the CapsNet used in this paper is depicted in figure 1. A sample data used for training

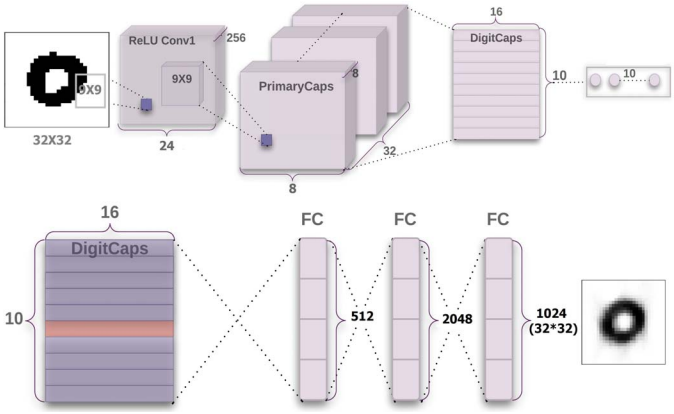
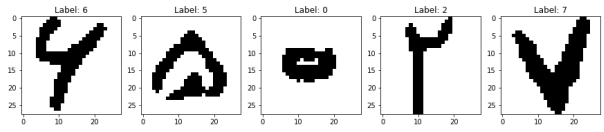
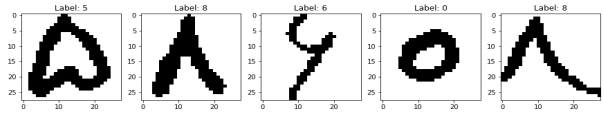


Fig. 1: Capsule Network digit recognizer and decoder

our CapsNet, and for testing it are depicted in figure2. The spread of training data in our dataset has been evaluated using *tSNE* algorithm [19], [20], with two different numbers of data samples of 1000 and 20000, respectively. These clusterings are

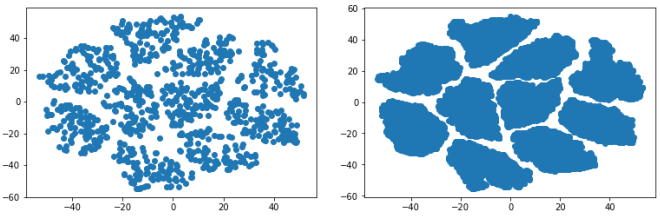


(a)



(b)

Fig. 2: A sample data taken from Hoda dataset; (a) A Training sample, (b) A Test sample



(a) (b)

Fig. 3: tSNE-based Clustering of Persian/Arabic handwritten digits using samples from the Hoda dataset, (a) 1000 samples, (b) 20000 samples

depicted in figure 3. A representation of the capsule data is extracted which is depicted in figure4. Moreover, a generated

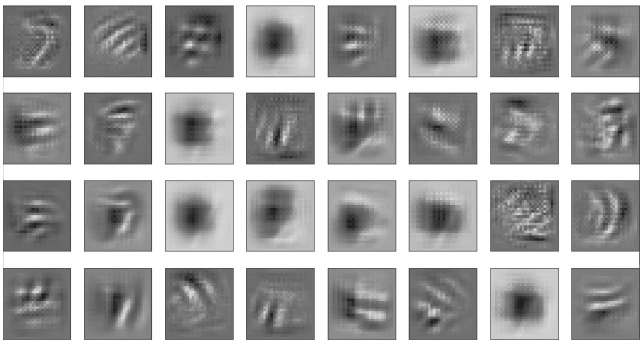


Fig. 4: Representation of the capsule data; 8 capsules in each row, versus 4 epochs in each column

image set from the CapsNet decoder as the initial value, the output of the decoder after epoch 1, and after epoch 2, for digit zero, are respectively depicted in figure 5.

Now, by feeding the test data in the trained network with the explained architecture, the normalized confusion matrix as in figure 6 was achieved.

The complete architecture of the capsule network and detailed parameters are depicted in figure 9. The confusion matrix of figure 6, is obtained using a 400-samples batch size. However, for most of the confusion matrices, the recognizer was 100% successful in classifying the digits. For those otherwise, this figure depicts the worst case happening in the recognition task.

As it was primarily expected, the capsule architecture has equivariance property, which makes it inherently robust against the rotation, transformation and different viewpoints. This property has been properly leveraged in this work to recognize the handwritten digits which are gathered from human in a real environment. As it was already proven to be successful in MNIST recognition task, here it is emphasized that the CapsNet architecture could be involved as the optimum recognizer, compared to the state-of-the-art.

The results of the proposed technique are further compared with some of the state-of-the-art methods which have been already applied on the same dataset, and are shown in Table I.

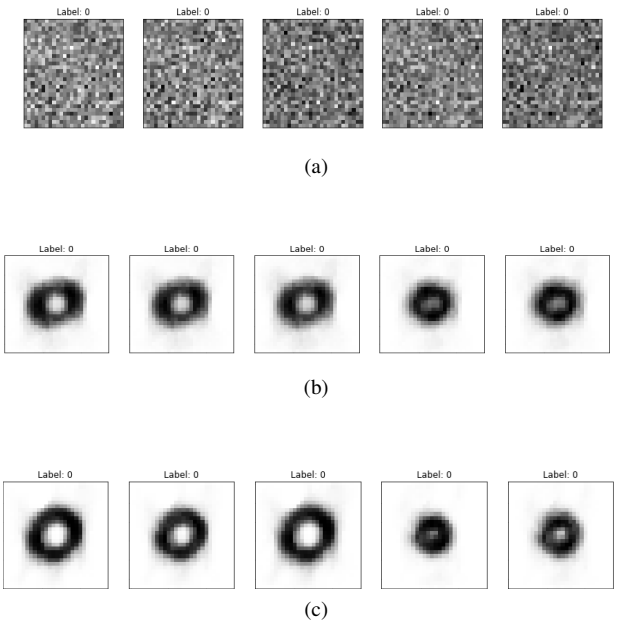


Fig. 5: CapsNet decoder outputs after: (a) initialization (b) epoch-1 (c) epoch-2

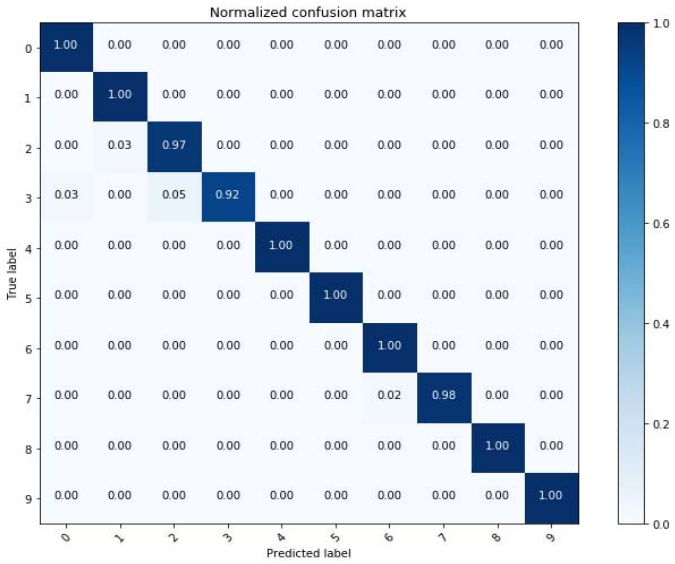


Fig. 6: Normalized confusion matrix for Persian/Arabic handwritten digits of **Hoda** dataset, using the proposed CapsNet

TABLE I: Comparison of algorithms on Hoda dataset

Algorithms	Dataset Size		Accuracy(%)	
	Train	Test	Train	Test
Sadri et al. [10]	7390	3035	—	94.14
Mozaffari et al. [11]	2240	1600	100	94.44
Alaei et al. [21]	60000	20000	99.99	98.71
Alei et al. [8]	60000	20000	99.99	99.02
Proposed	60000	20000	100	**99.87**

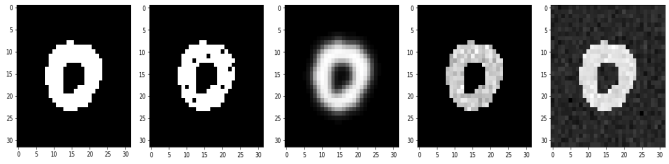


Fig. 7: Noisy images being augmented to the input data, also could not fool the classifier;(left to right): input image, pepper noise, Blurred image, Speckle noise, Gaussian Noise

Similar to Alaei et al. [8], we used 60000 samples for training our CapsNet, and 20000 samples out of Hoda dataset for testing the system. As it is clearly obvious through Table I, the results of our proposed system outperforms the other techniques, even during classification of the noisy input images. For that purpose, we further created some noisy images using different common noises (e.g., pepper noise, Blurring image, Speckle noise, Gaussian noise) and fed them through the classifier, as in Figure 7. Even for the noisy images the system could perfectly classify the input digits.

The total number of misclassified samples out of 20000 was about 26 samples. The total number of trainable parameters are $6,455,376$. These parameters are gathered in a model with almost 98.5 MBytes capacity. The train and test accuracy are also depicted in figure 8.

An advantage of the proposed system over the methods shown in Table I, is the robustness of it against the noise. All the other successful methods mentioned in Table I, use SVM classifier in the core of the algorithm. It is known from theory, that SVM classifier is highly sensible against the noise and outliers. Moreover, the Capsule network has the equivariance property which makes it robust against various transformations over the input images. Most of the conventional classifiers require pre-processing blocks to get rid of the initial transformations and then they are applied to the inputs. This inherent ability of CapsNets, justifies its usage despite the large model parameters it can impose to the model training.

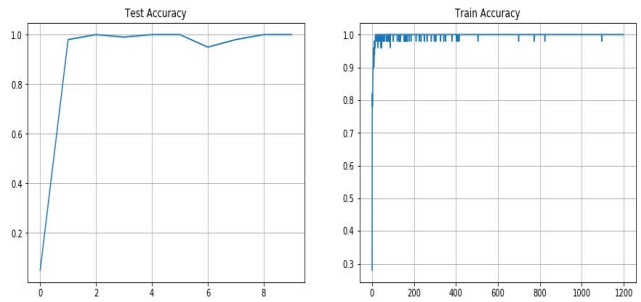


Fig. 8: (left) Test accuracy reaches 99.87 in less than 9 epochs, vs. (right) Train accuracy, for the proposed Capsule Network on **Hoda** dataset. The train accuracy reaches 100 % using minibatches of size 50 (totally 60,000 samples).

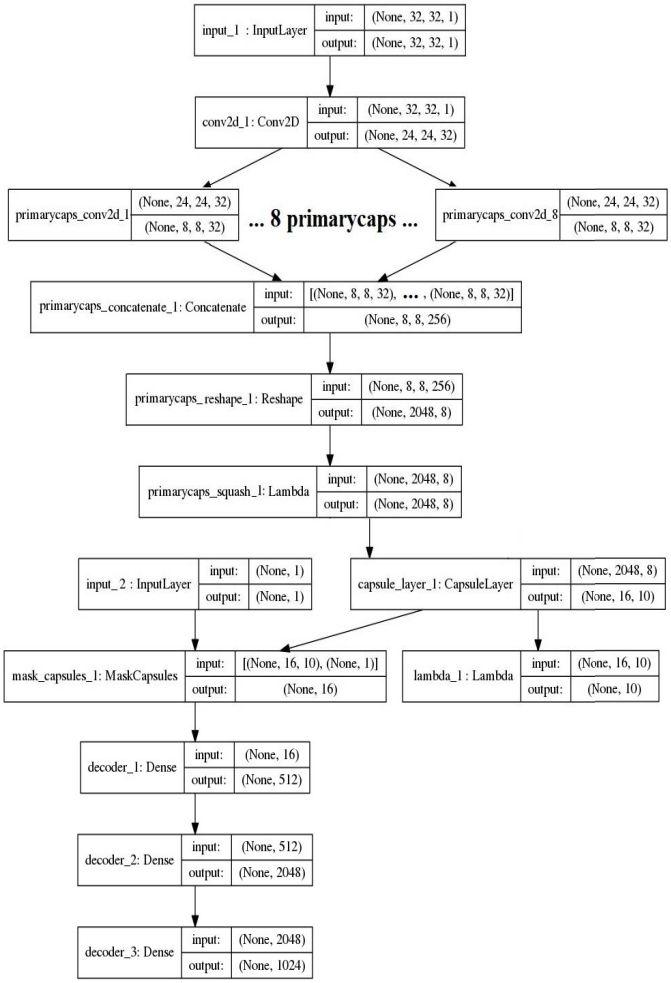


Fig. 9: Detailed Tree-graph of the capsule network architecture, including the input layer, convolution layers, primary capsules, decoders, and associated parameter settings.

IV. Conclusion

In this paper the problem of handwritten digit recognition for Persian/Arabic language, was addressed. For the first time, capsule network was employed and trained using EM algorithm. By comparing the results achieved on Hoda dataset, with the state-of-the-art on the same dataset, we could see the superiority of the capsule network on recognizing the handwritten digits, using all the previously applied methods. The power of the system against the noisy input images are further investigated, and the outputs have been proven to be robust against various noise effects and transformations.

Acknowledgment

Authors of this paper express special thanks to *Huadong Liao* for providing the Capslayer tools for faster implementation of the capsule network. Further, we acknowledge *Amir Sanayian* for providing the Hoda dataset loader.

References

[1] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.

[2] A. G. Hochuli, L. S. Oliveira, A. Britto Jr, and R. Sabourin, "Handwritten digit segmentation: Is it still necessary?" *Pattern Recognition*, vol. 78, pp. 1–11, 2018.

[3] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger *et al.*, "Comparison of learning algorithms for handwritten digit recognition," in *International conference on artificial neural networks*, vol. 60. Perth, Australia, 1995, pp. 53–60.

[4] M. Hamidi and A. Borji, "Invariance analysis of modified c2 features: case study—handwritten digit recognition," *Machine Vision and Applications*, vol. 21, no. 6, pp. 969–979, 2010.

[5] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition: benchmarking of state-of-the-art techniques," *Pattern recognition*, vol. 36, no. 10, pp. 2271–2285, 2003.

[6] H. Li, X. Guo, B. Dai, W. Ouyang, and X. Wang, "Neural network encapsulation," in *European Conference on Computer Vision*. Springer, 2018, pp. 266–282.

[7] H. Khosravi and E. Kabir, "Introducing a very large dataset of handwritten farsi digits and a study on their varieties," *Pattern recognition letters*, vol. 28, no. 10, pp. 1133–1141, 2007.

[8] A. Alaei, P. Nagabhushan, and U. Pal, "Fine classification of unconstrained handwritten persian/arabic numerals by removing confusion amongst similar classes," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*. IEEE, 2009, pp. 601–605.

[9] H. Salimi and D. Giveki, "Farsi/arabic handwritten digit recognition based on ensemble of svd classifiers and reliable multi-phase pso combination rule," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 16, no. 4, pp. 371–386, 2013.

[10] J. Sadri, C. Y. Suen, and T. D. Bui, "Application of support vector machines for recognition of handwritten arabic/persian digits," in *Proceedings of Second Iranian Conference on Machine Vision and Image Processing*, vol. 1, 2003, pp. 300–307.

[11] S. Mozaffari, K. Faez, and M. Ziaratban, "Structural decomposition and statistical description of farsi/arabic handwritten numeric characters," in *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. IEEE, 2005, pp. 237–241.

[12] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3856–3866.

[13] E. Xi, S. Bing, and Y. Jin, "Capsule network performance on complex data," *arXiv preprint arXiv:1712.03480*, 2017.

[14] C. Xiang, L. Zhang, Y. Tang, W. Zou, and C. Xu, "Ms-capsnet: A novel multi-scale capsule network," *IEEE Signal Processing Letters*, vol. 25, no. 12, pp. 1850–1854, 2018.

[15] J. O. Neill, "Siamese capsule networks," *arXiv preprint arXiv:1805.07242*, 2018.

[16] S. Sabour, N. Frosst, and G. E. Hinton, "Matrix capsules with em routing," in *6th International Conference on Learning Representations, ICLR*, 2018.

[17] R. LaLonde and U. Bagci, "Capsules for object segmentation," *arXiv preprint arXiv:1804.04241*, 2018.

[18] A. Lin, J. Li, and Z. Ma, "On learning and learned representation with dynamic routing in capsule networks," *arXiv preprint arXiv:1810.04041*, 2018.

[19] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[20] L. Van Der Maaten, "Accelerating t-sne using tree-based algorithms," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014.

[21] A. Alaei, U. Pal, and P. Nagabhushan, "Using modified contour features and svm based classifier for the recognition of persian/arabic handwritten numerals," in *Advances in Pattern Recognition, 2009. ICAPR'09. Seventh International Conference on*. IEEE, 2009, pp. 391–394.