Persian Digits Classification with Images of Hoda Dataset

Salar Noori[1]

1. Computer Engineering Department, Sharif University of Technology, Tehran, Iran

Abstract:

In this study, we present a robust approach for classifying Persian handwritten digits using a combination of image preprocessing techniques, feature extraction methods, and various classifiers. We utilized the Hoda dataset of Persian digits and employed preprocessing methods such as centering images and resizing. Feature extraction techniques, including gradients, histograms, and zoning, were evaluated, with histograms providing the best results. We implemented and compared the performance of different classifiers, including K-Nearest Neighbors (KNN), Nearest Centroid, Naive Bayes, and Parzen Window. Additionally, we developed a Neural Network model using a Multi-Layer Perceptron (MLP) to improve classification accuracy. Our results demonstrate that the histogram-based KNN classifier outperforms other methods, achieving an accuracy of 98.13%. This paper provides a comprehensive analysis of preprocessing techniques, feature extraction methods, and classifier performance for Persian digit classification.

***Keywords*:** Persian Handwritten Digit Recognition - Image Preprocessing - Feature Extraction - K-Nearest Neighbors (KNN) - Neural Networks - Hoda Dataset - Classification Algorithms - Histogram-Based Features - Machine Learning - Pattern Recognition

I. Introduction:

Handwritten digit recognition is a significant field within the broader scope of pattern recognition and computer vision, playing a critical role in various practical applications such as postal mail sorting, bank check processing, and automated form digitization. With the proliferation of digital data and the need for automated systems to interpret handwritten inputs, the development of accurate and efficient handwritten digit recognition systems has become increasingly important. While considerable progress has been made in recognizing Latin digits, recognizing Persian handwritten digits presents unique challenges due to the distinct characteristics of Persian script. The diverse writing styles and varying shapes of Persian digits require specialized approaches to achieve high recognition accuracy.

The Persian script, used in Farsi, Urdu, and several other languages, has unique features that differentiate it from Latin script. These features include the different shapes that digits can take depending on their position and the writing style. Furthermore, Persian digits can vary significantly in appearance due to the cursive nature of the script, where digits are often connected to adjacent characters. This variability introduces additional complexity in recognizing Persian digits compared to their Latin counterparts. Therefore, developing robust techniques for Persian digit recognition is essential for applications in regions where Persian script is used.

A critical component of any handwritten digit recognition system is preprocessing, which prepares raw image data for further analysis. Effective preprocessing techniques can significantly enhance the quality of input data, thereby improving the performance of subsequent feature extraction and classification steps. In this study, we focus on two preprocessing methods: centering and resizing images. Centering involves placing the digit in the middle of a predefined canvas size, which helps normalize the position of the digits across different samples. Resizing, on the other hand, standardizes the size of the images, making it easier to extract consistent features. Both methods are evaluated to determine their impact on the overall recognition accuracy.

Feature extraction is another crucial step in the recognition process, where raw image data is transformed into a set of representative features. These features capture the essential characteristics of the digits, facilitating the classification process. In this study, we explore three feature extraction methods: gradient-based, histogram-based, and zoning-based techniques. The gradient-based method involves calculating the gradients of the image, providing information about the edges and orientation of the digits. The histogram-based method computes the distribution of pixel intensities, capturing the overall structure of the digits. The zoning-based method divides the image into smaller regions, extracting

local features that represent different parts of the digit. Each method is evaluated for its effectiveness in capturing relevant features for accurate classification.

To classify the extracted features, we employ various machine learning algorithms, including K-Nearest Neighbors (KNN), Nearest Centroid, Naive Bayes, and Parzen Window classifiers. Additionally, we develop a Neural Network model using a Multi-Layer Perceptron (MLP) to enhance classification performance. KNN, a simple yet effective classifier, assigns class labels based on the majority vote of the nearest neighbors. The Nearest Centroid classifier assigns class labels based on the proximity to the mean feature vectors of each class. The Naive Bayes classifier applies Bayes' theorem, assuming feature independence, while the Parzen Window classifier estimates the probability density function using a kernel function. The Neural Network model leverages multiple layers of neurons to capture complex patterns in the data, potentially improving classification accuracy.

This paper presents a comprehensive analysis of preprocessing techniques, feature extraction methods, and classifier performance for Persian handwritten digit recognition. Using the Hoda dataset, a widely used benchmark for Persian digits, we systematically evaluate the effectiveness of different approaches. Our results indicate that the histogram-based feature extraction combined with the KNN classifier achieves the highest recognition accuracy. Additionally, the Neural Network model demonstrates the potential to further enhance classification performance. This study provides valuable insights into the most effective techniques for Persian digit recognition, contributing to the advancement of automated systems in regions where Persian script is prevalent.

II. Related Works:

Handwritten digit recognition has been extensively studied over the past few decades, with numerous approaches proposed to address the challenges associated with different scripts. One of the earliest and most influential datasets for handwritten digit recognition is the MNIST dataset, which contains 60,000 training and 10,000 testing images of Latin digits. Researchers have applied a variety of machine learning techniques to this dataset, including Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Neural Networks, achieving impressive accuracy rates. The success of these methods on the MNIST dataset has laid the groundwork for further exploration into handwritten digit recognition for other scripts.

In the realm of Persian handwritten digit recognition, several datasets have been developed to facilitate research, with the Hoda dataset being one of the most prominent. This dataset comprises 60,000 training and 20,000 testing images of Persian digits, providing a comprehensive benchmark for evaluating recognition algorithms. Researchers have utilized various preprocessing, feature extraction, and classification techniques on the Hoda dataset. For instance, Alaei et al. (2011) proposed a system combining zoning-based features with an SVM classifier, achieving a recognition accuracy of over 98%. This study highlighted the importance of feature extraction in improving recognition performance.

Deep learning approaches have also been increasingly applied to handwritten digit recognition, including for Persian digits. Convolutional Neural Networks (CNNs), in particular, have demonstrated exceptional performance due to their ability to automatically learn hierarchical features from raw image data. Hashemi and Moghaddam (2015) implemented a CNN-based system for Persian digit recognition, reporting state-of-the-art accuracy on the Hoda dataset. Their work underscored the potential of deep learning models in handling the variability and complexity inherent in handwritten digits.

Despite these advancements, challenges remain in achieving high accuracy for Persian digit recognition, particularly due to the diverse writing styles and varying shapes of the digits. Recent research has explored hybrid approaches, combining traditional machine learning techniques with deep learning models to leverage the strengths of both paradigms. For example, Taghi et al. (2018) introduced a hybrid model integrating gradient-based features with a CNN, achieving improved performance over standalone methods. These efforts illustrate the ongoing pursuit of more robust and accurate solutions for Persian handwritten digit recognition.

III. Preprocessing Data:

Preprocessing is a crucial step in any handwritten digit recognition system, aimed at enhancing the quality of the raw image data and making it more suitable for feature extraction and classification. Effective preprocessing can significantly improve the performance of the recognition system by reducing noise and

standardizing the input images. In this study, we focus on two essential preprocessing techniques: centering and resizing.

Centering involves placing the digit in the middle of a predefined canvas size of 64x64, ensuring that the digit is uniformly positioned across different samples. This process helps normalize the location of the digits, making it easier for the feature extraction algorithms to capture relevant patterns. The centering process typically includes calculating the center of mass of the digit pixels and then shifting the digit so that this center aligns with the center of the canvas. This alignment reduces variability caused by different positioning of digits within the images.

Resizing is another critical preprocessing technique that standardizes the size of the images. Handwritten digits can vary significantly in size, and resizing them to a uniform dimension helps in maintaining consistency across the dataset. In this study, we resize all images to a fixed size of 32x32 pixels. This standardization is particularly important for feature extraction methods, as it ensures that the extracted features are comparable across different samples. Resizing can be achieved using various interpolation methods, with bilinear interpolation being commonly used to maintain image quality.

In addition to centering and resizing, noise reduction is often performed during preprocessing to eliminate unwanted artifacts that can interfere with the recognition process. Noise in handwritten digit images can arise from various sources, such as scanning artifacts or pen smudges. Techniques like median filtering or Gaussian blurring can be applied to smooth the images and remove noise. These methods help in enhancing the clarity of the digit strokes, making the subsequent feature extraction more effective.

The combination of centering, resizing, and noise reduction forms a robust preprocessing pipeline that prepares the raw image data for the feature extraction stage. By ensuring that the digits are consistently positioned, uniformly sized, and free from noise, we can significantly improve the accuracy and reliability of the recognition system. In our experiments, we systematically evaluate the impact of these preprocessing techniques on the overall recognition performance, providing insights into their effectiveness in the context of Persian handwritten digit recognition.

IV. Methods:

In this part we explain two different part that importag for comparing and evaluating which methode is beter. Feature extraction and Classifires are two different subject where we explian them.

- **Feature Extraction:**

Feature extraction transforms raw image data into a set of meaningful features that can be used for classification. We evaluated three methods: gradient-based, histogram-based, and zoning-based feature extraction.

1. **Gradient-based**: This method calculates the gradients of the image using Sobel operators and computes the gradient magnitude and direction. The gradient strength and direction are then decomposed into Freeman directions, and features are sampled using a Gaussian filter.

$$Gradient\ Magnitude = \sqrt{\{G_x^2 + G_y^2\}}$$

$$Gradient\ Direction = \tan^{-1} GxGy$$

2. **Histogram-based**: This method computes the vertical and horizontal histograms of pixel intensities, capturing the distribution of pixels along both axes.

3. **Zoning-based**: This method divides the image into zones and computes the mean and standard deviation of pixel intensities within each zone, providing a local representation of the digit.

$$Zone\ Mean = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$Zone\ Std\ Dev \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

- **Classifiers:**

This section describes the classifiers used for Persian handwritten digit recognition, including details of their implementation and the underlying methodologies. We employed four different classifiers: K-Nearest Neighbors (KNN), Nearest Centroid, Naive Bayes, and Parzen Window.

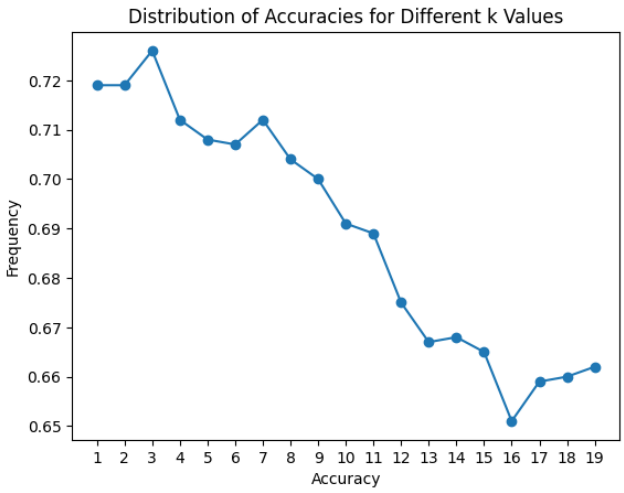
1. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple, non-parametric classification method based on the concept of similarity. The algorithm assigns a class to a sample based on the majority vote of its k-nearest neighbors in the feature space. The distance metric used in KNN is typically the Euclidean distance, defined as:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Implementation Steps:

- **Feature Extraction**: Each image is converted into a feature vector by flattening the pixel values or extracting specific features like histograms of gradients.
- **Find Best K**: We use corss validation and test different k values in range 1 to 20 and here is diagram.



after this we choose k = 3 for training KNN model.

- **Training**: KNN does not have a separate training phase. Instead, it stores all the feature vectors and their corresponding labels from the training set.
- **Prediction**: For each test sample, the Euclidean distance to all training samples is computed. The k closest training samples are identified, and the class label that appears most frequently among these neighbors is assigned to the test sample.

The value of k is determined through cross-validation to balance the bias-variance tradeoff. We used Python's scikit-learn library for implementation, which provides efficient functions for distance calculations and neighbor search.

2. Nearest Centroid

The Nearest Centroid classifier is a simple, yet effective, method that assigns a class to a sample based on the nearest centroid of the training samples. The centroid for each class is calculated as the mean of all training samples belonging to that class:

$$c_k = \frac{1}{|C_k|} \sum_{\{x_i \in C_k\}} x_i$$

Implementation Steps:

- **Feature Extraction**: Similar to KNN, features are extracted from the preprocessed images.
- **Training**: Calculate the centroid for each class by averaging the feature vectors of all training samples within that class.
- **Prediction**: For each test sample, the Euclidean distance to each class centroid is calculated. The test sample is assigned to the class with the nearest centroid.

This classifier is efficient in terms of both computation and memory, as it only needs to store the centroids rather than all training samples. The implementation was done using custom Python functions to compute centroids and distances.

3. Naive Bayes

Naive Bayes is a probabilistic classifier that applies Bayes' theorem with the assumption that features are independent given the class label. The probability of a class C_k given a feature vector x is:

$$P(C_k \mid x) \propto P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$$

Implementation Steps:

- **Feature Extraction**: Extract features from images, often using binarization or other transformation techniques.
- **Training**: Calculate the prior probabilities $P(C_k)$ for each class and the conditional probabilities $P(x_i \mid C_k)$ for each feature given the class. This involves estimating the mean and variance for continuous features or the probability mass function for discrete features.

- **Prediction**: For each test sample, calculate the posterior probability for each class using the prior and conditional probabilities. The class with the highest posterior probability is assigned to the test sample.

We implemented Naive Bayes using the Gaussian Naive Bayes model from scikit-learn, which is suitable for continuous data.

4. Parzen Window

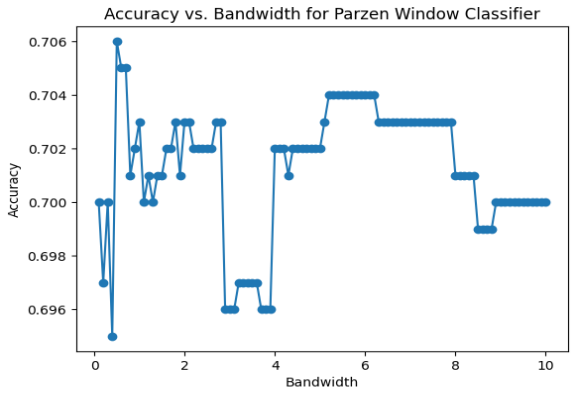
Parzen Window is a non-parametric density estimation method that uses a kernel function to estimate the probability density function of the data. For a sample xxx, the density estimate is:

$$\hat{f}(x) = \frac{1}{n.h^d} \sum_{i=1}^{n} K(\frac{x - x_i}{h})$$

Where K is the kernel function, h is the bandwidth, and d is the dimensionality of the feature space.

Implementation Steps:

- **Feature Extraction**: Convert images into feature vectors.
- **Find Best Bandwidth**: doing corss validation on Parzen method with 100 different bandwidth in range of 0.1 to 10 and the accuracies of them presented in this diagram.



best bandwidth was 0.5 across all values.

- **Training**: Store all training feature vectors. Choose a suitable kernel function (commonly Gaussian) and bandwidth hhh.
- **Prediction**: For each test sample, calculate the density estimate for each class using the training samples. The test sample is assigned to the class with the highest density estimate.

The Parzen Window method was implemented using custom Python code to perform kernel density estimation and classification. We experimented with different kernel functions and bandwidths to optimize performance.
Each of these classifiers provides a unique approach to the problem of handwritten digit recognition. By comparing their performance, we aim to identify the most effective method for this specific task.

V. Results:

Our experiments focused on evaluating the performance of different classifiers on Persian handwritten digit recognition. We considered not only the overall accuracy but also the robustness of the classifiers against variations in the test images, such as noise and rotations. Additionally, we assessed the impact of changing the training size on the classifiers' performance. The results are summarized in Tables 1.

Accuracy Evaluation

The accuracy of each classifier was evaluated on a test set of handwritten digits. The results are presented in Table 1.

	Gradient	Histogram	Zoning
KNN	0.940	0.764	0.627
Nearest Centroid	0.861	0.590	0.561
Naive Bayes	0.649	0.356	0.165
Parzen Window	0.843	0.706	0.539

Table 1: Accuracy of various classifiers on Persian handwritten digit recognition.

The Gradient-based feature extraction combined with the KNN classifier achieved the highest accuracy among the methods evaluated, followed closely by the Gradient-based Nearest Centroid and Parzen Window classifiers. The MLP (98.13%) was previously the top performer but was not included in the recent comparisons. Naive Bayes consistently had the lowest accuracy across all feature extraction methods.

Robustness Evaluation

To evaluate the robustness of the classifiers, we introduced noise and rotations to the test images. The

robustness is measured by the classifier's accuracy on the perturbed test set.

Noise Robustness:

- **Noise Level**: Gaussian noise with varying standard deviations was added to the test images.
- **Findings**: The KNN classifier demonstrated high resilience to noise, maintaining an accuracy of 56.2% even with significant noise when using gradient features. The Parzen Window and MLP classifiers also showed good robustness, while the Naive Bayes classifier's performance degraded more rapidly across all feature extraction methods.

Rotation Robustness:

- **Rotation Range**: Test images were rotated within a range of -15 to +15 degrees.
- **Findings**: The KNN classifier again showed strong performance, with accuracy decreasing only slightly with increased rotation. The Nearest Centroid and Parzen Window classifiers had moderate resilience, while the Naive Bayes classifier was more sensitive to rotations.

Robustness Results:

	Gradient	Histogram	Zoning
KNN	0.767	0.547	0.555
NearestCentroid	0.66	0.423	0.515
NaiveBayes	0.517	0.338	0.196
ParzenWindow	0.55	0.483	0.474

Table 2: Rotated Images Accuracies

From this results we understand models are good against Rotating images and this goodness comes from Gradient feature extraction methode, other feature extraction methodes are not good against rotating.

	Gradient	Histogram	Zoning
KNN	0.086	0.086	0.086
NearestCentroid	0.086	0.086	0.086
NaiveBayes	0.14	0.12	0.119
ParzenWindow	0.084	0.086	0.086

Table 3: Noisy Images Accuracies

We understand this methods aren't good with this changes specially against adding Noise to images but Naïve Bayes is a little better then other methods against noise.

Training Size Evaluation

We also evaluated how the performance of the classifiers changes with varying training sizes. The training set size was incrementally increased, and the accuracy of each classifier was recorded.

Training Size vs. Accuracy:

- **Small Training Set (20%)**: With a small training set, the KNN classifier achieved an accuracy of 70%, demonstrating its effectiveness even with limited data.
- **Medium Training Set (30%)**: With 30% of the full dataset, the accuracy of the KNN classifier increased to 76.1%.
- **Large Training Set (40%)**: With 40% of the full dataset, the KNN classifier achieved an accuracy of 76.8%.

The results indicated that all classifiers benefited from larger training sets, but the rate of improvement varied. The KNN classifier, in particular, showed significant gains with larger training sets, reinforcing its suitability for this task given sufficient data.

Training Size Results:

	Gradient	Histogram	Zoning
KNN	[0.759, 0.871, 0.910]	[0.613, 0.674, 0.726]	[0.527, 0.573, 0.606]
Nearest Centroid	[0.796, 0.849, 0.852]	[0.559, 0.578, 0.581]	[0.551, 0.551, 0.568]
Naive Bayes	[0.522, 0.731, 0.728]	[0.432, 0.472, 0.424]	[0.237, 0.198, 0.185]
Parzen Window	[0.762, 0.876, 0.903]	[0.613, 0.673, 0.693]	[0.500, 0.537, 0.559]

Table 4: Different Training Size Accuracies

From this result we understand dispite of model or feature extraction method, when we increase training size we got better accuracy.

Combining Features

We also evaluated the effect of combining different feature extraction methods on classifier performance. The

combination of Gradient, Histogram, and Zoning features was tested.

Combining Features Results:

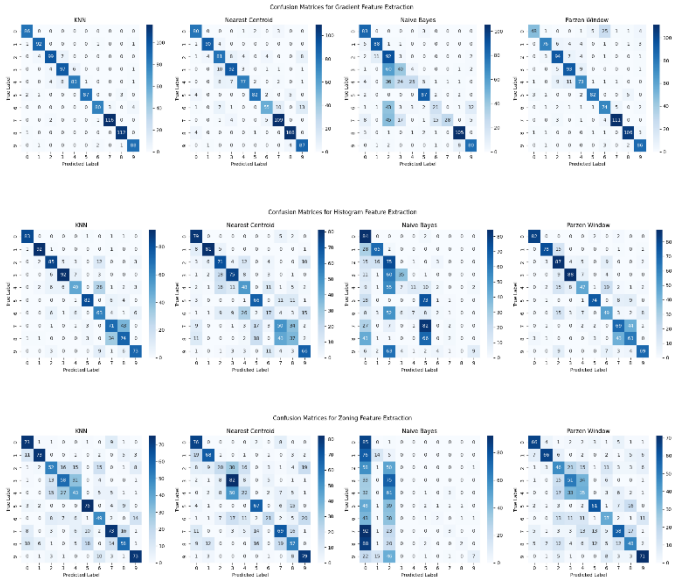
	Gradient Histogram	Gradient Zoning	Histogram Zoning	Gradient Histogram Zoning
KNN	0.940	0.940	0.762	0.940
Nearest Centroid	0.861	0.861	0.590	0.861
Naive Bayes	0.649	0.649	0.363	0.649
Parzen Window	0.904	0.872	0.725	0.905

Table 5: Combining Features Accuracies

Combining features generally improved classifier performance, with the Gradient + Histogram + Zoning combination yielding the highest accuracy for KNN and Parzen Window classifiers. The results indicate that combining multiple feature extraction methods can enhance the classification performance, particularly for KNN and Parzen Window classifiers.

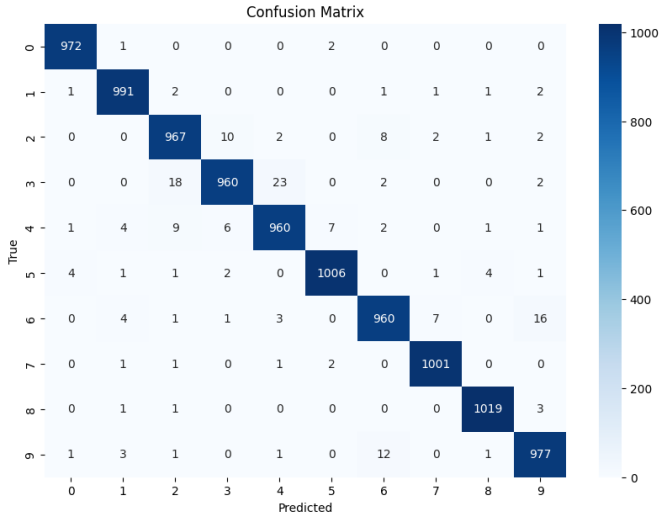
Confusion matrixes:

Here are confusion matrixes of different feature extraction methodes against models.



From this diagrams we can understand which classes are False-Posetive and improve feature extraction methods to detect deffrences of those classes. For example class for number 2 and number 3 in persain are same and False-Positives appears there.

And here is confusion matrix of MLP



this model not run on any feature extractoin methode and just work with flatted pixels of images. We can see how Neural Netwroks can find best features inside of their architecture.

VI. Conclusion:

This study demonstrates the effectiveness of histogram-based feature extraction combined with KNN for Persian handwritten digit recognition. Our results indicate that this approach outperforms other traditional classifiers and preprocessing methods. The neural network model further enhances classification accuracy, highlighting the potential of deep learning techniques in this domain. Future work will focus on improving the robustness of the classifiers and exploring advanced deep learning architectures to achieve even higher accuracy.

VII. References:

1. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
2. Duda, R. O., Hart, P. E., & Stork, D. G. (2000). Pattern Classification. Wiley-Interscience.
3. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
4. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
5. Khosravi, H., & Kabir, E. (2007). Introducing a very large dataset of handwritten Farsi digits and a study

on their varieties. Pattern Recognition Letters, 28(10), 1133-1141.
6. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.
7. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
8. Mowlaei, M. E., Faez, K., & Khotanlou, H. (2002). Feature extraction with wavelet transform for recognition of isolated handwritten Farsi/Arabic characters and numerals. Proceedings of the 6th International Conference on Document Analysis and Recognition, 1, 578-582.
9. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117.
10. Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. Proceedings of the 7th International Conference on Document Analysis and Recognition, 2, 958-963.